SY33-8041-1
File No. S370-21 (OS/VS, VM/370)

Systems

OS/VS–VM/370 Assembler Logic

IBM

Second Edition (March, 1974)

This is a major revision of, and obsoletes, SY33-8041-0 and Technical Newsletters SN33-8152 and SN33-8158. Support for VM/370 has been added. Other changes to the text and to illustrations are indicated by a vertical line to the left of the change.

This edition applies to release 3 of OS/VS1 and release 2 of OS/VS2, release 2 of VM/370 and to all subsequent modifications until otherwise indicated in new editions or Technical Newsletters. Changes are continually made to the specifications herein; before using this publication in connection with the operation of IBM systems, consult the IBM System/360 and System/370 Bibliography, Order No. GA22-6822 for the editions that are applicable and current.

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

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

A form is provided at the back of this publication for readers' comments. If the form has been removed, comments may be addressed to IBM Nordic Laboratory, Product Communications, Box 962, S-181 09 Lidingo 9, Sweden. Comments become the property of IBM.

Preface

This program logic manual is written for OS/VS - VM/370 customer engineers and programmers maintaining the Assembler. The manual describes the structure, logic, and operation of the assembler.

Prerequisites

This manual was written with the assumption that the reader has:

- a good knowledge of the assembler language, including its macro and conditional assembly facilities. This language is covered in OS/VS - DOS/VS - VM/370 Assembler Language, Order Number GC33-4021.
- a good knowledge of System/370 and System/360 machine instructions. Machine instructions are described in IBM System/370 Principles of Operation, Order Number GA22-7000, and IBM System/360 Principles of Operation, Order Number GA22-6821.
- a good knowledge of how to use the assembler. This is covered in the OS/VS - VM/370 Assembler Programmer's Guide, Order Number GC33-4021.

How this Manual is Organized

The "Introduction" contains a summary of general information about the program.

"Method of Operation" describes the functional objectives of the assembler. Method of Operation diagrams highlight the inputs, processing, and outputs of the assembler functions. The diagrams are accompanied by text describing the functions in more detail and cross-references to the program elements that perform the functions.

"Program Organization" describes how the program is divided into units. The section contains detailed charts of how the assembler phases use main storage and diagrams showing the flow of data and control between assembler phases.

The "Directory" serves as a cross-reference between items in the "Method of Operation" and "Program Organization" sections and to the microfiche listings.

"Data Areas" contains detailed layouts of data areas to help in interpreting storage dumps.

"Diagnostic Aids" contains information designed to be helpful in debugging.

The appendixes contain information about error message origin, macro and copy code usage, meta text flags, internal operation codes, entry points and EXTRN symbols, record formats, and the internal character set.

Additional Literature

OS/VS Supervisor Services & Macros, Order Number GC28-6646.

OS/VS Data Management Macro Instructions, Order Number GC26-3793.

OS/VS Data Management for System Programmers, Order Number GC28-6550.

Contents

Illustrations

Introduction

The OS/VS - VM/370 Assembler is the OS/VS - VM/370 assembler language processor. It is a three-pass assembler, with one pass over the source deck for editing, one pass for macro-generation and symbol resolution, and a third pass for final assembly.

Purpose and Function

The assembler translates a source program coded in assembler language into a relocatable machine language object program. The assembler assigns relative storage loctions to instructions and other program elements and performs auxiliary assembler functions specified by the programmer. The object modules produced by the assembler are in the format required by the linkage editor. They can be link-edited with object modules produced by other language processors.

Compatibility

The language supported by the OS/VS - VM/370 Assembler is compatible with the language of Assembler F. All programs which assemble error free on Assembler F will also assemble error free on the OS/VS - VM/370 Assembler. Because the language supported by the OS/VS - VM/370 Assembler has more capacity than that supported by Assembler F, some attribute values which are undefined in F will be replaced by the true values. These extensions and the extended SETC facility might, in odd cases, produce different results.

Language Supported

The language supported by the assembler is defined in the publication: *OS/VS - DOS/VS - VM/370 Assembler Language*, Order Number GC33-4010.

Environmental Characteristics

SYSTEM CONFIGURATION

The Assembler will operate on the minimum system configuration required for OS/VS and VM/370.

SYSTEM INTERFACE

All system dependent functions and operations are handled by the assembler's interface modules. The interface modules are:

DMSASM	VM/370 interfaces
DMSASD	VM/370 interfaces
IFOX0A	Driver routines
IFOX0B	Workfile I/O and core management
IFOX0C	Master common work area
IFOX0D	Assembler initialization
IFOX0E	Input common work area
IFOX0F	Input I/O module
IFOX0G	Output common work area
IFOX0H	Output I/O module
IFOX0I	Abort routines
IFOX0J	Assembler options parameters

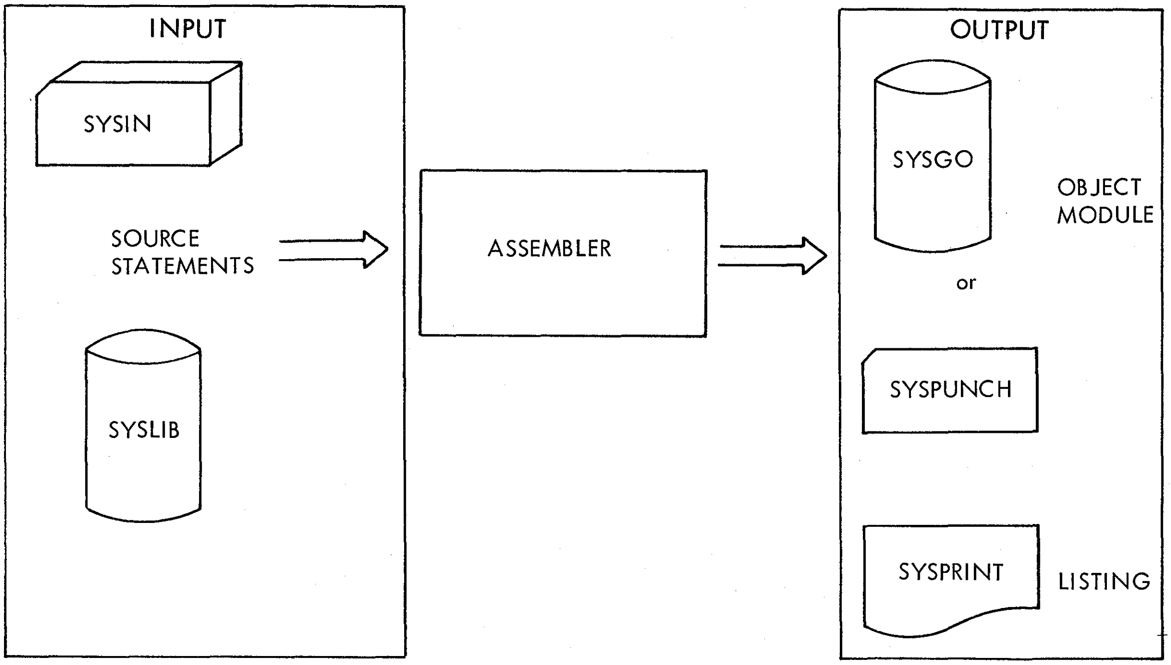
PHYSICAL CHARACTERISTICS

The assembler is made up of 16 reentrant load modules which reside on the link library.

Operational Considerations

INPUT AND OUTPUT

Input to the assembler is source code from SYSIN, SYSLIB, or a private library. Output is an object module and an optional deck and/or listing.



Control Information

As the assembler is a processing program operating under OS/VS, control information is passed to the operating system be means of job control statements. The assembler options are specified in the PARM field of the EXEC job control statement. For an explanation of these options, see OS/VS - VM/370 Assembler Programmer's Guide, Order Number GC33-4021.

Method of Operation

Purpose

The purpose of this section is:

- To give a functional description of the assembler.
- To provide a cross reference from any given description to the listing and to other parts of the manual.

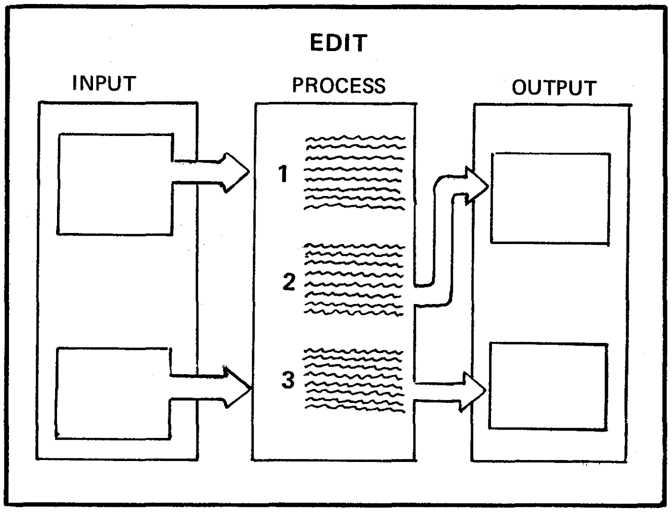
How this Section is Organized

This section consists of diagrams which are arranged in a hierarchy as shown in the foldout located at the back of the manual.

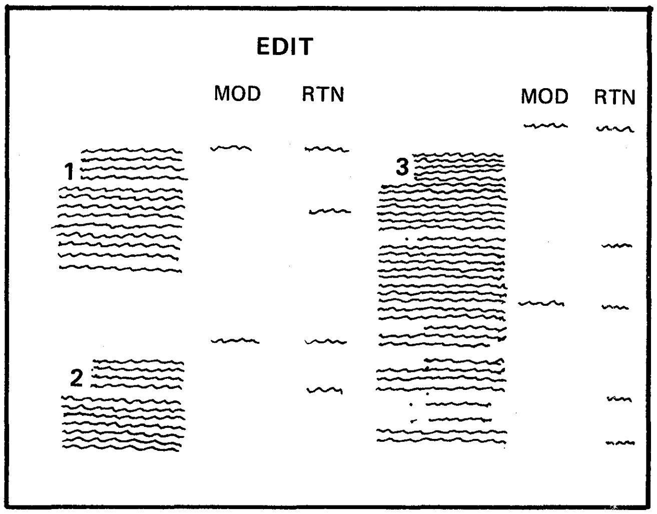
With each diagram is an "extended description" which contains detailed information about the function or subfunction shown in the diagram.

How to Read the Diagrams and Descriptions

Each diagram is divided into three parts: input, process, and output. The input part shows the data before it is processed; the process part shows, in abbreviated form, what is done to the data; and the output part shows what the data is after it has been processed.



Diagram



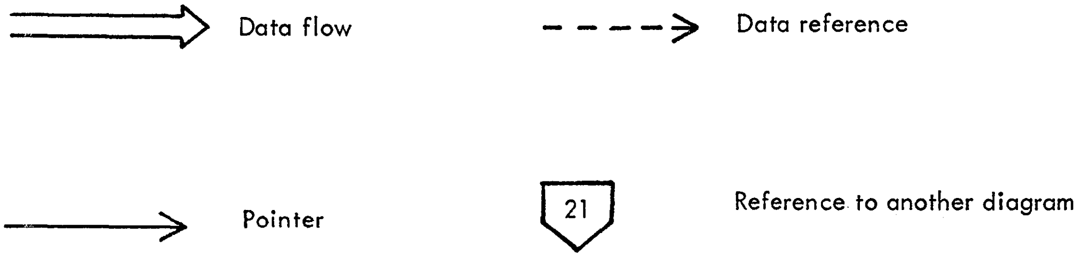
Extended Description

Data areas are identified on the diagrams in two ways: by main-storage address and by DSECT name. Data areas as shown on the diagrams are highly schematic. For complete and accurate data area layouts, see the section "Data Areas".

Many of the data areas and routines are mentioned in two or more diagrams. For a cross-reference of these items to the diagrams in this section, use the "Directory" section of the manual. The Directory also cross references the appropriate microfiche card if you wish to go directly to the listing.

The extended descriptions are keyed by process step to the diagrams and describe the process in more detail. In addition, the extended descriptions give the names of the module and routine that perform the function.

The following symbols are used in the diagrams:



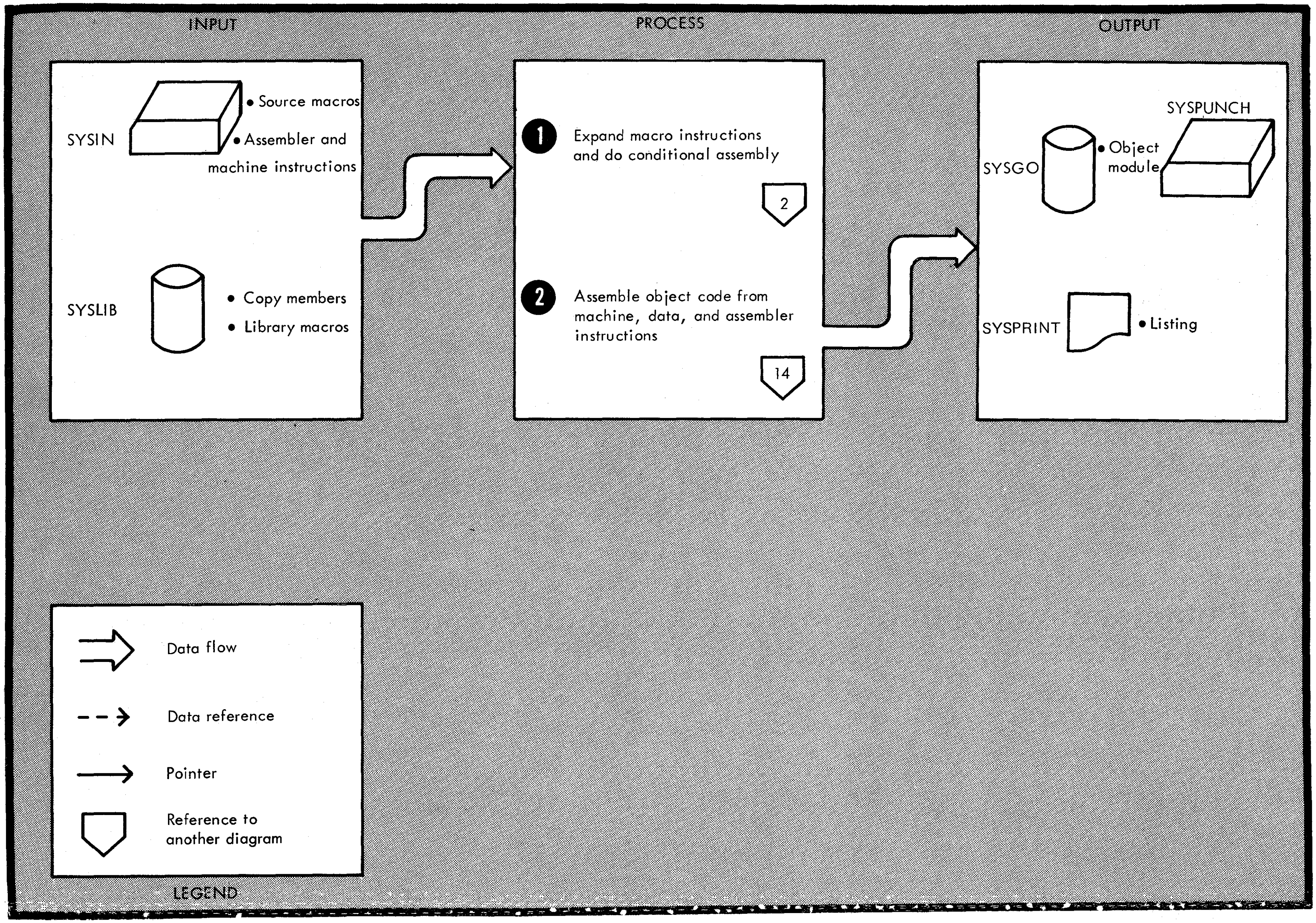
Relation of the Diagrams to Program Phases

Since the diagrams are broken down by function of the assembler, they are not organized exactly like the phases of the assembler. Below is a table showing which diagrams cover which phases.

Phase	Diagram
Initialization	27
Edit	3, 4, 5, 6, 7
Dictionary Interlude	8, 9, 10
Generation	11, 12, 13
Symbol Resolution	15, 16, 17, 18, 19, 20
Assembly	21, 22, 23, 24, 25
Post Processor	26
Diagnostic	21

Generate Object Code from Source Code

1


INPUT
PROCESS
OUTPUT
SYSIN
• Source macros
• Assembler and machine instructions
SYSLIB
• Copy members
• Library macros
1
Expand macro instructions and do conditional assembly
2
2
Assemble object code from machine, data, and assembler instructions
14
SYSPUNCH
SYSGO
• Object module
SYSPRINT
• Listing
Data flow
Data reference
Pointer
Reference to another diagram
LEGEND

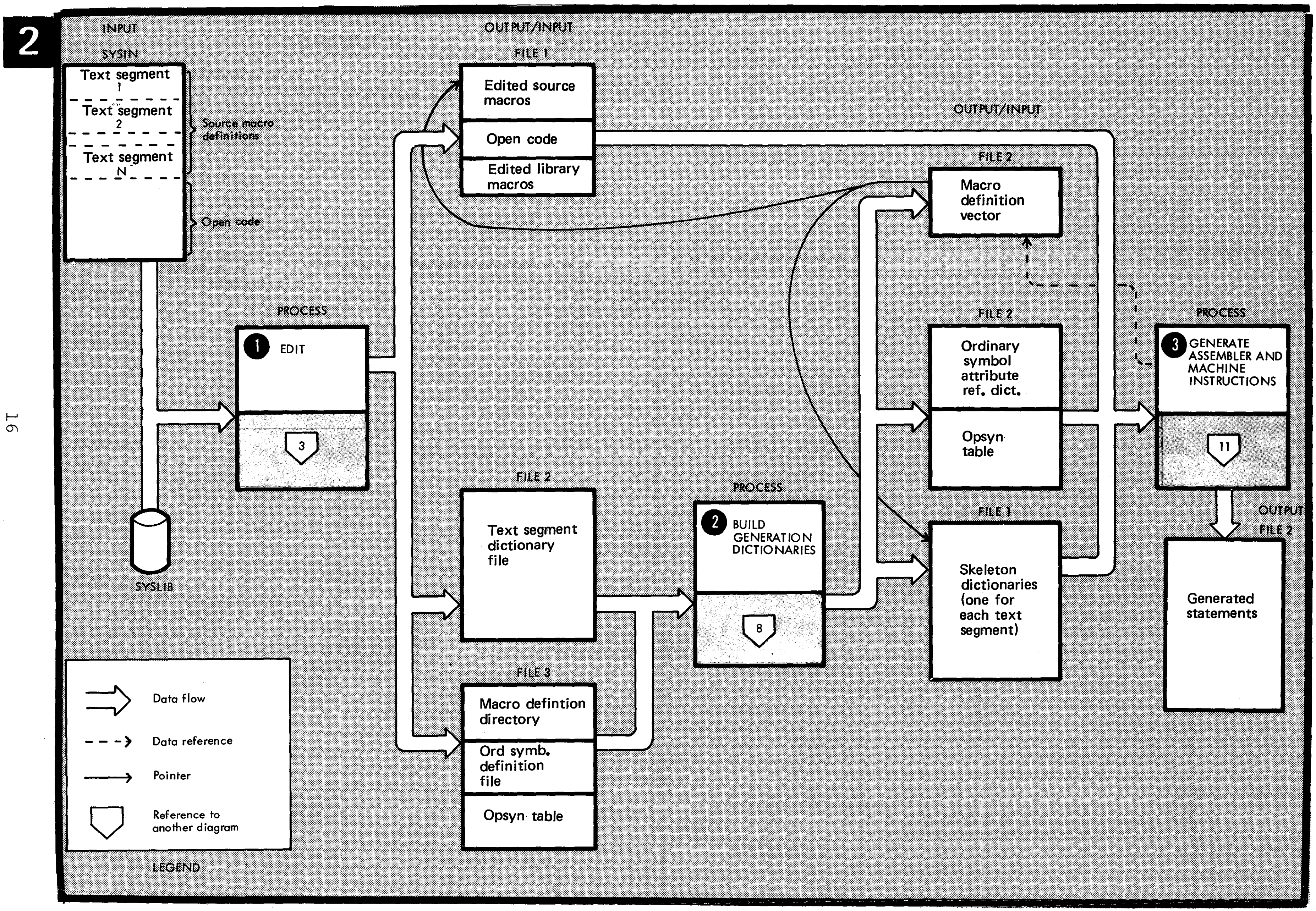
Generate Object Code from Source Code (cont.)

Input to the assembler is source statements in the following forms: SYSIN: source macro definitions and machine and assembler instructions; SYSLIB: COPY members (which may also contain macro definitions) and library macro definitions (either IBM-supplied or installation-written).

❶ Source statements are read and macro instructions expanded according to their definitions and the results of the conditional assembly. Conditional assembly in open code is also performed.

❷ When all macro instructions have been expanded and all conditional assembly performed, the source statements are assembled into object code. Output is an object module (either on SYSGO or SYSPUNCH) and a listing.

Expand Macro Instructions and Do Conditional Assembly



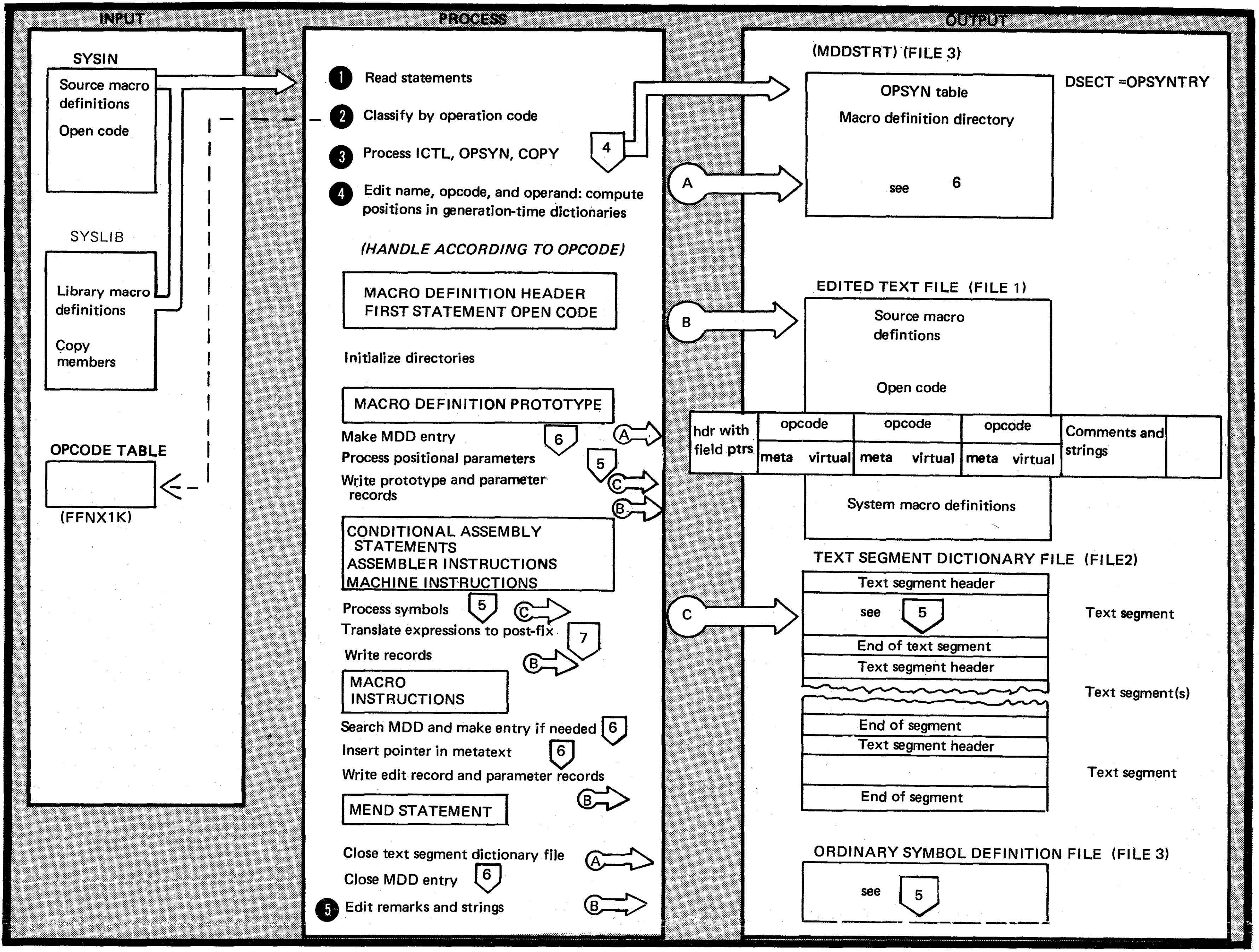
Expand Macro Instructions and Do Conditional Assembly (cont.)

	MODULE
1 Source statements are read from SYSIN and SYSLIB. They are formatted, and expressions are translated to postfix notation. Positions for symbol values in in generation-time dictionaries are computed and pointers to the positions inserted in the records.	IFNX1A IFNX1J IFNX1S
The edited records are written on the edited text file (file I) which is passed to Generate Assembler and Machine Instructions (Diagram 11).	
Another editing function is to collect information needed to build generation-time dictionaries. The sizes of the dictionaries are calculated and in some cases data is collected to fill them. This information is collected in the text segment dictionary file and in the macro definition directory (files 2 and 3).	

	MODULE
2 Information collected in the text segment dictionary file and the macro definition directory is used to build (and in some cases fill) the dictionaries to be used during generation. The macro definition vector, which serves as a link between a macro call, its definition, and the dictionaries necessary to expand the macro, is also built.	IFNX2A
3 The edited text file is read and the dictionaries are used to produce assembler and machine instructions from the macro instructions and conditional assembly instructions. The output contains no macros or conditional assembly statements.	IFNX3A

3

Edit



INPUT

SYSIN

Source macro definitions

Open code

SYSLIB

Library macro definitions

Copy members

OPCODE TABLE

(FFNX1K)

PROCESS

1 Read statements

2 Classify by operation code

3 Process ICTL, OPSYN, COPY 4

4 Edit name, opcode, and operand: compute positions in generation-time dictionaries

(HANDLE ACCORDING TO OPCODE)

MACRO DEFINITION HEADER
FIRST STATEMENT OPEN CODE

Initialize directories

MACRO DEFINITION PROTOTYPE

Make MDD entry 6 A

Process positional parameters 5

Write prototype and parameter records C B

CONDITIONAL ASSEMBLY STATEMENTS
ASSEMBLER INSTRUCTIONS
MACHINE INSTRUCTIONS

Process symbols 5 C

Translate expressions to post-fix 7

Write records B

MACRO INSTRUCTIONS

Search MDD and make entry if needed 6

Insert pointer in metatext 6

Write edit record and parameter records B

MEND STATEMENT

Close text segment dictionary file A

Close MDD entry 6

5 Edit remarks and strings B

OUTPUT

(MDDSTRT) (FILE 3)

DSECT =OPSYNTRY

OPSYN table

Macro definition directory

A

see 6

EDITED TEXT FILE (FILE 1)

B

Source macro defintions

Open code

hdr with field ptrs	opcode		opcode		opcode		Comments and strings	
	meta	virtual	meta	virtual	meta	virtual		

System macro definitions

TEXT SEGMENT DICTIONARY FILE (FILE2)

C

Text segment header

see 5

End of text segment

Text segment

Text segment header

Text segment(s)

End of segment

Text segment header

End of segment

Text segment

ORDINARY SYMBOL DEFINITION FILE (FILE 3)

see 5

Edit (cont.)

Editing consists of converting records into an internal format suitable for processing; inserting pointers to generation-time dictionaries for variable symbols, sequence symbols, and ordinary symbol attribute references; and translating expressions into postfix notation. Each record is split into "virtual" text (a copy of the input record separated by fields) and "metatext" (either a pointer to where a symbol´s value will be found at generation time or an expression translated into postfix notation for generation-time evaluation). The order of editing is opcode, name, operand, remarks and strings.

	MODULE	ROUTINE (LABEL)
1 Statements are read from SYSIN or from SYSLIB (in the case of COPY code and library macro definitions). Edit scans past the name field to the opcode field (but saves the name field). If the statement is a comment, the complete record is written immediately.	IFNX1A	READNEXT (RDSRC) (GSCAN)
2 The opcode is checked against the OPSYN and OPCODE tables. Errors in opcode or a statement´s position in the source file cause error messages to be generated. If the opcode is a variable symbol, the statement is processed as a machine instruction (see below).	IFNX1A IFNX1A IFNX1J	TBLOPS STMTSEQ (OPERCODE)
3 ICTL, OPSYN, and COPY statements are processed (see Diagram 4).	IFNX1A	TBLOPS (ICTL) (OPSYN) (COPY)

	MODULE	ROUTINE (LABEL)
4 The statements are handled according to their opcode:	IFNX1A	TBLOPS
Macro Definition Header First Statement of Open Code		
The variable symbol definition directory, the sequence symbol reference directory, the ordinary symbol attribute reference directory, and the text segment directory file are initialized (see Diagram 5).	IFNX1A IFNX1J IFNX1J	MACRO (MACRENT) (OPENENT)
Macro Definition Prototype		
• Make MDD Entry (see Diagram 6).	IFNX1A	PROTOIN
• Variable symbols (positioned and keyword parameters) in the operand are processed (see Diagram 5).	IFNX1A IFNX1J	VARSYM (VARBSYMD)
• The prototype record is then written on the edited text file. Also, one parameter record is written for each keyword parameter followed by an "end of all parameters" record.	IFNX1A	NEXTPM
Conditional Assembly Statements Assembler Instructions Machine Instructions		
• Variable symbols, sequence symbols, and ordinary symbol attributes are processed (see Diagram 5).	IFNX1J	VARBSYMD VARBSYMR SEQSYMBD SEQSYMBR ORDSYMBR
• Expressions are translated into postfix notation (see Diagram 7).	IFNX1S	
• Edit records are written on the edited text file.		

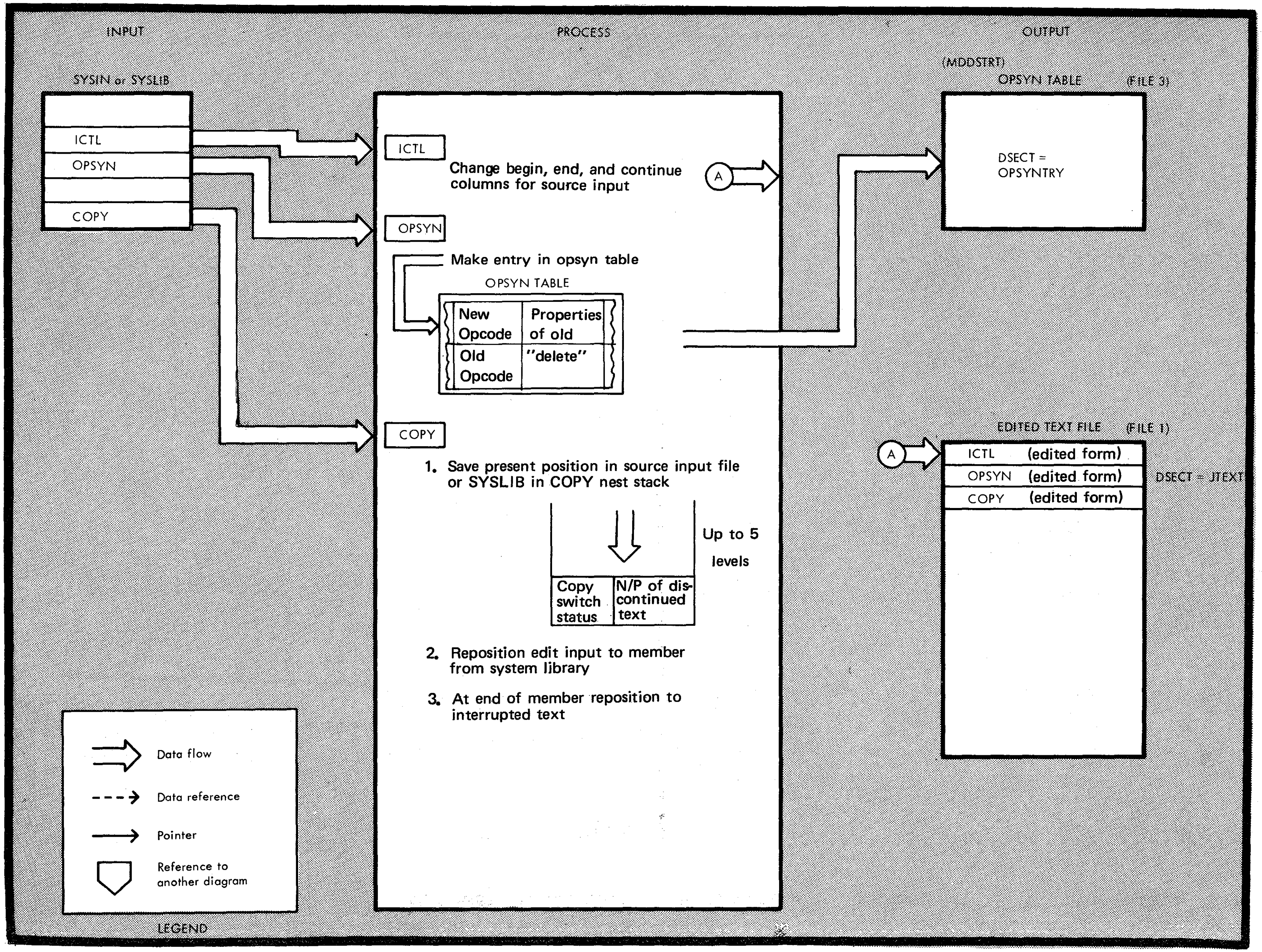
	MODULE	ROUTINE (LABEL)
Macro Instructions		
• Process according to Diagram 6.	IFNX1A	MCALLIN
• The record is put to the edited text file; also one parameter record for each parameter specified is written on the edited text file. Each ordinary symbol used as a parameter causes an ordinary symbol attribute reference to be logged (see Diagram 5). If a positional parameter is omitted, an "omitted parameter" record is written on the edited text file. An "end of all parameters" record follows the parameter records.	IFNX1A IFNX1J	NEXTPARM (ORDSYMBR)

	MODULE	ROUTINE (LABEL)
MEND Statement		
• The text segment dictionary file is closed.	IFNX1A	MEND
• The MDD entry for the text segment is closed (see Diagram 6).	IFNX1J	(MACREND)
5 The rest of the record (remarks and strings) is edited.	IFNX1A	WRAPFLD

This page intentionally left blank

Process ICTL, COPY, and OPSYN

4



ICTL: Change begin, end, and continue columns for source input

OPSYN: Make entry in opsyn table

COPY:

1. Save present position in source input file or SYSLIB in COPY nest stack
2. Reposition edit input to member from system library
3. At end of member reposition to interrupted text

Process ICTL, COPY, and OPSYN (cont.)

ICTL

An ICTL statement changes the beginning, end, and continue columns for source input. The (edited) ICTL record is put on the edited text file.

MODULE	ROUTINE (LABEL)
IFNX1A	ICTL

OPSYN

For every valid OPSYN an entry is made in the OPSYN table. Entries may be two forms: either the user wants to give a standard opcode a duplicate name and keep both opcode names as valid; or he wants to replace a standard opcode name with one of his own and wants the standard name to be invalid. The two types of entries are shown in the table. The OPSYN table is complete when a statement other than ICTL, OPSYN, print control, or comments is read. The (edited) OPSYN statements are written on the edited text file. The OPSYN table is built and kept in core during editing; it is then written onto file 3.

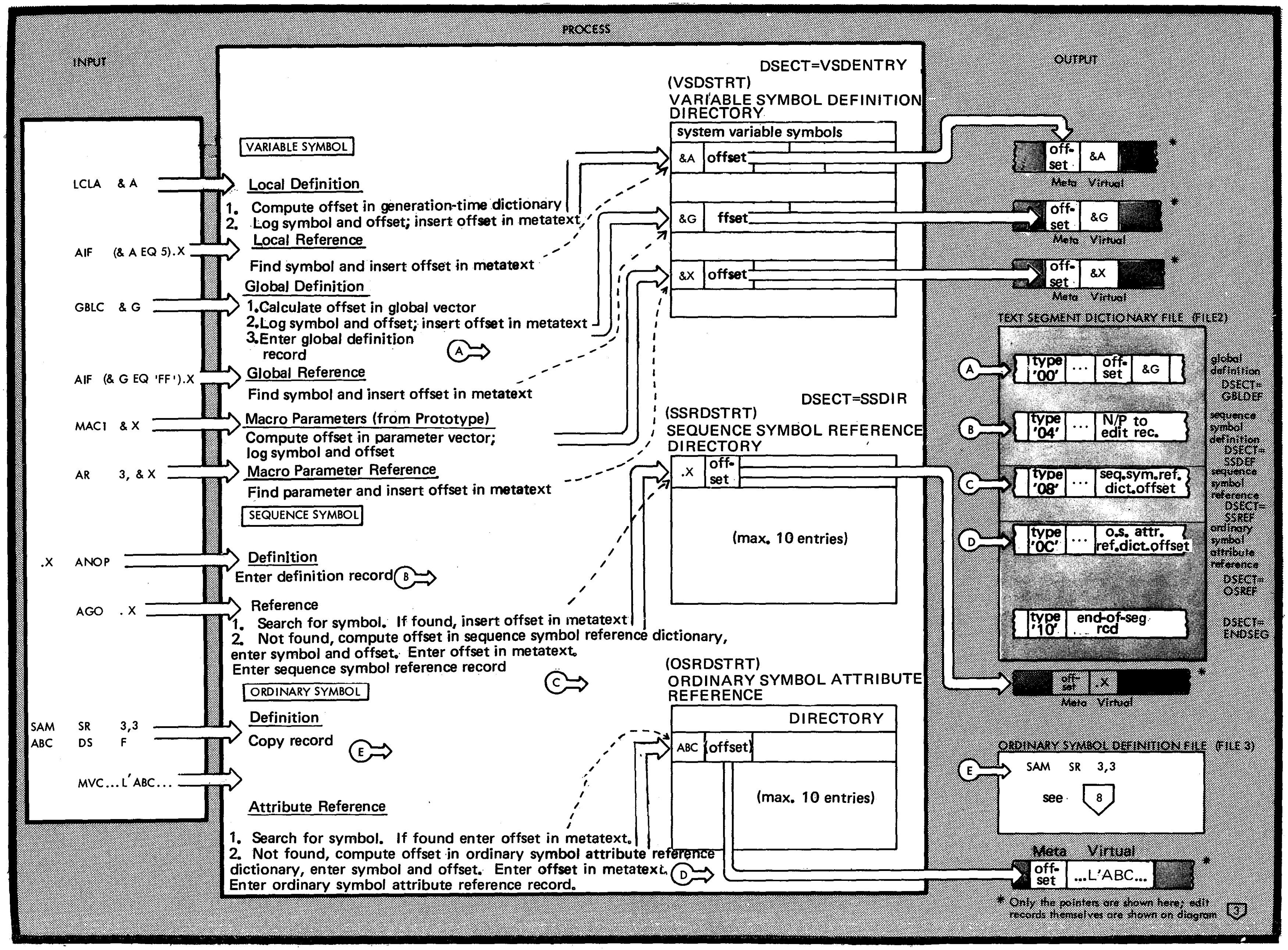
MODULE	ROUTINE (LABEL)
IFNX1A	OPSYN
IFNX1J	OPSYNBLD

COPY

Up to five levels of COPY nesting are allowed. When a COPY statement is encountered, the present position in the source input file (or in the system library) is saved in the COPY nest stack. Each entry in the stack consists of the copy level status switch and the N/P address of the discontinued text. The input is repositioned to edit the copy member from SYSLIB. At the end of the member, the input is repositioned to the interrupted location. The (edited) COPY statement itself is written on the edited text file.

MODULE	ROUTINE (LABEL)
IFNX1A	COPY
IFNX1A	CSTKENT
IFNX1A	CSTKEXT

Process Symbols



VARIABLE SYMBOL

Local Definition

1. Compute offset in generation-time dictionary
2. Log symbol and offset; insert offset in metatext

Local Reference

Find symbol and insert offset in metatext

Global Definition

1. Calculate offset in global vector
2. Log symbol and offset; insert offset in metatext
3. Enter global definition record

Global Reference

Find symbol and insert offset in metatext

Macro Parameters (from Prototype)

Compute offset in parameter vector; log symbol and offset

Macro Parameter Reference

Find parameter and insert offset in metatext

SEQUENCE SYMBOL

Definition

Enter definition record

Reference

1. Search for symbol. If found, insert offset in metatext
2. Not found, compute offset in sequence symbol reference dictionary, enter symbol and offset. Enter offset in metatext. Enter sequence symbol reference record

ORDINARY SYMBOL

Definition

Copy record

Attribute Reference

1. Search for symbol. If found enter offset in metatext.
2. Not found, compute offset in ordinary symbol attribute reference dictionary, enter symbol and offset. Enter offset in metatext. Enter ordinary symbol attribute reference record.

Process Symbols (cont.)

To edit statements containing symbols, it is necessary to:

- Compute the positions in generation-time dictionaries of variable symbols (including macro parameters), sequence symbols, and ordinary symbols with attribute references.

- Insert the offset of the symbol value (in the dictionary) in the record's metatext.

- Construct the text segment dictionary file, from which the generation-time dictionaries and vectors are built later.

  Note: Only symbols needed for macro expansion and conditional assembly (variable symbols, sequence symbols, and ordinary symbols with attribute references) are processed at this stage. See Diagram 15 for processing of ordinary symbols for assembly.

Three internal work areas are used: The variable symbol definition directory serves to keep track of which variable symbols have been defined. It contains system variable symbols, local and global variable symbols (and their offsets in generation-time dictionaries), and macro parameters (with their offsets in the macro parameter vector). The sequence symbol reference directory keeps track of references to sequence symbols and where their definition positions will be in the generation-time dictionary. The ordinary symbol attribute directory serves an exactly analogous role for ordinary symbols whose attributes have been referenced.

> Note: Both the sequence symbol reference dictionary and the ordinary attribute reference dictionary contain only 10 entries at a time; thus the value of a given symbol may appear more than once in the generation-time dictionary.

Symbols are processed according to type:	MODULE	ROUTINE (LABEL)
VARIABLE SYMBOL		
Local Definition		
1. The offset of the symbol's value in the generation-time local dictionary is computed from the symbol type and specified dimension.	IFNX1J	VARBSYMD (VSLOOKUP)
2. The symbol and its offset are then entered in the variable symbol definition directory; the offset is entered in the metatext of the edit record.	IFNX1J	VARDSYMD
Local Reference		
The symbol is found in the variable symbol definition directory and its offset inserted in the edit record's metatext.	IFNX1J	VARBSYMR (VSLOOKUP)
Global Definition		
1. The offset of the symbol's pointer in the global vector is calculated (each entry in the global vector is three bytes long).	IFNX1J	VARBSYMD (VSLOOKUP)
2. The global vector offset is then entered in the record's metatext.	IFNX1J	VARBSYMD
3. A global definition record (consisting of the symbol, its dimension, type, and offset in the global vector) is written on the text segment dictionary file.		
Global Reference		
The symbol is found in the variable symbol definition directory and its offset in the global vector inserted in the metatext.	IFNX1J	VARBSYMR VSLOOKUP

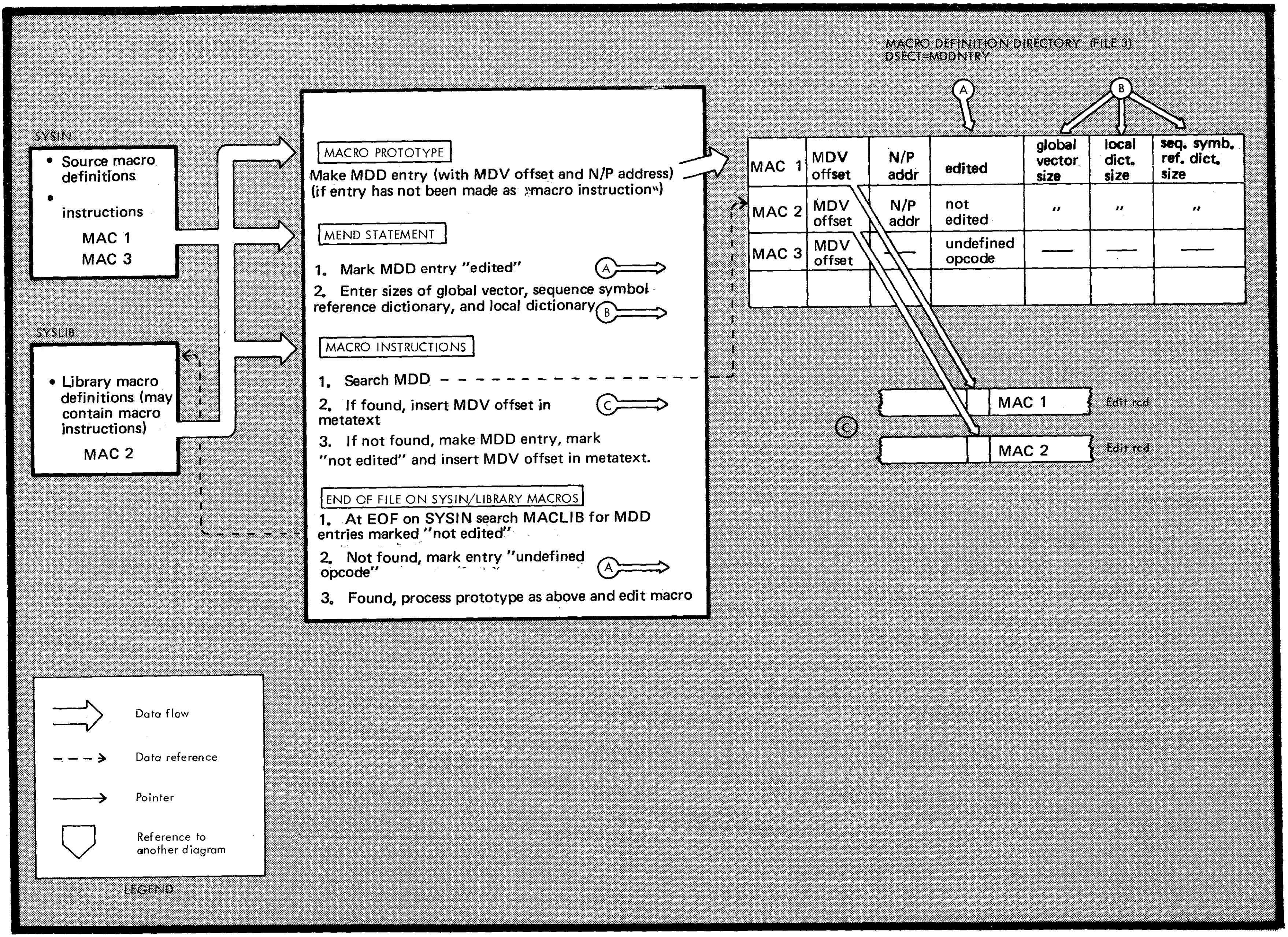
PROCESS SYMBOLS (cont.)

	MODULE	ROUTINE (LABEL)
Macro Parameters (from Prototype)		
The symbol´s offset in the generation-time parameter vector (see Diagram 12) is computed. The symbol and its vector offset are logged in the variable symbol definition directory and the offset inserted in the parameter record metatext.	IFNX1J	VARBSYMD VSLOOKUP
Macro Parameter Reference		
The symbol is found in the variable symbol definition directory and its parameter vector offset placed in the edit record´s metatext.	IFNX1J	VARBSYMR VSLOOKUP
SEQUENCE SYMBOL		
Definition		
A sequence symbol definition record (with N/P value of edit record) is written on the text segment dictionary file.	IFNX1J	SEQSYMBD
Reference		
1. The sequence symbol reference directory (first 10 entries) is searched. If the definition is found, its offset (in the sequence symbol reference dictionary) is inserted in the record´s metatext.	IFNX1J	SEQSYMBR COMNREF
2. If not found, (a) the symbol´s offset in the sequence symbol reference dictionary is computed and the symbol and its offset entered in the sequence symbol reference directory; (b) the offset is entered in the record metatext; (c) a sequence symbol reference record is written on the text segment dictionary file.	IFNX1J	COMNREF
ORDINARY SYMBOL		
Definition		
The record is copied onto the ordinary symbol definition file.	IFNX1J	ORDSYMBD
Attribute Reference		
1. The ordinary symbol attribute reference directory is searched. If the symbol is found, its offset in ordinary symbol attribute reference dictionary is inserted in the record´s metatext.	IFNX1J	ORDSYMBR COMNREF
2. If the symbol is not found, its offset in the ordinary symbol attribute reference dictionary is computed and the symbol and offset entered in the ordinary symbol attribute reference directory. The offset is entered into the record´s metatext. An "ordinary symbol attribute reference record" is written on the text segment dictionary file.	IFNX1J	COMNREF

This page intentionally left blank

Process Macros and Build Macro Definition Directory

6



SYSIN

- Source macro definitions
-
  instructions

MAC 1
MAC 3

SYSLIB

- Library macro definitions (may contain macro instructions)

MAC 2

MACRO PROTOTYPE

Make MDD entry (with MDV offset and N/P address) (if entry has not been made as "macro instruction")

MEND STATEMENT

1. Mark MDD entry "edited"
2. Enter sizes of global vector, sequence symbol reference dictionary, and local dictionary

MACRO INSTRUCTIONS

1. Search MDD
2. If found, insert MDV offset in metatext
3. If not found, make MDD entry, mark "not edited" and insert MDV offset in metatext.

END OF FILE ON SYSIN/LIBRARY MACROS

1. At EOF on SYSIN search MACLIB for MDD entries marked "not edited"
2. Not found, mark entry "undefined opcode"
3. Found, process prototype as above and edit macro

MACRO DEFINITION DIRECTORY (FILE 3)
DSECT=MDDNTRY

MAC 1	MDV offset	N/P addr	edited	global vector size	local dict. size	seq. symb. ref. dict. size
MAC 2	MDV offset	N/P addr	not edited	"	"	"
MAC 3	MDV offset	—	undefined opcode	—	—	—

Data flow

Data reference

Pointer

Reference to another diagram

LEGEND

Process Macros and Build Macro Definition Directory (cont.)

The macro definition directory (MDD) serves roughly the same function for macros as the variable symbol definition directory does for variable symbols: it keeps track of which macros have been defined and helps in assigning pointers to their generation-time definitions. Information from the MDD is later used to build the macro definition vector (MDV) (see Diagram 10).

Macro prototypes, macro instructions, end-of-file on SYSIN, and MEND all cause entries to be made in the MDD.

MACRO PROTOTYPE

	MODULE	ROUTINE (LABEL)
• The MDD is searched for a corresponding entry.		
• If found, the N/P address on the edit text file is added to the MDD entry. If not found, the macro name, its calculated offset in the MDV, and its N/P address on the edited text file are entered in the MDD.	IFNX1J	MACRENT

MEND Statement

	MODULE	ROUTINE (LABEL)
1. At the end of a macro definition (either source or library) the MDD entry is marked "edited".	IFNX1J	MACREND
2. The sizes of the global vector, the sequence symbol reference dictionary, and the local dictionary for the text segment are placed in the entry.		COMNEND

MACRO Instruction

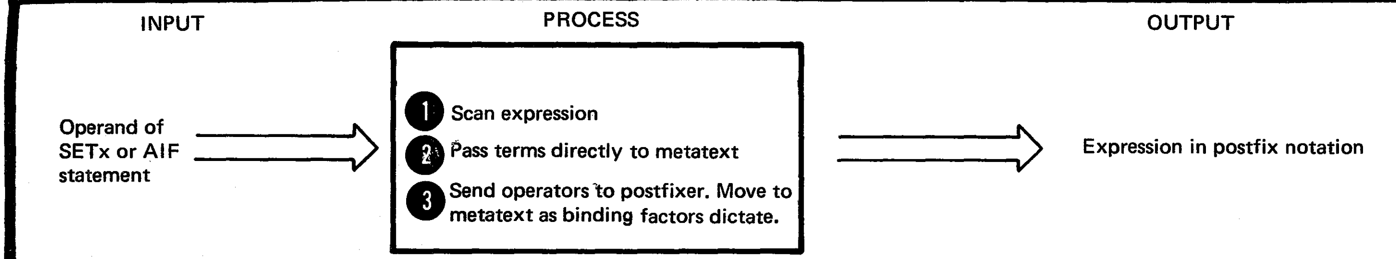
	MODULE	ROUTINE (LABEL)
1. When a macro instruction is encountered (either within a macro definition or open code), the MDD is searched for a corresponding entry.	IFNX1J	MACRNAME
2. If found, the MDV offset is inserted in the metatext of the macro instruction's edit record.	IFNX1J	MSCANA
3. If it is not found, the macro name and the next calculated MDV offset are entered in a MDD entry for the macro. Its MDV offset is inserted in the metatext. The MDD entry is marked "not edited".	IFNX1J	MACENTRY

END OF FILE on SYSIN (Library Macros)

	MODULE	ROUTINE (LABEL)
1. At EOF on SYSIN each entry in the MDD marked "not edited" is found. These entries are either library macros or undefined opcodes. SYSLIB is searched for corresponding entries.	IFNX1A IFNX1J IFNX1J IFNX1A IFNX1J	NEOFRTN OPENEND COMNEND ESYSMAC EDITSYSM
2. If not found, the MDD entry is marked "undefined opcode".	IFNX1J	MACREND
3. If the macro is found on SYSLIB, the prototype is edited as above and its N/P address on the edited text file placed in the MDD entry (the entry will be eventually marked "edited" by the MEND statement processing).	IFNX1J	MACREND

Convert Expressions to Postfix Notation

7



INPUT

Operand of SETx or AIF statement

PROCESS

1. Scan expression
2. Pass terms directly to metatext
3. Send operators to postfixer. Move to metatext as binding factors dictate.

OUTPUT

Expression in postfix notation

```
         MACRO
         MAC1     &PARM1,&PARM2,&PARM3
         LCLA     &X,&Y,&Z,&A
         LCLB     &M,&N,&B
         LCLC     &Q,&C(50)
&A       SETA     5
&B       SETB     1
:
         AIF      ('ABC' EQ ' &C(&A)' AND &A*&B+1 LT &PARM2) .LABEL
.LABEL   ANOP
         MEND
```

THIS EXPRESSION

('ABC' EQ ' &C(&A)' AND &A*&B+1 LT &PARM2) .LABEL

GIVES

THIS METATEXT

00	start character mode
2E	character string
03	length 3
0A	A
0B	B
0C	C
01	end character mode
00	start character mode
26	set C
000029	offset in local dictionary for &C
0032	dimension of &C
22	set A
000019	offset in local dictionary for &A
0000	0 (&A undimensioned)
17	dimension operator ( )
01	end character mode
0E	EQ operator
22	setA
000019	offset in local dictionary for &A
0000	0 (&A undimensioned)
24	setB
00001F	offset in local dictionary for &B
0000	0 (&B undimensioned)
08	*operator
2C	self-defining term
00000001	1 (value of self-defining term)
00	padding
0A	+ operator
34	positional parameter
000006	offset in positional parameter vector for &PARM2
0000	padding
01	LT operator
13	AND operator
1A	statement terminator
2A	sequence symbol (.LABEL)
000000	.LABEL's offset in sequence symbol ref. dict.
0000	padding
003A	meta text length

Convert Expressions to Postfix Notation (cont.)

Expressions are translated into postfix notation (also called reverse Polish notation). This is a form easier for the assembler to interpret during generation.

		MODULE	ROUTINE (LABEL)
1	Expressions are scanned	IFNX1A	METASCAN
2	Elements (that is, non-operators) are assigned identifiers are are inserted immediately into the metatext. Variable symbols and attribute references are processed as described in Diagram 5 and dictionary pointers entered.	IFNX1J	VARBSYMR ORDSYMBR
3	Operators are sent to the postfix routine, where they are put into a stack according to their "binding factor". This is a value assigned to each operator: the lower the binding factor, the earlier the operator is inserted into the metatext. Operators are assigned the following binding factors:	IFNX1S	POSTER

0	DIMENSION OPERATOR
1	STRING OPERATOR
2	DUPLICATION OPERATOR
3	PERIOD (CONCATENATION)
4	UNARY PLUS AND MINUS, TYPE, LENGTH, SCALE, INTEGER, COUNT AND NUMBER ATTRIBUTES
5	MULTIPLY, DIVIDE
6	ADD, SUBTRACT
7	GT, GE, LT, LE, EQ, NE
8	NOT
9	AND
10	OR
11	), COMMA
12	(
13	STATEMENT TERMINATOR

The first operator encountered is always entered in the stack. For all other operators, the operator's binding factor value is compared with that of the last operator entered into the stack. If the value being compared is lower than that of the last operator in the stack, the operator is placed in the stack. If the value being compared is higher than or equal to that of the last operator in the stack, the operator in the stack is removed and placed in the metatext. The value is then compared with the next element in the stack, and so forth.

"Start character mode" and "end character mode" operators are placed immediately into the metatext, bypassing the stack. When the end of the expression is reached, the edit phase passes an "expression end" operator to the stack. This operator has the highest binding factor and forces the remaining operators in the stack into the metatext. The "expression end" operator is placed last in the metatext to serve as a terminator.

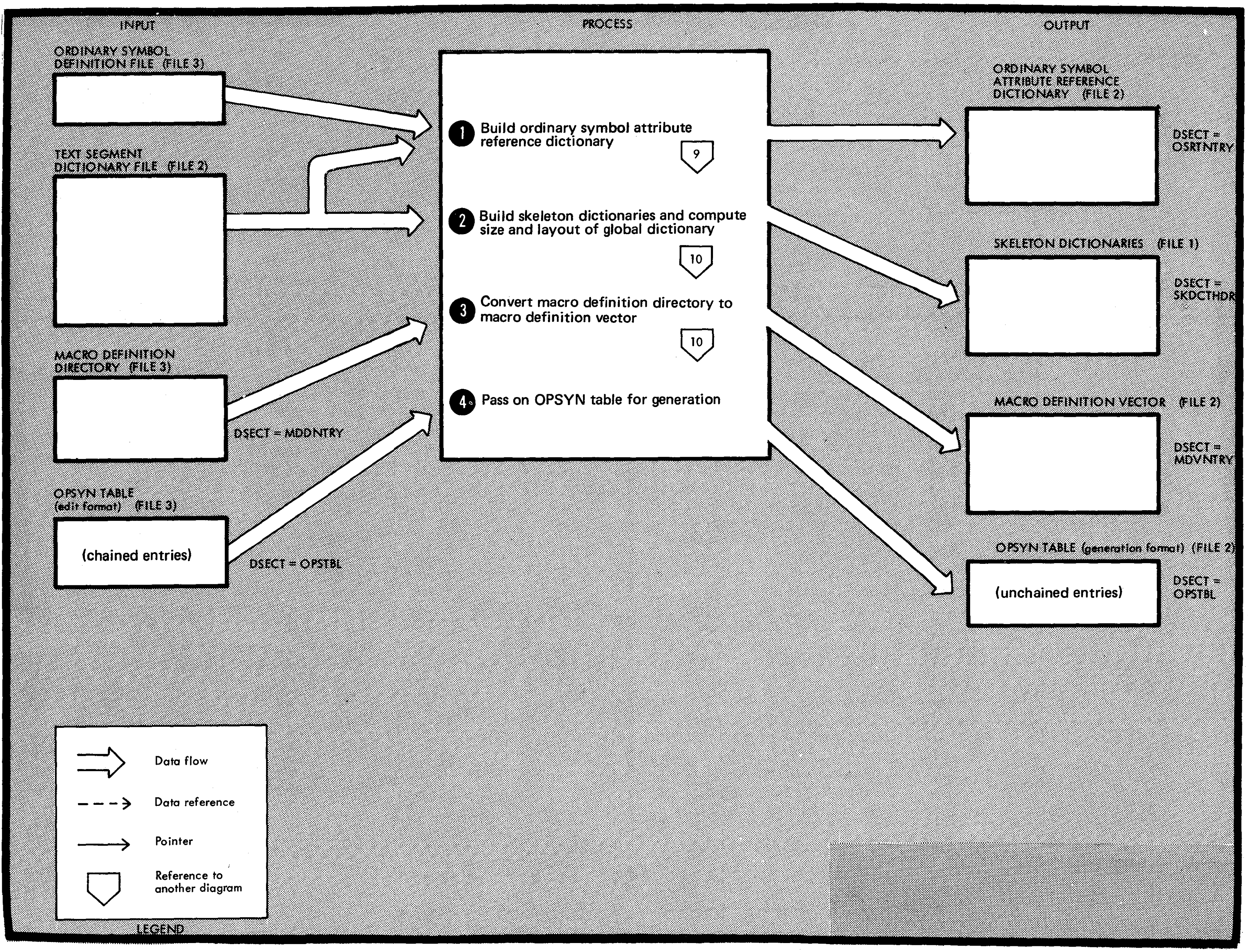
Processing of the expression shown on Diagram 7 proceeds as follows:

```
('ABC'  EQ  '&C(&A)'  AND  &A*&B + 1  LT  &PARM2) . LABEL
   1     2    3  4      5    6 7 8 9 10 11     12   13
```

		METATEXT	OPERATOR STACK
1	ABC is placed immediately in metatext	ABC	
2	EQ is entered in the operator stack	ABC	EQ
3	&C&A placed in metatext	ABC&C&A	EQ
4	()'s binding factor compared to EQ's binding factor; () is put in the stack	ABC&C&A	() EQ
5	AND's binding factor > ()'s binding factor. AND replaces () in stack; () goes to metatext. AND also replaces EQ in the stack; EQ goes to metatext.	ABC&C&A () EQ	AND
6	&A goes to metatext	ABC&C&A () EQ&A	AND
7	*'s binding factor < AND's binding factor. Enter * in stack	ABC&C&A () EQ&A	* AND
8	&B goes to metatext	ABC&C&A () EQ&A&B	* AND
9	+'s binding factor > *'s binding factor. * goes to metatext, + into stack. +'s binding factor < AND's binding factor; no change	ABC&C&A () EQ&A&B*	+ AND
10	1 goes into metatext		
11	LT's binding factor > +'s binding factor. + out, LT in. LT's binding factor < AND's; no change	ABC&C&A () EQ&A&B*1+	LT AND
12	&PARM2 into metatext	ABC&C&A () EQ&A&B*1+&PARM2	LT AND
13	END empties stack	ABC&C&A () EQ&A&B*1+&PARM2LTANDEND	

Build Generation-Time Dictionaries

8


INPUT
PROCESS
OUTPUT
ORDINARY SYMBOL DEFINITION FILE (FILE 3)
TEXT SEGMENT DICTIONARY FILE (FILE 2)
MACRO DEFINITION DIRECTORY (FILE 3)
DSECT = MDDNTRY
OPSYN TABLE (edit format) (FILE 3)
(chained entries)
DSECT = OPSTBL
9
10
10
ORDINARY SYMBOL ATTRIBUTE REFERENCE DICTIONARY (FILE 2)
DSECT = OSRTNTRY
SKELETON DICTIONARIES (FILE 1)
DSECT = SKDCTHDR
MACRO DEFINITION VECTOR (FILE 2)
DSECT = MDVNTRY
OPSYN TABLE (generation format) (FILE 2)
(unchained entries)
DSECT = OPSTBL
Data flow
Data reference
Pointer
Reference to another diagram
LEGEND

1. Build ordinary symbol attribute reference dictionary
2. Build skeleton dictionaries and compute size and layout of global dictionary
3. Convert macro definition directory to macro definition vector
4. Pass on OPSYN table for generation

Build Generation-Time Dictionaries (cont.)

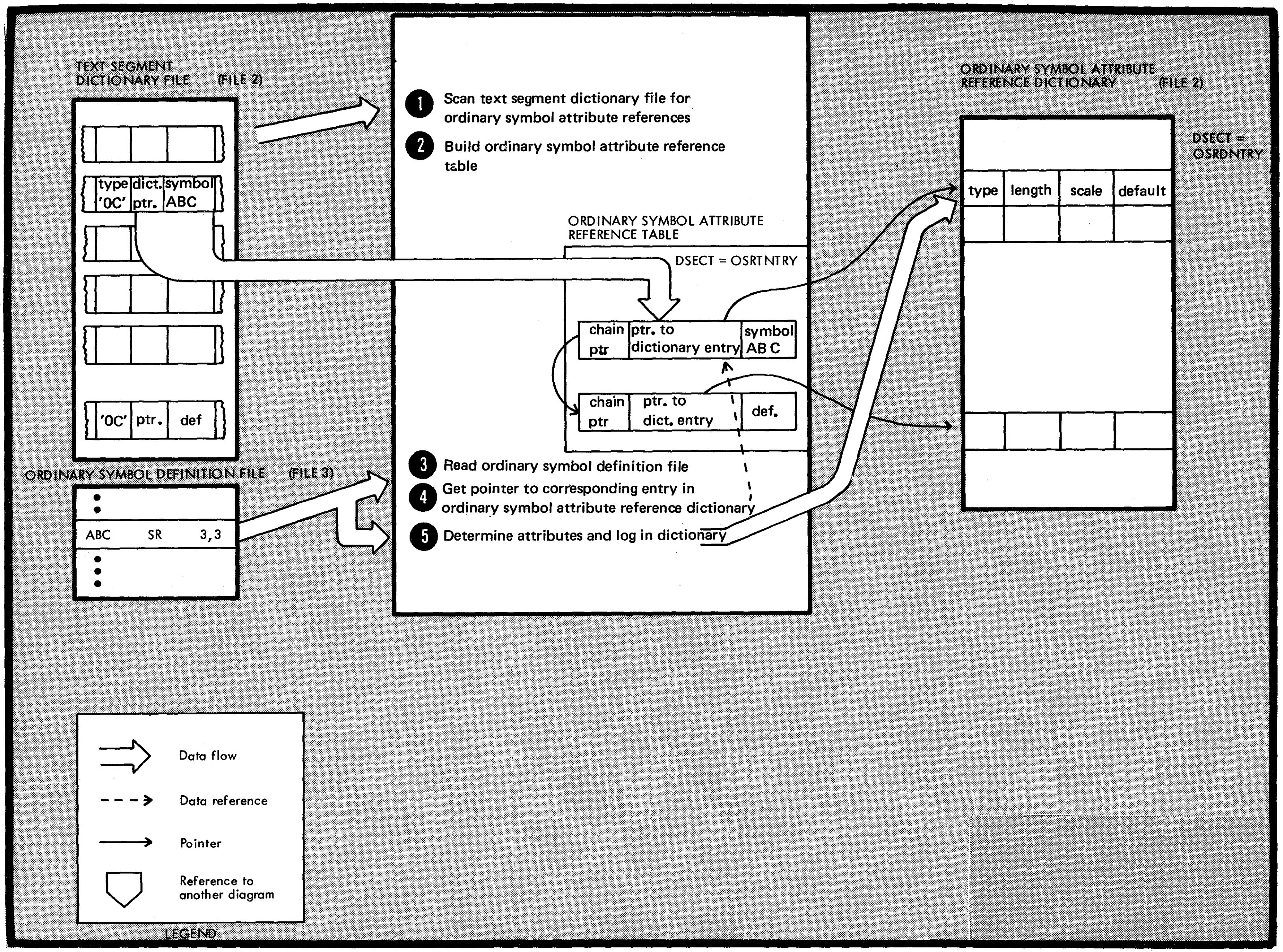
Selected information collected during editing is used to set up the dictionaries for use during generation.

		MODULE	ROUTINE (LABEL)
1	The ordinary symbol attribute reference dictionary is built by matching entries in the ordinary symbol definition file with corresponding entr ies in the text segment dictionary file (see Diagram 9).	IFNX2A	ORDREF ORDSYMBR
2	Skeleton dictionaries are set up for each text segment. Because each skeleton dictionary contains a global vector pointing to entries in a common (for all text segments) global dictionary, it is necessary to set up the global dictionary at the same time. See Diagram 10.	IFNX2A	SEQREF SEQDEF GBLDEF

		MODULE	ROUTINE (LABEL)
3	Information in the macro definition directory is split at this point. Part goes to the skeleton dictionary headers, and part goes to make up the macro definition vector. See Diagram 10.	IFNX2A	ENDSEGB
4	The OPSYN table is passed on for generation. Entries are unchained and the size and location of the table saved in COMMON.	IFNX2A	OPSYNBLD PUTOPSYN

Build Ordinary Symbol Attribute Reference Dictionary

9



1 Scan text segment dictionary file for ordinary symbol attribute references

2 Build ordinary symbol attribute reference table

3 Read ordinary symbol definition file

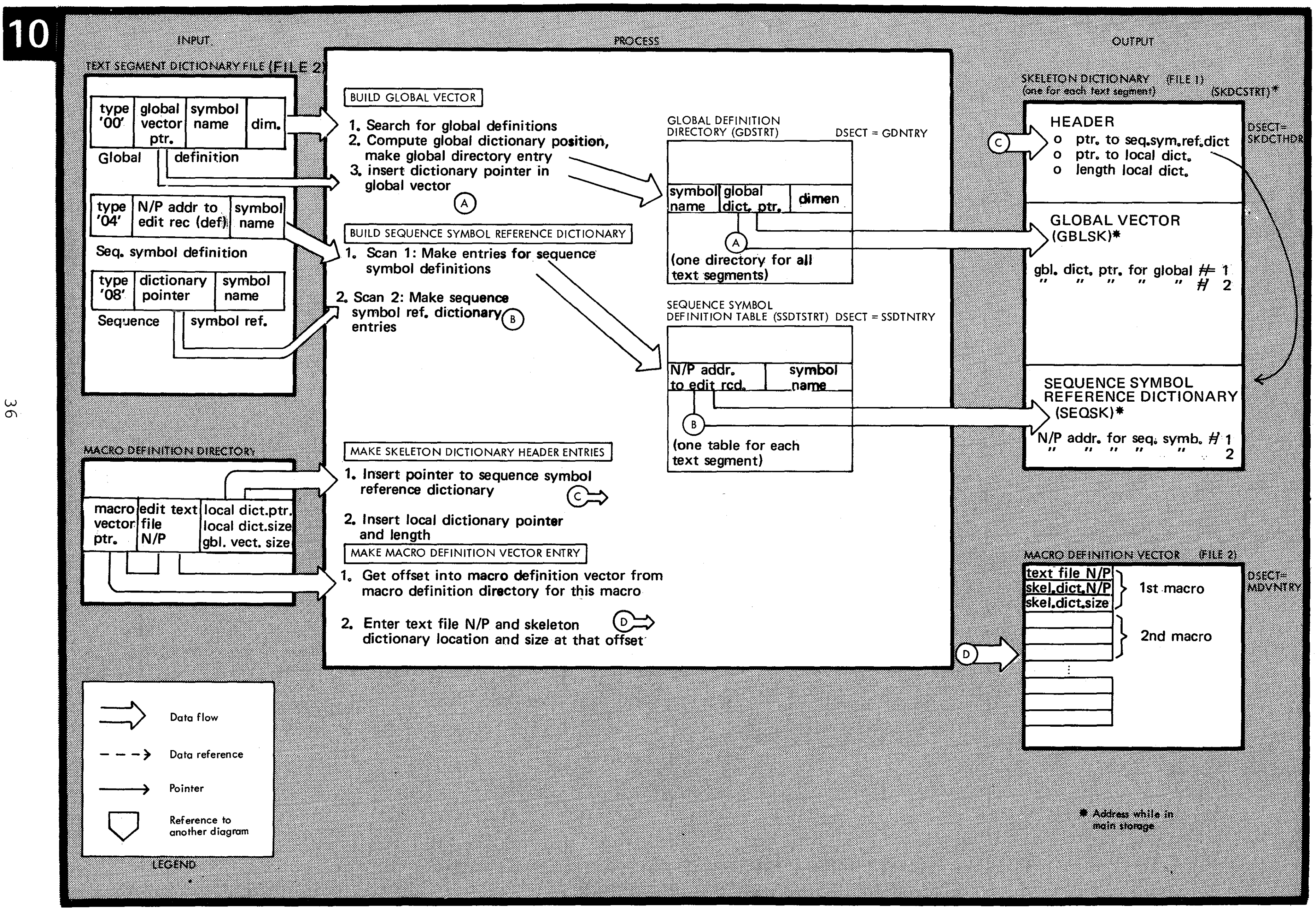
4 Get pointer to corresponding entry in ordinary symbol attribute reference dictionary

5 Determine attributes and log in dictionary

Build Ordinary Symbol Attribute Reference Dictionary (cont.)

		MODULE	ROUTINE (LABEL)
	Attributes of ordinary symbols are collected and placed in a dictionary to be used at generation.		
1	The text segment dictionary file is scanned and type 'OC' (ordinary symbol attribute required) records read.	IFNX2A	ORDREF
2	The symbol is hashed and inserted in the ordinary symbol reference table, along with a pointer to its eventual position in the ordinary symbol attribute reference dictionary. Entries in the table are chained.	IFNX2A	HASH
3	When all the 'OC' records for a given text segment are read, the ordinary symbol definition file containing the definition records for all ordinary symbols, is read.	IFNX2A	ORDSYMBR
4	The symbol from the ordinary symbol definition file is hashed and the ordinary symbol attribute reference table searched for a corresponding entry. If found (that is, if attributes are required), the symbol's position in the ordinary symbol attribute reference dictionary is obtained.	IFNX2A	HASH OSLUKUP
5	The opcode and operand are then scanned to determine the attributes, which are inserted in the dictionary at the positions given by the symbol's hash-table chain. Pointers to dictionary locations have already been placed in the metatext of edit records that require them (see Diagram 5). Note that one symbol may have several entries in the ordinary symbol attribute reference dictionary because of the 10-entry limitation of the edit-time ordinary symbol attribute reference directory.	IFNX2A	BRONTYP

Build Skeleton Dictionary and Macro Definition Vector



10

INPUT

TEXT SEGMENT DICTIONARY FILE (FILE 2)

type '00'	global vector ptr.	symbol name	dim.

Global definition

type '04'	N/P addr to edit rec (def)	symbol name

Seq. symbol definition

type '08'	dictionary pointer	symbol name

Sequence symbol ref.

MACRO DEFINITION DIRECTORY

macro vector ptr.	edit text file N/P	local dict.ptr. local dict.size gbl. vect. size

PROCESS

BUILD GLOBAL VECTOR

1. Search for global definitions
2. Compute global dictionary position, make global directory entry
3. insert dictionary pointer in global vector (A)

BUILD SEQUENCE SYMBOL REFERENCE DICTIONARY

1. Scan 1: Make entries for sequence symbol definitions
2. Scan 2: Make sequence symbol ref. dictionary entries (B)

MAKE SKELETON DICTIONARY HEADER ENTRIES

1. Insert pointer to sequence symbol reference dictionary (C)
2. Insert local dictionary pointer and length

MAKE MACRO DEFINITION VECTOR ENTRY

1. Get offset into macro definition vector from macro definition directory for this macro
2. Enter text file N/P and skeleton dictionary location and size at that offset (D)

GLOBAL DEFINITION DIRECTORY (GDSTRT) DSECT = GDNTRY

symbol name	global dict. ptr.	dimen

(A)

(one directory for all text segments)

SEQUENCE SYMBOL DEFINITION TABLE (SSDTSTRT) DSECT = SSDTNTRY

N/P addr. to edit rcd.	symbol name

(B)

(one table for each text segment)

OUTPUT

SKELETON DICTIONARY (FILE 1)
(one for each text segment) (SKDCSTRT)*

DSECT= SKDCTHDR

(C) HEADER
- o ptr. to seq.sym.ref.dict
- o ptr. to local dict.
- o length local dict.

GLOBAL VECTOR (GBLSK)*

gbl. dict. ptr. for global # 1
" " " " " # 2

SEQUENCE SYMBOL REFERENCE DICTIONARY (SEQSK)*

N/P addr. for seq. symb. # 1
" " " " " 2

MACRO DEFINITION VECTOR (FILE 2)

DSECT= MDVNTRY

(D)

text file N/P
skel.dict.N/P
skel.dict.size } 1st macro

} 2nd macro

⋮

* Address while in main storage

LEGEND

⇒ Data flow

- - -> Data reference

——> Pointer

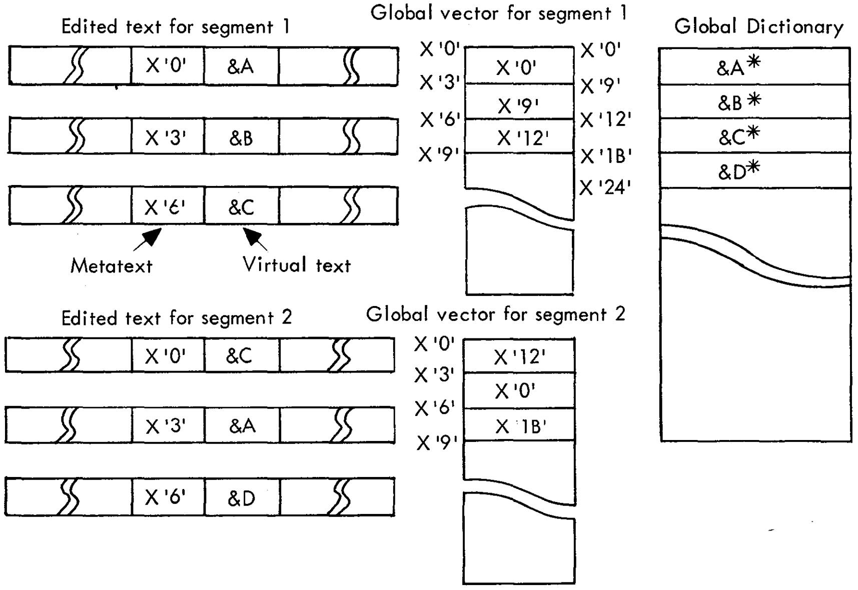
⬠ Reference to another diagram

Build Skeleton Dictionary and Macro Definition Vector (cont.)

A skeleton dictionary for a text segment consists of a header, a global vector, and a sequence symbol reference dictionary.

BUILD GLOBAL VECTOR

There is a global vector for each text segment. The relationship among global symbols, global vectors, and the global dictionary is shown below:



* The symbol itself does not appear in the dictionary. It is shown here only to indicate which locations are assigned to which symbols. &A, &B, &C and &D are assumed in this example to be GBLC variables, each of which takes up 9 bytes in the global dictionary. (If the symbols are longer than 9 bytes, a dictionary extension is used.)

	MODULE	ROUTINE (LABEL)
1. The text segment dictionary file is read for type "00" (global symbol definition) records.	IFNX2A	GBLDEF (GSHASHER)
2. For each new definition (that is, one not defined in a previous text segment) a position in the global dictionary is computed and the symbol with its position entered in the global definition directory (an in-core work area). The global definition directory is used to keep track of which global symbols have previously been defined and thus to insure that the global dictionary contains only one entry per symbol. The entries are accumulated from all text segments.		
3. The global vector offset for this symbol is obtained from the definition record. At that offset in the global vector an entry is made giving the position in the global dictionary.	IFNX2A	GBLDEF
BUILD SEQUENCE SYMBOL REFERENCE DICTIONARY		
Two passes over the text segment dictionary file are needed to build the sequence symbol reference dictionary.	IFNX2A	SEQDEF (SSHASHER)
1. On the first pass, the file is scanned for type "04" (sequence symbol definition) reccrds. An entry for each such record is made in the sequence symbol definition table (an in-core work area).		
2. On the second pass, type "08" records are read to obtain the offset in the sequence symbol reference dictionary for each sequence symbol that has been referenced. The N/P address of the symbol definition is then inserted, at the offset given in the type "08" record, in the sequence symbol reference dictionary.		
Note that only sequence symbols which are referenced are entered in the dictionary.		

BUILD SKELETON DICTIONARY AND MACRO DEFINITION VECTOR (cont.)

MAKE SKELETON DICTIONARY HEADER ENTRIES

MODULE	ROUTINE (LABEL)
IFNX2A	ENDSEGB

The skeleton dictionary header contains a pointer to the sequence symbol reference dictionary and a pointer to, and the size of, the local dictionary.

1. The pointer to the sequence symbol reference dictionary for the text segment is taken from the MDD and inserted in the skeleton dictionary header.

2. The size and location of the local dictionary for the text segment are also placed in the header.

BUILD MACRO DEFINITION VECTOR

MODULE	ROUTINE (LABEL)
IFNX2A	ENDSEGB

The macro definition vector (MDV) contains entries (one for each macro) consisting of the text file N/P address of the macro definition, the text file N/P address of the skeleton dictionary for that segment, and the size of the skeleton dictionary.

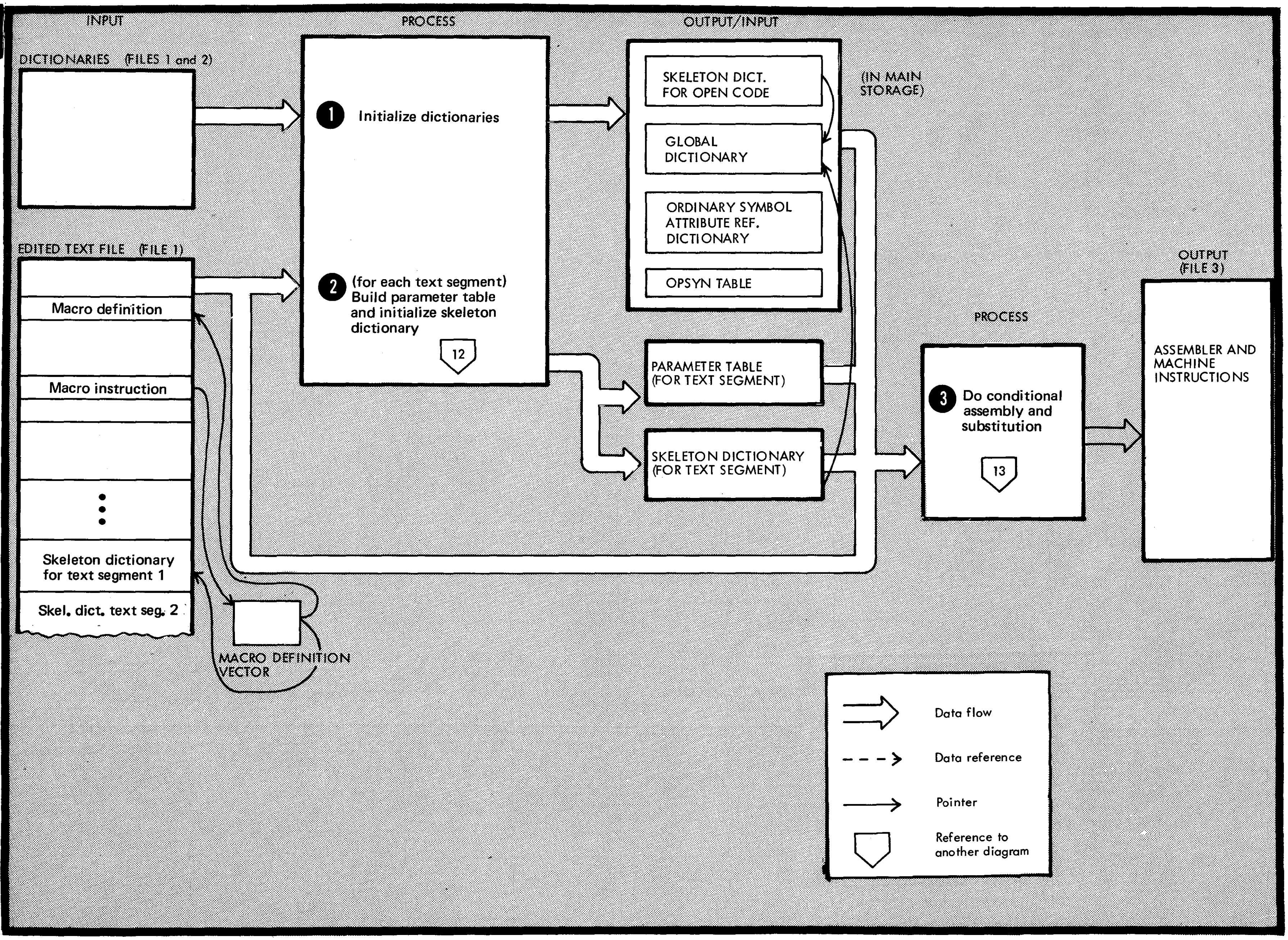
1. The offset in the MDV for the macro is obtained from the MDD.

2. The N/P address of the macro definition, the N/P address of the local dictionary, and the size of the local dictionary are entered in the MDV.

This page intentionally left blank

Generate Assembler and Machine Instructions

11


INPUT
PROCESS
OUTPUT/INPUT
DICTIONARIES (FILES 1 and 2)
1 Initialize dictionaries
SKELETON DICT. FOR OPEN CODE
GLOBAL DICTIONARY
ORDINARY SYMBOL ATTRIBUTE REF. DICTIONARY
OPSYN TABLE
(IN MAIN STORAGE)
EDITED TEXT FILE (FILE 1)
Macro definition
Macro instruction
Skeleton dictionary for text segment 1
Skel. dict. text seg. 2
2 (for each text segment) Build parameter table and initialize skeleton dictionary
12
PARAMETER TABLE (FOR TEXT SEGMENT)
SKELETON DICTIONARY (FOR TEXT SEGMENT)
PROCESS
3 Do conditional assembly and substitution
13
OUTPUT (FILE 3)
ASSEMBLER AND MACHINE INSTRUCTIONS
MACRO DEFINITION VECTOR
Data flow
Data reference
Pointer
Reference to another diagram

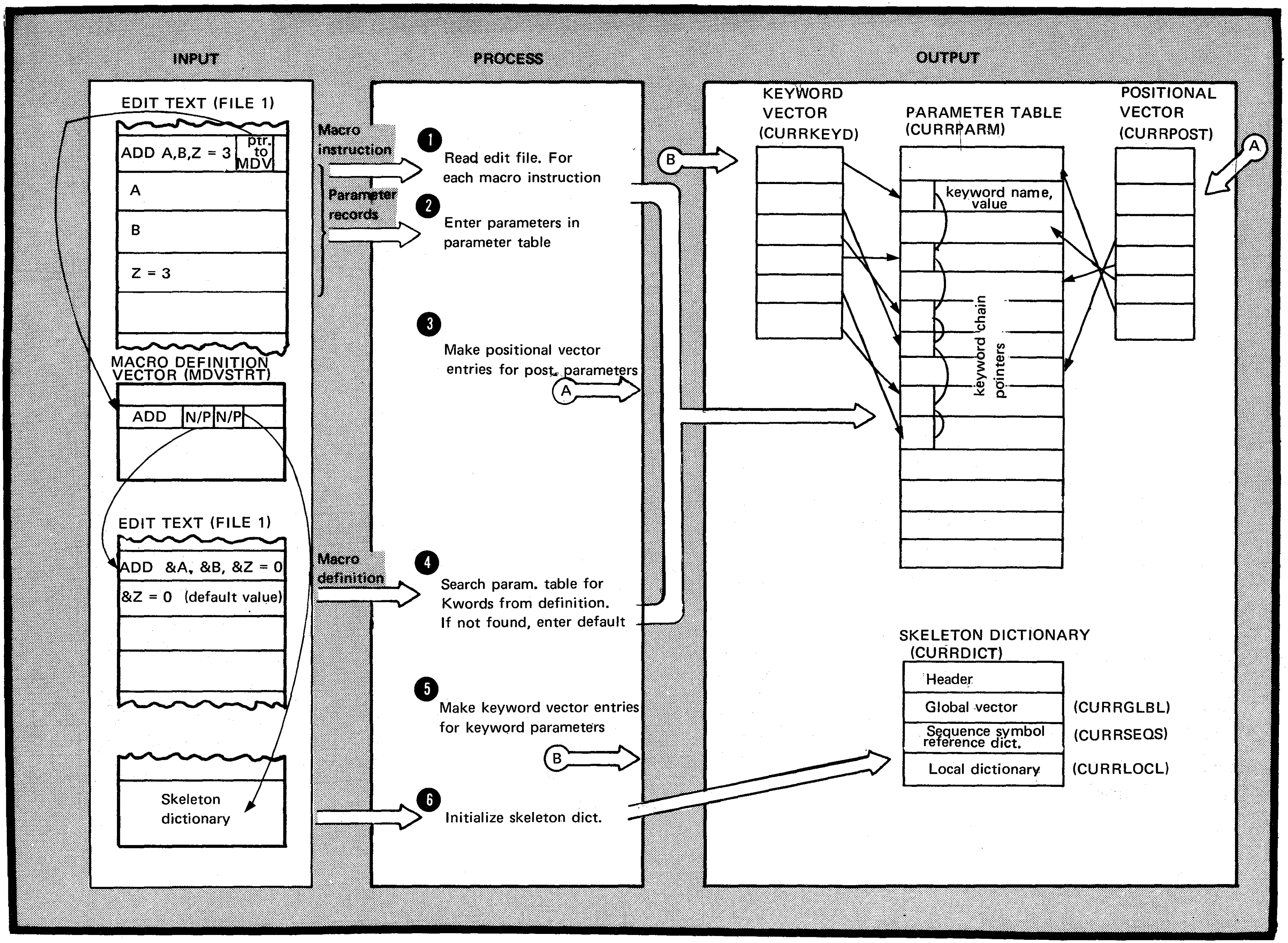
Generate Assembler and Machine Instructions (cont.)

	MODULE	ROUTINE (LABEL)
1 The macro definition vector, the ordinary symbol attribute reference dictionary, and the OPSYN table are read into main storage from file 2. The length of the global dictionary is retrieved from COMMON and it is initialized. The skeleton dictionary for open code is read from file 1. The local dictionary for open code is initialized. Edited text is read.	IFNX3N IFNX3A	PHASENTR MINPUT
2 As macro instructions are encountered, the pointer to the corresponding entry in the MDV is retrieved. The MDV entries consist of N/P addresses of the macro definition and of the skeleton dictionary for the text segment. Parts of the parameter table are built. The text file is repositioned to the macro definition.	IFNX3N	MACRCALL MACRPOST MACRKWRD CALLEND

	MODULE	ROUTINE (LABEL)
The parameter table for the text segment is then built. This table contains values for both positional and keyword parameters. (See Diagram 12.)	IFNX3N	PROTOKWD
The skeleton dictionary for the text segment is then read into main storage from file 1. Everything is now ready for expanding the macro instruction and performing the conditional assembly.	IFNX3N	PROTOEND
3 Macro definition (or conditional assembly) records from the edited text file are read and their pointers to dictionary entries used to fill and reference dictionaries. Expressions are evaluated and substitution performed. See Diagram 13. The output from this function is source statements free from macro instructions or conditional assembly statements.	IFNX3A IFNX3N	

Build Parameter Table and Initialize Skeleton Dictionary

12



1 Read edit file. For each macro instruction

2 Enter parameters in parameter table

3 Make positional vector entries for post. parameters

4 Search param. table for Kwords from definition. If not found, enter default

5 Make keyword vector entries for keyword parameters

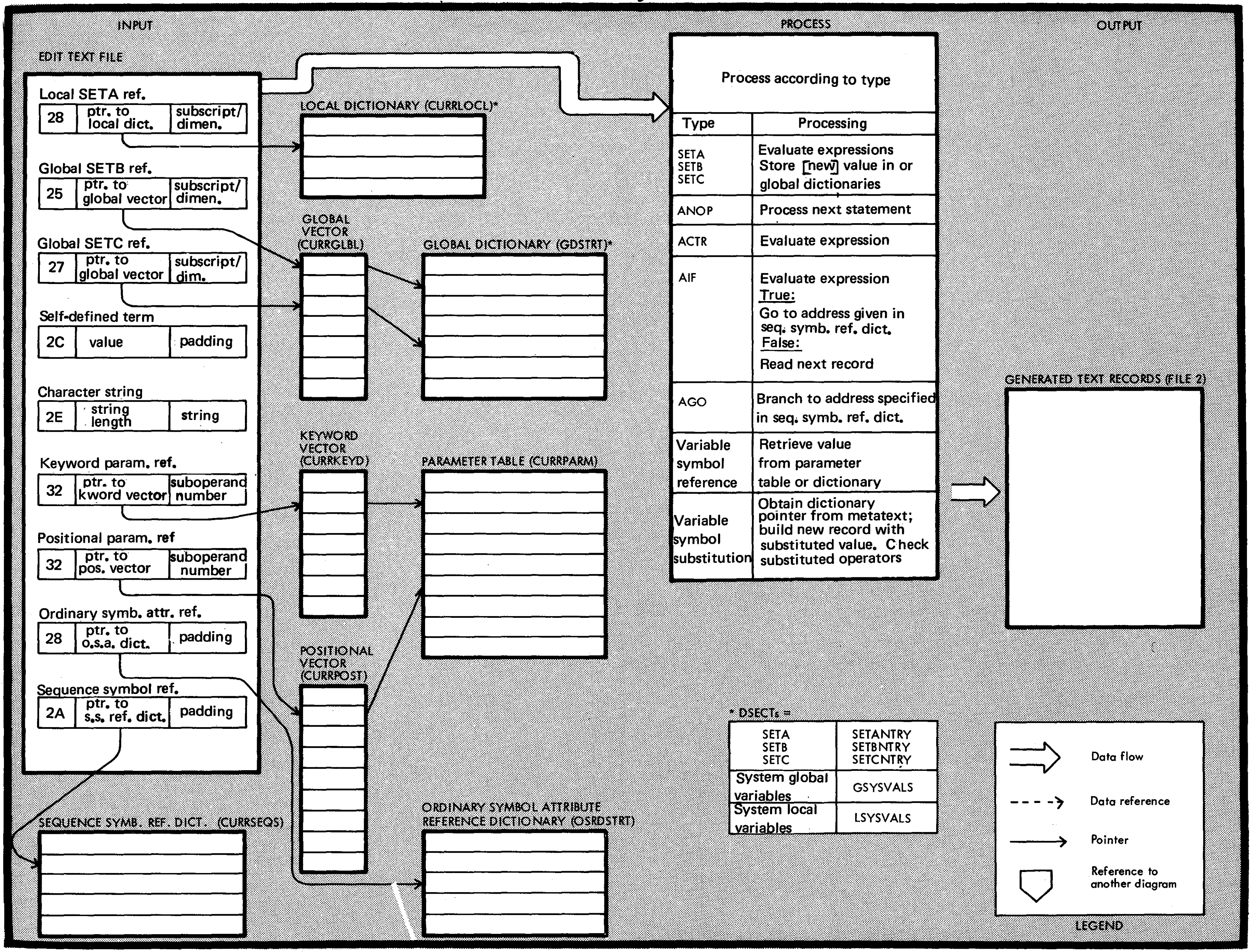
6 Initialize skeleton dict.

Build Parameter Table and Initialize Skeleton Dictionary (cont.)

		MODULE	ROUTINE (LABEL)
1	The edit file is read.		
2	For each macro instruction encountered, entries are made in the parameter table for each parameter record following the macro instruction. Keyword parameter values are chained.	IFNX3N	MACRKWRD MACRPOST
3	An entry is made in the positional vector for every positional parameter entered in the parameter table. The entries in the positional vector are addresses of the parameter value in the parameter table.	IFNX3N	MACRPOST
4	The text file is repositioned (using pointers to and from the MDV) to read the macro definition. The parameter table is searched (via the chained keyword entries) for keyword entries corresponding to the keyword parameter records in the definition. If they are not found, the default value is entered in the parameter table.	IFNX3N	CALLEND PROTOKWD
5	Entries are made in the keyword vector in the same way as for the positional vector.	IFNX3N	PROTOKWD
6	The MDV also contains the N/P address of the skeleton dictionary for the text segment. This dictionary is read into main storage and the local dictionary initialized.	IFNX3N	PROTOEND

13

Do Conditional Assembly and Substitution



Process according to type

Type	Processing
SETA SETB SETC	Evaluate expressions Store [new] value in or global dictionaries
ANOP	Process next statement
ACTR	Evaluate expression
AIF	Evaluate expression True: Go to address given in seq. symb. ref. dict. False: Read next record
AGO	Branch to address specified in seq. symb. ref. dict.
Variable symbol reference	Retrieve value from parameter table or dictionary
Variable symbol substitution	Obtain dictionary pointer from metatext; build new record with substituted value. Check substituted operators

* DSECTs =

SETA SETB SETC	SETANTRY SETBNTRY SETCNTRY
System global variables	GSYSVALS
System local variables	LSYSVALS

Do Conditional Assembly and Substitution (cont.)

	MODULE	ROUTINE (LABEL)
When a macro instruction or conditional assembly statements are encountered, it is necessary to do substitution for variable symbols and to perform the conditional assembly. In the case of a macro instruction, the input file is repositioned to the macro definition. Values of variable symbols are computed (in the case of expressions) from SETx statements and inserted in their dictionaries according to the dictionary pointers. These values can then be used either in substitution or in conditional assembly.	IFNX3A IFNX3N	

The records are processed according to type:

SETx

	MODULE	ROUTINE (LABEL)
The value of the operand is placed in the proper dictionary (local or global). If the operand is an expression, it is first evaluated.	IFNX3A	MSETA MSETB MSETC EVAL GENSTRNG RESOLVE
	IFNX3N	GBLDICTR LCLDICTR PARMTBLR ORDSYMBR GBLDICTS LCLDICTS

ANOP

No processing; the next instruction is processed.

ACTR

	MODULE	ROUTINE (LABEL)
The operand is evaluated and the value kept during processing of the current text segment.	IFNX3A	MACTR EVAL RESOLVE
	IFWX3N	GBLDICTR LCLDICTR PARMTBLR ORDSYMBR

AIF

	MODULE	ROUT!NE (LABEL)
The expression is evaluated. If true, the text file is repositioned to the N/P address given in the sequence symbol reference dictionary for the sequence symbol. If false, the next statement is processed.	IFNX3A	MAIF EVAL GENSTRNG RESOLVE
	IFNX3N	GBLDICTR LCLDICTR PARMTBLR ORDSYMBR SEQSYMBR

AGO

	MODULE	ROUT!NE (LABEL)
The text file is repositioned to the address given in the sequence symbol reference dictionary for the sequence symbol.	IFNX3A	MBRANCH1
	IFNX3N	SEQSYMBR

VARIABLE SYMBOL REFERENCE

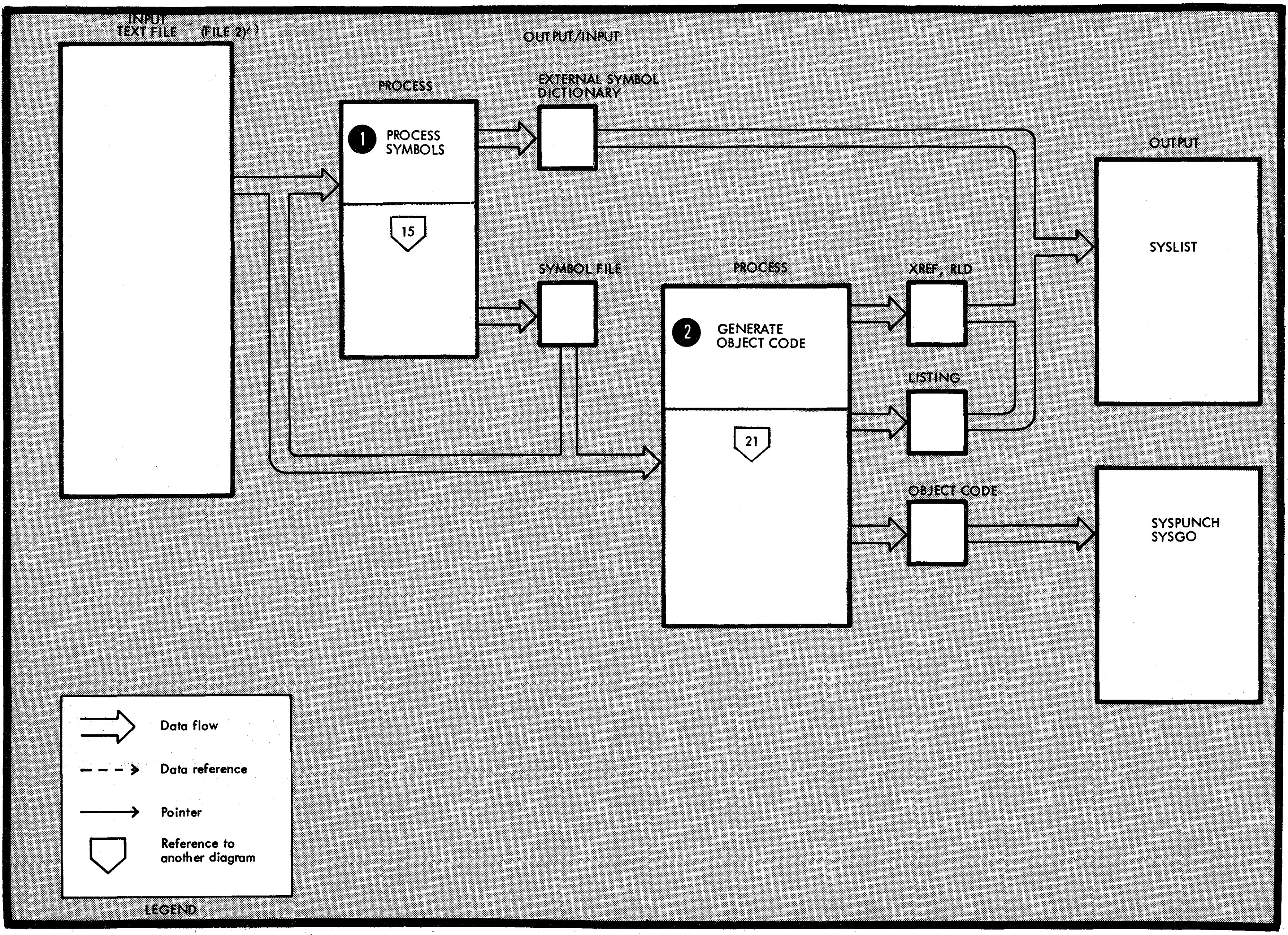
	MODULE	ROUT!NE (LABEL)
If a reference to a macro parameter, the value is retrieved from the parameter table. If a reference to a variable symbol, it is retrieved from the relevant dictionary.	IFNX3N	PARMTBLR LCLDICTR GBLDICTR

VARIABLE SYMBOL SUBSTITUTION

	MODULE	ROUT!NE (LABEL)
Evaluate as a SETC operand and move the value into the generated text record. If substitution is performed in the operator field, check against the OPSYN and opcode tables for validity.	IFNX3A	GENFLD GENSTRNG EVAL RESOLVE
	IFNX3N	GBLDICTR LCLDICTR PARMTBLR ORDSYMBR

Assemble Object Code from Machine, Data, and Assembler Instructions

14



Assemble Object Code from Machine, Data, and Assembler Instructions (cont.)

MODULE

After all macro instructions have been expanded and conditional assembly in open code performed, the assembler is ready to generate object code from the assembler and machine instructions. During generation symbol definitions and references are collated in the symbol file 1 (Work File 3).

❶ As each text record is processed, symbols are defined (that is, given addresses) and their definitions (and resolved references) collected in a symbol file. IFNX3B

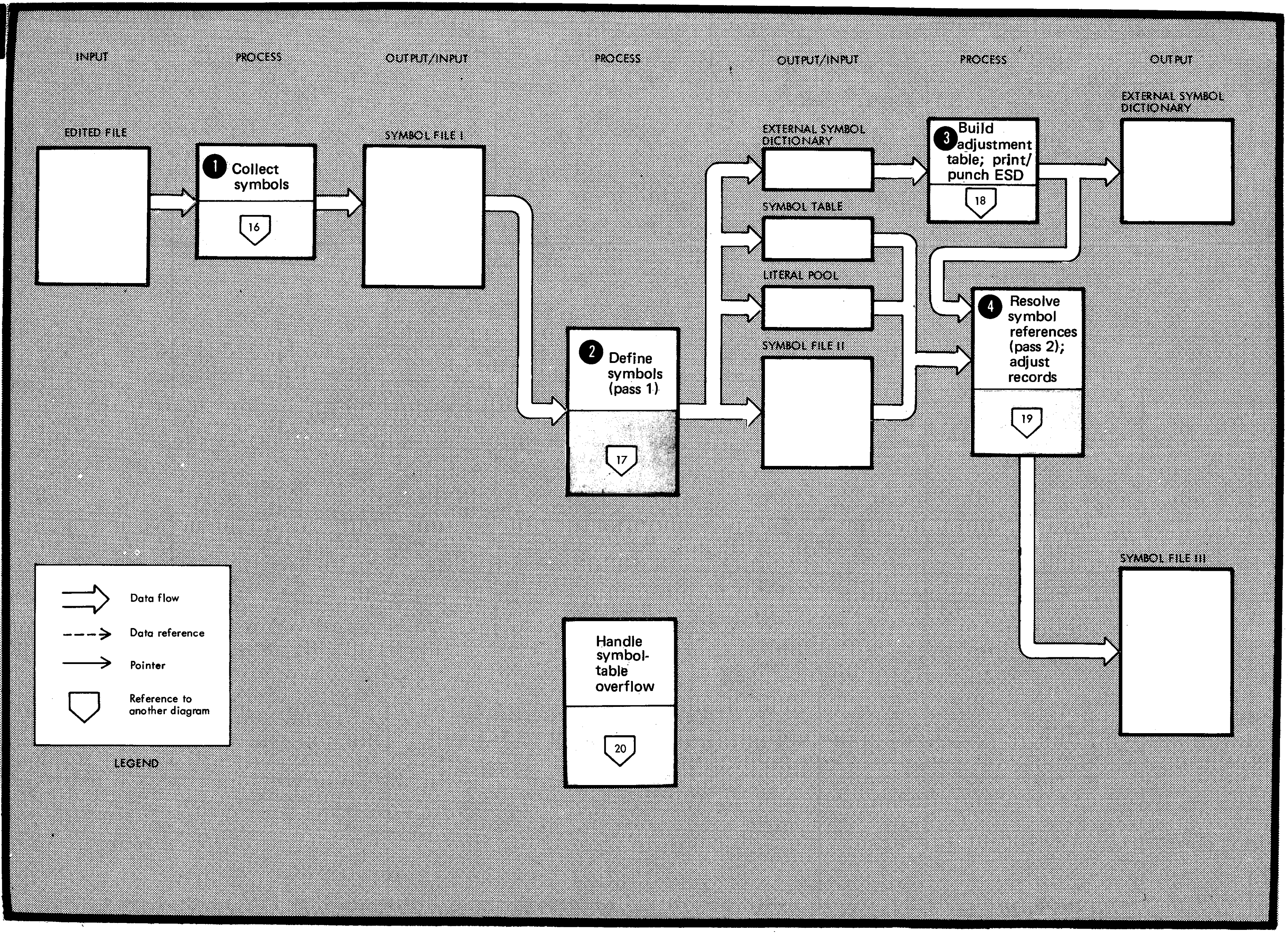
MODULE

Information for each control section is collected in the external symbol dictionary, which is passed directly to be printed. IFNX41

❷ The text file is read with the symbol file, to generate the object code. Output is object code, put to either SYSGO or SYSPUNCH, and a listing. IFNX51

Process Symbols

15


INPUT
PROCESS
OUTPUT/INPUT
PROCESS
OUTPUT/INPUT
PROCESS
OUTPUT
EDITED FILE
1 Collect symbols
16
SYMBOL FILE I
2 Define symbols (pass 1)
17
EXTERNAL SYMBOL DICTIONARY
SYMBOL TABLE
LITERAL POOL
SYMBOL FILE II
3 Build adjustment table; print/ punch ESD
18
EXTERNAL SYMBOL DICTIONARY
4 Resolve symbol references (pass 2); adjust records
19
SYMBOL FILE III
Handle symbol-table overflow
20
Data flow
Data reference
Pointer
Reference to another diagram
LEGEND

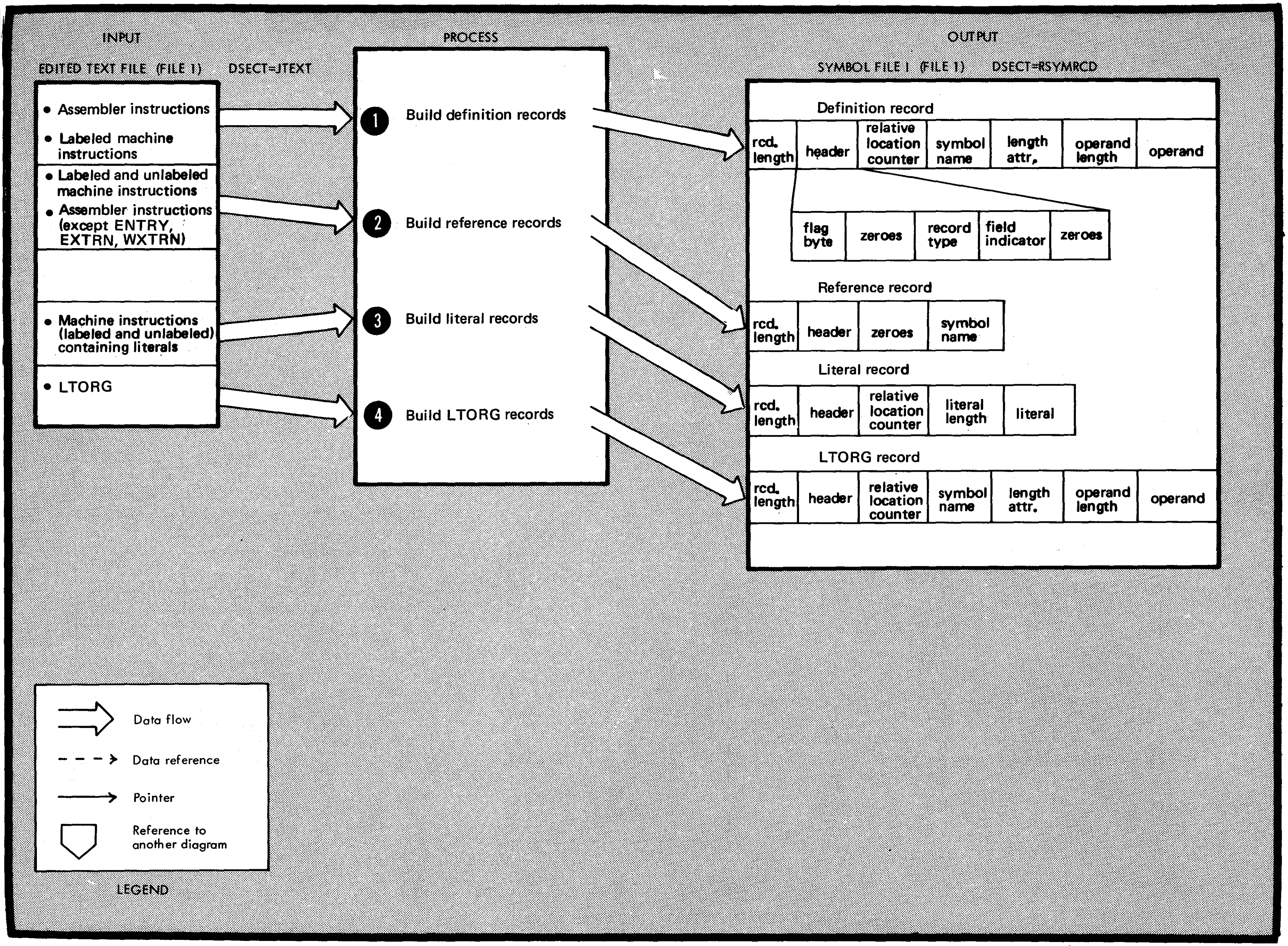
Process Symbols (cont.)

		MODULE	ROUTINE (LABEL)
1	The edited text is scanned and a sequential symbol file ("symbol file I") of all records necessary for symbol resolution produced. The file consists of symbol definitions, symbol references, literals, and other assembler operations affecting the ESD or location counter.	IFNX3B	
2	Symbol file I is scanned and the ESDID and location counter updated for all symbol definitions and references. Symbol definitions (ESDID and location counter values) are entered in the symbol table. External symbol dictionary entries are made for control sections, dummy control sections, external dummy sections, external symbols and entry-point symbols. The symbol table is searched for all references and the reference resolved if possible. A literal pool is built.	IFNX4M IFNX4D IFNX4E IFNX4S IFNX4V	

		MODULE	ROUTINE (LABEL)
3	The adjustment table is used to add the start value of a control section to a symbol's location counter value (for symbols defined in a control section that does not start at 0). It is also used to change the ESDID for all symbols defined in a DSECT referenced by a Q-type address constant.	IFNX4E	MAKESD
4	Symbol file II is scanned and symbol references resolved with the help of the symbol table. Literal references are resolved. Resolved symbol records are written on symbol file III. All ESDIDs and location-counter values are adjusted, if necessary.	IFNX4M IFNX4S	
5	Special handling is necessary if the symbol table overflows. See Diagram 20.	IFNX4M IFNX4D IFNX4E IFNX4S IFNX4V	

Collect Symbols

16



INPUT

EDITED TEXT FILE (FILE 1) DSECT=JTEXT

- Assembler instructions
- Labeled machine instructions
- Labeled and unlabeled machine instructions
- Assembler instructions (except ENTRY, EXTRN, WXTRN)
- Machine instructions (labeled and unlabeled) containing literals
- LTORG

PROCESS

1 Build definition records

2 Build reference records

3 Build literal records

4 Build LTORG records

OUTPUT

SYMBOL FILE I (FILE 1) DSECT=RSYMRCD

Definition record

rcd. length	header	relative location counter	symbol name	length attr.	operand length	operand

flag byte	zeroes	record type	field indicator	zeroes

Reference record

rcd. length	header	zeroes	symbol name

Literal record

rcd. length	header	relative location counter	literal length	literal

LTORG record

rcd. length	header	relative location counter	symbol name	length attr.	operand length	operand

Data flow

Data reference

Pointer

Reference to another diagram

LEGEND

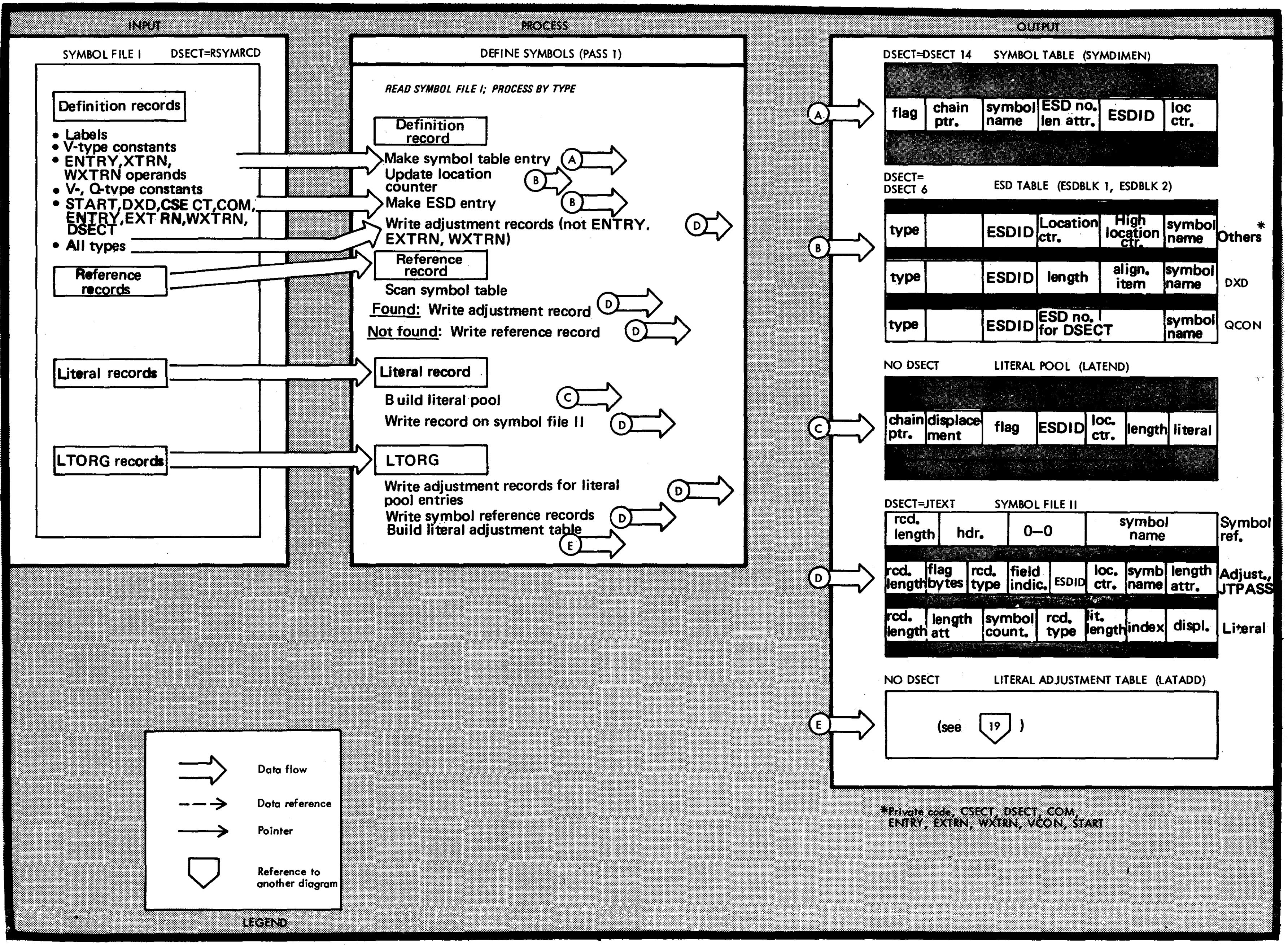
Collect Symbols (cont.)

	MODULE	ROUTINE (LABEL)
The edited text file (output from Generate) is read and all symbol definitions and references logged in symbol file I.	IFNX3B	
❶ A definition record is built for each assembler instruction and labeled machine instruction. The relative (that is, relative to the last definition or literal record) location counter value and length attribute are placed in the output record.	IFNX3B	

	MODULE	ROUTINE (LABEL)
❷ Symbol reference records are built for all machine and assembler instructions that have symbols in their operands, except for ENTRY, EXTRN, and WXTRN.	IFNX3B	
❸ Machine instructions are scanned for literals and literal records built. The relative location counter value is placed in the record.	IFNX3B	
❹ LTORG records are built when LTORG statements are encountered.	IFNX3B	

Define Symbols (Pass 1)

17



INPUT

SYMBOL FILE I DSECT=RSYMRCD

Definition records

- Labels
- V-type constants
- ENTRY,XTRN, WXTRN operands
- V-, Q-tvpe constants
- START,DXD,CSE CT,COM, ENTRY,EXT RN,WXTRN, DSECT
- All types

Reference records

Literal records

LTORG records

PROCESS

DEFINE SYMBOLS (PASS 1)

READ SYMBOL FILE I; PROCESS BY TYPE

Definition record

Make symbol table entry (A)
Update location counter (B)
Make ESD entry (B)
Write adjustment records (not ENTRY. EXTRN, WXTRN) (D)

Reference record

Scan symbol table
Found: Write adjustment record (D)
Not found: Write reference record (D)

Literal record

Build literal pool (C)
Write record on symbol file II (D)

LTORG

Write adjustment records for literal pool entries (D)
Write symbol reference records (D)
Build literal adjustment table (E)

OUTPUT

(A) DSECT=DSECT 14 SYMBOL TABLE (SYMDIMEN)

flag	chain ptr.	symbol name	ESD no. len attr.	ESDID	loc ctr.

(B) DSECT= DSECT 6 ESD TABLE (ESDBLK 1, ESDBLK 2)

type		ESDID	Location ctr.	High location ctr.	symbol name	Others*
type		ESDID	length	align. item	symbol name	DXD
type		ESDID	ESD no. for DSECT		symbol name	QCON

(C) NO DSECT LITERAL POOL (LATEND)

chain ptr.	displacement	flag	ESDID	loc. ctr.	length	literal

(D) DSECT=JTEXT SYMBOL FILE II

rcd. length	hdr.	0–0	symbol name	Symbol ref.

rcd. length	flag bytes	rcd. type	field indic.	ESDID	loc. ctr.	symb name	length attr.	Adjust., JTPASS

rcd. length	length att	symbol count.	rcd. type	lit. length	index	displ.	Literal

(E) NO DSECT LITERAL ADJUSTMENT TABLE (LATADD)

(see 19)

*Private code, CSECT, DSECT, COM, ENTRY, EXTRN, WXTRN, VCON, START

Define Symbols (Pass 1) (cont.)

	MODULE	ROUTINE (LABEL)
Records are read from symbol file I and processed as follows:		
DEFINITION RECORDS		
Make Symbol Table Entry. A symbol-table entry is made for all symbols defined in the name field of statements or in the operand field of EXTRN, WXTRN, and ENTRY statements.	IFNX4S	ENTER
Update Location Counter. The relative location counter value in the symbol definition record is added to the current location counter in the ESD. If a DS or DC statement, the operand is evaluated and the location counter updated by the length of the constant.	IFNX4M IFNX4D	
Make ESD Entry. ESD entries are made for each unique START, CSECT, DSECT, DXD, COM, ENTRY, WTRN, and EXTRN symbol and Q- and V- type address constants. ESDID and ESD numbers are assigned in ascending sequence from 1 for all entries. There are two series of ESDID numbers, both assigned in ascending sequence from 1. One set is assigned to DSECTs, the other to other entries of all types.	IFNX4E	BLDESD ENTRY EXTRN VCON QCON

	MODULE	ROUTINE (LABEL)
Write Adjustment Records. All labeled definition records and unlabeled START, CSECT, DSECT, COM, DC and DS records are changed to "adjustment records" (that is, marked for later adjustment -- see Diagram 19) and written on symbol file II. Current ESDID and location counter values are moved into the adjustment record, as are the length attributes of the symbols.	IFNX4M	WRITE
EXTRN and WTRN records, since they are not processed in pass 2, are changed to JTPASS (not needing adjustment) records.	IFNX4E	EXTRN
ENTRY statements receive special handling: if the symbol is not found in the symbol table, the ESDID and location counter value can be passed to the record and it needs only to be adjusted. If found, the type is changed to symbol reference and the symbol is resolved in pass 2.	IFNX4E	ENTRY
REFERENCE RECORDS		
The symbol table is searched for the symbol in the reference record. If the name is found, the reference can be resolved. Location counter value, ESDID, and length attribute are moved from the symbol table into the symbol record. Record type is changed to "adjustment" and the record written on symbol file II. If the name is not found, the record is written unchanged on symbol file II to be resolved in the next pass.	IFNX4M IFNX4S	SYMBOL FIND

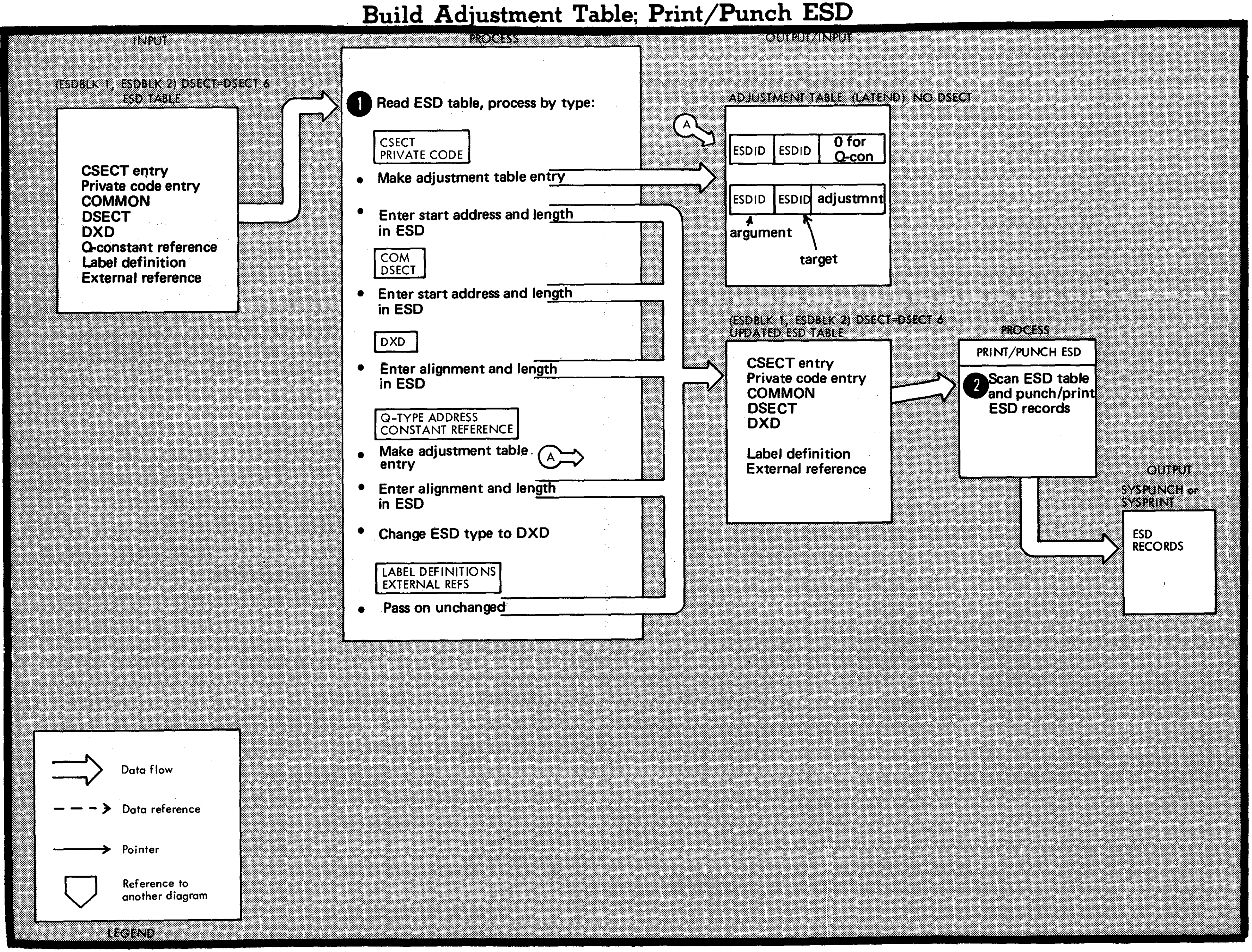
	MODULE	ROUTINE LABEL
LITERAL RECORDS		
Literal definitions are collected and the length of the generated constant computed. Each unique literal is then hashed and entered into one of the chains in the literal pool. When the literal is entered, or if it already is in the pool, its chain identification and the displacement of the literal within that chain are noted and written in the record. (The literal pool is a table containing a hash table and four chains of all the unique literals that have been defined since the start of the assembly or since the last LTORG statement. The hash table consists of four pointers, each the address of one of the chains. Each chain is terminated by a zero chain pointer.) Literal records are written on symbol file II.	IFNX4M IFNX4D	LITERAL

	MODULE	ROUTINE LABEL
LTORG		
Write Adjustment Record. The literal pool is scanned and an adjustment record written on symbol file II for each entry. The location counter is updated for each entry.	IFNX4M IFNX4D	LTORG LTDUMP
Write Symbol Reference Record. A symbol reference record is written on symbol file II for each symbol in the literal statement.	IFNX4E	REFER
Build Literal Adjustment Table. The literal adjustment table is built by adding the current location counter value to the chain lengths to get the starting addresses of the literal chains.	IFNX4D	LTORG

This page intentionally left blank

18

Build Adjustment Table; Print/Punch ESD



INPUT

(ESDBLK 1, ESDBLK 2) DSECT=DSECT 6
ESD TABLE

CSECT entry
Private code entry
COMMON
DSECT
DXD
Q-constant reference
Label definition
External reference

PROCESS

1 Read ESD table, process by type:

CSECT
PRIVATE CODE

- Make adjustment table entry
- Enter start address and length in ESD

COM
DSECT

- Enter start address and length in ESD

DXD

- Enter alignment and length in ESD

Q-TYPE ADDRESS
CONSTANT REFERENCE

- Make adjustment table entry A
- Enter alignment and length in ESD
- Change ESD type to DXD

LABEL DEFINITIONS
EXTERNAL REFS

- Pass on unchanged

OUTPUT/INPUT

ADJUSTMENT TABLE (LATEND) NO DSECT

A

ESDID	ESDID	0 for Q-con
ESDID	ESDID	adjustmnt

argument

target

(ESDBLK 1, ESDBLK 2) DSECT=DSECT 6
UPDATED ESD TABLE

CSECT entry
Private code entry
COMMON
DSECT
DXD

Label definition
External reference

PROCESS

PRINT/PUNCH ESD

2 Scan ESD table and punch/print ESD records

OUTPUT

SYSPUNCH or SYSPRINT

ESD
RECORDS

Data flow

Data reference

Pointer

Reference to another diagram

LEGEND

Build Adjustment Table; Print/Punch ESD (cont.)

MODULE ROUTINE (LABEL)

The adjustment table consists of two kinds of entries:

1. Those used to adjust location-counter values for symbols defined in a given CSECT or private code.

2. Those used to change the ESDID for all symbols defined in a DSECT and all references to these symbols if the DSECT is referenced by a Q-type address constant.

An entry consists of three parts: an argument ESDID, a target ESDID, and an adjustment factor. For (1), above, both the argument and the target ESDID are the ESDID of the CSECT or private code. The adjustment factor is the start address of the CSECT or private code. For (2) the adjustment factor is 0. The argument ESDID is the ESDID for the DSECT. The target ESDID is the ESDID for the Q-type constant reference.

❶ Read the ESD table and process by type:

CSECT
Private Code — IFNX4E MAKESD SUMESD

- The start address of the CSECT or private code is computed from the lengths of the previous control sections or the start value of private code. If the START address of the CSECT or private code is nonzero, the start address is entered into the adjustment factor and the ESDID moved from the ESD entry to both argument and target ESDID. — SUMCST

MODULE ROUTINE (LABEL)

- The start address and the length are entered in the ESD table. The length of the section is retrieved from the original ESD entry. — SUMCST

COM
DSECT

Same as the second step above, except that the start address is 0. — SUMCST

DXD

The alignment and length are entered into the ESD entry. for DXD, the alignment factor and the length are obtained from the ESD entry and the fields re-ordered (reversed). — SUMDXD

Q-type Address Constant Reference

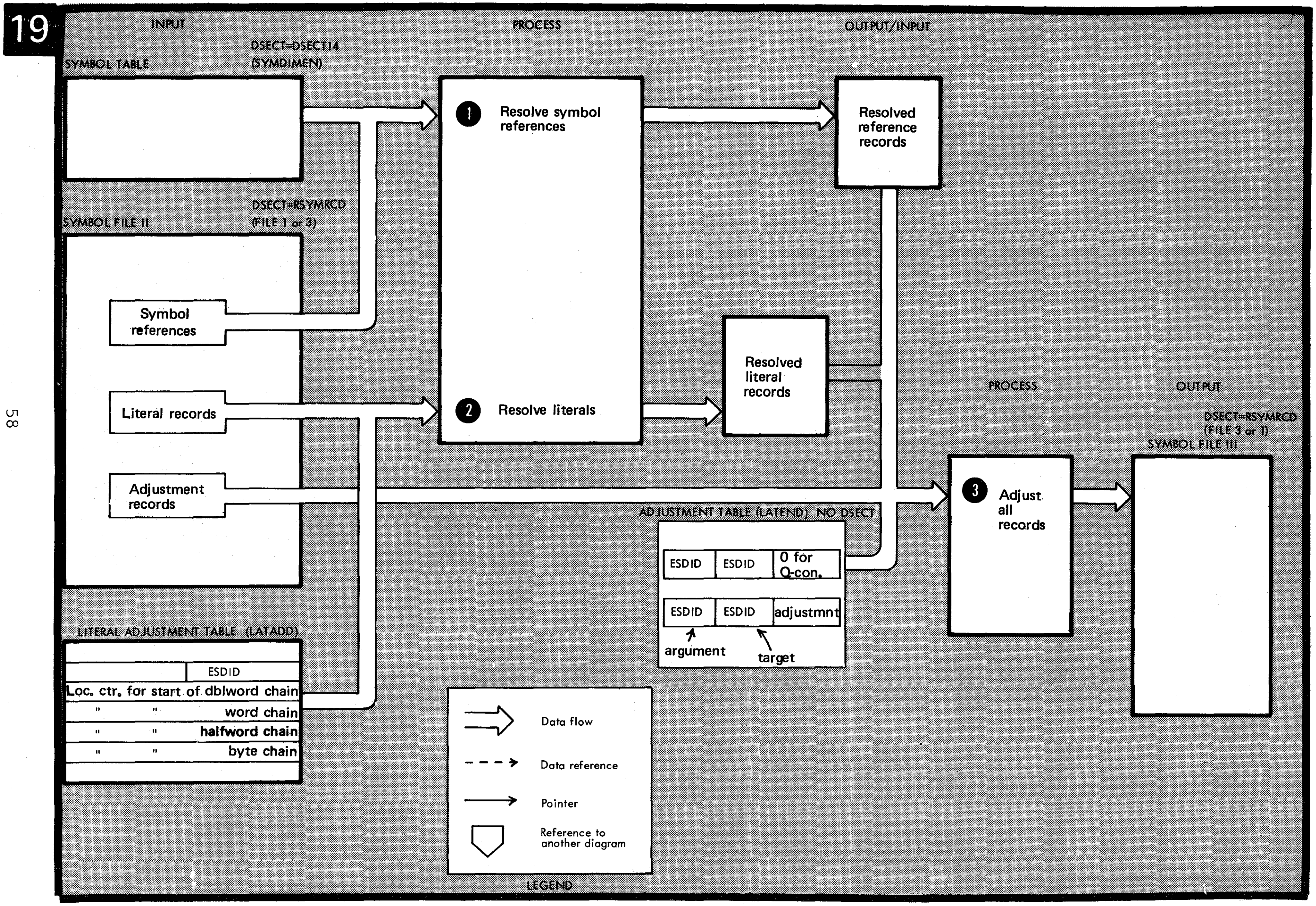
The alignment factor is set to 7. The length of the referenced DSECT is obtained from its ESD entry. The alignment factor and length are stored in the Q-type address constant reference ESD entry. — SUMDSD

Label Definitions
External References

These are passed from the old to the updated ESD table without change. — SUMGET

❷ The updated ESD table is scanned and a record printed or punched for all entries except DSECTs. — IFNX4E MAKGET

Resolve Symbol References (Pass 2); Adjust Records



ESDID	ESDID	0 for Q-con.
ESDID	ESDID	adjustmnt

	ESDID
Loc. ctr. for start of dblword chain	
" " word chain	
" " halfword chain	
" " byte chain	

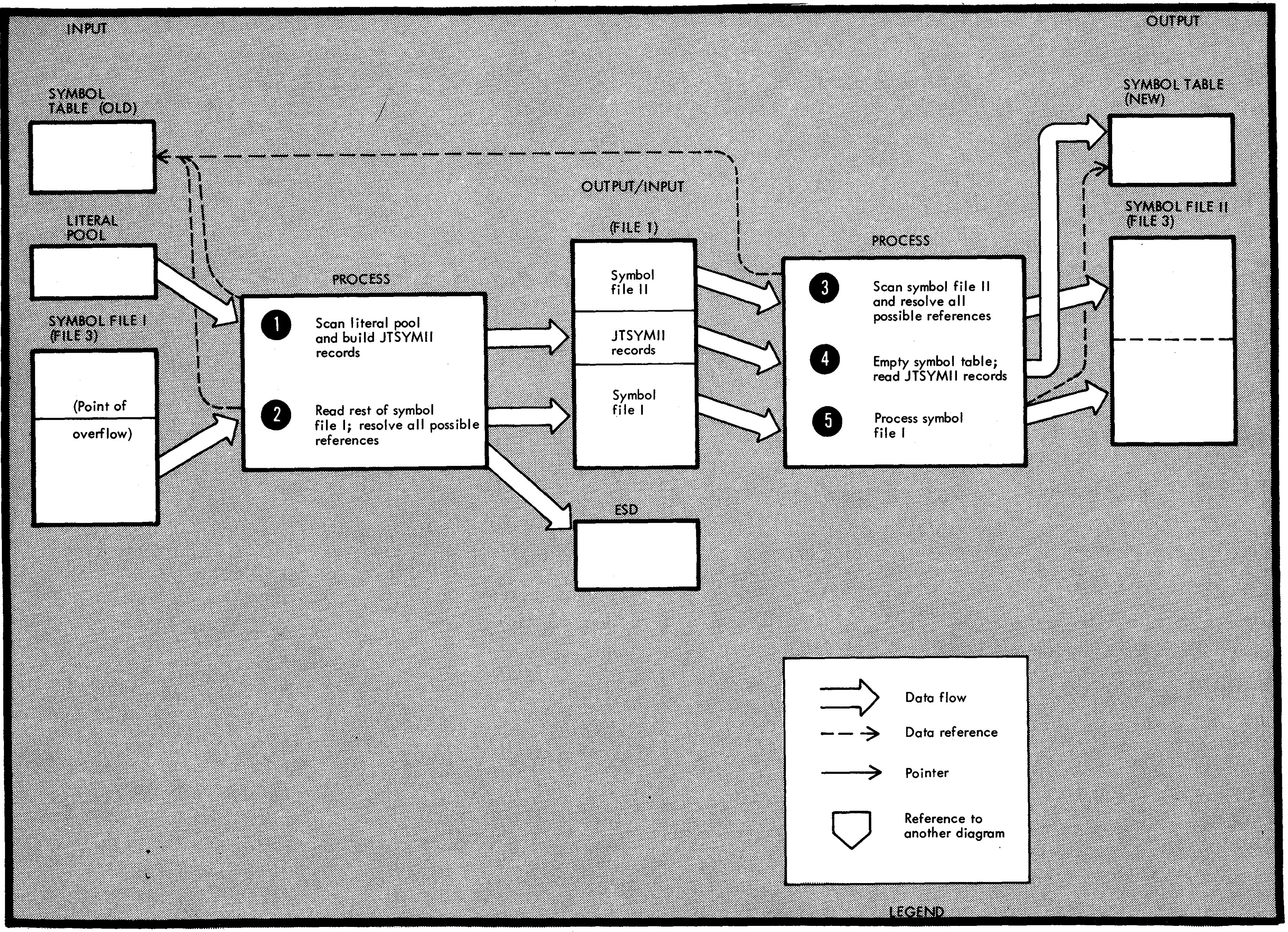
Resolve Symbol References (Pass 2); Adjust Records (cont.)

		MODULE	ROUTINE (LABEL)
1	Resolve symbol references. The symbol referenced in symbol file II is searched for in the symbol table. If it is not found, the record is transferred unchanged to symbol file III as an "undefined symbol record". If found, the location counter value, the ESDID, and the length attribute are moved from the symbol table entry to the symbol reference record and it is adjusted (see step 3, below).	IFNX4M IFNX4S	SYMBL FIND
2	Resolve literals.		
•	The pointer to the corresponding entry in the literal adjustment table is obtained from the literal record.	IFNX4M	SYMBL
•	The location-counter value to the start of the appropriate literal chain is obtained from the literal adjustment table.		LITRII
•	The displacement into the literal chain (obtained from the literal record) is added to the location counter value obtained in the previous step. The result is the resolved location-counter value for the literal.		

		MODULE	ROUTINE (LABEL)
•	The ESDID for the literal is obtained from the literal adjustment table and stored in the literal record.		
•	The record is adjusted (see step 3, below).		ADJUST
3	Adjust all records.	IFNX4M	ADJUST
•	The ESDID is obtained from the record.		
•	The adjustment table is searched for a corresponding argument ESDID. If a match is not found, the record is transferred to symbol file III.		
•	If a match is found, the ESDID in the corresponding target ESDID is stored in the symbol record.		
•	The corresponding adjustment factor is added to the location counter value in the symbol record.		
•	The record is written on symbol file III.		

Handle Symbol Table Overflow

20

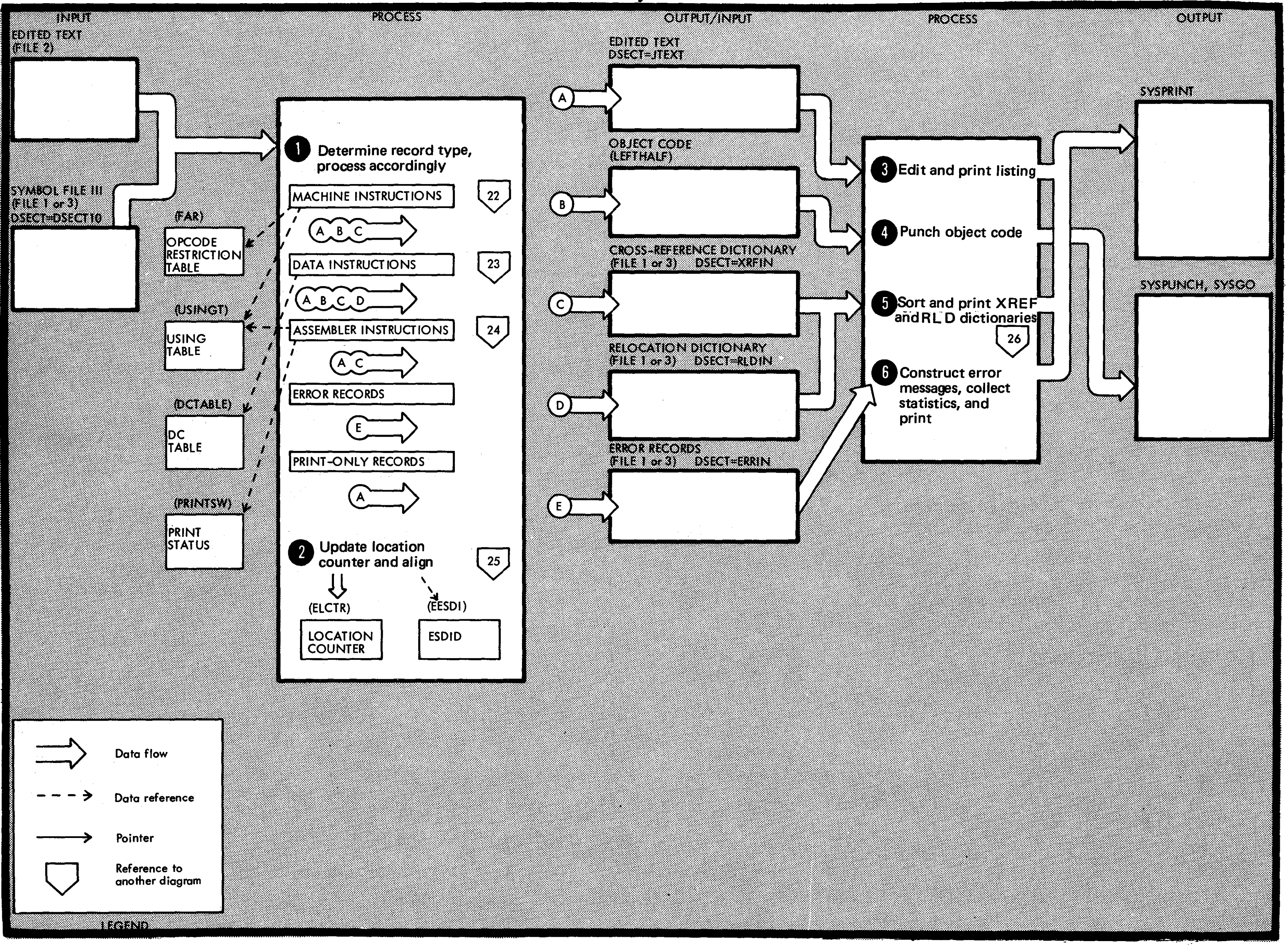


Handle Symbol Table Overflow (cont.)

	MODULE	ROUTINE (LABEL)
When, during symbol resolution (see Diagram 17), the symbol table is filled, it is necessary to take special action to process the rest of symbol file I.		
1 The literal pool and the symbol table are scanned for symbol table entries corresponding to symbols in literals. The symbol table entries (called JTSYMII records) are copied onto file I. These will later be written into the new symbol table to resolve literals.	IFNX4M IFNX4D IFNX4E	LTDUMP REFER
2 Symbol File I is read from the point at which the symbol table overflowed. The symbol table is searched for each symbol record. If the symbol is not found, the record is simply transferred. If it is found, it is processed by type:	IFNX4M	TRANSFER SEARCH ENTRY EXTRN
ENTRY records: if a CSECT name, the record type is changed to JTSYMBL and passed. If not a CSECT name, the ESDID and the location counter value are moved from the symbol table to the symbol record and the type changed to adjustment record. An ESD entry is made.	IFNX4S IFNX4E	FIND ENTRY EXTRN
Symbol Reference Records: the type is changed to adjustment.		
Others: the record is marked "symbol previously defined" and passed.		
3 After the rest of symbol file I has been written on file 1, symbol file II is scanned and written on file 3. All symbol references are resolved, if possible.	IFNX4M IFNX4S	SYMBL TRANSFER FIND
4 The old symbol table is now emptied of all entries that do not define ESD items and the JTSYMII records read into it.	IFNX4M	REHASH EOFIIS
5 The rest of file I (the remaining part of symbol file I) is now read and processed with the new symbol table (as in Diagram 17).	IFNX4S	SUBSET ENTER

21

Generate Object Code



Generate Object Code (cont.)

	MODULE	ROUTINE (LABEL)
Object code is built from statements read from the edited text file (file 2). When a symbol is encountered, the symbol file (JINFILE) is used to cross-reference the symbol and to resolve addresses. A relocation dictionary (RLD) entry is made for relocatable address constants.	IFNX5M	
Output is object code (to SYSPUNCH or SYSGO) and a listing (to SYSPRINT).		
1 Records are read from the edited text file and the type of statement determined from the operation code and the "FLAGA" field.	IFNX5C	TEXTGET
Processing proceeds according to record type (machine instruction, data instruction, assembler instructions, error records, print-only records).		
Machine Instructions (Diagram 22)		
Each instruction is processed according to its type and its operand restrictions (as listed in the opcode restriction table). Implicit addresses are resolved by means of the using table and cross-reference entries are made for all symbols and literals that appear in the statement.	IFNX5M	
Data Instructions (Diagram 23)		
Each DC, DS, CXD, and DXD instruction or literal definition is processed according to type. Cross-reference entries are made for symbols and literals. Relocation dictionary entries are made for relocatable address constants. CCW, REPRO, and PUNCH statement processing is also shown here because these statements, unlike other assembler statements, generate object code.	IFNX5A IFNX5D IFNX5F	

	MODULE	ROUTINE (LABEL)
Assembler Instructions (Diagram 24)		
These statements (which do not produce object code) are processed according to type.	IFNX5A	
Error Records		
The statement number assigned to a statement flagged during previous phases is inserted into the error record that follows the statement. Then the error record is written on file 1 or 3 for subsequent processing by diagnostic routines. Error records for errors detected in this phase are also built, the number of the statement in error inserted, and the record written on file 1 or 3.	IFNX5A IFNX5P	ERRORO
Print-Only Records		
These records (remarks, etc.) are edited and written directly on SYSPRINT.	IFNX5A	AOP350
2 Each instruction generating object code causes the location counter to be updated. In addition, the location counter is updated by ORG, CNOP, CSECT, DSECT, COM, and START assembler statements. Alignment is done for CCW, CNOP, LTORG, and CXD statements, as well as for DS, DC and machine instructions requiring it. (See Diagram 25.)	IFNX5A	LOCUPD
3 For each statement the object code built is packed and the virtual text is inserted into the print line together with the packed code. Depending on the linecount option given, new pages are made and headings are printed.	IFNX5P	

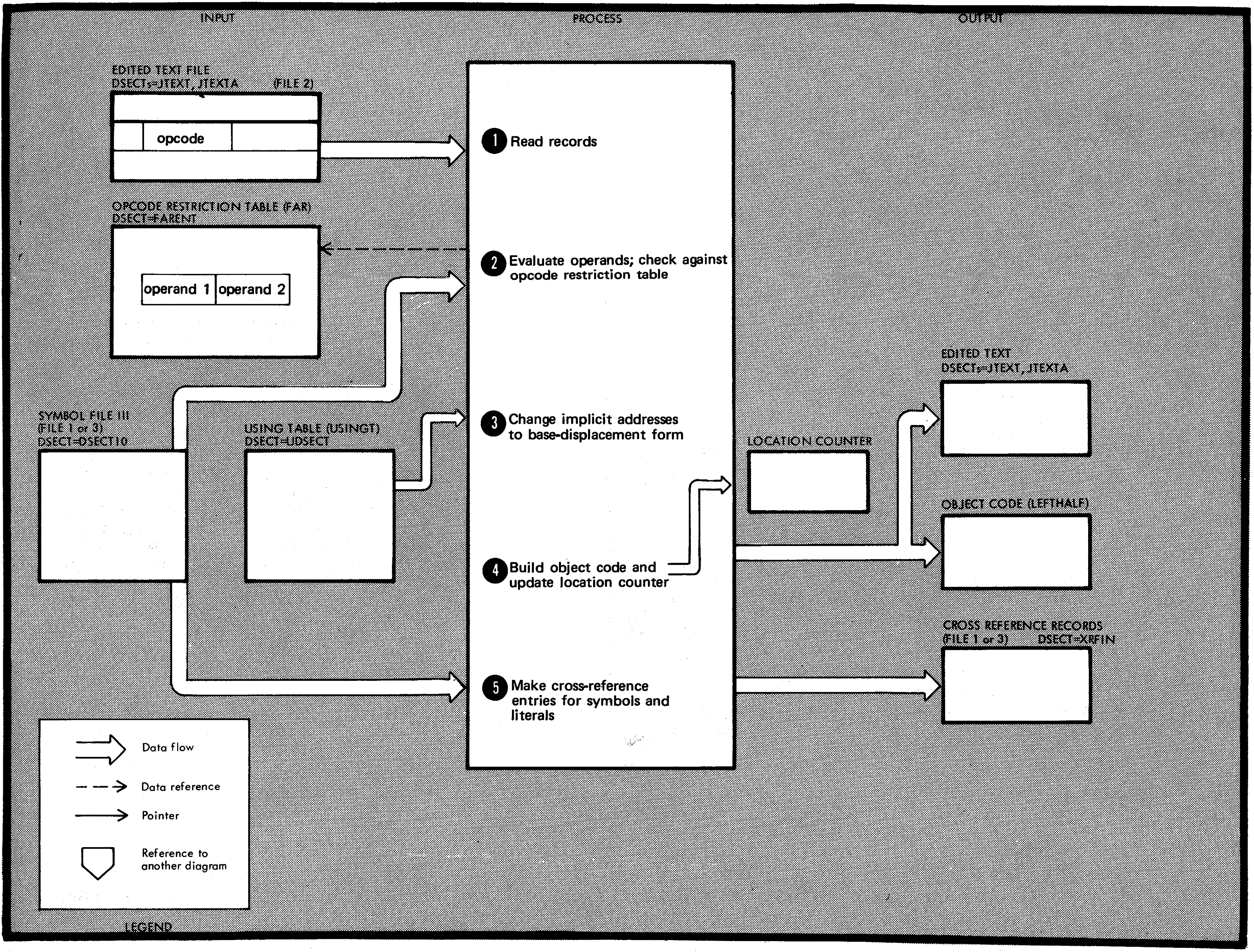
GENERATE OBJECT CODE (cont.)

		MODULE	ROUTINE LABEL
4	The object code is packed into the current record. When the 80 bytes are filled or the ESDID for the code changes, the current record is punched and a new one initiated.	IFNX5P	PUNRTN
5	RLD and XREF records are stored before they are edited and printed (see Diagram 26).	IFNX6A	
6	The error number in the edited record is used to locate the message text associated with the number from the table in IFNX6C. This text is scanned for $ and #, which indicate insertion. A $ indicates that an appended data field is to be inserted in the text. A # indicates that the text NEAR OPERAND COLUMN followed by the value of the column pointer is to be inserted.	IFNX6B	
	The line is edited to remove unnecessary blanks and the statement number inserted.		
	During printing of the error messages, the number of statements flagged is counted and the highest severity code encountered is saved.		

This page intentionally left blank

Process Machine Instructions

22


INPUT
PROCESS
OUTPUT
EDITED TEXT FILE
DSECTs=JTEXT, JTEXTA (FILE 2)
opcode
OPCODE RESTRICTION TABLE (FAR)
DSECT=FARENT
operand 1
operand 2
SYMBOL FILE III
(FILE 1 or 3)
DSECT=DSECT10
USING TABLE (USINGT)
DSECT=UDSECT
1 Read records
2 Evaluate operands; check against opcode restriction table
3 Change implicit addresses to base-displacement form
4 Build object code and update location counter
5 Make cross-reference entries for symbols and literals
LOCATION COUNTER
EDITED TEXT
DSECTs=JTEXT, JTEXTA
OBJECT CODE (LEFTHALF)
CROSS REFERENCE RECORDS
(FILE 1 or 3) DSECT=XRFIN
Data flow
Data reference
Pointer
Reference to another diagram
LEGEND

Process Machine Instructions (cont.)

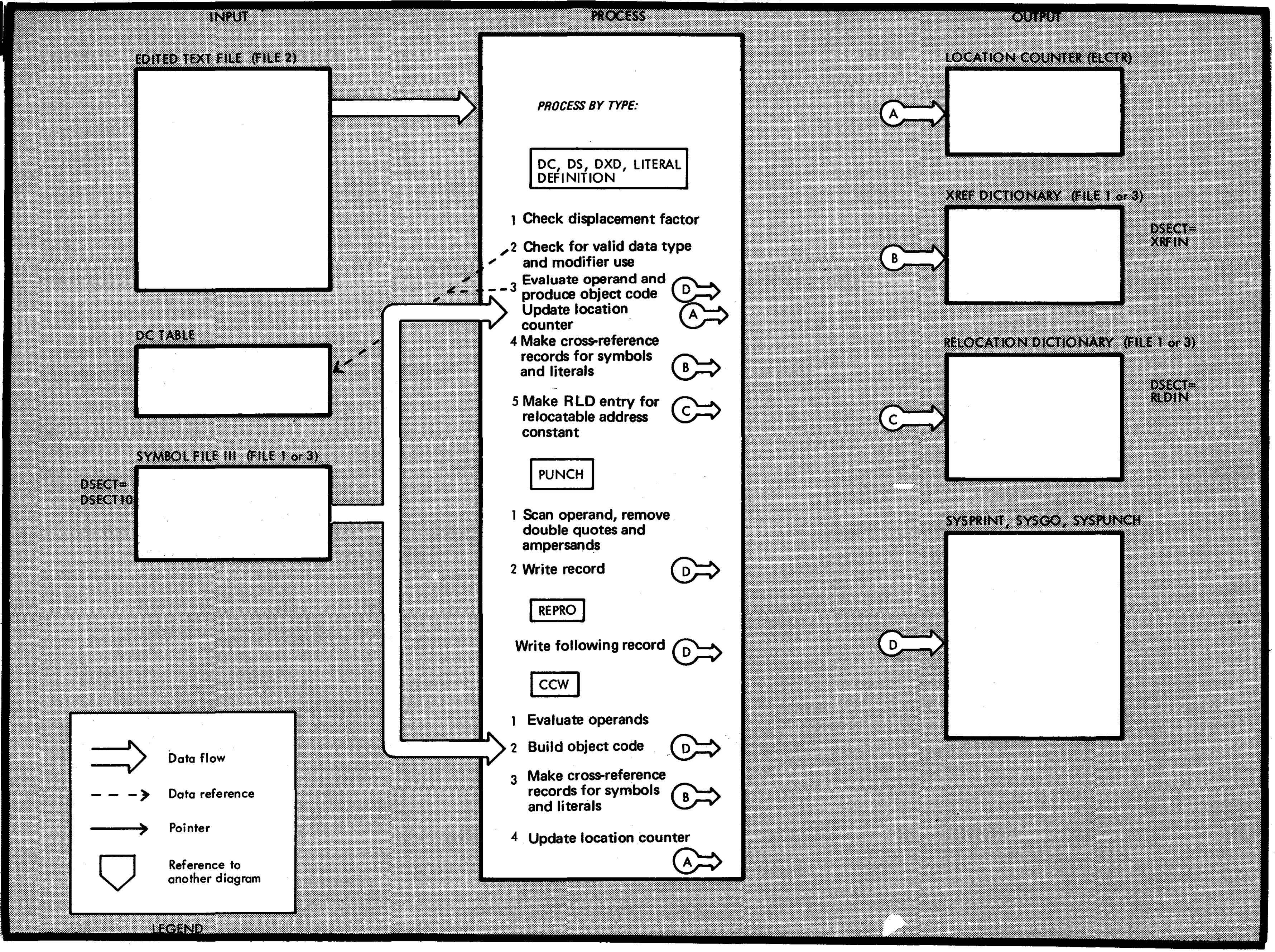
Two tables are used in the processing: the opcode restriction table (an in-core table containing data on opcodes) and the using table (containing available base registers and their associated values and ESDIDs). When a symbol is encountered, its definition is obtained from the symbol file; an entry to the cross reference dictionary is also made.

	MODULE	ROUTINE (LABEL)
❶ Records are read from the edited text file. The opcode byte in the edited record is used to find the associated entry in the opcode restriction table. The table contains one entry for each operand allowed. The operands can be classified as I, S, SX, and SL. One operand can contain both an immediate data portion (mask, register or length) and a storage data part (data address or implicit address).	IFNX5M	
❷ The operand is evaluated according to information in the opcode restriction table. This table contains information on operand type, allocation of fields in the object code, restrictions on divisibility and upper boundaries of immediate data, alignment, whether or not literals are allowed, and if execution of the instruction modifies storage.	IFNX5M	DRIVER

	MODULE	ROUTINE (LABEL)
❸ Implicit addresses are decomposed to base-displacement form by means of the using table. The table is searched for the register giving the smallest displacement among those available. If two registers give the same displacement, the higher numbered register is used.	IFNX5M	SPART
❹ Object code for the instruction is built and the location counter updated. In the listing, this code is printed at the left of the source statements.	IFNX5M	IASGN SPASGN
❺ When a symbol or literal is found in the edited text record, a record is read from the symbol file. From the information in this record a cross-reference record is built containing a flag telling if a definition or a reference record, the name of the symbol, the statement number, the length attribute value, and the location-counter value for the symbol. This information will later be used to build the cross-reference dictionary (Diagram 25).	IFNX5V	SYM

Process Data Instructions

23


INPUT
PROCESS
OUTPUT
EDITED TEXT FILE (FILE 2)
DC TABLE
SYMBOL FILE III (FILE 1 or 3)
DSECT=
DSECT10
LOCATION COUNTER (ELCTR)
XREF DICTIONARY (FILE 1 or 3)
DSECT=
XRFIN
RELOCATION DICTIONARY (FILE 1 or 3)
DSECT=
RLDIN
SYSPRINT, SYSGO, SYSPUNCH
A
B
C
D
Data flow
Data reference
Pointer
Reference to another diagram
LEGEND

PROCESS BY TYPE:

DC, DS, DXD, LITERAL DEFINITION

1 Check displacement factor

2 Check for valid data type and modifier use

3 Evaluate operand and produce object code (D)
Update location counter (A)

4 Make cross-reference records for symbols and literals (B)

5 Make RLD entry for relocatable address constant (C)

PUNCH

1 Scan operand, remove double quotes and ampersands

2 Write record (D)

REPRO

Write following record (D)

CCW

1 Evaluate operands

2 Build object code (D)

3 Make cross-reference records for symbols and literals (B)

4 Update location counter (A)

Process Data Intructions (cont.)

Included here are those assembler instructions which generate data in the object code: DC, DS, DXD, REPRO and CCW.

DC, DS, DXD, Literal Definition

	MODULE	ROUTINE (LABEL)
1 For DS, DXD, and DC instructions with duplication factor 0, no object code is built; if no duplication factor if given, the default value is 1.	IFNX5A	DS0100 DC0100 DXD100
2 A check is made to insure that the specified data type is valid and that the specified modifiers are within the ranges given in the DC table. If no modifiers are supplied, default values are used.	IFNX5D	DCEVAL
3 The last part of each entry in the DC table is a branch address to the routine handling the given data type. These routines scan the operands and evaluate them. Values of symbols are obtained from the corresponding symbol records.	IFNX5D	CKON DKON XKON AYKON BKON VKON PKON QKON ZKON SKON

	MODULE	ROUTINE (LABEL)
4 An entry is made in the cross-reference dictionary for symbols and literals.	IFNX5A	XREF
5 Relocation dictionary entries are made for address constants with relocatable expressions in the operand.	IFNX5A	RLDOUT

PUNCH, REPRO

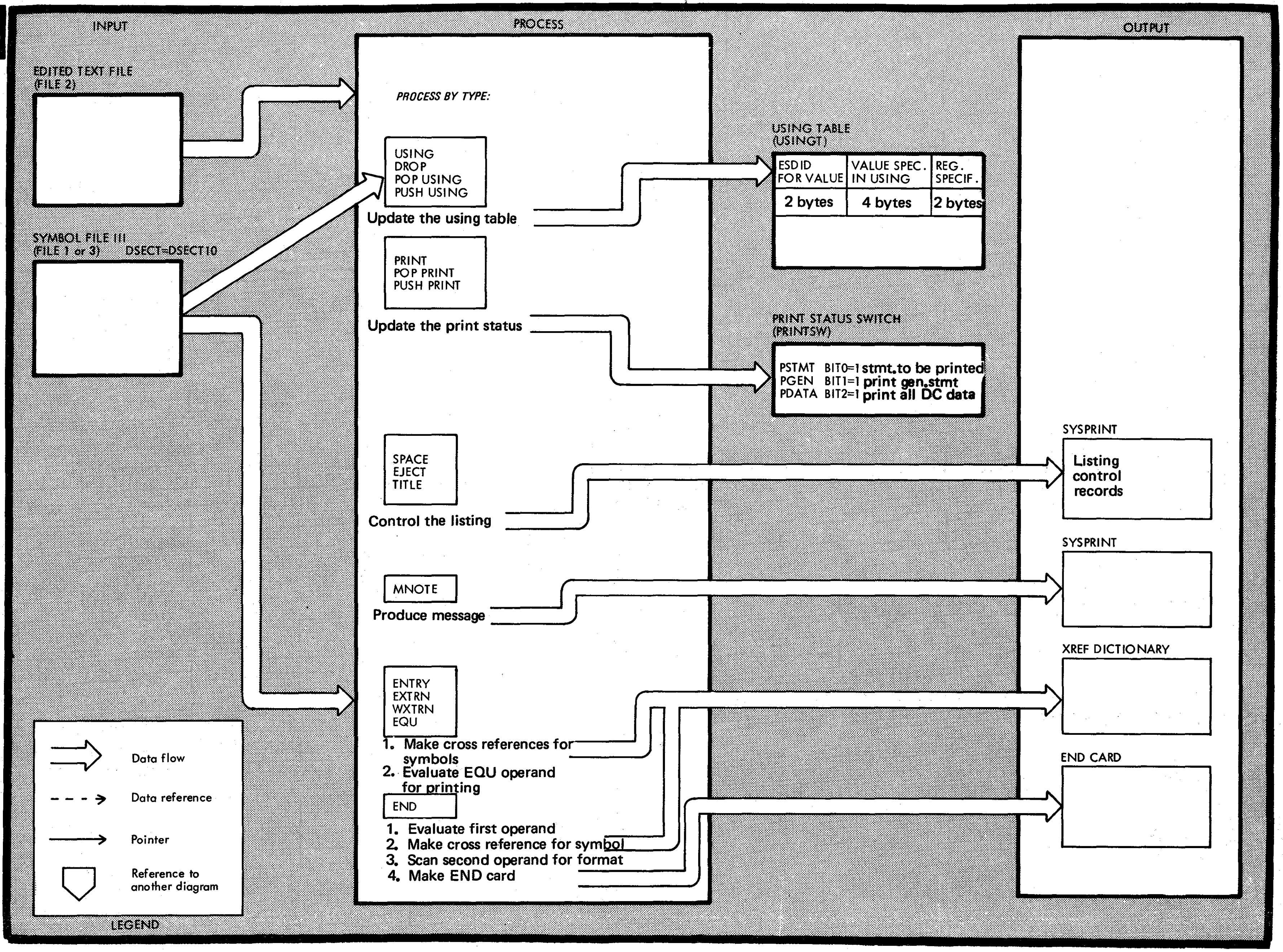
	MODULE	ROUTINE (LABEL)
The operand of a PUNCH statement and the input record following a REPRO statement is an 80-byte EBCDIC string included in the object code.	IFNX5A	REPRO0 PUNCH0

CCW

	MODULE	ROUTINE (LABEL)
The operand of a CCW instruction is evaluated and the result stored in an 8-byte object code record that is aligned to a double-word boundary. The location counter is updated accordingly.	IFNX5A	CCW100

Process Assembler Instructions

24



Process Assembler Instructions (cont.)

USING
DROP
POP USING
PUSH USING

	MODULE	ROUTINE (LABEL)
USING The operand is evaluated and the register or registers indicated are checked against the using table. The using table has one entry for each value that has been specified in a current USING statement. If the register was already in use, the earlier entry is dropped. The new entry or entries are made and the whole table sorted in descending order. The primary sort field is the ESDID, the secondary field is the value, and the tertiary sort field is the register number. If the ESDID is 0, the corresponding entry is an absolute USING.	IFNX5A	USING0
DROP The operand is scanned for registers to be dropped from the using table. If the operand is blank, all registers are dropped. Each register indicated causes a scan of the using table, and if it is found, the remaining entries are moved up in the table, writing over the dropped register(s).	IFNX5A	DROP00
PUSH USING The operand is scanned for USING (and PRINT -- see PUSH PRINT, below). If USING is found, the using table is saved in the PUSH stack for USING. A maximum of four copies of the using table can be saved. PUSH does not affect the current status of the using table.	IFNX5A	PUSH00
POP USING The operand is scanned for USING (and PRINT -- see POP PRINT, below). A USING value that has previously been saved is restored; the current value is destroyed. If the POP has not been preceded by a PUSH with a USING operand, a diagnostic message results.	IFNX5A	POP100

PRINT
POP PRINT
PUSH PRINT

	MODULE	ROUTINE (LABEL)
PRINT The current print options are saved. All print options are turned on. The print routine is called to list the PRINT statement and, on return, the print options are restored. The operand is then scanned and the print options updated accordingly.	IFNX5A	PRINT0
PUSH PRINT The print options are saved in the PUSH stack for PRINT. A maximum of four values of the print options can be saved. PUSH does not affect the current status of the PRINT options.	IFNX5A	PUSH00
POP PRINT The PRINT value that has been previously saved is restored. The current value is destroyed. If the POP has not been preceded by a PUSH PRINT, a diagnostic message is produced.	IFNX5A	POP100

TITLE
EJECT
SPACE

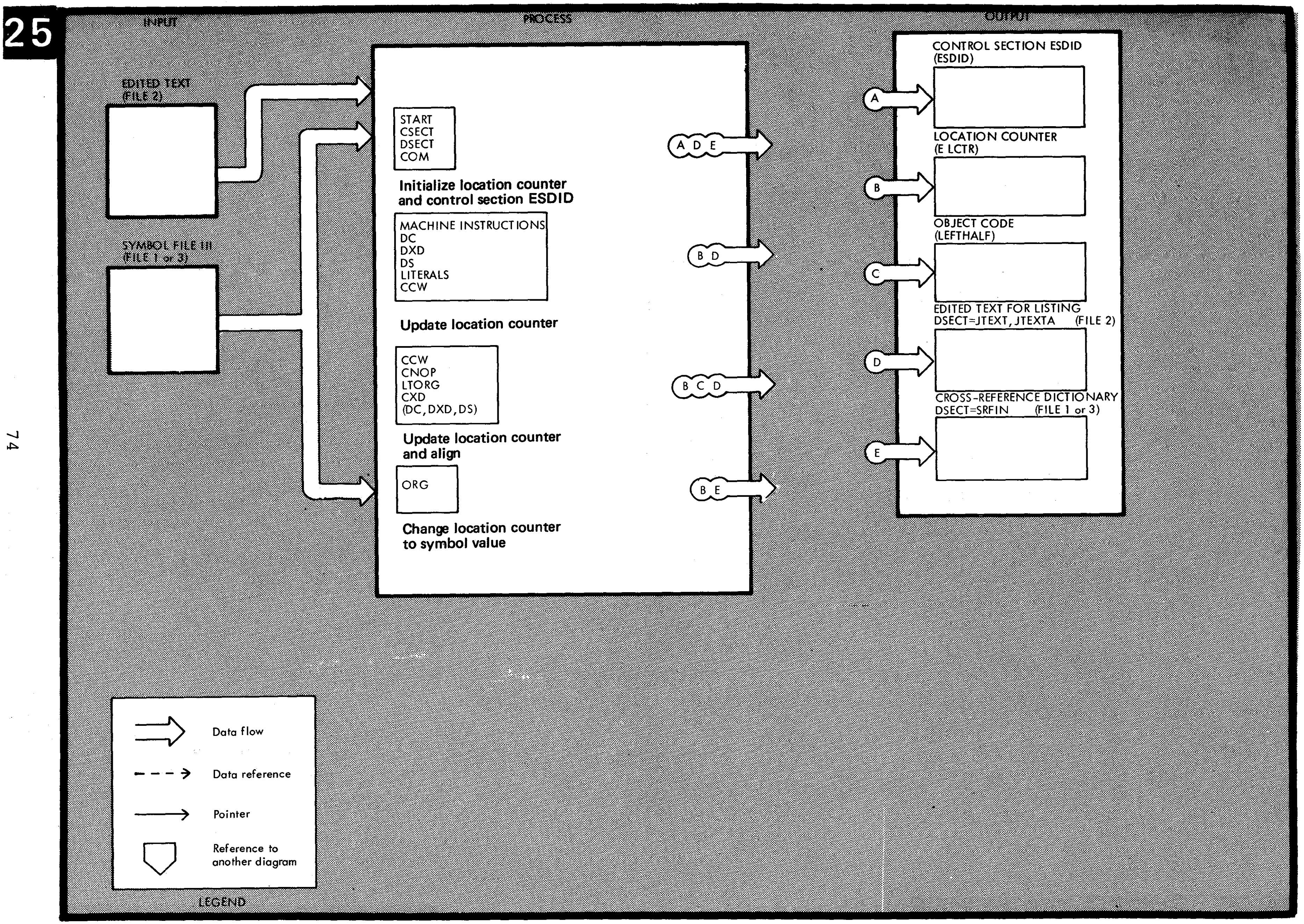
	MODULE	ROUTINE (LABEL)
TITLE The operand is scanned for duplicate ampersands and quotes (duplicates are eliminated). The title is saved, the carriage control index to the print routine is loaded into register 10, and register 11 is set to a negative number to indicate an eject. The print routine is then called.	IFNX5A	TITLE0
EJECT Register 10 is loaded with the carriage control index for the print routine. Register 11 is set to a negative number to indicate an eject and the print routine is called.	IFNX5A	EJECT0

	MODULE	ROUTINE LABEL
ENTRY EXTRN WXTRN EQU	IFNX5A	ENTRY0 EXTRN0 EQU100
These statements generate cross-reference dictionary entries. The first operand of EQU is evaluated to get a value to print.		
END		
The symbol (if any) in the operand is evaluated and the value is saved for the postprocessor. Literals (if any) cause alignment to a double word boundary; the literals are evaluated and printed after the END statement.	IFNX5A	END100

	MODULE	ROUTINE LABEL
SPACE		
The operand is scanned for a decimal value. If no operand is encountered, a value of 1 is loaded in register 11. Register 10 is loaded with the carriage control index and the print routine is called.	IFNX5A	SPACE0
MNOTE		
A message is generated. If a severity code is given, it is saved for statistics. Double quotes and ampersands are eliminated.	IFNX5A	MNOTE0

This page intentionally left blank

Update Location Counter



Update Location Counter (cont.)

	MODULE	ROUTINE (LABEL)
The location counter is updated by the following instructions:	IFNX5A	START0

START
CSECT
DSECT
COM

These instructions initialize the location counter and the control section ESDID with values from the symbol file record . Symbols are cross-referenced.

MACHINE INSTRUCTIONS
DC
DXD
DS
LITERALS
CCW

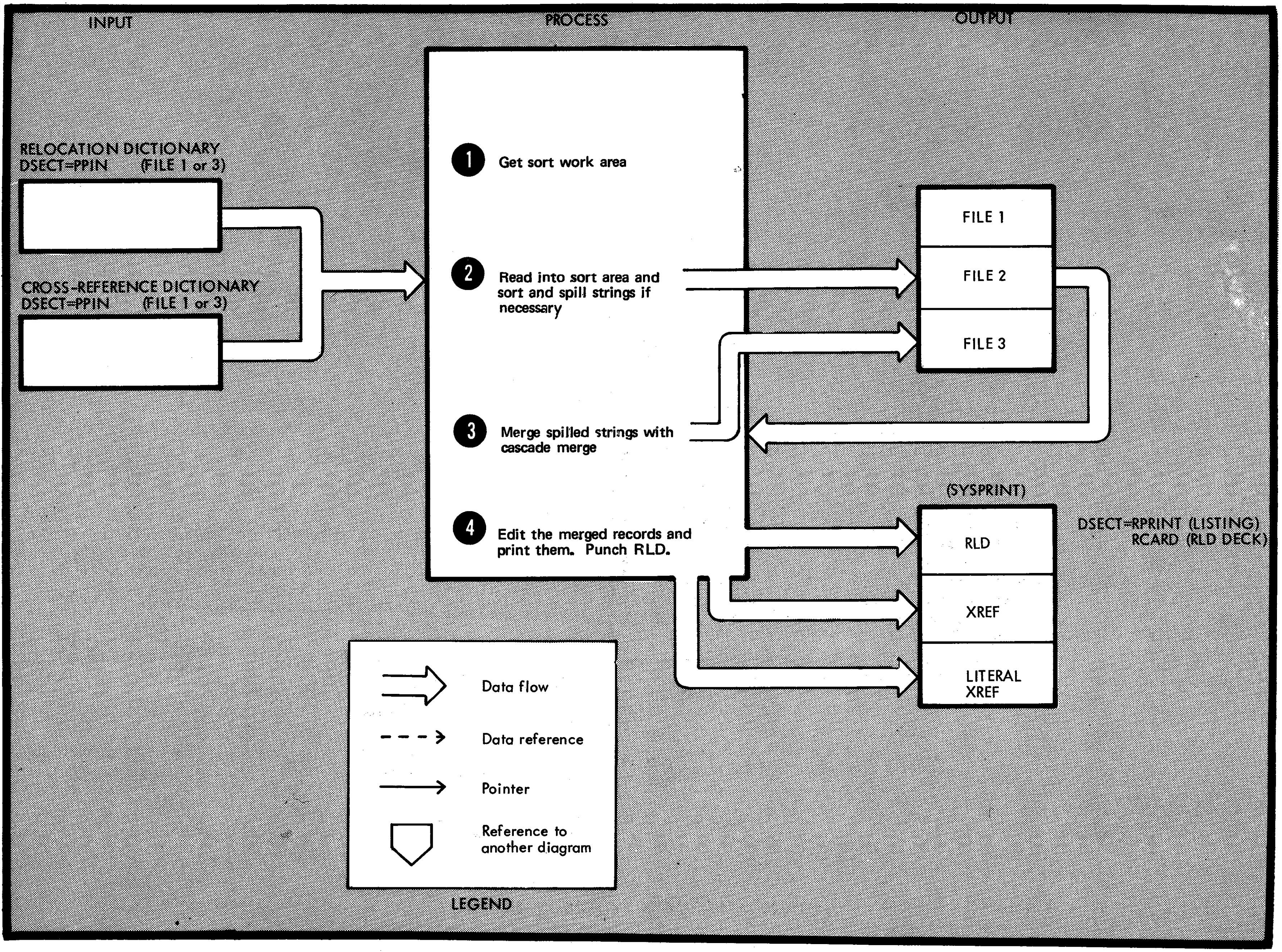
	MODULE	ROUTINE (LABEL)
After each machine instruction, and when object code is generated by other statements, the length of the generated code is added to the current location counter value. The result is saved as the "new" current location counter. If the NOALIGN option is not in effect, most instructions require alignment. Others, such as LTORG and CNOP, are specifically designed to effect alignment. Alignment consists of updating the location counter by the number of bytes needed (for example, a CXD instruction adds four bytes to the location counter. If the alignment is the result of a DC instruction, zeroes are added to the object code. A CNOP instruction fills the alignment bytes with 0700.)	IFNX5A IFNX5A	LOCUPD ALIGN

ORG

	MODULE	ROUTINE (LABEL)
The ORG instruction causes the location counter to take on the value given by the operand. The new value is taken from the symbol file record.	IFNX5A	ORG100

Sort RLD and XREF

26



1 Get sort work area

2 Read into sort area and sort and spill strings if necessary

3 Merge spilled strings with cascade merge

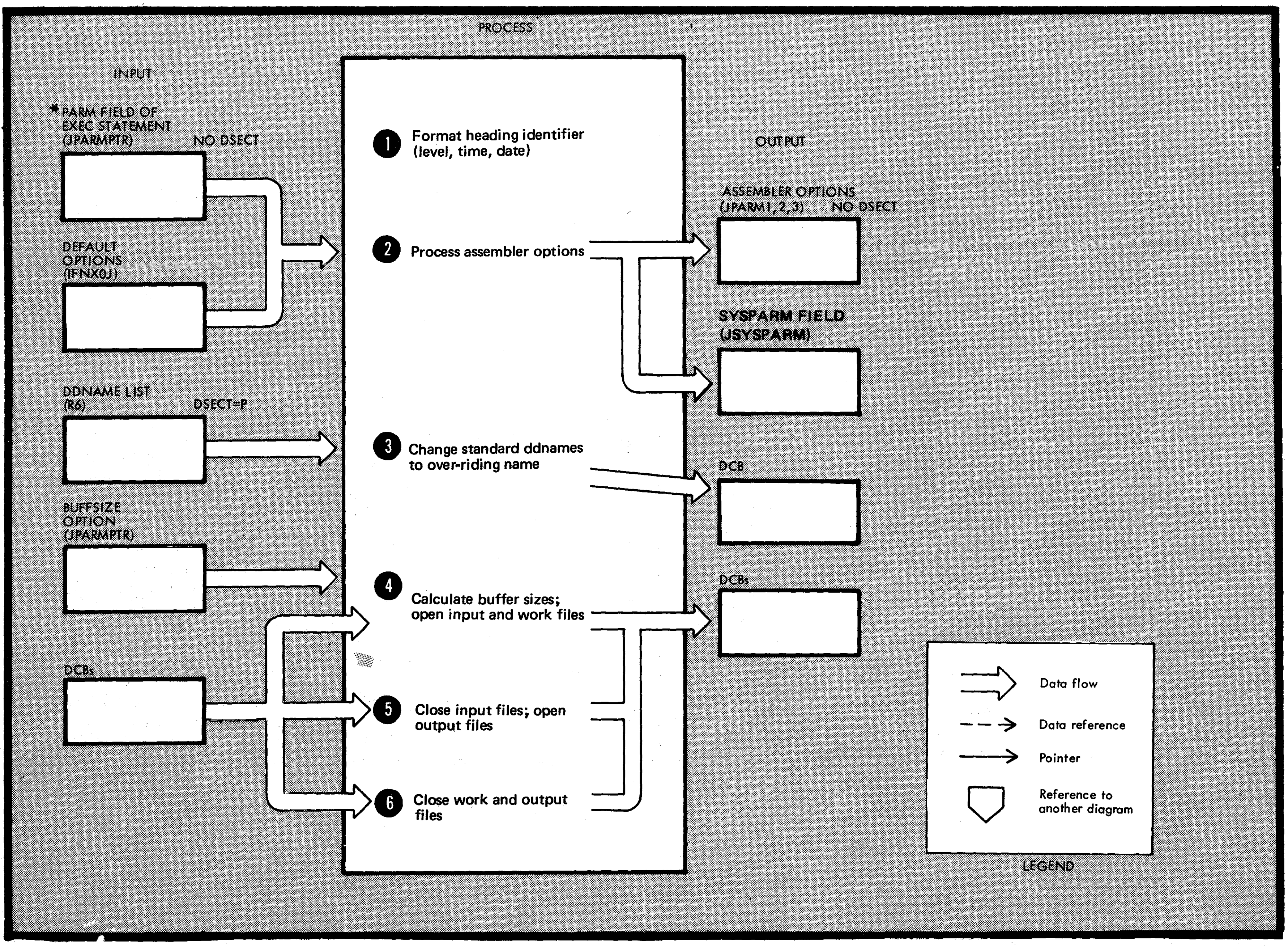
4 Edit the merged records and print them. Punch RLD.

Sort RLD and XREF (cont.)

	MODULE	ROUTINE (LABEL)
1 A GETMAIN is issued to obtain all available core. The area is divided into six buffers and a sort work area. The sort work area and buffers 5 and 6 are divided into a sort pointer area and a data area.	IFNX6A	X6AENT
2 Records are read into the sort data area and a four-byte entry for each record (its address) is made in the sort pointer area. Entries are made until the data area is filled or the input file is empty. The records are then sorted (Shell's sort) using bytes 4-17 of the record as the sort field. If there is more input it is spilled onto file 1 or 2.	IFNX6A	GTRGTR
3 A cascade merge is used to reduce the number of sorted strings. When two files contain one string each and the third file is empty, a final merge is done.	IFNX6A	MERGE
4 During the final merge the records are edited and put out. The records have been sorted in the order RLD, symbol XREF, and literal XREF. RLD records are simply formatted and printed (and punched). XREF symbol definition records have the symbol, its length, value, and definition fields fully inserted. A statement number is added to each reference record. If a reference record appears without being preceded by a definition, the symbol is marked "undefined" and the undefined text is inserted. If a record appears with the duplicate flag, a line with the message ** DUPLICATE ** is inserted. Literal XREF records are handled in the same way.	IFNX6A	OUTPUTS

Initialize

27


PROCESS
INPUT
*PARM FIELD OF EXEC STATEMENT (JPARMPTR) NO DSECT
DEFAULT OPTIONS (IFNX0J)
DDNAME LIST (R6) DSECT=P
BUFFSIZE OPTION (JPARMPTR)
DCBs
1 Format heading identifier (level, time, date)
2 Process assembler options
3 Change standard ddnames to over-riding name
4 Calculate buffer sizes; open input and work files
5 Close input files; open output files
6 Close work and output files
OUTPUT
ASSEMBLER OPTIONS (JPARM1,2,3) NO DSECT
SYSPARM FIELD (JSYSPARM)
DCB
DCBs
Data flow
Data reference
Pointer
Reference to another diagram
LEGEND

* Input, processing, and output of the initialization routine is essentially the same for VM/370 as for OS/VS. OS/VS requires the use of OS JCL EXEC statements for the passing of parameters; parameters are passed by means of the ASSEMBLE command in VM/370.

Initialize (cont.)

		MODULE
1	The time and date are obtained with a TIME macro. The level is contained in IFOX0A.	IFOX0D
2	Assembler options are obtained from the PARM field of the EXEC statement and from the default options.	IFOX0D
3	When the assembler has been invoked from another program, there may be overriding DDnames. Relevant DCBs are changed to correspond to the new names.	IFOX0D
4	The buffer sizes for workfiles are calculated. If no BUFSIZE option has been given, 37% of the region is allocated to buffers and 63% to generation-time dictionaries. If the BUFSIZE(MIN) option has been specified, each utility data set is allocated a single 790-byte buffer and the remaining storage allocated to dictionaries.	IFOX0A

		MODULE
	SYSIN and SYSLIB are opened.	IFOX0F
	SYSUT1, SYSUT2 and SYSUT3 are opened	IFOX0A
5	After all input has been read and processed, the input files are closed and the output files (SYSPUNCH, SYSGO, and SYSPUNCH) opened.	IFOX0F IFOX0H
6	Finally, all files are closed.	IFOX0A IFOX0H

This page intentionally left blank

Program Organization

This section describes how the program is divided into units. It contains detailed charts of how the assembler phases use main storage and diagrams showing the flow of data and control between assembler phases.

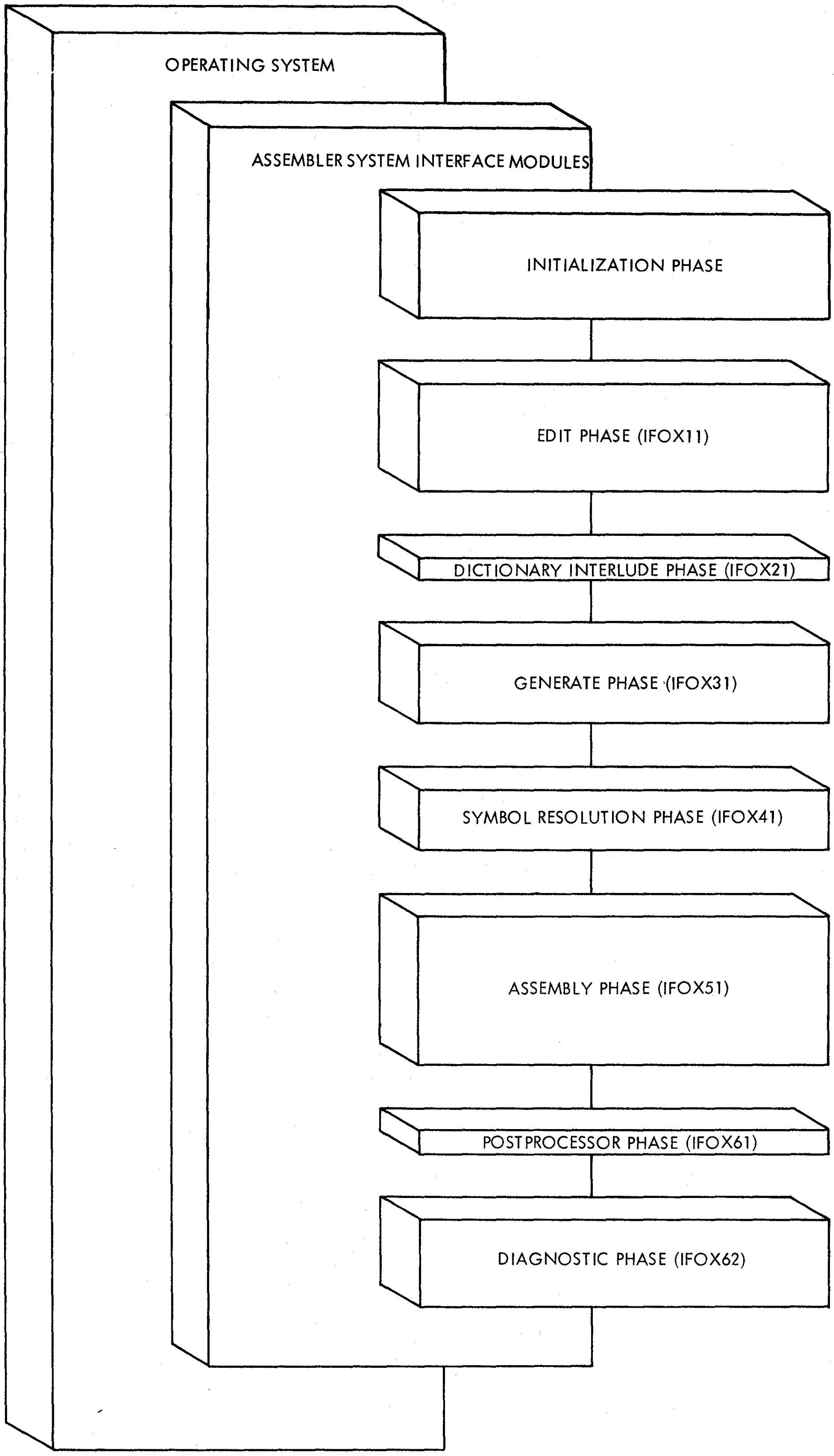

OPERATING SYSTEM
ASSEMBLER SYSTEM INTERFACE MODULES
INITIALIZATION PHASE
EDIT PHASE (IFOX11)
DICTIONARY INTERLUDE PHASE (IFOX21)
GENERATE PHASE (IFOX31)
SYMBOL RESOLUTION PHASE (IFOX41)
ASSEMBLY PHASE (IFOX51)
POSTPROCESSOR PHASE (IFOX61)
DIAGNOSTIC PHASE (IFOX62)

Figure 1. Logical Flow of Control

LOAD MODULE NAME	CSECT	OBJECT MODULE	OBJECT MODULE DESCRIPTION
ASSEMBLE	DMSASM*	DMSASM	VM/370 (CMS) Initialization Procedures for the System Assembler
	DMSASD*	DMSASM	VM/370 Auxiliary Directory for the System Assembler
IFOX00	IFOX0A00	IFOX0A	Driver Routines
	IFOX0B00	IFOX0B	Workfile I/O And Storage Management Routines
IFOX01	IFOX0C00	IFOX0C	Master Common Area
IFOX02	IFOX0D00	IFOX0D	Master Common Area Initialization Routines
	IFOX0J00	IFOX0J	Assembler Option Parameters
IFOX03	IFOX0E00	IFOX0E	Input DCB's And Module X0F Work Areas
IFOX04	IFOX0F00	IFOX0F	Input Routines
IFOX05	IFOX0G00	IFOX0G	Output DCB's And Module X0H Work Areas
IFOX06	IFOX0H00	IFOX0H	Output Routines
IFOX07	IFOX0I00	IFOX0I	Abort Routines
IFOX11	IFNX1A00 IFNX1A10 IFNX1A20 IFNX1A30	IFNX1A	Edit Phase Mainline Logic
	IFNX1KUN	IFNX1K	Edit Phase Operation Code Table
	IFNX1J00	IFNX1J	Edit Phase Dictionary Routines
	IFNX1S00	IFNX1S	Edit Phase Post-fix Routines
IFOX21	IFNX2A00 IFNX2A02	IFNX2A	Dictionary Interlude Phase
IFOX31	IFNX3A00 IFNX3A03	IFNX3A	Generate Phase Mainline Logic
	IFNX3B00	IFNX3B	Generate Phase Symbol Resolution Preprocessor
	IFNX3KUN	IFNX3K	Generate Phase Operation Code Table
	IFNX3N00	IFNX3N	Generate Phase Dictionary Routines
IFOX41	IFNX4D00	IFNX4D	Symbol Resolution Phase DS/DC Evaluation Routines
	IFNX4E00	IFNX4E	Symbol Resolution Phase ESD Routines
	IFNX4M00	IFNX4M	Symbol Resolution Phase Mainline Logic
	IFNX4S00	IFNX4S	Symbol Resolution Phase Symbol Table Routine
	IFNX4V00	IFNX4V	Symbol Resolution Phase Expression Evaluation
IFOX42	IFNX4N00	IFNX4N	Symbol Resolution Phase DS/DC Evaluation Routines (Test Option Specified)
	IFNX4E00	IFNX4E	Symbol Resolution Phase ESD Routines (Test Option Specified)
	IFNX4T00	IFNX4T	Symbol Resolution Phase Mainline Logic (Test Option Specified)
	IFNX4S00	IFNX4S	Symbol Resolution Phase Symbol Table Routine Test Option Specified)
	IFNX4V00	IFNX4V	Symbol Resolution Phase Expression Evaluation (Test Option Specified)

*DMSASM and DMSASD are VM/370 modules used to interface with the CMS component of VM/370.

Module Directory. This chart shows how the assembler is divided into program units, and how these program units are subdivided. The make up of each load module is shown in terms of the objects modules and CSECTS that comprise it. Furthermore, the module directory contains a description of each object module. For further and more detailed information see the Directory.

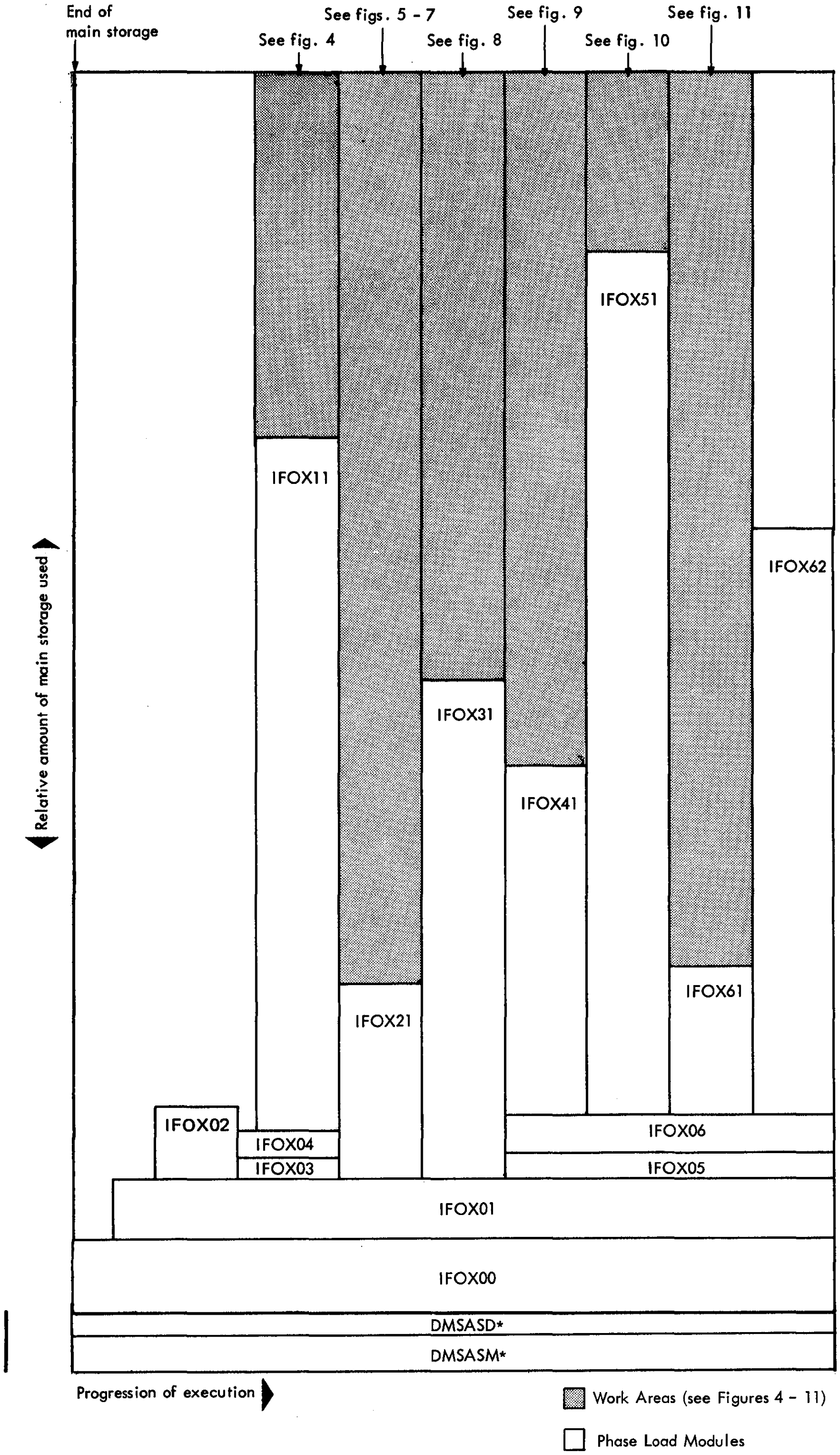
Figure 2. Module Directory (Part 1 of 2)

LOAD MODULE NAME	CSECT	OBJECT MODULE	ÓBJECT MODULE DESCRIPTION
IFOX51	IFNX5A00 IFNX5A20 IFNX5A30 IFNX5A40 IFNX5A50	IFNX5A	Assembly Phase Operation Code Processor
	IFNX5C00	IFNX5C	Assembly Phase Mainline Logic
	IFNX5D00	IFNX5D	Assembly Phase Constant Processor
	IFNX5F00	IFNX5F	Assembly Phase Fixed Point/Floating Point Conversion
	IFNX5L00	IFNX5L	Assembly Phase Error Logging Routine
	IFNX5M00	IFNX5M	Assembly Phase Machine op Processor
	IFNX5P00	IFNX5P	Assembly Phase Print Routine
	IFNX5V00	IFNX5V	Assembly Phase Expression Evaluation Routine
IFOX61	IFNX6A00	IFNX6A	Post Processor Phase
IFOX62	IFNX6B00 IFNX6B20	IFNX6B	Diagnostic Phase
	IFNX6C00	IFNX6C	Error Messages

Module Directory. This chart shows how the assembler is divided into program units, and how these program units are subdivided. The make up of each load module is shown in terms of the objects modules and CSECTS that comprise it. Furthermore, the module directory contains a description of each object module. For further and more detailed information see the Directory.

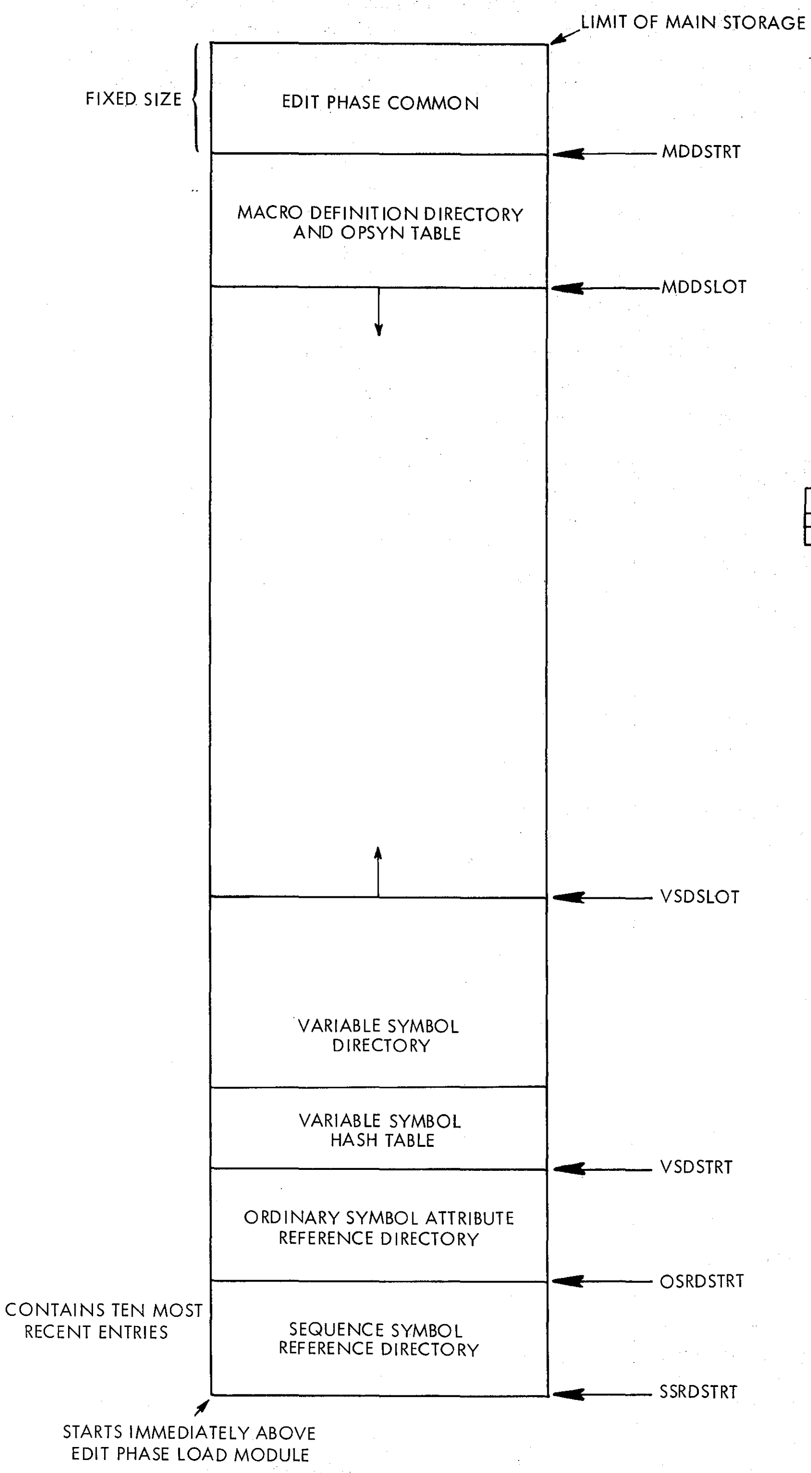
Figure 2. Module Directory (Part 2 of 2)

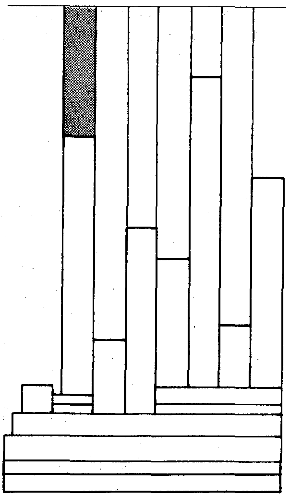
Main storage layout of the assembler. The vertical axis of this diagram represents the relative amount of main storage, and the heights of the bars representing the assembler phase load modules show the relative sizes of the different phases. The horizontal axis represents the progression of execution time, and therefore, at any point the diagram shows which load modules are in main storage. For example, when the Dictionary Interlude Phase (IFOX21) executes, with it in main storage are the Master Common Area (IFOX01) and the Driver Routines, Workfile I/O, and Storage Management Routines (IFOX00).



*These modules are used only with VM/370.

Figure 3. Main Storage Layout


LIMIT OF MAIN STORAGE
FIXED SIZE
EDIT PHASE COMMON
MDDSTRT
MACRO DEFINITION DIRECTORY AND OPSYN TABLE
MDDSLOT
VSDSLOT
VARIABLE SYMBOL DIRECTORY
VARIABLE SYMBOL HASH TABLE
VSDSTRT
ORDINARY SYMBOL ATTRIBUTE REFERENCE DIRECTORY
OSRDSTRT
CONTAINS TEN MOST RECENT ENTRIES
SEQUENCE SYMBOL REFERENCE DIRECTORY
SSRDSTRT
STARTS IMMEDIATELY ABOVE EDIT PHASE LOAD MODULE



Symbols are in the phase common

Figure 4. Edit Phase (IFOX11) Main Storage Work Area

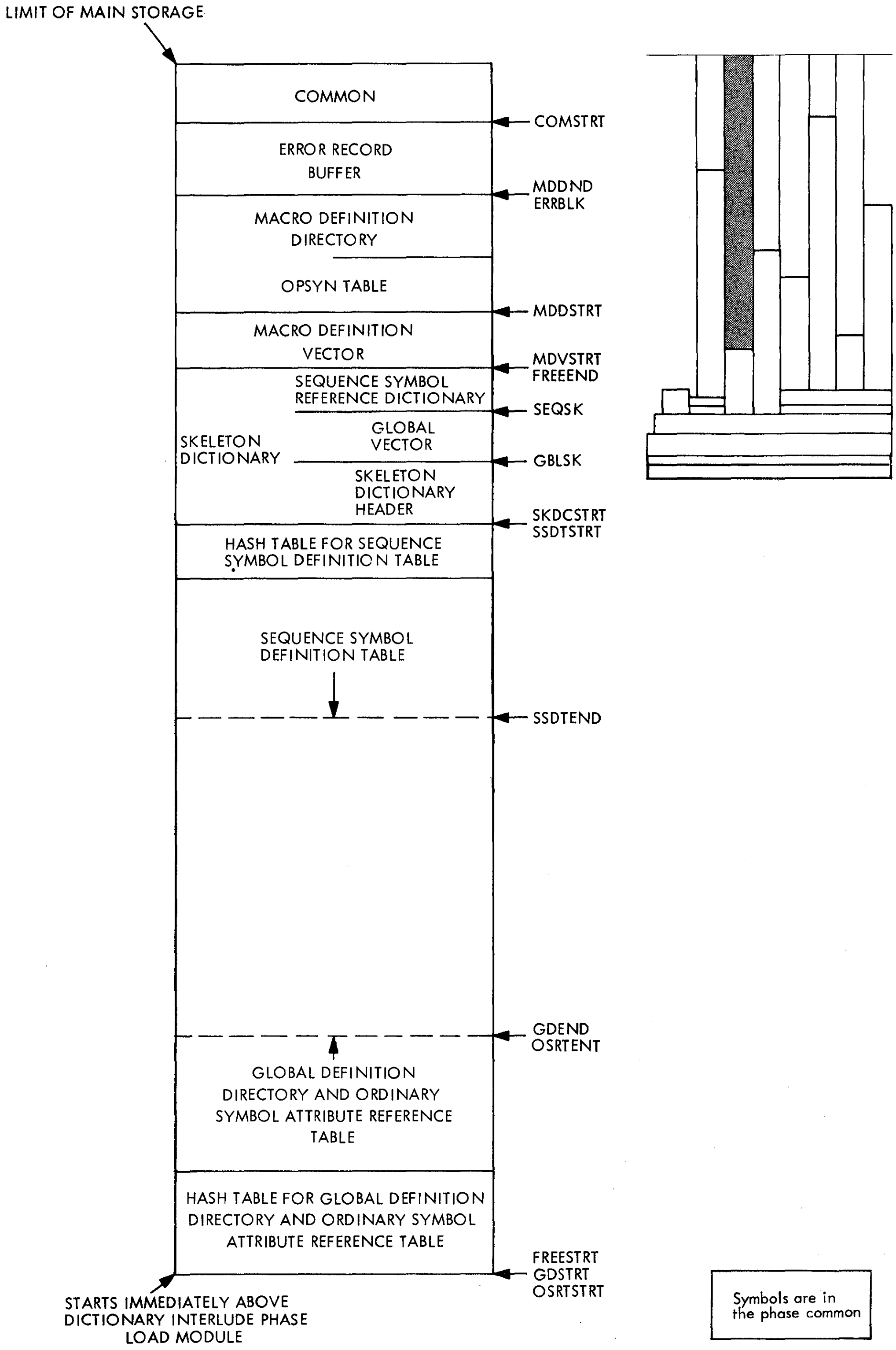

LIMIT OF MAIN STORAGE
COMMON
COMSTRT
ERROR RECORD BUFFER
MDDND
ERRBLK
MACRO DEFINITION DIRECTORY
OPSYN TABLE
MDDSTRT
MACRO DEFINITION VECTOR
MDVSTRT
FREEEND
SEQUENCE SYMBOL REFERENCE DICTIONARY
SEQSK
SKELETON DICTIONARY
GLOBAL VECTOR
GBLSK
SKELETON DICTIONARY HEADER
SKDCSTRT
SSDTSTRT
HASH TABLE FOR SEQUENCE SYMBOL DEFINITION TABLE
SEQUENCE SYMBOL DEFINITION TABLE
SSDTEND
GDEND
OSRTENT
GLOBAL DEFINITION DIRECTORY AND ORDINARY SYMBOL ATTRIBUTE REFERENCE TABLE
HASH TABLE FOR GLOBAL DEFINITION DIRECTORY AND ORDINARY SYMBOL ATTRIBUTE REFERENCE TABLE
FREESTRT
GDSTRT
OSRTSTRT
STARTS IMMEDIATELY ABOVE DICTIONARY INTERLUDE PHASE LOAD MODULE
Symbols are in the phase common

Figure 5. Dictionary Interlude Phase (IFOX21) Main Storage Work Area: 1 of 3 Process Skeleton Dictionaries

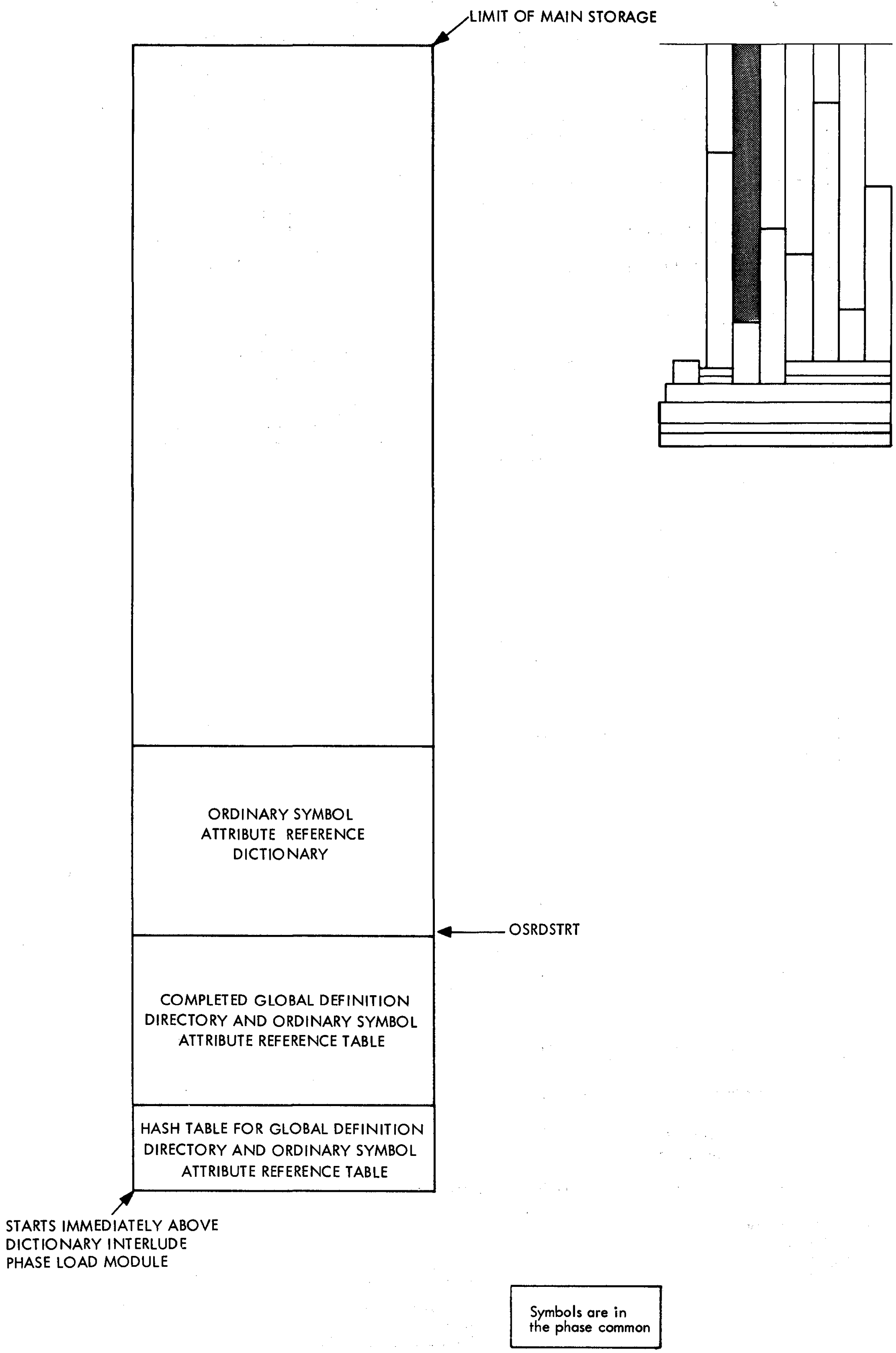
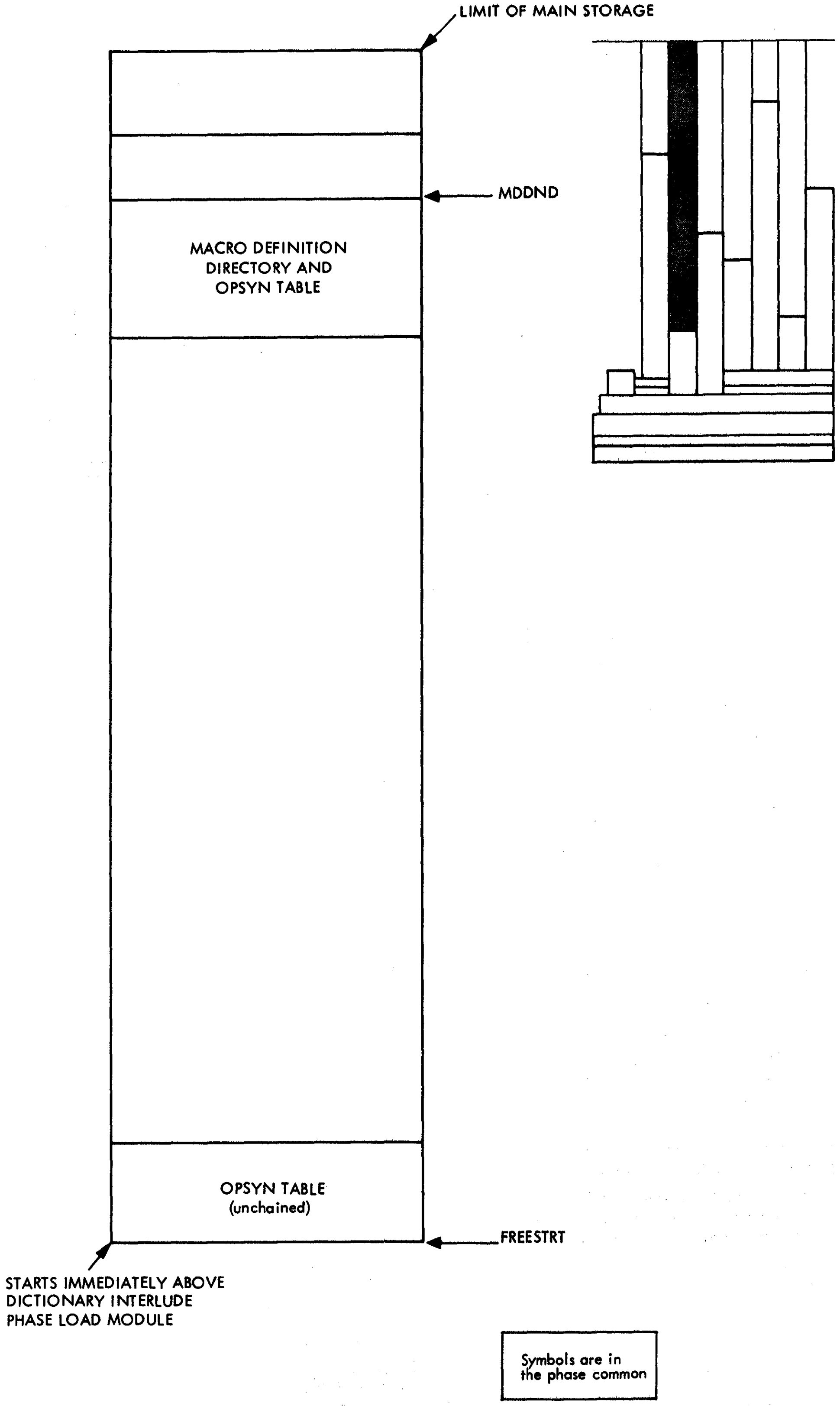

LIMIT OF MAIN STORAGE
ORDINARY SYMBOL ATTRIBUTE REFERENCE DICTIONARY
OSRDSTRT
COMPLETED GLOBAL DEFINITION DIRECTORY AND ORDINARY SYMBOL ATTRIBUTE REFERENCE TABLE
HASH TABLE FOR GLOBAL DEFINITION DIRECTORY AND ORDINARY SYMBOL ATTRIBUTE REFERENCE TABLE
STARTS IMMEDIATELY ABOVE DICTIONARY INTERLUDE PHASE LOAD MODULE
Symbols are in the phase common

Figure 6. Dictionary Interlude Phase (IFOX21)
Main Storage Work Area: 2 of 3
Build Ordinary Symbol Attribute Reference Dictionary


LIMIT OF MAIN STORAGE
MDDND
MACRO DEFINITION DIRECTORY AND OPSYN TABLE
OPSYN TABLE (unchained)
FREESTRT
STARTS IMMEDIATELY ABOVE DICTIONARY INTERLUDE PHASE LOAD MODULE

Symbols are in the phase common

Figure 7. Dictionary Interlude Phase (IFOX21) Main Storage Work Area: 3 of 3 Unchain Opsyn Table

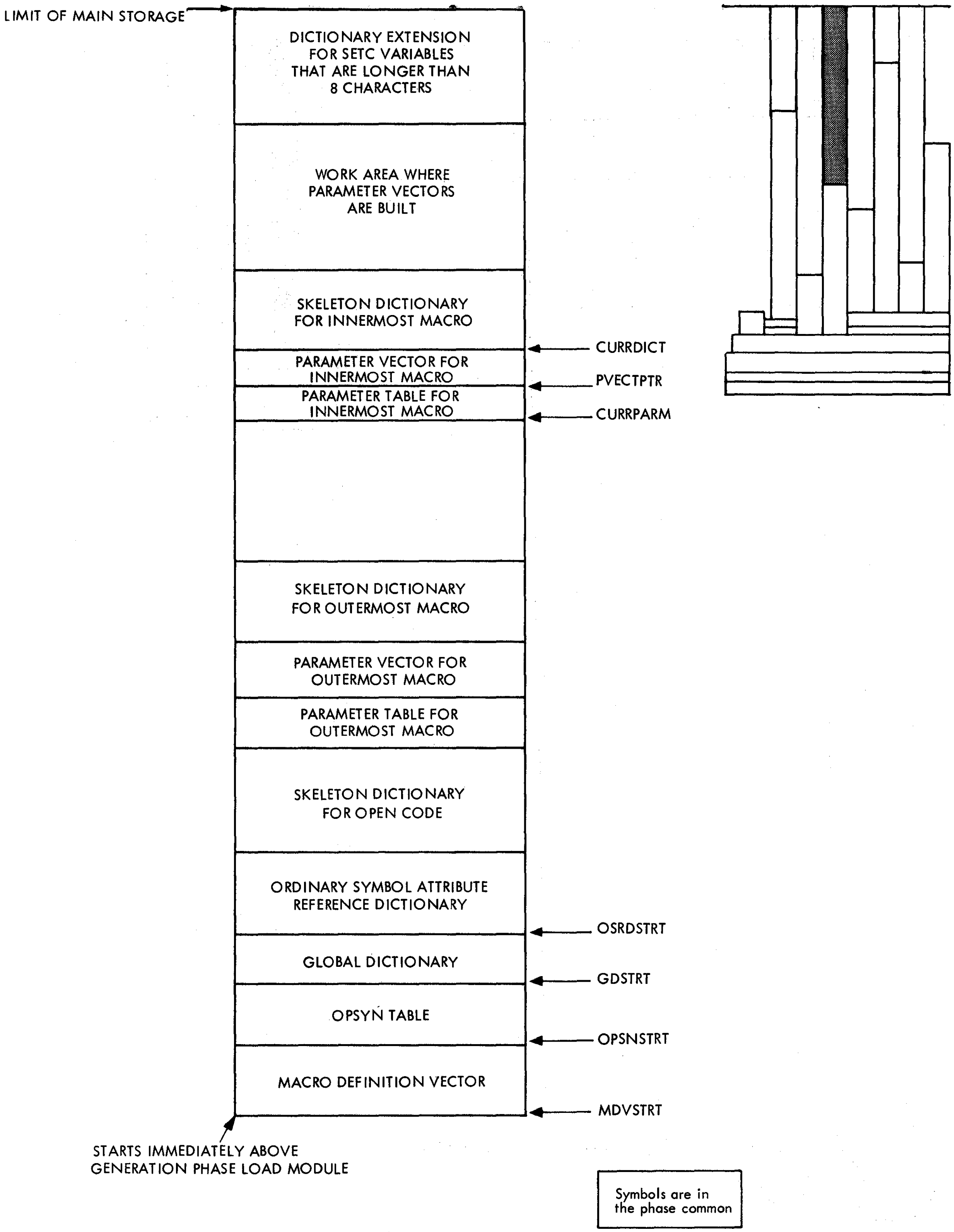
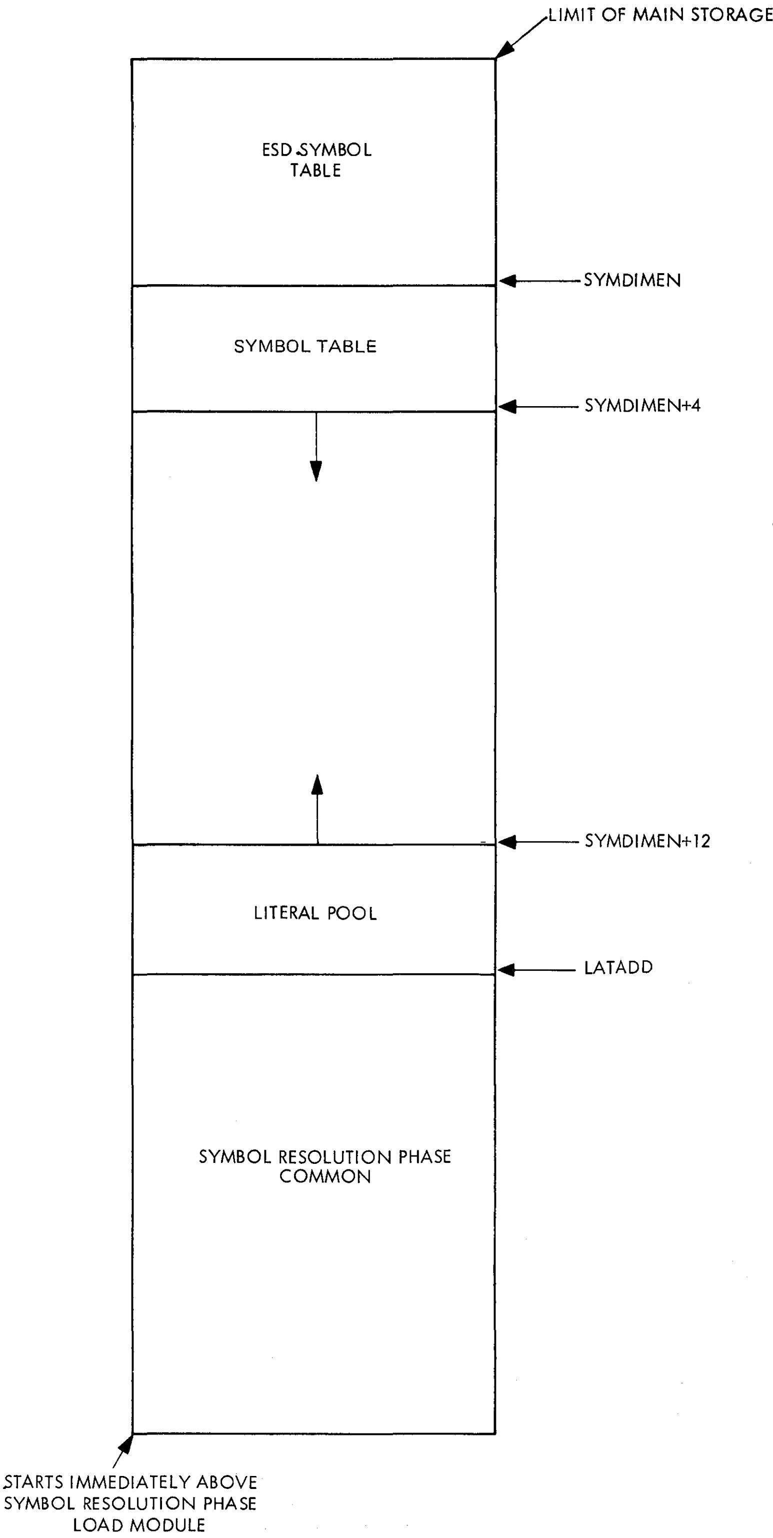
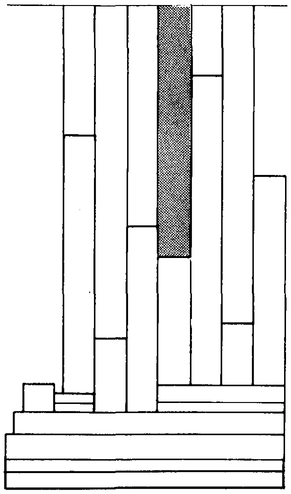

LIMIT OF MAIN STORAGE
DICTIONARY EXTENSION FOR SETC VARIABLES THAT ARE LONGER THAN 8 CHARACTERS
WORK AREA WHERE PARAMETER VECTORS ARE BUILT
SKELETON DICTIONARY FOR INNERMOST MACRO
CURRDICT
PARAMETER VECTOR FOR INNERMOST MACRO
PVECTPTR
PARAMETER TABLE FOR INNERMOST MACRO
CURRPARM
SKELETON DICTIONARY FOR OUTERMOST MACRO
PARAMETER VECTOR FOR OUTERMOST MACRO
PARAMETER TABLE FOR OUTERMOST MACRO
SKELETON DICTIONARY FOR OPEN CODE
ORDINARY SYMBOL ATTRIBUTE REFERENCE DICTIONARY
OSRDSTRT
GLOBAL DICTIONARY
GDSTRT
OPSYN TABLE
OPSNSTRT
MACRO DEFINITION VECTOR
MDVSTRT
STARTS IMMEDIATELY ABOVE GENERATION PHASE LOAD MODULE
Symbols are in the phase common

Figure 8. Generation Phase (IFOX31) Main Storage Work Area





Symbols are in the phase common

Figure 9. Symbol Resolution Phase (IFOX41) Main Storage Work Area

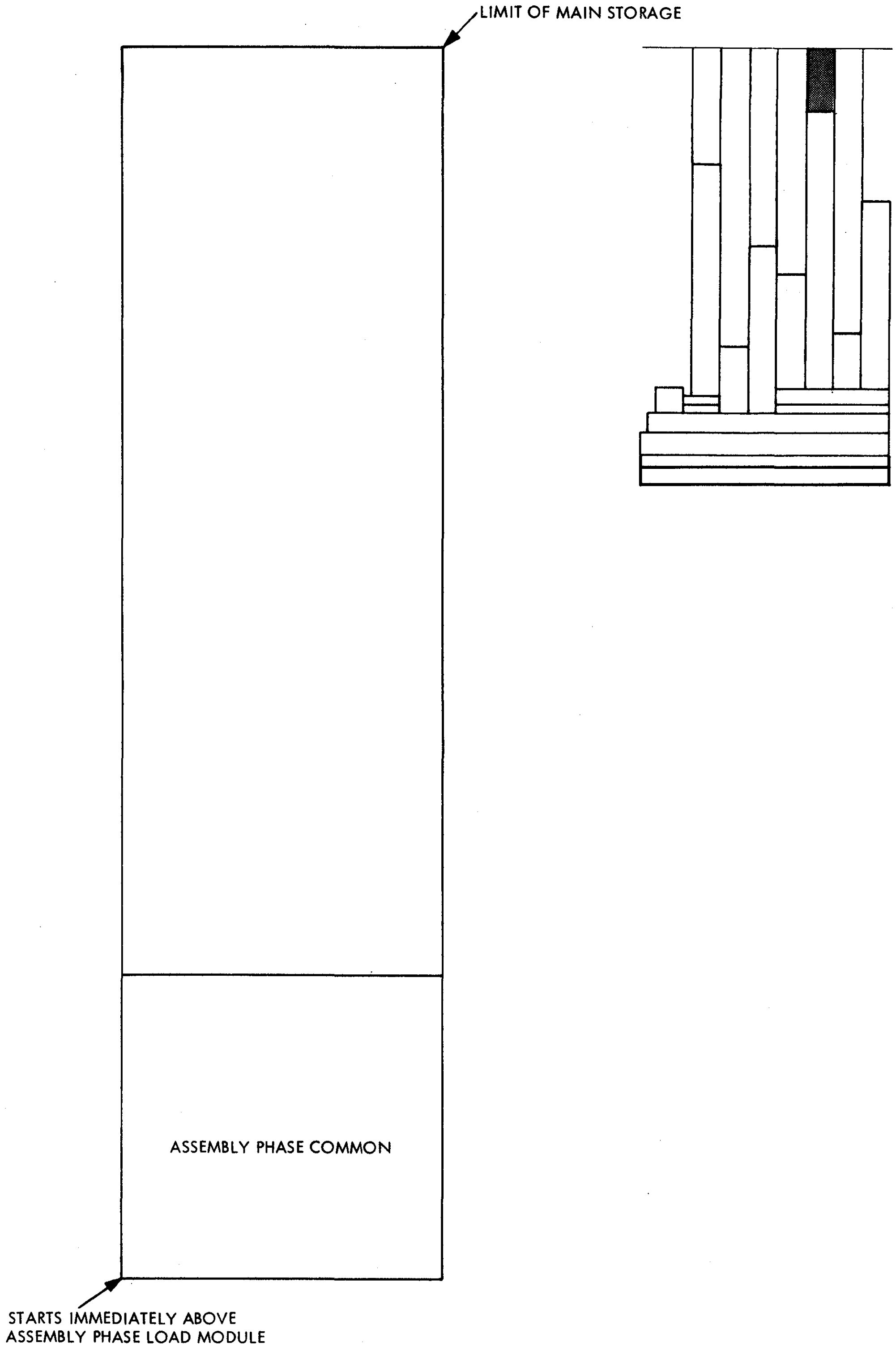
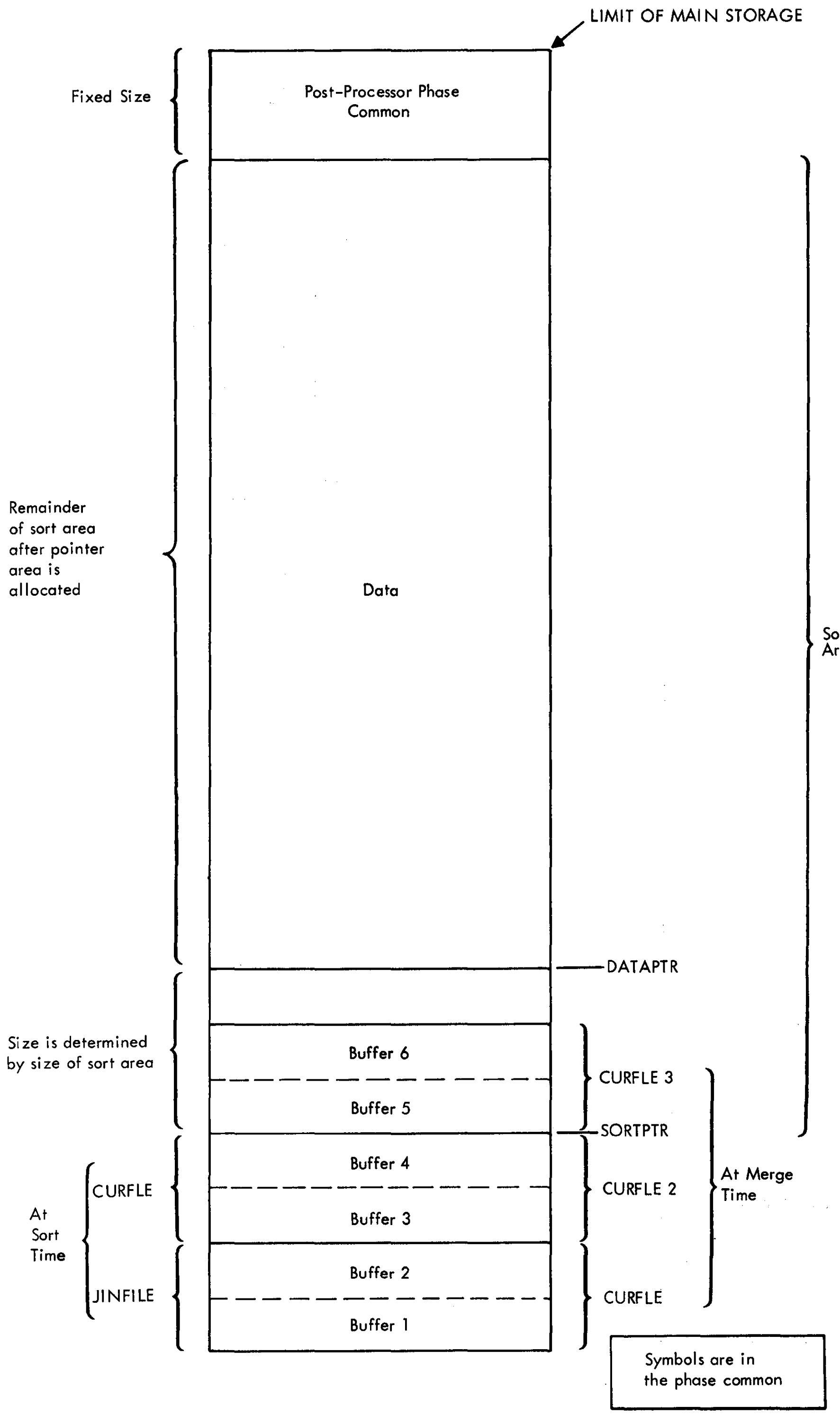


Figure 10. Assembly Phase (IFOX51)
Main Storage Work Area



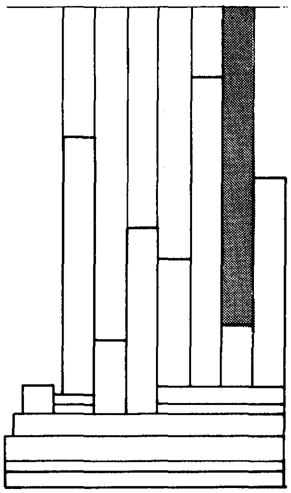


Figure 11. Post Processor Phase (IFOX61) Main Storage Work Area

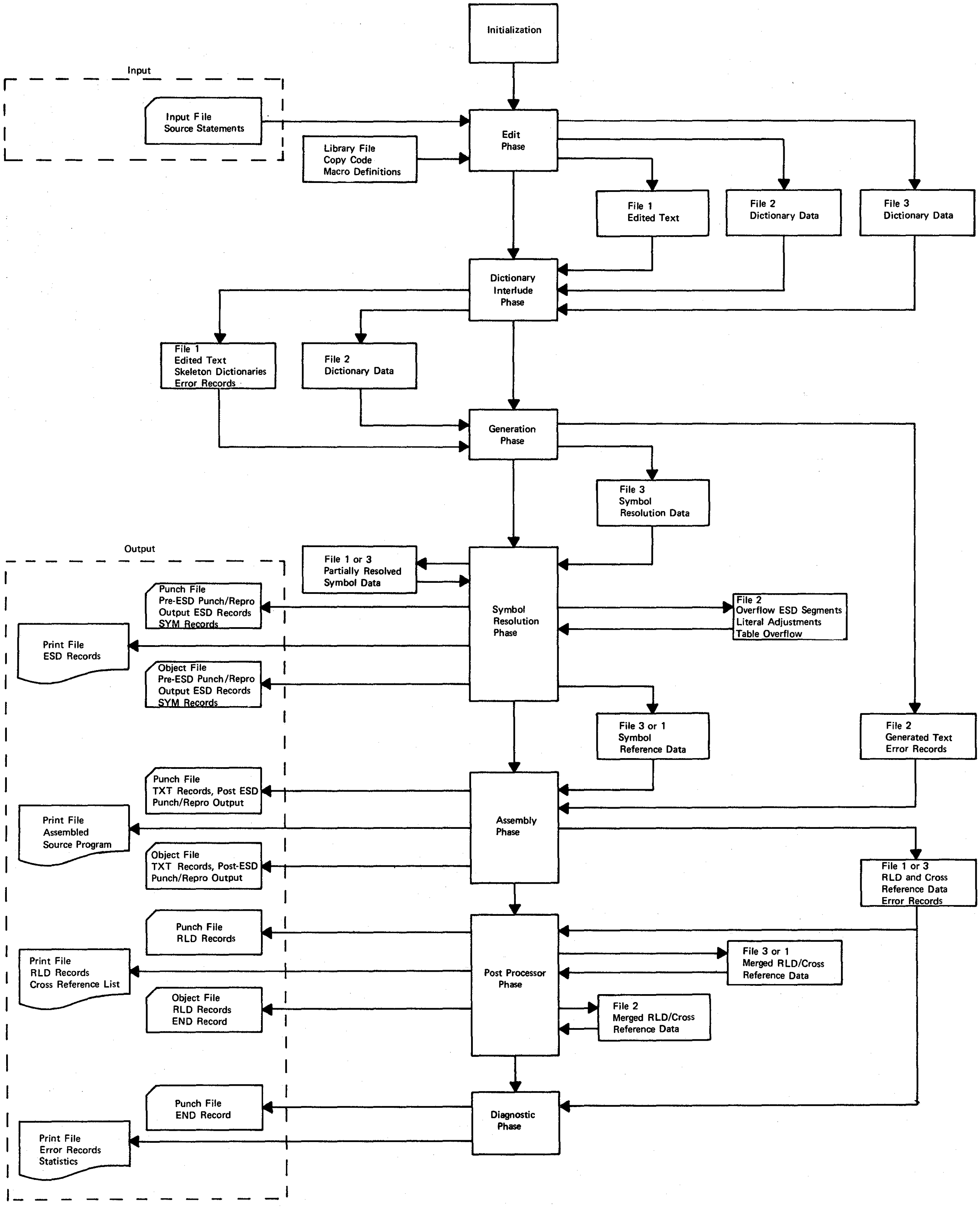


Figure 12. Assembler Data Flow

Data Areas

This section contains detailed layouts of data areas to help in interpreting storage dumps.

DSECT NAME: **EDSECT**

LOAD MODULE: IFOX11

SIZE: 1124

CREATED BY: IFNX1A

REFERENCED BY: IFNX1A,IFNX1J,IFNX1S

UPDATED BY: IFNX1A,IFNX1J,IFNX1S

FUNCTION: EDITOR COMMON

OPERATIONS DIAGRAMS:

DISPLMNT			FIELD	DESCRIPTION: CONTENTS,
DEC	(HEX)	SIZE	NAME	MEANING/USE
0	(0000)	1	SWITCH1	PROGRAM SWITCH
			SMDEF	BIT 0 - WITHIN MAC DEF (SET BY MACRO)
			SXPRTO	BIT 1 - PROTO EXPECTED (SET BY MACRO)
			SMISCN	BIT 2 - RETURN TO MISCAN
			SNOPSYN	BIT 3 - OPSYN NO LONGER ALLOWED
1	(0001)	1	SWITCH2	PROGRAM SWITCH
			SONECD	BIT 0 - READ ONE CARD (REPRO)
			SBYCNT	BIT 1 - BYPASS ALL CONTINUATIONS
			SONECT	BIT 2 - READ ONE CONTINUATION
			SALLCT	BIT 3 - READ ALL CONTINUATIONS
			SBYONE	BIT 4 - BYPASS ONE CARD IN EDITED FORM
			SCTLRTN	BIT 6 - RETURN TO CALLER
			SNOPND	BIT 7 - RETURN TO CALLER
2	(0002)	1	SWITCH3	PROGRAM SWITCH
			SCMTCT	BIT 0 - COMMENTS CONTINUED
			SNXTCT	BIT 1 - NEXT CD CNT'N OF THIS CD
			SPRVCT	BIT 2 - THIS CD CNT'N OF PREVIOUS CD
			SLSTCD	BIT 3 - LAST CARD
			SINEOF	BIT 4 - EOF ON SYSTEM INPUT
			SGBLCL	BIT 5 - PROC'G GBLX, LCLX STMT
			SMI	BIT 6 - EDITING MACRO INSTRUCTION
			SUBSOP	BIT 7 - SUBSTITUTED OP CODE FOUND
3	(0003)	1	SWITCH4	PROGRAM SWITCH
			SPGRMD	BIT 0 - PROCESSING PROGRAMMER MACRO
			SOPNCD	BIT 1 - IN OPEN CODE
			SSYSMD	BIT 2 - IN SYSTEM MACRO DEFINITION
			SICTL	BIT 3 - ICTL PROCESSED IN THIS RUN
			SNOACTR	BIT 4 -
			SABORT	BIT 5 -
			SKPMND	BIT 6 - SKIP TO MEND
			SKPEND	BIT 7 - SKIP TO END
4	(0004)	1	SWITCH5	PROGRAM SWITCH
			SCOPY	BIT 0 - COPY STATEMENT

DISPLMNT			FIELD	DESCRIPTION: CONTENTS,
DEC	(HEX)	SIZE	NAME	MEANING/USE
			SXMCRO	BIT 1 - EXPECT MACRO (EDITING MD'S)
			SFSTCD	BIT 2 - READ FIRST CARD
			SDINIT	BIT 3 - PREPARE TO INIT./CLOSE D'S
			SDENT	BIT 4 - PREPARE TO MAKE D ENTRY
			SUPDNT	BIT 5 - SUPPRESS DIRECTORY ENTRY
			SMDDENTR	BIT 6 - MDD ENTRY MADE FOR THIS MACRO
5	(0005)	1	SWITCH6	PROGRAM SWITCH
			SUBLST	BIT 0 - PROCESSING SUBLIST
			POSSUBL	BIT 1 - FIRST SCAN OF SUBLIST CANDIDATE
			SCNCAT	BIT 2 - CONCATENATION IN OPERAND
			SKWPRM	BIT 3 - PROCESSING KEYWORD PARAMETER
			PROTOCAL	BIT 4 - EDITING PROTO/MACRO CALL
			SKPNAME	BIT 5 - SKIP TO OP CODE FIELD
			SPRMER	BIT 6 - PARAMETER ERROR
			SENDST	BIT 7 - END STATEMENT ENCOUNTERED
6	(0006)	1	SWITCH7	PROGRAM SWITCH
			SNMFND	BIT 0 - NAME FOUND
			SNOFND	BIT 1 - FIELD NOT FOUND
			SNOSMCRO	BIT 2 - NO MACRO STMT IN SYS MAC DEF
			SBDPROTO	BIT 3 - BAD PROTOTYPE STATEMENT
			SNOSYSMD	BIT 4 - SYSTEM MAC DEF NOT FOUND
			SDTCMT	BIT 5 - .* TYPE COMMENTS
			SASTCMT	BIT 6 - * TYPE COMMENT
			STRCMT	BIT 7 - * TYPE COMMENT
7	(0007)	1	SWITCH8	PROGRAM SWITCH
			SENAME	BIT 5 - PRESENTLY EDITING NAME FIELD
			SEOPCD	BIT 6 - PRESENTLY EDITING OP CODE FIELD
			SEOPND	BIT 7 - PRESENTLY EDITING OPERAND FIELD
8	(0008)	1	SWITCH9	PROGRAM SWITCH
			SINCPY	BIT 0 - IN COPY CODE
			SISEQ	BIT 1 - SEQ CHECK (SET BY ISEQ)
			SNOCNT	BIT 2 - CNT'N NOT ALLOWED (SET BY ICTL)
			SMAC	BIT 3 - MACRO STMT COPIED AT THIS LEVEL
9	(0009)	1	AOTSW	PROGRAM SWITCH
			AOEND	BIT 0 - END STATEMENT
			AOMEND	BIT 1 - MEND STATEMENT
			AICOPY	BIT 2 - ICTL/COPY STATEMENT
			AOPSYN	BIT 3 - OPSYN STATEMENT
			AOCOPYX	BIT 4 - ILLEGAL WITHIN COPY CODE
			AOMACROX	BIT 5 - ILLEGAL WITHIN MACRO DEF
			AOPENCDX	BIT 6 - ILLEGAL WITHIN OPEN CODE
			AOKBTNPM	BIT 7 - ALLOWED BETWEEN PROG'R MACRO
10	(000A)	1	GSCNSW	PROGRAM SWITCH
			GQST	BIT 0 - ODD QUOTE CHECKER
			GSUBS	BIT 1 - FIELD NEEDS SUBSTITUTION
			GAIF	BIT 2 - AIF STATEMENT BEING SCANNED
			METSW	BIT 3 - META TEXT INDICATION
11	(000B)	1	PARMSTAT	PROGRAM SWITCH

DISPLMNT DEC	(HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
			DMIENT	BIT 0 - ENTERED FROM MIPRTOIN ROUTINE
			DUMOPND	BIT 1 - OPERAND TREATED AS DUMMY
			DSDTX	BIT 2 - DISALLOW SDT
			DLPRN	BIT 3 - LEFT PARENTHESIS WAS READ
			DECMA	BIT 4 - END OPERAND - COMMA PASSED
			DEEQL	BIT 5 - END OPERAND - EQUAL SIGN PAST
			DQUOT	BIT 6 - ODD QUOTE STATUS
			DNOCRD	BIT 7 - NEW CARD WAS READ
12	(000C)	1	NAMBYT	PROGRAM SWITCH
			NQTSTG	BIT 0 - QUOTED STRING
			NNALFA	BIT 1 - FIRST CARD NOT ALPHA
			NCNCAT	BIT 2 - CONCATENATION
			NMPURE	BIT 3 - IMPURITY (PASSED END COLUMN)
			NNTGER	BIT 4 - INTEGER (DECIMAL)
			NOSYM	BIT 5 - O SYM
			NSSYM	BIT 6 - SEQUENCE SYMBOL
			NVSYM	BIT 7 - V SYM
13	(000D)	1	GSUMRY	PROGRAM SWITCH
			RQTSTG	BIT 0 - QUOTED STRING
			RNALFA	BIT 1 - FIRST CHARACTER NOT ALPHA
			RCNCAT	BIT 2 - CONCATENATION
			RMPURE	BIT 3 - IMPURITY (PASSED END COLUMN
			RNTGER	BIT 4 - INTEGER (DECIMAL)
			ROSYM	BIT 5 - ORDINARY SYMBOL
			RSSYM	BIT 6 - SEQUENCE SYMBOL
			RVSYM	BIT 7 - VARIABLE SYMBOL
14	(000E)	1	MSERR	PROGRAM SWITCH
			MXVS	BIT 0 - INVALID VARIABLE SYMBOL
			MXRPRN	BIT 1 - EXCESSIVE RIGHT PARENTHESES
15	(000F)	1	SDENTR	DIRECTORY ENTRY INDEX
16	(0010)	2	SDENTR1	DIR INDEX FOR EXTRN/WXTRN OPND
18	(0012)	2	DDNDX	D ENTRY INDEX
20	(0014)	4	DSTGEND	DESTINATION AREA END PLUS 1
24	(0018)	4	ENDATA	END OF DATA IN WORK BUFFER
28	(001C)	4	FPTRSV	FIELD POINTER SAVE AREA
32	(0020)	4	INPUT	INPUT WORK BUFFER ADDRESS
36	(0024)	4	IPTRSV	INPUT BUFFER ADDRESS SAVE AREA
40	(0028)	4	IRTNSV	RETURNED ADDRESS SAVE AREA
44	(002C)	4	OUTADR	OUTPUT BUFFER LOCATION
48	(0030)	4	VECPTR	PARAM VECTOR POINTER SAVE AREA
52	(0034)	4	FSTGL	BEGIN OF STRING (PARAM)
56	(0038)	4	AERRSTK	ERROR MSG STACK ADDRESS
60	(003C)	4	ESTKNDX	ERROR MSG STACK INDEX
64	(0040)	4	DSTGBGN	DESTINATION AREA POINTER
68	(0044)	4	DSTGADJ	DEST. AREA POINTER AFTER ADJ.
72	(0048)	4	DSTGNDX	DESTINATION AREA INDEX
76	(004C)	4	STGNDX	DISPATCH AREA INDEX
80	(0050)	4	EDTSVX	RETURN/TLINK REG SAVE
84	(0054)	4	EDTSVY	RETURN POINTER SAVE AREA
88	(0058)	20	EDTSVZ	R15,R3 SAVE AREA
108	(006C)	4	OCPTRSV	OPCODE POINTER SAVE AREA
112	(0070)	4	INTERMET	INTERMEDIATE LOCATION IN MT
116	(0074)	4	MEZZOPTR	INTERMEDIATE LOC IN WORK AREA

DISPLMNT			FIELD	DESCRIPTION: CONTENTS,
DEC	(HEX)	SIZE	NAME	MEANING/USE
120	(0078)	4	OPNDPTR	OPERAND FIELD POINTER SAVED
124	(007C)	4	RTNSV	POINTER SAVE AREA
128	(0080)	4	MIOPNDSV	
132	(0084)	4	NAMP	SYMBOL LOCATION POINTER
136	(0088)	4	NAML	MOVE LENGTH OF THE SYMBOL
140	(008C)	4	NAMP1	SYMBOL PTR TO EXTRN/WXTRN OPND
144	(0090)	4	NAML1	ADDT'L SYMBOL LENGTH SAVE AREA
148	(0094)	4	NOTESV1	NOTED VALUE SAVE AREA 1
152	(0098)	4	NOTESV2	NOTED VALUE SAVE AREA
156	(009C)	40	SEQSV	SEQ FIELD SAVE AREA
196	(00C4)	4	COPYSV2	SWITCH SAVE AREA
200	(00C8)	28	COPYSV3	ICTL FORMAT SAVE AREA
228	(00E4)	24	COPYSV4	
252	(00FC)	4	HICVAL	SDT HIGH CHAR VAL
256	(0100)	4	TBGLN	PREBEGIN STRING LENGTH
260	(0104)	4	TSRCLN	DATA PORTION TRUE LENGTH
264	(0108)	4	TCNTLN	CONTINUATION FLD TRUE LENGTH
268	(010C)	4	PBGLEN	PREBEGIN STRING LENGTH MINUS 1
272	(0110)	4	PNDLEN	POSTEND STRING LENGTH MINUS 1
276	(0114)	4	ENDCOL	END COLUMN MINUS 1
280	(0118)	8	SMACNAM	MACRO NAME SAVE AREA
288	(0120)	8	COPYCODE	COPY CODE
296	(0128)	2	COPYLN	COPY CODE LENGTH
298	(012A)	2	DSTGLN	STRING LENGTH
300	(012C)	2	OCSAVE	INTERNAL OP CODE SAVE AREA
302	(012E)	2	PRNLVL	PAREN LEVEL COUNTER
304	(0130)	2	FLAGBT	FLAG BYTE SAVE AREA
306	(0132)	2	DTLENG	DATA LENGTH
308	(0134)	2	OPNDCTR	OPERAND COUNTER
310	(0136)	2	MINDIF	DIF BETWEEN MINPUT AND INPUT
312	(0138)	4	MTXTP	MI/PROTO META TEXT POINTER
316	(013C)	4	MINPUT	CURRENT MI DATA AREA POINTER
320	(0140)	4	MINPSTD	STANDARD MINPUT SAVED
324	(0144)	4	STNPSTD	STANDARD INPUT SAVED
328	(0148)	4	MINPADJ	ADJUSTED MINPUT SAVED
332	(014C)	4	STNPADJ	ADJUSTED INPUT SAVED
336	(0150)	4	OPCDPTR	OP CODE FIELD POINTER
340	(0154)	4	ENDWKA	END OF DATA AREA PLUS 1
344	(0158)	4	MREGSV	EDSECT BASE REG SAVED
348	(015C)	4	SVENDWKA	SAVE END OF DATA AREA+1
352	(0160)	4	COLCTR	COLUMN COUNTER
356	(0164)	4	OFPTRSV	INDEXP SAVE AREA
360	(0168)	4	SVMINDIF	SAVE STANDARD MINDIF
364	(016C)	4	RAVSP	RSTACK NEXT AVAILABLE LOCATION
368	(0170)	4	NRSTK	END OF RSTACK + 1
372	(0174)	200	RSTACK	MAXIMUM OF 25 ENTRIES
572	(023C)	56	CSTK	COPY CODE RECURSION STACK
628	(0274)	4	NCSTK	5 ENDING ADDRESS OF CSTK+1
632	(0278)	4	BCSTK	6 CSTK BEGIN ADDRESS
636	(027C)	4	CSTKADR	7 CSTK NEXT AVAILABLE LOCATION
640	(0280)	64	SAVMALL	REGISTER SAVE AREA
704	(02C0)	2	ERRCNT	ERROR MSG COUNT - MAX 5 MSGS.
706	(02C2)	66	ERRSTK	ERROR MSG STACK
772	(0304)	4	SVLAST	LAST STACK ELEMENT POINTER
776	(0308)	4	ALAST	START OF STACK--CONSTANT
780	(030C)	1	TEMPOP	OPERATOR
781	(030D)	1	TEMPBIND	BINDING FACTOR
782	(030E)	72	STACK	MAXIMUM OF 35 OPERATORS IN
854	(0356)	1	VSFLG	SET VAR TYPE SAVED FOR NAME
856	(0358)	1	STGCNT	STRING COUNTER

DISPLMNT DEC	 (HEX)	 SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
857	(0359)	1	CNTCTR	CONTINUATION CARD COUNTER
858	(035A)	42	SEQSVT	SEQ FIELD - COMPARE V. SEQSV
900	(0384)	4	ADJSV	RETURN POINTER SAVED HERE
904	(0388)	4	VSRSV	VSRTN RETURN LINKAGE
908	(038C)	4	VSRSV1	HEADER DATA POINTER
912	(0390)	4	MPOPSV	MPOPND ROUTINE RETURN LINKAGE
916	(0394)	4	NEXPSV	RETURN LINKAGE SAVED
920	(0398)	8	SUBSAVE	MPOPSV/NEXPSV SAVE AREA
928	(03A0)	4	REGSAVE3	REGISTER SAVE AREA
932	(03A4)	6	DNTERR	ENTRY POINT TO LOG ERROR
938	(03AA)	1	DSEVCD	SEVERITY CODE
939	(03AB)	5	DERRCD	ERROR CODE
944	(03B0)	4	FREESTRT	PTR TO START OF DICT WORK AREA
948	(03B4)	4	VSDSTRT	PTR TO START OF VARB SYMB DIR
952	(03B8)	4	MDDSTRT	PTR TO START OF MACR DEFN DIR
956	(03BC)	4	SSRDSTRT	PTR TO START OF SEQ SYMB REF DT
960	(03C0)	4	VSDSLOT	PTR TO NEXT AVAIL VSD ENTRY
964	(03C4)	4	OSRDSTRT	PTR TO START OF ORD SYMB REF DT
968	(03C8)	4	MDDSLOT	PTR TO NEXT AVAIL MDD ENTRY
972	(03CC)	4	CURMDDPT	PTR TO CURRENT MDD ENTRY
976	(03D0)	4	REGSAVE1	REGISTER SAVE AREA
980	(03D4)	4	GTMVALOC	MACRO DEFINITION VECTOR LENGTH
984	(03D8)	4	HIBYTE0	FULL WORD WORK AREA
988	(03DC)	4	MDDCHN	MASTER LINK, CHAINED MOD ENTRYS
992	(03E0)	4	MDDCNT	NUMBER OF MDD ENTRYS
996	(03E4)	4	OPSCHN	MASTER LINK, CHAINED OPSYN ENTR
1000	(03E8)	4	GTPVALOC	POSITIONAL PARAM VECTOR LENGTH
1004	(03EC)	4	GTKVALOC	KEYWORD PARAM VECTOR LENGTH
1008	(03F0)	4	GTLDALOC	LOCAL DICTIONARY LENGTH
1012	(03F4)	4	GTGVALOC	GLOBAL VECTOR LENGTH
1016	(03F8)	4	GTSDALOC	SEQ SYMB REFER DICT LENGTH
1020	(03FC)	2	SSRAPDIS	DISPL IN SSRD FOR NEXT ENTRY
1022	(03FE)	1	SWITCHA	PROGRAM SWITCH
			FNDFLG	BIT 0 - MATCHING DIRECT ENTRY FOUND
			NOTEFIL2	BIT 1 - NOTE OF NEXT RECORD REQ'D
			LSTSYSMS	BIT 2 - SYSTEM MACRO EDIT COMPLETED
			ITERSW	BIT 3 - SYSTEM VARIABLE DEFINITIONS
1023	(03FF)	1	FSWITCH	FIRST RECORD WRITTEN NOTED
1024	(0400)	4	GTODALOC	ORD SYMB REF DICT LENGTH
1028	(0404)	2	OSRAPDIS	DISPL IN OSRD FOR NEXT ENTRY
1030	(0406)	2	SSDLNGTH	LENGTH OF SSRD ENTRY
1032	(0408)	1	SSFLGVAL	TEXT FLAG FOR SEQ SYMB REFER
1033	(0409)	1	SREFTYPE	RECORD TYPE, SEQ SYMB REFER
1034	(040A)	4		FILLER FOR ALIGNMENT (REQ'D)
1038	(040E)	2	OSDLNGTH	LENGTH OF OSRD ENTRY
1040	(0410)	1	OSFLGVAL	TEXT FLAG FOR ORD SYMB REFER
1041	(0411)	3	OREFTYPE	RECORD TYPE, ORD SYMB REFER
1044	(0414)	4	REGSAVE2	REGISTER SAVE AREA
1048	(0418)	4	PIOPARMB	FULL I/O AREA LENGTH
1052	(041C)	4	PIOPARMA	CURRENT I/O AREA ADDRESS
1056	(0420)	2	PIOPARMC	CURRENT I/O AREA LENGTH
1058	(0422)	1	IOCID	PROGRAM SWITCH
			IZRO	BIT 0 - IOCLNG - OPCODE
			IONE	BIT 1 - LENGTH REDEFINED
			ITWO	BIT 2 - IOCTYD - OPCODE
			ITRE	BIT 3 - TYPE REDINED

DISPLMNT			FIELD	DESCRIPTION: CONTENTS,
DEC	(HEX)	SIZE	NAME	MEANING/USE
1059	(0423)	1	CONCODE	PROGRAM SWITCH
			B0	BIT 0 - NOT USED
			B1	BIT 1 - NOT USED
			B2	BIT 2 - NOT USED
			B3	BIT 3 - NOT USED
			B4	BIT 4 - NOT USED
			B5	BIT 5 - NOT USED
			B6	BIT 6 - NOT USED
			B7	BIT 7 - NOT USED
1060	(0424)	1	ATTRSV	PROGRAM SWITCH
			AT0	BIT 0 -
			AT1	BIT 1 -
			AT2	BIT 2 -
			AT3	BIT 3 -
			AT4	BIT 4 -
			AT5	BIT 5 -
			AT6	BIT 6 -
			AT7	BIT 7 -
1061	(0425)	1	MCALL	PROGRAM SWITCH
			MCLA	BIT 0 - SETA TYPE
			MCLC	BIT 1 - SETC TYPE
			MCMPLX	BIT 2 - COMPLEX STATE
			MSLST	BIT 4 - SYSLIST
1062	(0426)	1	FLGBYT	PROGRAM SWITCH
			VTYP1	BIT 0 - 0&1: 00 GLOBAL; 10-NOT DEFINED
			VTYP2	BIT 1 - 01 LOCAL; 11-PARAMETER
			VPTYP	BIT 2 - POSITIONAL/KEYWORD
			VSNS	BIT 3 - SYSTEM/NON-SYSTEM
			VSLS	BIT 4 - SYSLIST/NON-SYSLIST
			VDIM	BIT 5 - DIMENSIOND/NON-DIMENSIONED
			VSTP1	BIT 6 - SUBTYPE
			VSTP2	BIT 7 - SUBTYPE
1063	(0427)	8	NOTESAVE	NOTE OF START OF MACRO DEFINTN
1071	(042F)	9	SAVENOTE	NOTE OF START OF DICT DATA FILE
1080	(0438)	32	REGSTACK	REGISTER SAVE AREA
1112	(0458)	16		PATCH AREA
1128	(0468)		ENDEDSCT	END OF MODULE COMMON AREA

FIELD NAME	DISPLACEMENT DECIMAL	(HEX)
ADJSV	900	(384)
*AERRSTK	56	(38)
AICOPY	9	(9)
ALAST	776	(308)
AOCOPYX	9	(9)
AOEND	9	(9)
AOKBTNPM	9	(9)
AOMACROX	9	(9)
AOMEND	9	(9)
AOPENCDX	9	(9)
AOPSYN	9	(9)
AOTSW	9	(9)
ATTRSV	1060	(424)
AT0	1060	(424)
AT1	1060	(424)
AT2	1060	(424)
AT3	1060	(424)
AT4	1060	(424)
AT5	1060	(424)
AT6	1060	(424)
AT7	1060	(424)
*BCSTK	632	(278)
B0	1059	(423)
B1	1059	(423)
B2	1059	(423)
B3	1059	(423)
B4	1059	(423)
B5	1059	(423)
B6	1059	(423)
B7	1059	(423)
CNTCTR	857	(359)
COLCTR	352	(160)
CONCODE	1059	(423)
COPYCODE	288	(120)
COPYLN	296	(128)
COPYSV2	196	(C4)
COPYSV3	200	(C8)
COPYSV4	228	(E4)
CSTK	572	(23C)
*CSTKADR	636	(27C)
CURMDDPT	972	(3CC)
DDNDX	18	(12)
DECMA	11	(B)
DEEQL	11	(B)
*DERRCD	939	(3AB)
DLPRN	11	(B)
DMIENT	11	(B)
DNOCRD	11	(B)
DNTERR	932	(3A4)
DQUOT	11	(B)
DSDTX	11	(B)
*DSEVCD	938	(3AA)
DSTGADJ	68	(44)
DSTGBGN	64	(40)
*DSTGEND	20	(14)
DSTGLN	298	(12A)
DSTGNDX	72	(48)

*POINTER

FIELD NAME	DISPLACEMENT DECIMAL	(HEX)
DTLENG	306	(132)
DUMOPND	11	(B)
EDTSVX	80	(50)
EDTSVY	84	(54)
EDTSVZ	88	(58)
*ENDATA	24	(18)
ENDCOL	276	(114)
ENDEDSCT	1128	(468)
ENDWKA	340	(154)
ERRCNT	704	(2C0)
ERRSTK	706	(2C2)
*ESTKNDX	60	(3C)
FLAGBT	304	(130)
FLGBYT	1062	(426)
FNDFLG	1022	(3FE)
*FPTRSV	28	(1C)
FREESTRT	944	(3B0)
FSTGL	52	(34)
FSWITCH	1023	(3FF)
GAIF	10	(A)
GQST	10	(A)
GSCNSW	10	(A)
GSUBS	10	(A)
GSUMRY	13	(D)
GTGVALOC	1012	(3F4)
GTKVALOC	1004	(3EC)
GTLDALOC	1008	(3F0)
GTMVALOC	980	(3D4)
GTODALOC	1024	(400)
GTPVALOC	1000	(3E8)
GTSDALOC	1016	(3F8)
HIBYTE0	984	(3D8)
HICVAL	252	(FC)
*INPUT	32	(20)
INTERMET	112	(70)
IOCID	1058	(422)
IONE	1058	(422)
*IPTRSV	36	(24)
*IRTNSV	40	(28)
ITERSW	1022	(3FE)
ITRE	1058	(422)
ITWO	1058	(422)
IZRO	1058	(422)
LSTSYSMS	1022	(3FE)
MCALL	1061	(425)
MCLA	1061	(425)
MCLC	1061	(425)
MCMPLX	1061	(425)
MDDCHN	988	(3DC)
MDDCNT	992	(3E0)
MDDSLOT	968	(3C8)
MDDSTRT	952	(3B8)
METSW	10	(A)
MEZZOPTR	116	(74)
MINDIF	310	(136)
MINPADJ	328	(148)
MINPSTD	320	(140)
MINPUT	316	(13C)

*POINTER

FIELD NAME	DISPLACEMENT DECIMAL	(HEX)
MIOPNDSV	128	(80)
MPOPSV	912	(390)
MREGSV	344	(158)
MSERR	14	(E)
MSLST	1061	(425)
MTXTP	312	(138)
MXRPRN	14	(E)
MXVS	14	(E)
NAMBYT	12	(C)
NAML	136	(88)
NAML1	144	(90)
NCNCAT	12	(C)
*NCSTK	628	(274)
NEXPSV	916	(394)
NMPURE	12	(C)
NNALFA	12	(C)
NNTGER	12	(C)
NOSYM	12	(C)
NOTEFIL2	1022	(3FE)
NOTESAVE	1063	(427)
NOTESV1	148	(94)
NOTESV2	152	(98)
NQTSTG	12	(C)
*NRSTK	368	(170)
NSSYM	12	(C)
NVSYM	12	(C)
OCPTRSV	108	(6C)
OCSAVE	300	(12C)
OFPTRSV	356	(164)
OPCDPTR	336	(150)
OPNDCTR	308	(134)
OPNDPTR	120	(78)
OPSCHN	996	(3E4)
OREFTYPE	1041	(411)
OSDLNGTH	1038	(40E)
OSFLGVAL	1040	(410)
OSRAPDIS	1028	(404)
OSRDSTRT	964	(3C4)
*OUTADR	44	(2C)
PARMSTAT	11	(B)
PBGLEN	268	(10C)
PIOPARMA	1052	(41C)
PIOPARMB	1048	(418)
PIOPARMC	1056	(420)
PNDLEN	272	(110)
POSSUBL	5	(5)
PRNLVL	302	(12E)
PROTOCAL	5	(5)
*RAVSP	364	(16C)
RCNCAT	13	(D)
REGSAVE1	976	(3D0)
REGSAVE2	1044	(414)
REGSAVE3	928	(3A0)
REGSTACK	1080	(438)
RMPURE	13	(D)
RNALFA	13	(D)
RNTGER	13	(D)
ROSYM	13	(D)

*POINTER

FIELD NAME	DISPLACEMENT DECIMAL	(HEX)
RQTSTG	13	(D)
RSSYM	13	(D)
RSTACK	372	(174)
RTNSV	124	(7C)
RVSYM	13	(D)
SABORT	3	(3)
SALLCT	1	(1)
SASTCMT	6	(6)
SAVENOTE	1071	(42F)
SAVMALL	640	(280)
SBDPROTO	6	(6)
SBYCNT	1	(1)
SBYONE	1	(1)
SCMTCT	2	(2)
SCNCAT	5	(5)
SCOPY	4	(4)
SCTLRTN	1	(1)
SDENT	4	(4)
SDENTR	15	(F)
SDENTR1	16	(10)
SDINIT	4	(4)
SDTCMT	6	(6)
SENAME	7	(7)
SENDST	5	(5)
SEOPCD	7	(7)
SEOPND	7	(7)
SEQSV	156	(9C)
SEQSVT	858	(35A)
SFSTCD	4	(4)
SGBLCL	2	(2)
SICTL	3	(3)
SINCPY	8	(8)
SINEOF	2	(2)
SISEQ	8	(8)
SKPEND	3	(3)
SKPMND	3	(3)
SKPNAME	5	(5)
SKWPRM	5	(5)
SLSTCD	2	(2)
SMAC	8	(8)
SMACNAM	280	(118)
SMDDENTR	4	(4)
SMDEF	0	(0)
SMI	2	(2)
SMISCN	0	(0)
SNMFND	6	(6)
SNOACTR	3	(3)
SNOCNT	8	(8)
SNOFND	6	(6)
SNOPND	1	(1)
SNOPSYN	0	(0)
SNOSMCRO	6	(6)
SNOSYSMD	6	(6)
SNXTCT	2	(2)
SONECD	1	(1)
SONECT	1	(1)
SOPNCD	3	(3)
SPGRMD	3	(3)

*POINTER

FIELD	DISPLACEMENT	
NAME	DECIMAL	(HEX)
SPRMER	5	(5)
SPRVCT	2	(2)
SREFTYPE	1033	(409)
SSDLNGTH	1030	(406)
SSFLGVAL	1032	(408)
SSRAPDIS	1020	(3FC)
SSRDSTRT	956	(3BC)
SSYSMD	3	(3)
STACK	782	(30E)
STGCNT	856	(358)
STGNDX	76	(4C)
STNPADJ	332	(14C)
STNPSTD	324	(144)
STRCMT	6	(6)
SUBLST	5	(5)
SUBSAVE	920	(398)
SUBSOP	2	(2)
SUPDNT	4	(4)
SVENDWKA	348	(15C)
SVLAST	772	(304)
SVMINDIF	360	(168)
SWITCHA	1022	(3FE)
SWITCH1	0	(0)
SWITCH2	1	(1)
SWITCH3	2	(2)
SWITCH4	3	(3)
SWITCH5	4	(4)
SWITCH6	5	(5)
SWITCH7	6	(6)
SWITCH8	7	(7)
SWITCH9	8	(8)
SXMCRO	4	(4)
SXPRTO	0	(0)
TBGLN	256	(100)
TCNTLN	264	(108)
TEMPBIND	781	(30D)
TEMPOP	780	(30C)
TSRCLN	260	(104)
VDIM	1062	(426)
VECPTR	48	(30)
VPTYP	1062	(426)
VSDSLOT	960	(3C0)
VSDSTRT	948	(3B4)
VSFLG	854	(356)
VSLS	1062	(426)
VSNS	1062	(426)
VSRSV	904	(388)
VSRSV1	908	(38C)
VSTP1	1062	(426)
VSTP2	1062	(426)
VTYP1	1062	(426)
VTYP2	1062	(426)

*POINTER

DSECT NAME: **ENDFIL**

LOAD MODULE: IFOX11

SIZE: 3

CREATED BY: IFNX1J

REFERENCED BY: IFNX2A

UPDATED BY:

FUNCTION: END-OF-SEGMENT RECORD FOR TEST DICTIONARY FILE

OPERATIONS DIAGRAMS:

DISPLMNT DEC (HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE	
0 (0000)	2	EFILRL	RECORD LENGTH	2 BYTES
2 (0002)	1	EFILRT	X'10' RECORD TYPE	1 BYTE

DSECT NAME: **ENDSEG**

LOAD MODULE: IFOX11

SIZE: 3

CREATED BY: IFNX1J

REFERENCED BY: IFNX2A

UPDATED BY:

FUNCTION: END-OF-SEGMENT RECORD FOR TEST SEGMENT DICTIONARY FILE

OPERATIONS DIAGRAMS: 5

DISPLMNT DEC (HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
0 (0000)	2	ESEGRL	RECORD LENGTH 2 BYTES
2 (0002)	1	ESEGRT	X'10' RECORD TYPE 1 BYTE

DSECT NAME: **ERRIN**

LOAD MODULE: IFOX51

SIZE: 22

CREATED BY: IFNX5C

REFERENCED BY: IFNX5V

UPDATED BY:

FUNCTION: ERROR INDICATOR

OPERATIONS DIAGRAMS: 21

DISPLMNT DEC	 (HEX)	 SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
0	(0000)	2	ERRLEN	ERROR RECORD LENGTH
2	(0002)	1	ERRID	ERROR IDENTIFIER
3	(0003)	1	NUMERR	NUMBER OF ERRORS
4	(0004)	2	ERRSTMT	ERROR STATMENT NUMBER
6	(0006)	1	ERRNUM	ERROR NUMBER
7	(0007)	15	ERRFLD	REST OF ERRORS

FIELD NAME	DISPLACEMENT DECIMAL	 (HEX)
ERRFLD	7	(7)
ERRID	2	(2)
ERRLEN	0	(0)
ERRNUM	6	(6)
ERRSTMT	4	(4)
NUMERR	3	(3)

*POINTER

DSECT NAME: **ERRMESS**

LOAD MODULE: IFOX11

SIZE: 11

CREATED BY: IFNX1A

REFERENCED BY: IFNX1A,IFNX1J,IFNX1S,IFNX3A,IFNX3N

UPDATED BY:

FUNCTION: ENTRY IN ERROR MESSAGE STACK

OPERATIONS DIAGRAMS:

DISPLMNT DEC	(HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
0	(0000)	0	EMSGSVTY	ERROR MSG SEVERITY CODE
1	(0001)	1	EMSGCODE	ERROR MSG CODE
2	(0002)	1	ENTRYLNG	ERROR MSG ENTRY LENGTH
3	(0003)	8	EMSGNTRY	ERROR MSG ENTRY

DSECT NAME: **FARENT**

LOAD MODULE: IFOX51

SIZE: 3

CREATED BY: IFNX5M

REFERENCED BY: IFNX5M

UPDATED BY:

FUNCTION: MAPS OPCODE RESTRICTIONS

OPERATIONS DIAGRAMS: 21, 22

DISPLMNT DEC (HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
0 (0000)	1	FMT	PROGRAM SWITCH
		FSNLIT	BIT 0 - NO LITERAL
		FILEN	BIT 4 - LENGTH FIELD
		FIAL1	BIT 6 - FIRST BIT OF FIALOC
1 (0001)	1	RIST	PROGRAM SWITCH
		RIDEC	BIT 0 - DECIMAL DIGIT
1 (0001)	1	RSST	PROGRAM SWITCH
		RSMOD	BIT 0 - STORAGE MODIFIED
		RSALW	BIT 4 - ALIGNMENT ALWAYS CHECKED

FIELD NAME	DISPLACEMENT DECIMAL	 (HEX)
FENT	0	(0)
FIAL1	0	(0)
FILEN	0	(0)
FMT	0	(0)
FSNLIT	0	(0)
RIDEC	1	(1)
RIST	1	(1)
RSALW	1	(1)
RSMOD	1	(1)
RSST	1	(1)
VEOP	0	(0)

*POINTER

DSECT NAME: **GBLDEF**

LOAD MODULE: IFOX11

SIZE: 7-13

CREATED BY: IFNX1J

REFERENCED BY: IFNX2A

UPDATED BY: IFNX2A

FUNCTION: GLOBAL DEFINITION RECORD FOR TEXT
SEGMENT DICTIONARY FILE

OPERATIONS DIAGRAMS: 5

DISPLMNT DEC (HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE	
0 (0000)	2	GDEFRL	RECORD LENGTH	2 BYTES
2 (0002)	1	GDEFRT	X'00' RECORD TYPE	1 BYTE
3 (0003)	1	GDEFF	FLAGS*	1 BYTE
4 (0004)	1	GDEFSL	SYMBOL LENGTH	1 BYTE
0 (0000)	1	GDEFTF	TEXT FLAG VALUE	1 BYTE
1 (0001)	3	GDEFVP	VECTOR POINTER	3 BYTES
4 (0004)	2	GDEFD	DIMENSION	2 BYTES

*SEE FLGBYT IN EDSECT

FIELD NAME	DISPLACEMENT DECIMAL	 (HEX)
GDEFD	4	(4)
GDEFF	3	(3)
GDEFRL	0	(0)
GDEFRT	2	(2)
GDEFSL	4	(4)
GDEFTF	0	(0)
GDEFVP	1	(1)

*POINTER

DSECT NAME: **GBLNTRY**

LOAD MODULE: IFOX11

SIZE: 13-19

CREATED BY: IFNX1J

REFERENCED BY: IFNX1J

UPDATED BY:

FUNCTION: GLOBAL VARIABLE ENTRY IN VARIABLE SYMBOL DEFINITION DIRECTORY (IN-CORE WORK TABLE)

OPERATIONS DIAGRAMS: 5

DISPLMNT DEC	(HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE	
0	(0000)	3	GCHAIN	CHAIN POINTER	3 BYTES
3	(0003)	1	GFLAGS	FLAGS	1 BYTE
4	(0004)	1	GLNGTH	SYMBOL LENGTH	1 BYTE
5	(0005)	2-8	GSYMBL	VARIABLE SYMBOL	
0	(0000)	1	GTFVAL	TEXT FLAG VALUE	1 BYTE
1	(0001)	3	GVECTR	VECTOR POINTER	3 BYTES
4	(0004)	2	GDIMEN	DIMENSION	2 BYTES

FIELD NAME	DISPLACEMENT DECIMAL	(HEX)
GCHAIN	0	(0)
GDIMEN	4	(4)
GFLAGS	3	(3)
GLNGTH	4	(4)
GTFVAL	0	(0)
GVECTR	1	(1)

*POINTER

DSECT NAME: **GDNTRY**

LOAD MODULE: IFOX21

SIZE: 13-19

CREATED BY: INFX2A

REFERENCED BY: IFNX2A

UPDATED BY:

FUNCTION: GLOBAL DEFINITION DIRECTORY ENTRY (IN-CORE WORK TABLE)

OPERATIONS DIAGRAMS: 10

DISPLMNT DEC (HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
0 (0000)	3	GDCP	CHAIN POINTER 3 BYTES
3 (0003)	1	GDFL	PROGRAM SWITCH
		GTYP1	BIT 0 -
		GTYP2	BIT 1 -
		GPTYP	BIT 2 -
		GSNS	BIT 3 -
		GSLS	BIT 4 -
		GDIM	BIT 5 - DIMENSIONEC IF 1
		GSTP1	BIT 6 - 6 & 7 SUBTYPE: 00 A-TYPE
		GSTP2	BIT 7 - 01 B-TYPE 10 PARAMETER 11 C-TYPE
4 (0004)	1	GDSL	SYMBOL LENGTH 1 BYTE
5 (0005)	2-8	GDSYM	SYMBOL LENGTH
0 (0000)	1	GDTFV	TEXT FLAG VALUE 1 BYTE
1 (0001)	3	GDDP	G.T. DICT. PTR 3 BYTES
4 (0004)	2	GDDM	DIMENSION 2 BYTES

FIELD NAME	DISPLACEMENT DECIMAL	(HEX)
GDCP	0	(0)
GDDM	4	(4)
GDDP	1	(1)
GDFL	3	(3)
GDIM	3	(3)
GDTFV	0	(0)
GPTYP	3	(3)
GSLS	3	(3)
GSNS	3	(3)
GSTP1	3	(3)
GSTP2	3	(3)
GTYP1	3	(3)
GTYP2	3	(3)

*POINTER

DSECT NAME: J

LOAD MODULE: IFOX00

SIZE: 1272

CREATED BY: IFNX0A

REFERENCED BY: ALL MODULES

UPDATED BY: SEE MICROFICHE

FUNCTION: COMMON

OPERATIONS DIAGRAMS:

DISPLMNT			FIELD	DESCRIPTION: CONTENTS,
DEC	(HEX)	SIZE	NAME	MEANING/USE
0	(0000)	0	JCOMMON	BEGINNING OF COMMON
0	(0000)	72	JSAVE	SYSTEM SAVE AREA
72	(0048)	56	JFLEBLK1	FILE BLOCK 1
128	(0080)	56	JFLEBLK2	FILE BLOCK 2
184	(00B8)	56	JFLEBLK3	FILE BLOCK 3
240	(00F0)	2	JMAXRL1	MAX RL FOR FILE 1
242	(00F2)	2	JMAXRL2	MAX RL FOR FILE 2
244	(00F4)	2	JMAXRL3	MAX RL FOR FILE 3
246	(00F6)	2	JMAXRL	MIN OF MAX RL FOR ALL FILES
248	(00F8)	4	JADINCM	ADDRESS OF INPUT COMMON
252	(00FC)	4	JADOUTCM	ADDRESS OF OUTPUT COMMON
256	(0100)	0	JPHNAME	PHASE NAME OF LAST PHASE LOADED
256	(0100)	3	JPHPREF	PHASE NAME PREFIX
259	(0103)	3	JPHSUFF	PHASE NAME SUFFIX
262	(0106)	2	JPHBLANK	TWO BLANKS
264	(0108)	0	JLVTMDT	ASM LEVEL, TIME, DATE
264	(0108)	10		SAME
274	(0112)	5	JSYSTIME	HH.MM
279	(0117)	1		BLANK
280	(0118)	8	JSYSDATE	MM/DD/YY OR DD/MM/YY
288	(0120)	1	JDECKIDL	LENGTH OF DECK ID (0 THRU 8)
289	(0121)	8	JDECKID	INTERNAL DECK ID
297	(0129)	0	JPARMS	MSGLEVEL AND LINECOUNT
297	(0129)	1	JMSGL	MSGLEVEL=
298	(012A)	2	JLNCT	LINECNT=
300	(012C)	4	JSYSPARM	SYSPARM POINTER
304	(0130)	4	JPARMPTR	ADDR OF TRANS PARM (IF PRESENT)
308	(0134)	4	JPARM	OPTION PARMS (PARM 1,2,3,4)
308	(0134)	1	JPARM1	PROGRAM SWITCH
			JLIST	BIT 0 - PRINT LISTING
			JXREF	BIT 1 - PRINT XREF
			JESD	BIT 2 - PRINT ESD'S
			JRLD	BIT 3 - PRINT RLD'S
			JDECK	BIT 4 - PUNCH DECK
			JLINK	BIT 5 - WRITE OBJECT MODULE
			JTEST	BIT 6 - PUNCH SYMBOL TABLE
309	(0135)	1	JPARM2	PROGRAM SWITCH
			JRENT	BIT 0 - REENTRANT CHECKING
			JALGN	BIT 1 - ALIGNMENT CHECKING
			JSYSMAC	BIT 2 - PRINT SYSTEM MACROS

DISPLMNT DEC	 (HEX)	 SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
			JALOGIC	BIT 3 - PRINT ASSEMBLER LOGIC
			JMLOGIC	BIT 4 - PRINT MACRO LOGIC
			JCALLS	BIT 5 - PRINT INNER MACRO CALLS
310	(0136)	1	JPARM3	PROGRAM SWITCH
			JTERM	BIT 0 - PRINT TO TERMINAL
			JSTMT	BIT 1 - PRINT STMT NO. ON TERM
			JNUM	BIT 2 - PRINT SEQ NO. ON TERM
			JMINBUF	BIT 4 - MINIMUM BUFFERS OR
			JLNCTKEY	BIT 6 - FIXED LINECNT
			JMSGLKEY	BIT 7 - FIXED MSGLEVEL
311	(0137)	1	JPARM4	PROGRAM SWITCH
			JPREFIX	CL3 -
312	(0138)	3	JPREFIX	COMPONENT NAME
315	(013B)	1	JWARNFLG	PROGRAM SWITCH
			JYCON	BIT 0 - RELOCATABLE YCON
			JREENTR	BIT 1 - REENTRANT CHK FAILED
316	(013C)	1	JRECCHK	PROGRAM SWITCH
			JRLDCHK	BIT 0 - RLD RECORDS PRESENT
			JXREFCHK	BIT 1 - XREF RECORDS PRESENT
			JERRCHK	BIT 2 - ERROR RECORDS PRESENT
			JESDCHK	BIT 3 - ESDID PRESENT ON END
			JENDCHK	BIT 4 - PUNCH END CARD
317	(013D)	1	JINDERRF	PROGRAM SWITCH
			JMISLIN	BIT 0 - MISSING SYSLIN DD CARD
			JMISPCH	BIT 1 - MISSING SYSPUNCH DD CARD
			JINVOPT	BIT 2 - INVALID OPTION
			JESDOFLO	BIT 3 - ESD OVERFLOW
			JMISPRT	BIT 4 - MISSING SYSPRINT DD CARD
318	(013E)	1	JPDFLAG	PROGRAM SWITCH
			JDUMPX0	BIT 0 - DUMP PHASE X0
			JDUMPX1	BIT 1 - DUMP PHASE X1
			JDUMPX2	BIT 2 - DUMP PHASE X2
			JDUMPX3	BIT 3 - DUMP PHASE X3
			JDUMPX4	BIT 4 - DUMP PHASE X4
			JDUMPX5	BIT 5 - DUMP PHASE X5
			JDUMPX6	BIT 6 - DUMP PHASE X6
319	(013F)	1	JINFLAG	PROGRAM SWITCH
			JIN2ND	BIT 0 - ENTERED JININIT ONCE
			JINLIB	BIT 1 - INPUT FROM LIBRARY
320	(0140)	1	JOUTFLAG	PROGRAM SWITCH
			JOUT2ND	BIT 0 - ENTERED JOUTINIT ONCE
			JNOSEQPH	BIT 1 - DON'T SEQ PUNCH

DISPLMNT			FIELD	DESCRIPTION: CONTENTS,
DEC	(HEX)	SIZE	NAME	MEANING/USE
321	(0141)	1	JMLCFLAG	PROGRAM SWITCH
			JPT4STAR	BIT 0 - POINT TO START OF FILE
			JPT4READ	BIT 1 - READ TO FOLLOW POINT
			JPT4WRIT	BIT 2 - WRITE TO FOLLOW POINT
			JPT4GET	BIT 3 - GET TO FOLLOW POINT
			JDBLALL	BIT 4 - FILES CAN BE DBLBUF
324	(0144)	4	JMLC	ADDRESS OF MAIN LINE CONTROL
328	(0148)	4	JINMLC	ADDR OF INPUT MAIN LINE CONTROL
332	(014C)	4	JOUTMLC	ADDR OF OUTPUT MAIN LINE CONTROL
336	(0150)	4	JPDUMP	ADDRESS OF PDUMP ROUTINE
340	(0154)	8	JNOTEVAL	VALUE FROM JNOTE
348	(015C)	4	JRECIN	NUMBER OF RECORDS FROM SYSIN
352	(0160)	4	JRECLIB	NUMBER OF RECORDS FROM LIBRARY
356	(0164)	4	JRECPCH	NUMBER OF CARDS PUNCHED
360	(0168)	4	JRECPRT	NUMBER OF LINES PRINTED
364	(016C)	4	JSLEN	LENGTH OF AREA (JEOS-JBOS)
368	(0170)	4	JBOS	BEGINNING OF AVAILABLE CORE
372	(0174)	4	JEOS	NEXT AVAILABLE GETCORE AREA
376	(0178)	4	JCLVLPTR	CURRENT SAVE LEVEL PTR
380	(017C)	4		SIZE OF ONE SAVE AREA
384	(0180)	320	JSAVETBL	PUSH/POP SAVE AREA
704	(02C0)	4	JABORT	ABORT ROUTINE LINKAGE
708	(02C4)	4	JAABORT	ADDR OF ABORT ROUTINE
712	(02C8)	4	JSYSOPEN	WORKFILE OPEN
716	(02CC)	4	JSYSCLOS	WORKFILE CLOSE
720	(02D0)	4	JCONTCL	CONTINUE COLUMN
722	(02D2)	2	JENDCOL	END COLUMN
724	(02D4)	4	ENTRPUTL	ENTRY POINT OF PUTLINE
728	(02D8)	8	JDWORD	DOUBLE WORD OF TEMP STORAGE
736	(02E0)	4	JFWORD1	TWO FULL WORDS
740	(02E4)	4	JFWORD2	OF TEMP STORAGE
744	(02E8)	2	JHWORD1	TWO HALF WORDS
746	(02EA)	2	JHWORD2	OF TEMP STORAGE
748	(02EC)	4	JSRCLN	DATA PORTION MOVE LENGTH (1-7)
752	(02F0)	4	JBEGCL	BEGIN COLUMN MINUS 1 (2-7)
756	(02F4)	4	JCTCHR	CONT CHR COLUMN MINUS 1 (3-7)
760	(02F8)	4	JSEQCL	SEQ FLD BEGIN COL MINUS 1 (4-7)
764	(02FC)	4	JSEQLN	SEQ FLD MOVE LENGTH (5-7)
768	(0300)	4	JCTBGN	CONT COLUMN MINUS 1 (6-7)
772	(0304)	4	JCTLN	CONT FLD MOVE LENGTH (7-7)
776	(0308)	2	JINFILE	INPUT FILE NO. FOR X4,X5,X6
778	(030A)	2	JOUTFILE	OUTPUT FILE NO. FOR X4,X5,X6
780	(030C)	4	JENTRYPT	ENTRY POINT ADDR FOR END CARD
784	(0310)	2	JESDID	ESDID FOR OBJECT END CARD
786	(0312)	2	JPAGENO	PAGE NUMBER FOR LISTING
788	(0314)	56	JDPASS	COMMUN. BETWEEN X2A AND X3N
844	(034C)	1	JSEVER	HIGHEST SEVERITY FOR X5, X6
845	(034D)	1	JPRTONLY	CATASTROPHIC ERROR IN X4
846	(034E)	1	JSW0013	PROGRAM SWITCH
			JSYSGEN	BIT 0 - OFF INDICATES SYSGEN MODE
847	(034F)	1		(NOT USED)
848	(0350)	2	JLITLNG	MAXIMUM LITERAL LENGTH
850	(0352)	51	JTBLTRT	TRANSLATE AND TEST TABLE
901	(0385)	259	JTRTABLE	SELF MAPPING TRANSLATE TABLE
1160	(0488)	72	JSAFE	SAVE AREA FOR PDUMPS
1232	(04D0)	40	JIDR	IDR
1272	(04F8)		JCOMEND	END OF COMMON

FIELD	DISPLACEMENT	
NAME	DECIMAL	(HEX)
*ENTRPUTL	724	(2D4)
*JAABORT	708	(2C4)
*JABORT	704	(2C0)
*JADINCM	248	(F8)
*JADOUTCM	252	(FC)
JALGN	309	(135)
JALOGIC	309	(135)
JBEGCL	752	(2F0)
*JBOS	368	(170)
JCALLS	309	(135)
*JCLVLPTR	376	(178)
JCOMEND	1272	(4F8)
JCOMMON	0	(0)
JCONTCL	720	(2D0)
JCTBGN	768	(300)
JCTCHR	756	(2F4)
JCTLN	772	(304)
JDBLALL	321	(141)
JDECK	308	(134)
JDECKID	289	(121)
JDECKIDL	288	(120)
JDPASS	788	(314)
JDUMPX0	318	(13E)
JDUMPX1	318	(13E)
JDUMPX2	318	(13E)
JDUMPX3	318	(13E)
JDUMPX4	318	(13E)
JDUMPX5	318	(13E)
JDUMPX6	318	(13E)
JDWORD	728	(2D8)
JENDCHK	316	(13C)
JENDCOL	722	(2D2)
*JENTRYPT	780	(30C)
*JEOS	372	(174)
JERRCHK	316	(13C)
JESD	308	(134)
JESDCHK	316	(13C)
JESDID	784	(310)
JESDOFLO	317	(13D)
*JFLEBLK1	72	(48)
*JFLEBLK2	128	(80)
*JFLEBLK3	184	(B8)
JFWORD1	736	(2E0)
JFWORD2	740	(2E4)
JHWORD1	744	(2E8)
JHWORD2	746	(2EA)
JIDR	1232	(4D0)
JINDERRF	317	(13D)
JINFILE	776	(308)
JINFLAG	319	(13F)
JINLIB	319	(13F)
*JINMLC	328	(148)
JINVOPT	317	(13D)
JIN2ND	319	(13F)
JLINK	308	(134)
JLIST	308	(134)
JLITLNG	848	(350)
JLNCT	298	(12A)

*POINTER

FIELD	DISPLACEMENT	
NAME	DECIMAL	(HEX)
JLNCTKEY	310	(136)
JLVTMDT	264	(108)
JMAXRL	246	(F6)
JMAXRL1	240	(F0)
JMAXRL2	242	(F2)
JMAXRL3	244	(F4)
JMINBUF	310	(136)
JMISLIN	317	(13D)
JMISPCH	317	(13D)
JMISPRT	317	(13D)
*JMLC	324	(144)
JMLCFLAG	321	(141)
JMLOGIC	309	(135)
JMSGL	297	(129)
JMSGLKEY	310	(136)
JNOSEQPH	320	(140)
JNOTEVAL	340	(154)
JNUM	310	(136)
JOUTFILE	778	(30A)
JOUTFLAG	320	(140)
*JOUTMLC	332	(14C)
JOUT2ND	320	(140)
*JPARM	308	(134)
*JPARMPTR	304	(130)
JPARMS	297	(129)
JPARM1	308	(134)
JPARM2	309	(135)
JPARM3	310	(136)
JPARM4	311	(137)
JPARM4	311	(137)
JPDFLAG	318	(13E)
*JPDUMP	336	(150)
JPHBLANK	262	(106)
*JPHNAME	256	(100)
JPHPREF	256	(100)
JPHSUFF	259	(103)
JPRTONLY	845	(34D)
JPT4GET	321	(141)
JPT4READ	321	(141)
JPT4STAR	321	(141)
JPT4WRIT	321	(141)
JRECCHK	316	(13C)
JRECIN	348	(15C)
JRECLIB	352	(160)
JRECPCH	356	(164)
JRECPRT	360	(168)
JREENTR	315	(13B)
JRENT	309	(135)
JRLD	308	(134)
JRLDCHK	316	(13C)
JSAFE	1160	(488)
JSAVE	0	(0)
JSAVETBL	384	(180)
JSEQCL	760	(2F8)
JSEQLN	764	(2FC)
JSEVER	844	(34C)
*JSLEN	364	(16C)
JSRCLN	748	(2EC)

*POINTER

FIELD NAME	DISPLACEMENT DECIMAL	 (HEX)
JSTMT	310	(136)
*JSYSCLOS	716	(2CC)
JSYSDATE	280	(118)
JSYSGEN	846	(34E)
JSYSMAC	309	(135)
*JSYSOPEN	712	(2C8)
*JSYSPARM	300	(12C)
JSYSTIME	274	(112)
JTBLTRT	850	(352)
JTERM	310	(136)
JTEST	308	(134)
JTRTABLE	901	(385)
JWARNFLG	315	(13B)
JXREF	308	(134)
JXREFCHK	316	(13C)
JYCON	315	(13B)

*POINTER

DSECT NAME: **JERRCD**

LOAD MODULE: IFOX02

SIZE: 12-92

CREATED BY: IFNX1A,IFNX3A

REFERENCED BY: IFNX5A

UPDATED BY: IFNX5A

FUNCTION: DEFINES THE INPUT RECORD FORMAT IN PHASE 5

OPERATIONS DIAGRAMS:

DISPLMNT DEC	 (HEX)	 SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
0	(0000)	0	JERECL	LENGTH
2	(0002)	2	JEFLGA	PROGRAM SWITCH - FLAG A
			JEPSOP	BIT 0 -
3	(0003)	1	JEFLGB	PROGRAM SWITCH - FLAG B
			JEPRPOS	BIT 4 - ERR MSG PRINT POSITION
4	(0004)	0	VJEOPCOD	X'37' INTERNAL OPCODE FOR ERROR RECORD
5	(0005)	1	JECOLPTR	POINTER
6	(0006)	1	JESTMTNO	STATEMENT NUMBER
9	(0009)	3	JEERCOD	ERROR AND SEVERITY CODE
9	(0009)	0	JESEV	SEVERITY CODE
10	(000A)	1	JERCDE	ERROR CODE
11	(000B)	1	JENODATA	NO. OF 8 BYTE DATA ITEMS
12	(000C)	0-80	JEDATA	DATA ITEMS MAXIMUM OF 10 EXACTLY 8 BYTES EACH

FIELD NAME	DISPLACEMENT DECIMAL	 (HEX)
JECOLPTR	5	(5)
JEERCOD	9	(9)
JEFLGA	2	(2)
JEFLGB	3	(3)
JENODATA	11	(B)
JEPRPOS	3	(3)
JEPSOP	2	(2)
JERCDE	10	(A)
JERECL	0	(0)
JESEV	9	(9)
JESTMTNO	6	(6)
VJEEOF	4	(4)
VJEOPCOD	4	(4)

*POINTER

DSECT NAME: **JFLEBLK**

LOAD MODULE: IFOX00

SIZE: 47

CREATED BY: IFOX0C

REFERENCED BY: IFOX0A,IFOX0B,IFOX0C,IFOX0D

UPDATED BY: IFOX0B

FUNCTION: HOLDS UTILITY FILE INFORMATION

OPERATIONS DIAGRAMS:

DISPLMNT DEC	(HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
0	(0000)	20	JDECB	EVENT CONTROL BLOCK
20	(0014)	4	JTCLOSE	TCLOSE PARM LIST
24	(0018)	4	JFLE	ADDR OF FILE DEFINITION
28	(001C)	4	JBUFFER	ADDR OF ALTERNATE BUFFER
32	(0020)	4	JBUF	ADDR OF BUFFER
36	(0024)	2	JRL	RECORD LENGTH
38	(0026)	2	JBUFNDX	BUFFER DISPLACEMENT (INDEX)
40	(0028)	1	JIOFLAG	PROGRAM SWITCH
			JPUTLPND	BIT 0 - PUTL PENDING
			JGETLPND	BIT 1 - GETL PENDING
			JGETLPNT	BIT 2 - GETL TO FOLLOW POINT
			JGETLSBF	BIT 3 - POINT (GETL) WITHIN SAME BUFFER
			JNOTED	BIT 4 - NOTE VALUE OF LAST RECORD NOTED
			JDBLBUF	BIT 5 - OUTPUT IS DOUBLE BUFFERED
			JCHKFILE	BIT 6 - FILE NEEDS TO BE CHECKED
41	(0029)	6	JLSTNOTE	NOTE VALUE OF LAST RECORD NOTED

FIELD NAME	DISPLACEMENT DECIMAL	(HEX)
*JBUF	32	(20)
*JBUFFER	28	(1C)
JBUFNDX	38	(26)
JCHKFILE	40	(28)
JDBLBUF	40	(28)
JDECB	0	(0)
*JFLE	24	(18)
JGETLPND	40	(28)
JGETLPNT	40	(28)
JGETLSBF	40	(28)
JIOFLAG	40	(28)
JLSTNOTE	41	(29)
JNOTED	40	(28)
JPUTLPND	40	(28)
JRL	36	(24)
*JTCLOSE	20	(14)

*POINTER

DSECT NAME: **JINCOM**

LOAD MODULE: IFOX04

SIZE: 48

CREATED BY: IFOX0E

REFERENCED BY: IFOX0E,IFOX0F,IFOX0I

UPDATED BY:

FUNCTION: HOLDS INPUT FILE INFORMATION

OPERATIONS DIAGRAMS:

DISPLMNT DEC	(HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
0	(0000)	4	JSYSIN	ADDR OF FILE DEF FOR INPUT
4	(0004)	4	JSYSLIB	ADDR OF FILE DEF FOR LIBRARY
8	(0008)	4	JINOPEN	ADDR OF OPEN PARM LIST
12	(000C)	4	JINCLOS	ADDR OF CLOSE PARM LIST
16	(0010)	20	JLIBDECB	EVENT CONTROL BLOCK
36	(0024)	4	JLIBBUF	ADDR OF LIBRARY BUFFER
40	(0028)	2	JBLKSIZE	BLOCK SIZE OF CURRENT LIB REC
42	(002A)	2	JLIBNDX	BUFFER INDEX INTO LIB BUFFER
44	(002C)	1	JINSW	PROGRAM SWITCH
			JREADPT	BIT 0 - SPECIAL READ FOR POINT
48	(0030)	0	JINCMEND	END OF INPUT COMMON

DSECT NAME: **JOUTCOM**

LOAD MODULE: IFOX05

SIZE: 37

CREATED BY: IFOX0G

REFERENCED BY: IFOX0G,IFOX0H,IFOX0I

UPDATED BY: IFOX0H

FUNCTION: HOLDS OUTPUT FILE INFORMATION

OPERATIONS DIAGRAMS:

DISPLMNT DEC	 (HEX)	 SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
0	(0000)	0	JSYSLST	ADDR OF FILE DEF FOR PRINT FILE
4	(0004)	4	JSYSPCH	ADDR OF FILE DEF FOR PUNCH FILE
8	(0008)	4	JSYSLNK	ADDR OF FILE DEF FOR LINK FILE
		4	JSYSTPM	ADDR OF FILE DEF FOR TERM FILE
12	(000C)	4	JSYSTRM	ADDR OF FILE DEF FOR TERM FILE
16	(0010)	4	JOUTOPEN	ADDR OF OPEN PARM LIST
20	(0014)	4	JOUTCLOS	ADDR OF CLOSE PARM LIST
24	(0018)	4	JCURPRT	ADDR OF CURRENT PRINT BUFFER
28	(001C)	4	JCURTAM	ADDR OF CURRENT TERM PRINT BUFFER
30	(001E)	2	JCURPCH	ADDR OF CURRENT PUNCH BUFFER
34	(0022)	2	JDECKSEQ	DECK SEQUENCE NUMBER
36	(0024)	1	JOUTSW	PROGRAM SWITCH
			BYPASPRT	BIT 0 - 1ST PRINT SWITCH
			BYPASPCH	BIT 1 - 1ST PUNCH SWITCH
			CLOSPRT	BIT 2 - FINAL PRINT SWITCH
			CLOSPCH	BIT 3 - FINAL PUNCH SWITCH
			NOSEQ	BIT 4 - DON'T SEQ PUNCHED OUTPUT
			BYPASTRM	BIT 5 - FIRST TERM PRINT SWITCH
			CLOSTRM	BIT 6 - FINAL TERM PRINT SWITCH
37	(0025)	0	JOUTCMD	

FIELD NAME	DISPLACEMENT DECIMAL	 (HEX)
BYPASPCH	30	(1E)
BYPASPRT	30	(1E)
CLOSPCH	30	(1E)
CLOSPRT	30	(1E)
*JCURPCH	24	(18)
*JCURPRT	20	(14)
JDECKSEQ	28	(1C)
*JOUTCLOS	16	(10)
JOUTCMND	32	(20)
*JOUTOPEN	12	(C)
JOUTSW	30	(1E)
*JSYSLNK	8	(8)
*JSYSLST	0	(0)
*JSYSPCH	4	(4)
NOSEQ	30	(1E)

*POINTER

DSECT NAME: **JTEXT**

LOAD MODULE: IFOX11

SIZE: 19

CREATED BY: IFNX1A

REFERENCED BY: IFNX1A,IFNX1J,IFNX2A,IFNX3A,IFNX3B,IFNX3N,IFNX4D,
IFNX4E,IFNX4M,IFNX4N,IFNX4S,IFNX4T,IFNX5C,IFNX5F

UPDATED BY:

FUNCTION: EDITED TEXT RECORD FIXED PART

OPERATIONS DIAGRAMS: MOST

DISPLMNT			FIELD	DESCRIPTION: CONTENTS,
DEC	(HEX)	SIZE	NAME	MEANING/USE
0	(0000)	2	JTRLI	RECORD LENGTH INDICATOR
2	(0002)	1	JTFLGA	PROGRAM SWITCH
			JPSOP	BIT 0 - PSEUDO-OP FLAG
			JEXTB	BIT 1 - EXTENDED OPCODE FLAG
			JINPC	BIT 2 - INITIALIZE PRIVATE CODE
			JINHB	BIT 3 - INHIBIT BIT
			JDEF	BIT 4 - DEFINITION RECORD
			JREF	BIT 5 - SCAN FOR SYMBOL REFERENCES
			JREQOP	BIT 6 - OPERAND REQUIRED FOR INTERLUDE
			JDCSX	BIT 7 - ON FOR DC, DS, AND DXD ONLY
2	(0002)	1	JTFLGA1	PROGRAM SWITCH
			JPRESD	BIT 5 - ON FOR PRE-ESD PUNCH & REPRO
			JLN4	BIT 6 - INSTRUCTION LENGTH
			JLN2	BIT 7 - INSTRUCTION LENGTH
3	(0003)	1	JTFLGB	PROGRAM SWITCH
			JPRONLY	BIT 0 - PRINT ONLY
			JERR	BIT 1 - DEAD STATEMENT
			JNOCNT	BIT 2 - DO NOT ASSIGN STATEMENT NUMBER
			JGEN	BIT 3 - STATEMENT IS GENERATED
			JNMERR	BIT 4 - INVALID NAME FIELD
			JSUBNAME	BIT 5 - SUBSTITUTION REQUIRED-NAME
			JSUBOPCD	BIT 6 - SUBSTITUTION REQUIRED-OPCODE
			JSUBOPND	BIT 7 - SUBSTITUTION REQUIRED-OPERAND
4	(0004)	1	JTIOP	INTERNAL OP CODES, 1ST BYTE OF OPCODE
4	(0004)	0	JTIOP1	SEE APPENDIX C. INTERNAL OPERATION CODES
4	(0004)	0	VJTICTL	00
4	(0004)	0	VJTISEQ	01
4	(0004)	0	VJTOPSYN	02
4	(0004)	0	VJTCOPY	03
4	(0004)	0	VJTANOP	04
4	(0004)	0	VJTGBLA	05
4	(0004)	0	VJTGBLB	06
4	(0004)	0	VJTGBLC	07
4	(0004)	0	VJTLCLA	08
4	(0004)	0	VJTLCLB	09
4	(0004)	0	VJTLCLC	0A

DISPLMNT DEC	 (HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
4	(0004)	0	VJTMACRO	0B
4	(0004)	0	VLOGENOP	0B
4	(0004)	0	VJTACTR	0C
4	(0004)	0	VJTAGO	0D
4	(0004)	0	VJTAGOB	0D
4	(0004)	0	VJTAIF	0E
4	(0004)	0	VJTAIFB	0E
4	(0004)	0	VJTSETA	0F
4	(0004)	0	VJTSETB	10
4	(0004)	0	VJTSETC	11
4	(0004)	0	VJTMEXIT	12
4	(0004)	0	VJTMEND	13
4	(0004)	0	VJTCALL	14
4	(0004)	0	VJTCPKEY	15
4	(0004)	0	VJTCPPOS	16
4	(0004)	0	VJTPROTO	17
4	(0004)	0	VJTPREP	17
4	(0004)	0	VJTPPKEY	18
4	(0004)	0	VJTPPCH	18
4	(0004)	0	VJTPPPOS	19
4	(0004)	0	VJTINPC	19
4	(0004)	0	VJTPEND	1A
4	(0004)	0	VJTPMOP	1A
4	(0004)	0	VJTEND	1B
4	(0004)	0	VHIGENOP	1B
4	(0004)	0	VLOREFOP	1B
4	(0004)	0	VLODEFOP	1B
4	(0004)	0	VJTDXD	1C
4	(0004)	0	VJTEQU	1D
4	(0004)	0	VJTORG	1E
4	(0004)	0	VJTCNOP	1F
4	(0004)	0	VJTCCW	20
4	(0004)	0	VJTDC	21
4	(0004)	0	VJTDS	22
4	(0004)	0	VJTSTART	23
4	(0004)	0	VHIREFOP	23
4	(0004)	0	VJTCSECT	24
4	(0004)	0	VJTDSECT	25
4	(0004)	0	VJTCOM	26
4	(0004)	0	VJTENTRY	27
4	(0004)	0	VJTLTLC	27
4	(0004)	0	VJTEXTRN	28
4	(0004)	0	VJTLTDC	28
4	(0004)	0	VJTWXTRN	29
4	(0004)	0	VJTLTND	29
4	(0004)	0	VJTCXD	2A
4	(0004)	0	VJTLTORG	2B
4	(0004)	0	VHIDEFOP	2B
4	(0004)	0	VJTLITR	2C
4	(0004)	0	VJTSYMBL	2D
4	(0004)	0	VJTPUNCH	2E
4	(0004)	0	VJTEOFII	2E
4	(0004)	0	VJTREPRO	2F
4	(0004)	0	VJTLITII	2F
4	(0004)	0	VJTPUSH	30
4	(0004)	0	VJTLTEND	30
4	(0004)	0	VJTPOP	31
4	(0004)	0	VJTADJII	31
4	(0004)	0	VJTPRINT	32
4	(0004)	0	VJTPASS	32
4	(0004)	0	VJTUSING	33

DISPLMNT DEC (HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
4 (0004)	0	VJTSYMII	33
4 (0004)	0	VJTDROP	34
4 (0004)	0	VJTCMNT	35
4 (0004)	0	VJTHCMNT	36
4 (0004)	0	VJTERROR	37
4 (0004)	0	VJTSPACE	38
4 (0004)	0	VLONOPRN	38
4 (0004)	0	VJTEJECT	39
4 (0004)	0	VJTTITLE	3A
4 (0004)	0	VJTMNOTE	3B
4 (0004)	0	VJTSICTL	3C
4 (0004)	0	VJTEEOF	FE
4 (0004)	0	VJTEOF	FF
4 (0004)	0		
5 (0005)	1	JTIOP2	SECOND BYTE OF OPCODE
6 (0006)	1	JTNMP	NAME FIELD POINTER
8 (0008)	2	JTOCP	OPCODE POINTER FIELD
10 (000A)	2	JTOPP	OPERAND POINTER FIELD
12 (000C)	2	JTCPR	COMMENTS POINTER FIELD
14 (000E)	2	JTSPR	STRING POINTER FIELD
16 (0010)	2	JTSYMCNT	NUMBER OF SYMBOLS IN OPERAND
17 (0011)	2		ZEROS, POINT TO HERE IF FIELD NOT PRESENT

FIELD NAME	DISPLACEMENT DECIMAL	(HEX)
JDCSX	2	(2)
JDEF	2	(2)
JERR	3	(3)
JEXTB	2	(2)
JGEN	3	(3)
JINHB	2	(2)
JINPC	2	(2)
JLN2	2	(2)
JLN4	2	(2)
JNMERR	3	(3)
JNOCNT	3	(3)
JPRESD	2	(2)
JPRONLY	3	(3)
JPSOP	2	(2)
JREF	2	(2)
JREQOP	2	(2)
JSUBNAME	3	(3)
JSUBOPCD	3	(3)
JSUBOPND	3	(3)
JTCPR	12	(C)
JTFLGA	2	(2)
JTFLGA1	2	(2)
JTFLGB	3	(3)
JTIOP	4	(4)
JTIOP1	4	(4)
JTIOP2	5	(5)
JTNMP	6	(6)
JTOCP	8	(8)
JTOPP	10	(A)
JTRLI	0	(0)
JTSPR	14	(E)
JTSYMCNT	16	(10)
VHIDEFOP	4	(4)
VHIGENOP	4	(4)
VHIREFOP	4	(4)
VJTACTR	4	(4)
VJTADJII	4	(4)
VJTAGO	4	(4)
VJTAGOB	4	(4)
VJTAIF	4	(4)
VJTAIFB	4	(4)
VJTANOP	4	(4)
VJTCALL	4	(4)
VJTCCW	4	(4)
VJTCMNT	4	(4)
VJTCNOP	4	(4)
VJTCOM	4	(4)
VJTCOPY	4	(4)
VJTCPKEY	4	(4)
VJTCPPOS	4	(4)
VJTCSECT	4	(4)
VJTCXD	4	(4)
VJTDC	4	(4)
VJTDROP	4	(4)
VJTDS	4	(4)
VJTDSECT	4	(4)
VJTDXD	4	(4)
VJTEEOF	4	(4)
VJTEJECT	4	(4)

*POINTER

FIELD NAME	DISPLACEMENT DECIMAL	(HEX)
VJTEND	4	(4)
VJTENTRY	4	(4)
VJTEOF	4	(4)
VJTEOFII	4	(4)
VJTEQU	4	(4)
VJTERROR	4	(4)
VJTEXTRN	4	(4)
VJTGBLA	4	(4)
VJTGBLB	4	(4)
VJTGBLC	4	(4)
VJTHCMNT	4	(4)
VJTICTL	4	(4)
VJTINPC	4	(4)
VJTISEQ	4	(4)
VJTLCLA	4	(4)
VJTLCLB	4	(4)
VJTLCLC	4	(4)
VJTLITII	4	(4)
VJTLITR	4	(4)
VJTLTDC	4	(4)
VJTLTEND	4	(4)
VJTLTLC	4	(4)
VJTLTND	4	(4)
VJTLTORG	4	(4)
VJTMACRO	4	(4)
VJTMEND	4	(4)
VJTMEXIT	4	(4)
VJTMNOTE	4	(4)
VJTOPSYN	4	(4)
VJTORG	4	(4)
VJTPASS	4	(4)
VJTPEND	4	(4)
VJTPMOP	4	(4)
VJTPOP	4	(4)
VJTPPCH	4	(4)
VJTPPKEY	4	(4)
VJTPPPOS	4	(4)
VJTPREP	4	(4)
VJTPRINT	4	(4)
VJTPROTO	4	(4)
VJTPUNCH	4	(4)
VJTPUSH	4	(4)
VJTREPRO	4	(4)
VJTSETA	4	(4)
VJTSETB	4	(4)
VJTSETC	4	(4)
VJTSICTL	4	(4)
VJTSPACE	4	(4)
VJTSTART	4	(4)
VJTSYMBL	4	(4)
VJTSYMII	4	(4)
VJTTITLE	4	(4)
VJTUSING	4	(4)
VJTWXTRN	4	(4)
VLODEFOP	4	(4)
VLOGENOP	4	(4)
VLONOPRN	4	(4)
VLOREFOP	4	(4)

*POINTER

DSECT NAME: **JTEXTA**

LOAD MODULE: IFOX11

SIZE: VARIABLE

CREATED BY: IFNX1J

REFERENCED BY: IFNX1J,IFNX2A,IFNX3A,IFNX3B,IFNX3N,IFNX4D,IFNX4E,IFNX4M,
IFNX4N,IFNX4S,IFNX4T,IFNX4V,IFNX5C,IFNX5M,IFNX5P

UPDATED BY:

FUNCTION: EDITED TEXT RECORD, VARIABLE PART

OPERATIONS DIAGRAMS: MOST

DISPLMNT			FIELD	DESCRIPTION: CONTENTS,
DEC	(HEX)	SIZE	NAME	MEANING/USE
0	(0000)	1	JTNMO	PROGRAM SWITCH
			JTNMOCD	BIT 0 - NAME POINTER--REAL PTR FOLLOWS
1	(0001)	1	JTNML	LENGTH OF NAME FIELD
0	(0000)	1	JTOCO	PROGRAM SWITCH
			JTOCOCD	BIT 0 - OPCODE POINTER REAL POINTER
1	(0001)	1	JTOCL	OP CODE LENGTH
0	(0000)	1	JTOPO	PROGRAM SWITCH
			JTOPOCD	BIT 0 - POINTER--REAL JTOPO FOLLOWS
1	(0001)	1	JTOPL	OPERAND LENGTH
0	(0000)	1	JTCOP	COMMENT OUTPUT POINTER
1	(0001)	1	JTCML	COMMENT LENGTH
0	(0000)	1	JTSTC	STRING COUNT
1	(0001)	1	JTSTO	STRING 1 OUTPUT COLUMN POINTER
2	(0002)	1	JTSTL	STRING 1 LENGTH
0	(0000)	1	JTSTO2	STRING 2 OUTPUT COLUMN POINTER
1	(0001)	1	JTSTL2	STRING 2 LENGTH

FIELD	DISPLACEMENT	
NAME	DECIMAL	(HEX)
JTCML	1	(1)
JTCOP	0	(0)
JTNML	1	(1)
JTNMO	0	(0)
JTNMOCD	0	(0)
JTOCL	1	(1)
JTOCO	0	(0)
JTOCOCD	0	(0)
JTOPL	1	(1)
JTOPO	0	(0)
JTOPOCD	0	(0)
JTSTC	0	(0)
JTSTL	2	(2)
JTSTL2	1	(1)
JTSTO	1	(1)
JTSTO2	0	(0)

*POINTER

DSECT NAME: **LCLNTRY**

LOAD MODULE: IFOX11

SIZE: 13-19

CREATED BY: IFNX1J

REFERENCED BY: IFNX1J

UPDATED BY:

FUNCTION: LOCAL VARIABLE ENTRY IN VARIABLE SYMBOL DEFINITION DIRECTORY

OPERATIONS DIAGRAMS: 5

DISPLMNT DEC	(HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE	
0	(0000)	3	LCHAIN	CHAIN POINTER	3 BYTES
3	(0003)	1	LFLAGS	FLAGS (SEE VSDENTRY)	1 BYTE
4	(0004)	1	LLNGTH	SYMBOL LENGTH	1 BYTE
5	(0005)	2-8	LSYMBL	VARIABLE SYMBOL	
0	(0000)	1	LTFVAL	META TEXT FLAG VALUE	1 BYTE
1	(0001)	3	LDICTR	LOCAL DICTIONARY PTR	3 BYTES
4	(0004)	2	LDIMEN	DIMENSION	2 BYTES

DSECT NAME: **MDDNTRY**

LOAD MODULE: IFOX11

SIZE: 40

CREATED BY: IFNX1J

REFERENCED BY: IFNX1J,IFNX2A

UPDATED BY:

FUNCTION: MACRO DEFINITION DIRECTORY ENTRY

OPERATIONS DIAGRAMS: 3, 6, 8, 10

DISPLMNT DEC	 (HEX)	 SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE	
0	(0000)	3	MCHAIN	CHAIN PTR	3 BYTES
3	(0003)	1	MFLAGS	PROGRAM SWITCH	
			TSEDIT	BIT 0 - ON-SEGMENT EDITED	
			OCTS	BIT 1 - ON-OPEN CODE ENTRY	
			FLUSH	BIT 2 - ON-MACRO FLUSHED	
			DELETE	BIT 3 - ON-MACRO DELETED VIA OPSYN	
			MNL1	BIT 5 - MACRO NAME LENGTH	
			MNL2	BIT 6 -	
			MNL3	BIT 7 -	
4	(0004)	8	MSYMBL	SYMBOL (PADDED)	8 BYTES
12	(000C)	3	MVECTR	VECTOR POINTER	3 BYTES
15	(000F)	8	MTXTNP	TEXT FILE N/P	8 BYTES
23	(0017)	8	MTSDNP	DICT FILE N/P	8 BYTES
31	(001F)	3	MGBLSZ	GBL VCTR SIZE	3 BYTES
34	(0022)	3	MSEQSZ	SEQ SYM DICT SIZE	3 BYTES
37	(0025)	3	MLCLSZ	LCL DICT SIZE	3 BYTES

FIELD NAME	DISPLACEMENT DECIMAL	 (HEX)
DELETE	3	(3)
FLUSH	3	(3)
MCHAIN	0	(0)
MFLAGS	3	(3)
MGBLSZ	31	(1F)
MLCLSZ	37	(25)
MNL1	3	(3)
MSEQSZ	34	(22)
MSYMBL	4	(4)
MTSDNP	23	(17)
MTXTNP	15	(F)
MVECTR	12	(C)
OCTS	3	(3)
TSEDIT	3	(3)

*POINTER

DSECT NAME: **MDVNTRY**

LOAD MODULE: IFOX21

SIZE: 19

CREATED BY: IFNX3N

REFERENCED BY: IFNX2A,IFNX3N

UPDATED BY:

FUNCTION: MAPS THE MACRO DEFINITION VECTOR ENTRY

OPERATIONS DIAGRAMS: 8, 10

DISPLMNT DEC (HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
0 (0000)	8	MNPTXT	TEXT FILE N/P 8 BYTES
8 (0008)	8	MNPSD	SKEL DICTION N/P 8 BYTES
16 (0010)	3	MSDL	SKEL DICT LENGTH 3 BYTES

DSECT NAME: **OPNTRY**

LOAD MODULE: IFOX11

SIZE: 3

CREATED BY: IFNX1K

REFERENCED BY: IFNX1J,IFNX3A

UPDATED BY:

FUNCTION: OPCODE TABLE ENTRY MAP

OPERATIONS DIAGRAMS: 3, 13

DISPLMNT DEC	 (HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE	
0	(0000)	2	OCHAIN	CHAIN POINTER	2 BYTES
2	(0002)	1	OFLAGS	FLAGS	1 BYTE
0	(0000)	0	OMNEM	MNEMONIC	1 BYTE
0	(0000)	1	OFLAGA	SWITCH CODES	1 BYTE
1	(0001)	1	OINTCD	INTERNAL OPCOD	1 BYTE
2	(0002)	1	OINTCD2	INTERNAL OPCODE BYTE	
3	(0003)	1	OMASK	MASK, EXT MNEMS	1 BYTE

DSECT NAME: **OPSTBL**

LOAD MODULE: IFOX21

SIZE: 13

CREATED BY: IFNX2A

REFERENCED BY: IFNX3A

UPDATED BY:

FUNCTION: OPSYN TABLE ENTRY

OPERATIONS DIAGRAMS: 8, 13

DISPLMNT DEC	 (HEX)	 SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
0	(0000)	1	OPSFLGS	FLAGS
			BIT 5	DELETED OPSYN ENTRY
1	(0001)	3	OPSTATTS	ATTRIBUTES
4	(0004)	1	OPSTNL	NAME LENGTH
5	(0005)	8	OPSTNAM	NAME

DSECT NAME: **OPSYNTRY**

LOAD MODULE: IFOX11

SIZE: 16

CREATED BY: IFNX1J

REFERENCED BY: IFNX1J,IFNX2A

UPDATED BY:

FUNCTION: OPSYN TABLE ENTRY

OPERATIONS DIAGRAMS: 3, 4, 8

DISPLMNT DEC (HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
0 (0000)	3	OPSYNCH	CHAIN POINTER 3 BYTES
3 (0003)	1	OPSYNFLG	PROGRAM SWITCH
		ODEL	BIT 5 - DELETED OPSYN ENTRY
		OPREV	BIT 6 - PREVIOUS OPSYN ENTRY
4 (0004)	3	OPSYNATT	ATTRIBUTES
7 (0007)	1	ONAMEL	NAME LENGTH
8 (0008)	8	ONAME	NAME

FIELD NAME	DISPLACEMENT DECIMAL	 (HEX)
ODEL	3	(3)
OMAC	3	(3)
ONAME	8	(8)
ONAMEL	7	(7)
OPREV	3	(3)
OPSYNCH	0	(0)
OPSYNCHN	4	(4)
OPSYNFLG	3	(3)

*POINTER

DSECT NAME: **OSDIR**

LOAD MODULE: IFOX11

SIZE: 14

CREATED BY: IFNX1J

REFERENCED BY: IFNX1J

UPDATED BY:

FUNCTION: ORDINARY SYMBOL ATTRIBUTE REFERENCE DIRECTORY ENTRY
(IN-CORE WORK TABLE; MAXIMUM 10 ENTRIES)

OPERATIONS DIAGRAMS: 5

DISPLMNT DEC	 (HEX)	 SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
0	(0000)	8	OSSYM	ORD SYMB (PADDED) 8 BYTES
8	(0008)	1	OTFVAL	TEXT FLAG VALUE 1 BYTE
9	(0009)	3	OSRDP	DICT POINTER 3 BYTES
12	(000C)	2	OSPAD	PADDING 2 BYTES

DSECT NAME: **OSRDNTRY**

LOAD MODULE: IFOX11

SIZE: 6

CREATED BY: IFNX2A

REFERENCED BY: IFNX3A

UPDATED BY:

FUNCTION: ORDINARY SYMBOL ATTRIBUTE REFERENCE DICTIONARY ENTRY

OPERATIONS DIAGRAMS: 8, 9

DISPLMNT DEC (HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
0 (0000)	1	TATTRIB	TYPE ATTRIBUTE 1 BYTE
1 (0001)	2	LATTRIB	LENGTH ATTRIBUTE 2 BYTES
3 (0003)	2	SATTRIB	SCALE ATTRIBUTE 2 BYTES
5 (0005)	1	ATTRIB	PROGRAM SWITCH
		TDEFALT	BIT 0 - TYPE ATTRIB IS DEFAULT VALUE
		LDEFALT	BIT 1 - LENGTH ATTRIB IS DEFAULT VALUE
		SDEFALT	BIT 2 - SCALE ATTRIB IS DEFAULT VALE
		UDEFALT	BIT 3 - UNDEFINED SYMBOL ATTRIBUTE REFERENCES

DSECT NAME: **OSREF**

LOAD MODULE: IFOX11

SIZE: 8-15

CREATED BY: IFNX1J

REFERENCED BY: IFNX1J,IFNX2A

UPDATED BY:

FUNCTION: MAPS ORDINARY SYMBOL ATTRIBUTE FOR TEXT SEGMENT DICTIONARY FILE

OPERATIONS DIAGRAMS: 5

DISPLMNT			FIELD	DESCRIPTION: CONTENTS,	
DEC	(HEX)	SIZE	NAME	MEANING/USE	
0	(0000)	2	OREFRL	RECORD LENGTH	2 BYTES
2	(0002)	1	OREFRT	RECORD TYPE	1 BYTE
3	(0003)	3	OREFDP	DICTIONARY PTR	3 BYTES
6	(0006)	1	OREFSL	SYMBOL LENGTH	1 BYTE
7	(0007)	1-8	OREFOS	ORDINARY SYMBOL	1-8 BYTES

DSECT NAME: **OSRTNTRY**

LOAD MODULE: IFOX21

SIZE: 8-15

CREATED BY: IFNX2A

REFERENCED BY: IFNX2A

UPDATED BY:

FUNCTION: ORDINARY SYMBOL ATTRIBUTE REFERENCE TABLE ENTRY
(IN-CORE WORK TABLE)

OPERATIONS DIAGRAMS:

DISPLMNT			FIELD	DESCRIPTION: CONTENTS,	
DEC	(HEX)	SIZE	NAME	MEANING/USE	
0	(0000)	3	OSRTCP	CHAIN POINTER	3 BYTES
3	(0003)	3	OSRTDP	DICTIONARY PTR	3 BYTES
6	(0006)	1	OSRTSL	SYMBOL LENGTH	1 BYTE
7	(0007)	1-8	OSRTOS	ORDINARY SYMBOL	

DSECT NAME: **P**

LOAD MODULE: IFOX02

SIZE: 98

CREATED BY: IFOX0D

REFERENCED BY: IFOX0D,IFOX0F,IFOX0I

UPDATED BY:

FUNCTION: HOLDS ANY ALTERNATE DD NAMES USED

OPERATIONS DIAGRAMS: 27

DISPLMNT DEC	 (HEX)	 SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
0	(0000)	2	PLEN	LENGTH OF LIST
2	(0002)	24		(NOT APPLICABLE)
26	(001A)	8	PSYSLIB	SYSLIB DDNAME
34	(0022)	8	PSYSIN	SYSIN DDNAME
42	(002A)	8	PSYSPRIN	SYSPRINT DDNAME
50	(0032)	8	PSYSPUNC	SYSPUNCH DDNAME
58	(003A)	8	PSYSUT1	SYSUT1 DDNAME
66	(0042)	8	PSYSUT2	SYSUT2 DDNAME
74	(004A)	8	PSYSUT3	SYSUT3 DDNAME
82	(0052)	8	PSYSGO	SYSGO DDNAME
90	(005A)	8	PSYSTERM	

FIELD NAME	DISPLACEMENT DECIMAL	 (HEX)
PLEN	0	(0)
PSYSGO	82	(52)
PSYSIN	34	(22)
PSYSLIB	26	(1A)
PSYSPRIN	42	(2A)
PSYSPUNC	50	(32)
PSYSUT1	58	(3A)
PSYSUT2	66	(42)
PSYSUT3	74	(4A)

*POINTER

DSECT NAME: **PPIN**

LOAD MODULE: IFOX51

SIZE: 62

CREATED BY: IFNX5A, IFNX5M

REFERENCED BY: IFNX5A,IFNX5C,IFNX5M,IFNX5V,IFNX6A

UPDATED BY:

FUNCTION: RLD AND XREF WHEN SORTED

OPERATIONS DIAGRAMS: 26

DISPLMNT DEC	(HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
0	(0000)	2	PPRLI	RECORD LENGTH
2	(0002)	2	PPFLG	FLAGS
4	(0004)	2	PPIOC	INTERNAL OPCODE
0	(0000)	2	RLDLEN	RLD RECORD LENGTH
2	(0002)	2	RFLAG	FLAG
4	(0004)	2	ROPCDE	OPCODE BYTES
6	(0006)	2	POSID	POSITION ESD/ID
8	(0008)	2	RELID	RELOCATION ESD/ID
10	(000A)	3	RLDVAL	RLD SYMBOL ADDRESS
13	(000D)	3	RLDFLG	RLD FLAG
16	(0010)	3		FULL-WORD ALIGNMENT
0	(0000)	2	XRECLN	XREF RECORD LENGTH
2	(0002)	2	XFLAG	FLAG
4	(0004)	2	XOPCDE	OPCODE
6	(0006)	8	XRFSYM	XREF SYMBOL
14	(000E)	1	XRFFLG	XREF FLAG, BASE, DEF, DUP,UNDEF
15	(000F)	2	XRFSTM	XREF STATEMENT NUMBER
17	(0011)	2	XRFLEN	XREF LENGTH
19	(0013)	4	XRFVAL	XREF VALUE
6	(0006)	4	LITLOCTR	LITERAL LOCATION COUNTER
10	(000A)	2	LITESDID	LITERAL ESD ID
12	(000C)	7	LITPOLID	LITERAL POOL ID
19	(0013)	1	LITDTL	LITERAL DATA LENGTH

FIELD NAME	DISPLACEMENT DECIMAL	(HEX)
LITDTL	19	(13)
LITESDID	10	(A)
LITLOCTR	6	(6)
LITPOLID	12	(C)
POSID	6	(6)
PPFLG	2	(2)
PPIOC	4	(4)
PPRLI	0	(0)
RELID	8	(8)
RFLAG	2	(2)
RLDFLG	13	(D)
RLDLEN	0	(0)

FIELD NAME	DISPLACEMENT DECIMAL	(HEX)
RLDVAL	10	(A)
ROPCDE	4	(4)
XFLAG	2	(2)
XOPCDE	4	(4)
XRECLN	0	(0)
XRFFLG	14	(E)
XRFLEN	17	(11)
XRFSTM	15	(F)
XRFSYM	6	(6)
XRFVAL	19	(13)

*POINTER

DSECT NAME: **PRMNTRY**

LOAD MODULE: IFOX11

SIZE: 13-19

CREATED BY: IFNX1J

REFERENCED BY: IFNX1J

UPDATED BY:

FUNCTION: MACRO PARAMETER ENTRY IN VARIABLE SYMBOL
DIRECTORY (IN-CORE WORK TABLE)

OPERATIONS DIAGRAMS: 5

DISPLMNT DEC	(HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE	
0	(0000)	3	PCHAIN	CHAIN POINTER	3 BYTES
3	(0003)	1	PFLAGS	FLAGS	1 BYTE
4	(0004)	1	PLNGTH	SYMBOL LENGTH	1 BYTE
5	(0005)	2-8	PSYMBL	VARIABLE SYMBOL	
0	(0000)	1	PTFVAL	TEXT FLAG VALUE	1 BYTE
1	(0001)	3	PVECTR	VECTOR POINTER	3 BYTES
4	(0004)	2	PPAD	PADDING	2 BYTES

DSECT NAME: **RCARD**

LOAD MODULE: IFOX61

SIZE: 80

CREATED BY: IFNX6A

REFERENCED BY: IFNX6A

UPDATED BY:

FUNCTION: RLD DECK

OPERATIONS DIAGRAMS: 26

DISPLMNT DEC	 (HEX)	 SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
0	(0000)	1	CARDID	RLD CARD LAYOUT
1	(0001)	3	RLDNAM	RLD NAME
4	(0004)	6		
10	(000A)	2	RLDBYT	NUMBER OF BYTES IN DATA FIELD
12	(000C)	4		
16	(0010)	56	RLDFLD	RLD DATA FIELD (VARIABLE)
72	(0048)	4	DECKID	ID AND
76	(004C)	4	SEQNUM	SEQUENCE FIELD

DSECT NAME: **RLDIN**

LOAD MODULE: IFOX51

SIZE: 14

CREATED BY: IFNX5A,IFNX5M

REFERENCED BY: IFNX5C,IFNX5M,IFNX5V,IFNX6A

UPDATED BY:

FUNCTION: RLD RECORD LAYOUT

OPERATIONS DIAGRAMS: 21, 23

DISPLMNT DEC (HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
0 (0000)	2	RLDLEN	RLD RECORD LENGTH
2 (0002)	2	RFLAG	FLAG
4 (0004)	2	ROPCDE	OPCODE BYTES
6 (0006)	2	POSID	POSITION ESD/ID
8 (0008)	2	RELID	RELOCATION ESD/ID
10 (000A)	3	RLDVAL	RLD SYMBOL ADDRESS
13 (000D)	1	RLDFLG	RLD FLAG

DSECT NAME: **RPRINT**

LOAD MODULE: IFOX61

SIZE: 187

CREATED BY: IFNX6A

REFERENCED BY: IFNX6A

UPDATED BY: INFX6A

FUNCTION: OUTPUT RECORD FORMAT

OPERATIONS DIAGRAMS:

DISPLMNT			FIELD	DESCRIPTION: CONTENTS,
DEC	(HEX)	SIZE	NAME	MEANING/USE
0	(0000)	1	RCNTRL	RLD PRINT CONTROL BYTE
1	(0001)	1		
2	(0002)	4	POSOUT	POSITION ESD/ID
6	(0006)	5		
11	(000B)	4	RELOUT	RELOCATION ESD/ID
15	(000F)	6		
21	(0015)	2	FLGOUT	RLD FLAG
23	(0017)	5		
28	(001C)	6	VALOUT	RLD SYMBOL ADDRESS
34	(0022)	0		
0	(0000)	1	XCNTRL	XREF PRINT CONTROL BYTE
1	(0001)	8	XSYMOUT	XREF SYMBOL
9	(0009)	1		
10	(000A)	5	XLENOUT	LENGTH OF XREF
15	(000F)	1		
16	(0010)	8	XVALOUT	VALUE OF XREF
24	(0018)	1		
25	(0019)	5	XDEFOUT	ADDRESS WHERE XREF DEFINED
30	(001E)	2	XDE	
32	(0020)	0	XRFREF	REFERENCES TO SYMBOL
32	(0020)	5	XRFENT	XREF REFERENCE ENTRY
37	(0025)	5		SEPARATOR
0	(0000)	1	LCNTRL	LIT XREF CONTROL CHAR
0	(0000)	1	CONTROL	LIST CONTROL CHARACTER VALUES
1	(0000)	1	VEJBYTE	
2	(0000)	1	VSPACE1	
3	(0000)	1	VSPACE2	
4	(0000)	1	VSPACE3	
1	(0001)	4	TITLE	TITLE
5	(0005)	38		BLANKS
43	(002B)	8	LHDPTR	LIT XREF HEADING PTR
51	(0033)	21	HDGPTR	RLD OR XREF PAGE IDENTIFIER
72	(0048)	15		
97	(0061)	15	DTEPTR	DATE
112	(0070)	4	PGEPTR	PAGE
116	(0074)	5	PGENUM	PAGE NUMBER

FIELD NAME	DISPLACEMENT DECIMAL	(HEX)
CONTROL	0	(0)
DTEPTR	97	(61)
FLGOUT	21	(15)
LCNTRL	0	(0)
LHDPTR	43	(2B)
POSOUT	2	(2)
RCNTRL	0	(0)
RELOUT	11	(B)
TITLE	1	(1)
VALOUT	28	(1C)
VEJBYTE	0	(0)
VSPACE1	0	(0)
VSPACE2	0	(0)
VSPACE3	0	(0)
XCNTRL	0	(0)
XDE	30	(1E)
XDEFOUT	25	(19)
XLENOUT	10	(A)
XRFENT	32	(20)
XRFREF	32	(20)
XSYMOUT	1	(1)
XVALOUT	16	(10)

*POINTER

DSECT NAME: **RSYMRCD**

LOAD MODULE: IFOX31

SIZE: 22

CREATED BY: IFNX3B

REFERENCED BY: IFNX3A,IFNX3B,IFNX4D,IFNX4E,IFNX4M,IFNX4N,IFNX4V,
IFNX5A,IFNX5C,IFNX5D,IFNX5L,IFNX5M,IFNX5N

UPDATED BY:

FUNCTION: MAPS THE SYMBOL TABLES

OPERATIONS DIAGRAMS: 16, 17, 19

DISPLMNT			FIELD	DESCRIPTION: CONTENTS,
DEC	(HEX)	SIZE	NAME	MEANING/USE
0	(0000)	2	RRCDL	RECORD LENGTH
2	(0002)	1	RFLGA	PROGRAM SWITCH
			RPSOP	BIT 0 - PSEUDO OP
3	(0003)	1	RFLGB	PROGRAM SWITCH
			ENTRYSW1	BIT 3 - ENTRY ITEM PENDING
			ESDNRSW1	BIT 4 -
			CSECTSW1	BIT 5 -
			DSECTSW1	BIT 6 -
			DSCOMSW1	BIT 7 -
4	(0004)	1	RTYPE	RECORD TYPE. (SEE 'JTEXT')
5	(0005)	1	RFLDI	PROGRAM SWITCH
			ESDOFLO	BIT 0 -
			DEFINED	BIT 1 - SYMBOL DEFINED, NO ERRO
			PRIORDEF	BIT 2 - PREVIOUSLY DEFINED SYMB
			RFIELDN	BIT 3 - NAME FIELD APPENDED
			RFIELDX	BIT 4 - FIELD 'A' OR 'B' APPEND
6	(0006)	1	RSWTS	PROGRAM SWITCH
			DSW1	BIT 1 - PXD
			CSW1	BIT 2 - COM
6	(0006)	2	RESDI	ESDID ASSOCIATED WITH VALUE
8	(0008)	4	RLCTR	VALUE
12	(000C)	0	RLNGA	
12	(000C)	8	RNAME	SYMBOL
20	(0014)	2	RLNGQ	SYMBOL LENGTH ATTRIBUTE
22	(0016)	0	RLNGB	
0	(0000)	12	RITEM	
12	(000C)	12	RSYMC1	COMMON SEGMENT
6	(0006)	6	RSYMC2	COMMON SEGMENT
6	(0006)	6	RESDC	COMMON SEGMENT

FIELD NAME	DISPLACEMENT DECIMAL	(HEX)
CSECTSW1	3	(3)
CSW1	6	(6)
DEFINED	5	(5)
DSCOMSW1	3	(3)
DSECTSW1	3	(3)
DSW1	6	(6)
ENTRYSW1	3	(3)
ESDNRSW1	3	(3)
ESDOFLO	5	(5)
PRIORDEF	5	(5)
RESDC	6	(6)
RESDI	6	(6)
RFIELDN	5	(5)
RFIELDX	5	(5)
RFLDI	5	(5)
RFLGA	2	(2)
RFLGB	3	(3)
RITEM	0	(0)
RLCTR	8	(8)
RLNGA	12	(C)
RLNGB	22	(16)
RLNGQ	20	(14)
RNAME	12	(C)
RPSOP	2	(2)
RRCDL	0	(0)
RSWTS	6	(6)
RSYMC1	12	(C)
RSYMC2	6	(6)
RTYPE	4	(4)

*POINTER

DSECT NAME: **SKDCTHDR**

LOAD MODULE: IFOX11

SIZE: 33

CREATED BY: IFNX1S

REFERENCED BY: IFNX2A,IFNX3N

UPDATED BY:

FUNCTION: MAPS THE HEADER FOR THE SKELETON DICTIONARY

OPERATIONS DIAGRAMS: 8, 10

DISPLMNT DEC	(HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
0	(0000)	3	SKSRDPT	DISPL SEQ SYM DIC 3 BYTES
3	(0003)	3	SKLDADR	LCL DICT PTR 3 BYTES
6	(0006)	3	SKLDLNG	LCL DICT LENGTH 3 BYTES
9	(0009)	3	SKMPADR	MACRO PARAM PTR 3 BYTES
12	(000C)	3	SKKVADR	KEYWD VECTR PTR 3 BYTES
15	(000F)	3	SKADNLD	DICT ADR NXT LVL 4 BYTES
19	(0013)	8	SKNPNLT	TEXT N/P NXT LVL 8 BYTES
27	(001B)	4	SKACTRV	ACTR VALUE 4 BYTES
31	(001F)	2	SKNOFSLS	N'&SYSLIST 2 BYTES

FIELD NAME	DISPLACEMENT DECIMAL	(HEX)
SKACTRV	27	(1B)
SKADNLD	15	(F)
SKKVADR	12	(C)
SKLDADR	3	(3)
SKLDLNG	6	(6)
SKMPADR	9	(9)
SKNOFSLS	31	(1F)
SKNPNLT	19	(13)
SKSRDPT	0	(0)

*POINTER

DSECT NAME: **SSDEF**

LOAD MODULE: IFOX11

SIZE: 14-20

CREATED BY: IFNX1J

REFERENCED BY: IFNX1J,IFNX2A

UPDATED BY:

FUNCTION: MAPS SEQUENCE SYMBOL DEFINITION ENTRY FOR TEXT SEGMENT DICTIONARY FILE

OPERATIONS DIAGRAMS: 5

DISPLMNT DEC	(HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE	
0	(0000)	2	SDEFRL	RECORD LENGTH	2 BYTES
2	(0002)	1	SDEFRT	X'04' RECORD TYPE	1 BYTE
3	(0003)	8	SDEFNP	NOTE/POINT ADDR	8 BYTES
11	(000B)	1	SDEFSL	SYMBOL LENGTH	1 BYTE
12	(000C)	2-8	SDEFSS	SEQUENCE SYMBOL	2-8 BYTES

DSECT NAME: **SSDIR**

LOAD MODULE: IFOX11

SIZE: 14

CREATED BY: IFNX1J

REFERENCED BY: IFNX1J

UPDATED BY:

FUNCTION: SEQUENCE SYMBOL REFERENCE DIRECTORY ENTRY
(IN-CORE WORK TABLE; MAXIMUM 10 RECORDS)

OPERATIONS DIAGRAMS: 5

DISPLMNT DEC	 (HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE	
0	(0000)	8	SSSYM	SEQ SYMB (PADDED)	8 BYTES
8	(0008)	1	STFVAL	TEXT FLAG VALUE	1 BYTE
9	(0009)	3	SSRDP	DICT POINTER	3 BYTES
12	(000C)	2	.SSPAD	PADDING	2 BYTES

DSECT NAME: **SSDTNTRY**

LOAD MODULE: IFOX21

SIZE: 14-20

CREATED BY: IFNX2A

REFERENCED BY: IFNX2A

UPDATED BY:

FUNCTION: SEQUENCE SYMBOL DEFINITION TABLE (IN-CORE WORK TABLE)

OPERATIONS DIAGRAMS: 5

DISPLMNT DEC	(HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE	
0	(0000)	3	SSDTCP	CHAIN POINTER	3 BYTES
3	(0003)	8	SSDTNP	NOTE/POINT ADDR	8 BYTES
11	(000B)	1	SSDTSL	SYMBOL LENGTH	1 BYTE
12	(000C)	2-8	SSDTSY	SEQUENCE SYMBOL	

DSECT NAME: **SSREF**

LOAD MODULE: IFOX11

SIZE: 9-15

CREATED BY: IFNX1J

REFERENCED BY: IFNX1J,IFNX2A

UPDATED BY:

FUNCTION: MAPS SEQUENCE SYMBOL REFERENCE ENTRY FOR TEXT SEGMENT DICTIONARY FILE

OPERATIONS DIAGRAMS: 5

DISPLMNT DEC	(HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE	
0	(0000)	2	SREFRL	RECORD LENGTH	2 BYTES
2	(0002)	1	SREFRT	X'08' RECORD TYPE	1 BYTE
3	(0003)	3	SREFDP	DICTIONARY PTR	3 BYTES
6	(0006)	1	SREFSL	SYMBOL LENGTH	1 BYTE
7	(0007)	2-8	SREFSS	SEQUENCE SYMBOL	2-8 BYTES

DSECT NAME: **UDSECT**

LOAD MODULE: IFOX51

SIZE: 10

CREATED BY: IFNX5A

REFERENCED BY: IFNX5C,IFNX5F,IFNX5V

UPDATED BY: IFNX5A

FUNCTION: ENTRY IN USING TABLE

OPERATIONS DIAGRAMS: 22

DISPLMNT DEC (HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
2 (0002)	2	UESD	ESDID FOR USING VALUE
4 (0004)	4	UVAL	VALUE SPECIFIED IN USING STMT
8 (0008)	4	UREG	REGISTER

DSECT NAME: **VSDENTRY**

LOAD MODULE: IFOX11

SIZE: 13-19

CREATED BY: IFNX1J

REFERENCED BY: IFNX1J

UPDATED BY:

FUNCTION: MAPS ENTRIES OF ALL TYPES IN VARIABLE SYMBOL DEFINITION DIRECTORY (IN-CORE WORK TABLE)

OPERATIONS DIAGRAMS: 5

DISPLMNT DEC (HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE	
0 (0000)	3	VCHAIN	CHAIN POINTER	3 BYTES
3 (0003)	1	VFLAGS	SEE "FLGBYT" IN EDSECT	1 BYTE
4 (0004)	1	VLNGTH	SYMBOL LENGTH	1 BYTE
5 (0005)	2-8	VSYMBL	VARIABLE SYMBOL	
0 (0000)	1	VTFVAL	META TEXT FLAG	1 BYTE
1 (0001)	1	VGVECTR	GBL VECTOR PTR	3 BYTES
1 (0001)	3	VLDICTR	LCL DICTNRY PTR	3 BYTES
1 (0001)	3	VPVECTR	PARAM VCTR PTR	3 BYTES
4 (0004)	3	VGDIMEN	GBL DIMEN/SUBSC	2 BYTES
4 (0004)	2	VLDIMEN	LCL DIMEN/SUBSC	2 BYTES
4 (0004)	2	VPPAD	PARAM TERM PAD	2 BYTES

FIELD NAME	DISPLACEMENT DECIMAL	(HEX)
VCHAIN	0	(0)
VFLAGS	3	(3)
VGDIMEN	4	(4)
VGVECTR	1	(1)
VLDICTR	1	(1)
VLDIMEN	4	(4)
VLNGTH	4	(4)
VPPAD	4	(4)
VPVECTR	1	(1)
VTFVAL	0	(0)

*POINTER

DSECT NAME: **XRFIN**

LOAD MODULE: IFOX51

SIZE: VARIABLE

CREATED BY: IFNX5A

REFERENCED BY: IFNX5C,IFNX5M,IFNX5V,IFNX6A

UPDATED BY:

FUNCTION: CROSS REFERENCE RECORD MAP

OPERATIONS DIAGRAMS: 21, 22, 23, 25

DISPLMNT DEC (HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
0 (0000)	2	XRECLN	XREF RECORD LENGTH
2 (0002)	2	XFLAG	FLAG
4 (0004)	2	XOPCDE	OPCODE
6 (0006)	8	XRFSYM	XREF SYMBOL
14 (000E)	1	XRFFLG	XREF FLAG, BASE, DEF, DUP,UNDEF
15 (000F)	2	XRFSTM	XREF STATEMENT NUMBER
17 (0011)	2	XRFLEN	XREF LENGTH
19 (0013)	2	XRFVAL	XREF VALUE

FIELD NAME	DISPLACEMENT DECIMAL	(HEX)
XFLAG	2	(2)
XOPCDE	4	(4)
XRECLN	0	(0)
XRFFLG	14	(E)
XRFLEN	17	(11)
XRFSTM	15	(F)
XRFSYM	6	(6)
XRFVAL	19	(13)

*POINTER

DSECT NAME: **X5COM**

LOAD MODULE: IFOX51

SIZE: 2316

CREATED BY: IFNX5C

REFERENCED BY: IFNX5A-IFNX5V

UPDATED BY: IFNX5A-IFNX5V

FUNCTION: ASSEMBLY PHASE COMMON

OPERATIONS DIAGRAMS:

DISPLMNT			FIELD	DESCRIPTION: CONTENTS,
DEC	(HEX)	SIZE	NAME	MEANING/USE
0	(0000)	12	ELCTR	CURRENT LOCATION COUNTER
12	(000C)	4	TXTPTR	TEXT POINTER
16	(0010)	4	STMTN	STATEMENT NUMBER-LIKEWISE
20	(0014)	4	LNCNT	LINE COUNT-INIT TO 1
24	(0018)	0	LEFTHF	ALF INPUT TO PRINT ROUTINE
24	(0018)	4	LOCATN	LOCATION OF ENTRY
28	(001C)	0	DCDATA	FIELD FOR DC INPUT DATA
28	(001C)	1	LHOPCD	OPCODE FOR MACHINE INSTRUCTION
29	(001D)	1	LHIMD	
30	(001E)	2	BASDS1	BASE-DISPLACEMENT 1
32	(0020)	4	BASDS2	BASE-DISPLACEMENT 2
36	(0024)	4	ADDRS1	ADDRESS OF FIRST OPERAND
40	(0028)	4	ADDRS2	ADDRESS OF SECOND OPERAND
44	(002C)	1	LHFLGS	PROGRAM SWITCH
			ENTDC	BIT 0 - ENTRY IS A DC
			ENTALN	BIT 1 - ENTRY IS ALIGNMENT
			LHLNG	BIT 2 - LENGTH OF OUTPUT DATA
			EOUBIT	BIT 3 - EQU ORG OR USING BIT
46	(002E)	2	CRDCNT	BYTE COUNT IN TEXT (TXT) CARD
48	(0030)	4	SYMDEF	POINTER TO DEFINITION DATA
52	(0034)	2	SYMXRF	COUNT OF SYMBOLS XREF-ED THIS STMT
54	(0036)	1	SYMCNT	COUNT OF SYMBOLS THIS STATEMENT
55	(0037)	1	AOPF	PROGRAM SWITCH
			USSRT	BIT 0 - USING SORT FLAG
			OPNPRS	BIT 1 - OPERAND PRESENT FLAG
			DCCOMP	BIT 2 - DC COMPLETE SWITCH
			DCSTRT	BIT 3 - OUTPUT OF DC ALREADY STARTED
			DCMOP	BIT 4 - DC FINAL MOP-UP SWITCH
			DCSWH	BIT 5 - DC SWITCH
			NAMPRS	BIT 6 - NAME PRESENT SWITCH
56	(0038)	1	DCEVSW	PROGRAM SWITCH
			DSSW	BIT 0 - INDICATOR FOR STATEMENT A DS
			DXDSW	BIT 1 - INDICATOR FOR STATEMENT A DX
			DLOCTREF	BIT 2 - SWITCH FOR L'*
			NOTEWL	BIT 3 - INDICATE A NOTE MAY BE NECESSARY
			NOTEHS	BIT 4 - INDICATE A POINT WILL BE NECESSARY
			LITRSW	BIT 5 - LITERAL PRESENT SWITCH

DISPLMNT			FIELD	DESCRIPTION: CONTENTS,
DEC	(HEX)	SIZE	NAME	MEANING/USE
			XRFNO	BIT 6 - TURN OFF XREF
			DUPEVAL	BIT 7 - DUPLICATE EVAL OF DATA CONSTant
57	(0039)	1	X5SW1	PROGRAM SWITCH
			WRPFLG	BIT 0 - WRAP OF LOCATION COUNTER FLAG
			LTDECV	BIT 1 - LITERAL IN MACHINE OP
			COLOVLP	BIT 2 - COLUMN PTR OVERLAPPED
			ERRBIT	BIT 3 - ERROR LOGGING BIT
			TWASLC	BIT 4 - TITLE WAS LAST ENTRY
			PERR	BIT 5 - ERRORS DETECTED IN LAST STMT
			TPTEXT	BIT 6 - TITLE PUNCH OPERAND VALIDITY
			PRPP	BIT 7 - FORCE PRINT OF POP OR PUSH
58	(003A)	2	USPHL	USING PUSH-DOWN LEVEL
60	(003C)	0	SWITCHES	
60	(003C)	1	PRINTSW	PROGRAM SWITCH
			PSTMT	BIT 0 - PRINT STATEMENT
			PGEN	BIT 1 - SWITCH TO PRINT GENERATED TEXT
			PDATA	BIT 2 - SWITCH TO PRINT DC DATA
61	(003D)	1	CARDP	PROGRAM SWITCH
			CDPTR1	BIT 0 - CARD POINTER FLAG
62	(003E)	1	FLDSW	PROGRAM SWITCH
			PNAME	BIT 0 - NAME FIELD IN STMT
			POPER	BIT 1 - OPCODE FIELD IN STMT
			POPND	BIT 2 - OPERAND FIELD IN STMT
			PCOMM	BIT 3 - COMMENTS FIELD IN STMT
63	(003F)	1	X5VSW	PROGRAM SWITCH
			VLIT	BIT 0 - LITERAL IN EXPR - SET BY X5V
			ZAPIT	BIT 1 - SET TO INDICATE ZERO LEFTHF
64	(0040)	1	X5MSW	PROGRAM SWITCH
			E2PR	BIT 0 - EXPRESSION 2 PRESENT
			E3PR	BIT 1 - EXPRESSION 3 PRESENT
			E1ERR	BIT 2 - EXPR1 COMPLEXLY RELOCATABLE
			E2ERR	BIT 3 - EXPR 2 SIMPLY OR COMPL REL
			E3ERR	BIT 4 - EXPR 3 SIMPLY OR COMPL REL
			TOOMANY	BIT 5 - TOO MANY OPERANDS *
			LEAVE	BIT 6 - LEAVE X5M (SYNTAX ERROR)
			ABSUS	BIT 7 - A USING WITH ABS VALUE EXIST
65	(0041)	1	X5ASW	PROGRAM SWITCH
			MNOPRT	BIT 0 - MNOTE NOT TO BE PRINTED
			REPCARD	BIT 1 - PRINT ONLY REPRO CARD EXPECTED
66	(0042)	2	LISTSW	TO ISOLATE THE JLIST SWITCH
68	(0044)	2	EESDI	CURRENT ESDID
70	(0046)	106	JTITLE	TITLE
176	(00B0)	8	DWORD1	
184	(00B8)	8	DWORD2	
192	(00C0)	4	FNTEND	END OF FAR INSTRUCTION ENTRY
196	(00C4)	4	OPNADR	OPERAND POINTER
200	(00C8)	4	STRADR	STRING GROUP POINTER

DISPLMNT			FIELD	DESCRIPTION: CONTENTS,
DEC	(HEX)	SIZE	NAME	MEANING/USE
204	(00CC)	4	PRNSAVE	LINK REGISTER SAVE WORD
208	(00D0)	4	FLDSAVE	LINK REGISTER SAVE WORD
212	(00D4)	12	X5LSAV	LOG ERROR REGISTER SAVE
224	(00E0)	0	EXP2	EXPRESSION 2 VALUE
224	(00E0)	3		
227	(00E3)	1	I	INDEX OR LENGTH FIELD
228	(00E4)	0	EXP3	EXPRESSION 3 VALUE
228	(00E4)	3		
231	(00E7)	1	BASEX	BASE REGISTER
232	(00E8)	2	DISPL	DISPLACEMENT FIELD
234	(00EA)	4	LQ1	LENGTH ATTRIBUTE OF EXPR1
238	(00EE)	136	USINGT	CURRENT USING TABLE
374	(0176)	544		SPACE FOR PUSHED TABLES
918	(0396)	10	PRPU	PRINT PUSHDOWN AREA
928	(03A0)	2	HWD	SCRATCH HALFWORD
930	(03A2)	2	JBGNCL	BEGIN COLUMN (FROM ICTL)
932	(03A4)	2	JCNTCL	CONTINUE COLUMN (ICTL)
934	(03A6)	2	JENDCL	END COL-1 (ALSO ICTL)
936	(03A8)	2	CDSTMT	CARD-WITHIN-STATEMENT COUNTER
938	(03AA)	2	SALOC	S PART ALLOC (HALFWORD NUMBER)
940	(03AC)	2	LHWORK	WORK AREA TO UNPACK LEFT HALF
940	(03AC)	8	ULOCO	LOCATION
948	(03B4)	4	UOPCOD	OPCODE + SECOND BYTE
952	(03B8)	4	UBASD1	BASE-DISPLACEMENT 1
956	(03BC)	4	UBASD2	BASE-DISPLACEMENT 2
960	(03C0)	4	UGARB	GARBAGE
964	(03C4)	8	UADR1	ADDRESS 1
972	(03CC)	8	UADR2	ADDRESS 2
980	(03D4)	4		
984	(03D8)	4	PRNSV1	PRINT BUFFER SAVE AREA
988	(03DC)	2	CRDLAC	BYTE COUNT IN LAST CARD
990	(03DE)	2	COLSAV	COLUMN PTR SAVE AREA
992	(03E0)	4	CRDPTR	POINTER TO TEXT (TXT) CARD
996	(03E4)	4	CRDVAL	VALUE OF TEXT (TXT) CARD
1000	(03E8)	4	LOCLEN	ENTRY LENGTH FOR LOCATION UPDAT
1004	(03EC)	8	NOTEVAL	NOTE POINT SAVE
1012	(03F4)	4	LITRLC	LITERAL LOCATION COUNTER
1016	(03F8)	2	LITPID	LITERAL POOL ID
1018	(03FA)	2	LITRSD	LITERAL ESD
1020	(03FC)	32	WORKAREA	DEC CONVERT I/O AREA
1052	(041C)	12	PRNTSV	REGISTER SAVE FOR PRINT
1064	(0428)	8	PREGSV	PRINT SAVE AREA
1072	(0430)	4	DUPF	DUP-FACTOR STORAGE
1076	(0434)	4	BITMOD	CONSTANT LENGTH IN BITS
1080	(0438)	4	STRTLC	BIT LC SAVE
1084	(043C)	4	KLENGTH	CONSTANT SCAN-LENGTH IN BITS
1088	(0440)	4	OUTSTART	CONVERTED OUTPUT ADDRESS
1092	(0444)	4	FULLWD	SAVE WORD
1096	(0448)	4	KONSTRT	CONSTANT FIELD START
1100	(044C)	2	KCOUNT	CONSTANT COUNT
1102	(044E)	1	LMODSW	EXPLICIT-LENGTH FLAG
1103	(044F)	1	SKLOG	ERROR-LOG BYPASS FLAG
1104	(0450)	1	ZDUPSW	ZERO DUP-FACTOR FLAG
1105	(0451)	1	SIGNSW	MINUS-SIGN FLAG
1106	(0452)	1	MTSW	EMPTY DS FLAG
1108	(0454)	4	TEMPLC	TEMPORARY LOCATION-COUNTER
1112	(0458)	4	BITLC	LOCATION-COUNTER IN BITS
1116	(045C)	4	XREFYES	$XREF SAVE
1120	(0460)	2	OPNDCT	OPERAND COUNT
1122	(0462)	2	OBITS	CURRENT OUTPUT-BIT COUNT
1124	(0464)	1	DCPRSW	PRINTSW SAVE

DISPLMNT			FIELD	DESCRIPTION: CONTENTS,
DEC	(HEX)	SIZE	NAME	MEANING/USE
1125	(0465)	1	DUMSW	DS OR DXD FLAG
1126	(0466)	1	FSTPSW	FIRST-TIME PRINT FLAG
1127	(0467)	1	TUBEOP	INTERLUDE BAD OPERAND NUMBER
1128	(0468)	1	SELFDEFN	INITIATE SELF-DEFINING VALUE EXPECTED
1128	(0468)	0	VSELFDEF	
1128	(0468)	1		
1129	(0469)	1	EVALMODE	EVALUATION SWITCH
1130	(046A)	2	CLCLNG	ACTUAL LENGTH
1132	(046C)	4	IMPLNG	DEFAULT LENGTH
1136	(0470)	0	EVALREGS	
1136	(0470)	4	ATPTR	TERM STACK POINTER
1140	(0474)	4	ALPTR	REL LIST POINTER
1144	(0478)	4	AOPTR	OPERATOR STACK POINTER
1148	(047C)	4	ERRPTR	ERROR COLUMN POINTER
1152	(0480)	8	OPNEND	END OF OPERAND
1160	(0488)	0		CAUSE DOUBLE WORD ALIGNMENT
1160	(0488)	40	RLIST	RELOCATION LIST
1200	(04B0)	0	EVALWORK	WORK AREA FOR X5V
1200	(04B0)	0	FIRST	INDICATE FIRST TERM IN STACK
1201	(04B1)	1	STATUS	EVALUATION STATUS INFO
1201	(04B1)	0	VSTATUS1	BINARY OPERATOR
1201	(04B1)	0	VSTATUS2	CALCULATION DONE
1201	(04B1)	1		
1202	(04B2)	6		
1208	(04B8)	0	RELOCTR	COUNTER OF RELOC TERMS
1208	(04B8)	0	VNORELOC	
1208	(04B8)	1		
1209	(04B9)	1		
1210	(04BA)	6		
1216	(04C0)	1	EVALSW	
1217	(04C1)	0	EVALSW1	
1217	(04C1)	0	VCOMPLEX	COMPLETELY RELOC TERMS
1217	(04C1)	1		
1218	(04C2)	6		
1224	(04C8)	0	PARENCNT	NMBR OF UNBALANCED LEFT PARAMS
1224	(04C8)	1		
1225	(04C9)	1	VNOPAREN	
1225	(04C9)	0	VMAXPARN	
1225	(04C9)	1		
1226	(04CA)	6		
1232	(04D0)	0	SHIFTN	SHIFT VALUES
1232	(C4D0)	0	VSHIFTB	
1232	(04D0)	0	VSHIFTD	
1232	(04D0)	0	VSHIFTH	
1232	(04D0)	0	VSHIFTC	
1232	(04D0)	1		
1233	(04D1)	0	VMAXCHAR	NUMBER OF CHAR IN STRING
1233	(04D1)	0	VMAXHEX	
1233	(04D1)	0	VMAXDEC	
1233	(04D1)	0	VMAXBIT	
1233	(04D1)	-33		
1200	(04B0)	120	TERMS	TERMSTACK
1320	(0528)	0	VENDPARN	
1320	(0528)	1		
1321	(0529)	28	OPRNS	OPERATOR STACK
1349	(0545)	1		NOT USED
1350	(0546)	1	XSSAV	TEST ESDID

DISPLMNT DEC (HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
1352 (0548)	2		ADDITIONAL WORK AREA DC EVAL
1416 (0588)	64	LCTRSAV	SAVE AREA
1432 (0598)	16	LHSAVE	LEFT HALF SAVE AREA
1456 (05B0)	24	ENDSTMNO	STATEMENT NO. OF END STATEMENT
1460 (05B4)	4	X5ATEMP	TEMP TITLE AND PUNCH BLD AREA
1716 (06B4)	256	X5ALIT	LITERAL RECORD BUILD AREA
2016 (07E0)	300	DCLNG	ACCUM OBJECT LENGTH OF BAD DC

FIELD NAME	DISPLACEMENT DECIMAL	(HEX)
ABSUS	64	(40)
ADDRS1	36	(24)
ADDRS2	40	(28)
*ALPTR	1140	(474)
AOPF	55	(37)
*AOPTR	1144	(478)
*ATPTR	1136	(470)
BASDS1	30	(1E)
BASDS2	32	(20)
BASEX	231	(E7)
BITLC	1112	(458)
BITMOD	1076	(434)
CARDP	61	(3D)
CDPTR1	61	(3D)
CDSTMT	936	(3A8)
CLCLNG	1130	(46A)
COLOVLP	57	(39)
COLSAV	990	(3DE)
CRDCNT	46	(2E)
CRDLAC	988	(3DC)
CRDPTR	992	(3E0)
CRDVAL	996	(3E4)
DCCOMP	55	(37)
DCDATA	28	(1C)
DCEVSW	56	(38)
DCMOP	55	(37)
DCPRSW	1124	(464)
DCSTRT	55	(37)
DCSWH	55	(37)
DISPL	232	(E8)
DLOCTREF	56	(38)
DSSW	56	(38)
DUMSW	1125	(465)
DUPEVAL	56	(38)
DUPF	1072	(430)
DWORD1	176	(B0)
DWORD2	184	(B8)
DXDSW	56	(38)
EESDI	68	(44)
ELCTR	0	(0)
ENTALN	44	(2C)
ENTDC	44	(2C)
EOUBIT	44	(2C)

*POINTER

FIELD NAME	DISPLACEMENT DECIMAL	(HEX)
ERRBIT	57	(39)
ERRPTR	1148	(47C)
EVALMODE	1129	(469)
EVALREGS	1136	(470)
EVALSW	1216	(4C0)
EVALSW1	1217	(4C1)
EVALWORK	1200	(4B0)
EXP2	224	(E0)
EXP3	228	(E4)
E1ERR	64	(40)
E2ERR	64	(40)
E2PR	64	(40)
E3ERR	64	(40)
E3PR	64	(40)
FIRST	1200	(4B0)
FLDSAVE	208	(D0)
FLDSW	62	(3E)
FNTEND	192	(C0)
FSTPSW	1126	(466)
FULLWD	1092	(444)
HWD	928	(3A0)
I	227	(E3)
IMPLNG	1132	(46C)
JBGNCL	930	(3A2)
JCNTCL	932	(3A4)
JENDCL	934	(3A6)
JTITLE	70	(46)
KCOUNT	1100	(44C)
KLENGTH	1084	(43C)
KONSTRT	1096	(448)
LEAVE	64	(40)
LEFTHF	24	(18)
LHFLGS	44	(2C)
LHIMD	29	(1D)
LHLNG	44	(2C)
LHOPCD	28	(1C)
LHWORK	940	(3AC)
LISTSW	66	(42)
LITPID	1016	(3F8)
LITRLC	1012	(3F4)
LITRSD	1018	(3FA)
LITRSW	56	(38)
LMODSW	1102	(44E)
LNCNT	20	(14)

*POINTER

FIELD NAME	DISPLACEMENT DECIMAL	(HEX)
LOCATN	24	(18)
LOCLEN	1000	(3E8)
LQ1	234	(EA)
LTDECV	57	(39)
MNOPRT	65	(41)
MTSW	1106	(452)
NAMPRS	55	(37)
NOTEHS	56	(38)
NOTEVAL	1004	(3EC)
NOTEWL	56	(38)
OBITS	1122	(462)
OPNADR	196	(C4)
OPNDCT	1120	(460)
OPNEND	1152	(480)
OPNPRS	55	(37)
OPRNS	1321	(529)
OUTSTART	1088	(440)
PARENCNT	1224	(4C8)
PCOMM	62	(3E)
PDATA	60	(3C)
PERR	57	(39)
PGEN	60	(3C)
PNAME	62	(3E)
POPER	62	(3E)
POPND	62	(3E)
PREGSV	1064	(428)
PRINTSW	60	(3C)
PRNSAVE	204	(CC)
PRNSV1	984	(3D8)
PRNTSV	1052	(41C)
PRPP	57	(39)
PRPU	918	(396)
PSTMT	60	(3C)
RELOCTR	1208	(4B8)
REPCARD	65	(41)
RLIST	1160	(488)
SALOC	938	(3AA)
SELFDEFN	1128	(468)
SHIFTN	1232	(4D0)
SIGNSW	1105	(451)
SKLOG	1103	(44F)
STATUS	1201	(4B1)
STMTN	16	(10)
STRADR	200	(C8)
STRTLC	1080	(438)
SWITCHES	60	(3C)
SYMCNT	54	(36)
SYMDEF	48	(30)

*POINTER

FIELD NAME	DISPLACEMENT DECIMAL	(HEX)
SYMXRF	52	(34)
TEMPLC	1108	(454)
TERMS	1200	(4B0)
TOOMANY	64	(40)
TPTEXT	57	(39)
TUBEOP	1127	(467)
TWASLC	57	(39)
TXTPTR	12	(C)
UADR1	964	(3C4)
UADR2	972	(3CC)
UBASD1	952	(3B8)
UBASD2	956	(3BC)
UGARB	960	(3C0)
ULOCO	940	(3AC)
UOPCOD	948	(3B4)
USINGT	238	(EE)
USPHL	58	(3A)
USSRT	55	(37)
VCOMPLEX	1217	(4C1)
VENDPARN	1320	(528)
VLIT	63	(3F)
VMAXBIT	1233	(4D1)
VMAXCHAR	1233	(4D1)
VMAXDEC	1233	(4D1)
VMAXHEX	1233	(4D1)
VMAXPARN	1225	(4C9)
VNOPAREN	1225	(4C9)
VNORELOC	1208	(4B8)
VSELFDEF	1128	(468)
VSHIFTB	1232	(4D0)
VSHIFTC	1232	(4D0)
VSHIFTD	1232	(4D0)
VSHIFTH	1232	(4D0)
VSTATUS1	1201	(4B1)
VSTATUS2	1201	(4B1)
WORKAREA	1020	(3FC)
WRPFLG	57	(39)
XREFYES	1116	(45C)
XRFNO	56	(38)
XSSAV	1350	(546)
X5ASW	65	(41)
X5LSAV	212	(D4)
X5MSW	64	(40)
X5SW1	57	(39)
X5VSW	63	(3F)
ZAPIT	63	(3F)
ZDUPSW	1104	(450)

*POINTER.

Data Area Directory

FIELD	DSECT	DISPLACEMENT DECIMAL	(HEX)
ABSUS	X5COM	64	(40)
ADDRS1	X5COM	36	(24)
ADDRS2	X5COM	40	(28)
ADJSV	EDSECT	900	(384)
*AERRSTK	EDSECT	56	(38)
AICOPY	EDSECT	9	(9)
ALAST	EDSECT	776	(308)
*ALPTR	X5COM	1140	(474)
AOCOPYX	EDSECT	9	(9)
AOEND	EDSECT	9	(9)
AOKBTNPM	EDSECT	9	(9)
AOMACROX	EDSECT	9	(9)
AOMEND	EDSECT	9	(9)
AOPENCDX	EDSECT	9	(9)
AOPF	X5COM	55	(37)
AOPSYN	EDSECT	9	(9)
*AOPTR	X5COM	1144	(478)
AOTSW	EDSECT	9	(9)
*ATPTR	X5COM	1136	(470)
ATTRSV	EDSECT	1060	(424)
AT0	EDSECT	1060	(424)
AT1	EDSECT	1060	(424)
AT2	EDSECT	1060	(424)
AT3	EDSECT	1060	(424)
AT4	EDSECT	1060	(424)
AT5	EDSECT	1060	(424)
AT6	EDSECT	1060	(424)
AT7	EDSECT	1060	(424)
BASDS1	X5COM	30	(1E)
BASDS2	X5COM	32	(20)
BASEX	X5COM	231	(E7)
*BCSTK	EDSECT	632	(278)
BITLC	X5COM	1112	(458)
BITMOD	X5COM	1076	(434)
BYPASPCH	JOUTCOM	30	(1E)
BYPASPRT	JOUTCOM	30	(1E)
B0	EDSECT	1059	(423)
B1	EDSECT	1059	(423)
B2	EDSECT	1059	(423)
B3	EDSECT	1059	(423)
B4	EDSECT	1059	(423)
B5	EDSECT	1059	(423)
B6	EDSECT	1059	(423)
B7	EDSECT	1059	(423)
CARDID	RCARD	0	(0)
CARDP	X5COM	61	(3D)
CDPTR1	X5COM	61	(3D)
CDSTMT	X5COM	936	(3A8)
CLCLNG	X5COM	1130	(46A)
CLOSPCH	JOUTCOM	30	(1E)
CLOSPRT	JOUTCOM	30	(1E)
CNTCTR	EDSECT	857	(359)

*POINTER.

FIELD	DSECT	DISPLACEMENT DECIMAL	(HEX)
COLCTR	EDSECT	352	(160)
COLOVLP	X5COM	57	(39)
COLSAV	X5COM	990	(3DE)
CONCODE	EDSECT	1059	(423)
CONTROL	RPRINT	0	(0)
COPYCODE	EDSECT	288	(120)
COPYLN	EDSECT	296	(128)
COPYSV2	EDSECT	196	(C4)
COPYSV3	EDSECT	200	(C8)
COPYSV4	EDSECT	228	(E4)
CRDCNT	X5COM	46	(2E)
CRDLAC	X5COM	988	(3DC)
CRDPTR	X5COM	992	(3E0)
CRDVAL	X5COM	996	(3E4)
CSECTSW1	RSYMRCD	3	(3)
CSTK	EDSECT	572	(23C)
*CSTKADR	EDSECT	636	(27C)
CSW1	RSYMRCD	6	(6)
CURMDDPT	EDSECT	972	(3CC)
DCCOMP	X5COM	55	(37)
DCDATA	X5COM	28	(1C)
DCEVSW	X5COM	56	(38)
DCMOP	X5COM	55	(37)
DCPRSW	X5COM	1124	(464)
DCSTRT	X5COM	55	(37)
DCSWH	X5COM	55	(37)
DDNDX	EDSECT	18	(12)
DECKID	RCARD	72	(48)
DECMA	EDSECT	11	(B)
DEEQL	EDSECT	11	(B)
DEFINED	RSYMRCD	5	(5)
DELETE	MDDNTRY	3	(3)
*DERRCD	EDSECT	939	(3AB)
DISPL	X5COM	232	(E8)
DLOCTREF	X5COM	56	(38)
DLPRN	EDSECT	11	(B)
DMIENT	EDSECT	11	(B)
DNOCRD	EDSECT	11	(B)
DNTERR	EDSECT	932	(3A4)
DQUOT	EDSECT	11	(B)
DSCOMSW1	RSYMRCD	3	(3)
DSDTX	EDSECT	11	(B)
DSECTSW1	RSYMRCD	3	(3)
*DSEVCD	EDSECT	938	(3AA)
DSSW	X5COM	56	(38)
DSTGADJ	EDSECT	68	(44)
DSTGBGN	EDSECT	64	(40)
*DSTGEND	EDSECT	20	(14)
DSTGLN	EDSECT	298	(12A)
DSTGNDX	EDSECT	72	(48)
DSW1	RSYMRCD	6	(6)
DTEPTR	RPRINT	97	(61)
DTLENG	EDSECT	306	(132)

*POINTER.

FIELD	DSECT	DISPLACEMENT	
		DECIMAL	(HEX)
DUMOPND	EDSECT	11	(B)
DUMSW	X5COM	1125	(465)
DUPEVAL	X5COM	56	(38)
DUPF	X5COM	1072	(430)
DWORD1	X5COM	176	(B0)
DWORD2	X5COM	184	(B8)
DXDSW	X5COM	56	(38)
EDTSVX	EDSECT	80	(50)
EDTSVY	EDSECT	84	(54)
EDTSVZ	EDSECT	88	(58)
EESDI	X5COM	68	(44)
EFILRL	ENDFIL	0	(0)
EFILRT	ENDFIL	2	(2)
ELCTR	X5COM	0	(0)
EMSGCODE	ERRMESS	1	(1)
EMSGNTRY	ERRMESS	3	(3)
*ENDATA	EDSECT	24	(18)
ENDCOL	EDSECT	276	(114)
ENDEDSCT	EDSECT	1128	(468)
ENDWKA	EDSECT	340	(154)
ENTALN	X5COM	44	(2C)
ENTDC	X5COM	44	(2C)
*ENTRPUTL	J	724	(2D4)
ENTRYLNG	ERRMESS	2	(2)
ENTRYSW1	RSYMRCD	3	(3)
EOUBIT	X5COM	44	(2C)
ERRBIT	X5COM	57	(39)
ERRCNT	EDSECT	704	(2C0)
ERRFLD	ERRIN	7	(7)
ERRID	ERRIN	2	(2)
ERRLEN	ERRIN	0	(0)
ERRNUM	ERRIN	6	(6)
ERRPTR	X5COM	1148	(47C)
ERRSTK	EDSECT	706	(2C2)
ERRSTMT	ERRIN	4	(4)
ESDNRSW1	RSYMRCD	3	(3)
ESDOFLO	RSYMRCD	5	(5)
ESEGRL	ENDSEG	0	(0)
ESEGRT	ENDSEG	2	(2)
*ESTKNDX	EDSECT	60	(3C)
EVALMODE	X5COM	1129	(469)
EVALREGS	X5COM	1136	(470)
EVALSW	X5COM	1216	(4C0)
EVALSW1	X5COM	1217	(4C1)
EVALWORK	X5COM	1200	(4B0)
EXP2	X5COM	224	(E0)
EXP3	X5COM	228	(E4)
E1ERR	X5COM	64	(40)
E2ERR	X5COM	64	(40)
E2PR	X5COM	64	(40)
E3ERR	X5COM	64	(40)
E3PR	X5COM	64	(40)
FENT	FARENT	0	(0)
FIAL1	FARENT	0	(0)
FILEN	FARENT	0	(0)
FIRST	X5COM	1200	(4B0)
FLAGBT	EDSECT	304	(130)

*POINTER.

FIELD	DSECT	DISPLACEMENT	
		DECIMAL	(HEX)
FLDSAVE	X5COM	208	(D0)
FLDSW	X5COM	62	(3E)
FLGBYT	EDSECT	1062	(426)
FLGOUT	RPRINT	21	(15)
FLUSH	MDDNTRY	3	(3)
FMT	FARENT	0	(0)
FNDFLG	EDSECT	1022	(3FE)
FNTEND	X5COM	192	(C0)
*FPTRSV	EDSECT	28	(1C)
FREESTRT	EDSECT	944	(3B0)
FSNLIT	FARENT	0	(0)
FSTGL	EDSECT	52	(34)
FSTPSW	X5COM	1126	(466)
FSWITCH	EDSECT	1023	(3FF)
FULLWD	X5COM	1092	(444)
GAIF	EDSECT	10	(A)
GCHAIN	GBLNTRY	0	(0)
GDCP	GDNTRY	0	(0)
GDDM	GDNTRY	4	(4)
GDDP	GDNTRY	1	(1)
GDEFD	GBLDEF	4	(4)
GDEFF	GBLDEF	3	(3)
GDEFRL	GBLDEF	0	(0)
GDEFRT	GBLDEF	2	(2)
GDEFSL	GBLDEF	4	(4)
GDEFTF	GBLDEF	0	(0)
GDEFVP	GBLDEF	1	(1)
GDFL	GDNTRY	3	(3)
GDIM	GDNTRY	3	(3)
GDIMEN	GBLNTRY	4	(4)
GDTFV	GDNTRY	0	(0)
GFLAGS	GBLNTRY	3	(3)
GLNGTH	GBLNTRY	4	(4)
GPTYP	GDNTRY	3	(3)
GQST	EDSECT	10	(A)
GSCNSW	EDSECT	10	(A)
GSLS	GDNTRY	3	(3)
GSNS	GDNTRY	3	(3)
GSTP1	GDNTRY	3	(3)
GSTP2	GDNTRY	3	(3)
GSUBS	EDSECT	10	(A)
GSUMRY	EDSECT	13	(D)
GTFVAL	GBLNTRY	0	(0)
GTGVALOC	EDSECT	1012	(3F4)
GTKVALOC	EDSECT	1004	(3EC)
GTLDALOC	EDSECT	1008	(3F0)
GTMVALOC	EDSECT	980	(3D4)
GTODALOC	EDSECT	1024	(400)
GTPVALOC	EDSECT	1000	(3E8)
GTSDALOC	EDSECT	1016	(3F8)
GTYP1	GDNTRY	3	(3)
GTYP2	GDNTRY	3	(3)
GVECTR	GBLNTRY	1	(1)
HIBYTE0	EDSECT	984	(3D8)
HICVAL	EDSECT	252	(FC)
HWD	X5COM	928	(3A0)
I	X5COM	227	(E3)
IMPLNG	X5COM	1132	(46C)

*POINTER.

FIELD	DSECT	DISPLACEMENT DECIMAL	 (HEX)
*INPUT	EDSECT	32	(20)
INTERMET	EDSECT	112	(70)
IOCID	EDSECT	1058	(422)
IONE	EDSECT	1058	(422)
*IPTRSV	EDSECT	36	(24)
*IRTNSV	EDSECT	40	(28)
ITERSW	EDSECT	1022	(3FE)
ITRE	EDSECT	1058	(422)
ITWO	EDSECT	1058	(422)
IZRO	EDSECT	1058	(422)
*JAABORT	J	708	(2C4)
*JABORT	J	704	(2C0)
*JADINCM	J	248	(F8)
*JADOUTCM	J	252	(FC)
JALGN	J	309	(135)
JALOGIC	J	309	(135)
JBEGCL	J	752	(2F0)
JBGNCL	X5COM	930	(3A2)
*JBOS	J	368	(170)
*JBUF	JFLEBLK	32	(20)
*JBUFFER	JFLEBLK	28	(1C)
JBUFNDX	JFLEBLK	38	(26)
JCALLS	J	309	(135)
JCHKFILE	JFLEBLK	40	(28)
*JCLVLPTR	J	376	(178)
JCNTCL	X5COM	932	(3A4)
JCOMEND	J	1272	(4F8)
JCOMMON	J	0	(0)
JCONTCL	J	720	(2D0)
JCTBGN	J	768	(300)
JCTCHR	J	756	(2F4)
JCTLN	J	772	(304)
*JCURPCH	JOUTCOM	24	(18)
*JCURPRT	JOUTCOM	20	(14)
JDBLALL	J	321	(141)
JDBLBUF	JFLEBLK	40	(28)
JDCSX	JTEXT	2	(2)
JDECB	JFLEBLK	0	(0)
JDECK	J	308	(134)
JDECKID	J	289	(121)
JDECKIDL	J	288	(120)
JDECKSEQ	JOUTCOM	28	(1C)
JDEF	JTEXT	2	(2)
JDPASS	J	788	(314)
JDUMPX0	J	318	(13E)
JDUMPX1	J	318	(13E)
JDUMPX2	J	318	(13E)
JDUMPX3	J	318	(13E)
JDUMPX4	J	318	(13E)
JDUMPX5	J	318	(13E)
JDUMPX6	J	318	(13E)
JDWORD	J	728	(2D8)
JECOLPTR	JERRCD	5	(5)
JEERCOD	JERRCD	9	(9)
JEFLGA	JERRCD	2	(2)
JEFLGB	JERRCD	3	(3)
JENDCHK	J	316	(13C)
JENDCL	X5COM	934	(3A6)

*POINTER.

FIELD	DSECT	DISPLACEMENT DECIMAL	 (HEX)
JENDCOL	J	722	(2D2)
JENODATA	JERRCD	11	(B)
*JENTRYPT	J	780	(30C)
*JEOS	J	372	(174)
JEPRPOS	JERRCD	3	(3)
JEPSOP	JERRCD	2	(2)
JERCDE	JERRCD	10	(A)
JERECL	JERRCD	0	(0)
JERR	JTEXT	3	(3)
JERRCHK	J	316	(13C)
JESD	J	308	(134)
JESDCHK	J	316	(13C)
JESDID	J	784	(310)
JESDOFLO	J	317	(13D)
JESEV	JERRCD	9	(9)
JESTMTNO	JERRCD	6	(6)
JEXTB	JTEXT	2	(2)
*JFLE	JFLEBLK	24	(18)
*JFLEBLK1	J	72	(48)
*JFLEBLK2	J	128	(80)
*JFLEBLK3	J	184	(B8)
JFWORD1	J	736	(2E0)
JFWORD2	J	740	(2E4)
JGEN	JTEXT	3	(3)
JGETLPND	JFLEBLK	40	(28)
JGETLPNT	JFLEBLK	40	(28)
JGETLSBF	JFLEBLK	40	(28)
JHWORD1	J	744	(2E8)
JHWORD2	J	746	(2EA)
JIDR	J	1232	(4D0)
JINDERRF	J	317	(13D)
JINFILE	J	776	(308)
JINFLAG	J	319	(13F)
JINHB	JTEXT	2	(2)
JINLIB	J	319	(13F)
*JINMLC	J	328	(148)
JINPC	JTEXT	2	(2)
JINVOPT	J	317	(13D)
JIN2ND	J	319	(13F)
JIOFLAG	JFLEBLK	40	(28)
JLINK	J	308	(134)
JLIST	J	308	(134)
JLITLNG	J	848	(350)
JLNCT	J	298	(12A)
JLNCTKEY	J	310	(136)
JLN2	JTEXT	2	(2)
JLN4	JTEXT	2	(2)
JLSTNOTE	JFLEBLK	41	(29)
JLVTMDT	J	264	(108)
JMAXRL	J	246	(F6)
JMAXRL1	J	240	(F0)
JMAXRL2	J	242	(F2)
JMAXRL3	J	244	(F4)
JMINBUF	J	310	(136)
JMISLIN	J	317	(13D)
JMISPCH	J	317	(13D)
JMISPRT	J	317	(13D)
*JMLC	J	324	(144)

*POINTER.

FIELD	DSECT	DISPLACEMENT	
		DECIMAL	(HEX)
JMLCFLAG	J	321	(141)
JMLOGIC	J	309	(135)
JMSGL	J	297	(129)
JMSGLKEY	J	310	(136)
JNMERR	JTEXT	3	(3)
JNOCNT	JTEXT	3	(3)
JNOSEQPH	J	320	(140)
JNOTED	JFLEBLK	40	(28)
JNOTEVAL	J	340	(154)
JNUM	J	310	(136)
*JOUTCLOS	JOUTCOM	16	(10)
JOUTCMND	JOUTCOM	32	(20)
JOUTFILE	J	778	(30A)
JOUTFLAG	J	320	(140)
*JOUTMLC	J	332	(14C)
*JOUTOPEN	JOUTCOM	12	(C)
JOUTSW	JOUTCOM	30	(1E)
JOUT2ND	J	320	(140)
*JPARM	J	308	(134)
*JPARMPTR	J	304	(130)
JPARMS	J	297	(129)
JPARM1	J	308	(134)
JPARM2	J	309	(135)
JPARM3	J	310	(136)
JPARM4	J	311	(137)
JPARM4	J	311	(137)
JPDFLAG	J	318	(13E)
*JPDUMP	J	336	(150)
JPHBLANK	J	262	(106)
*JPHNAME	J	256	(100)
JPHPREF	J	256	(100)
JPHSUFF	J	259	(103)
JPRESD	JTEXT	2	(2)
JPRONLY	JTEXT	3	(3)
JPRTONLY	J	845	(34D)
JPSOP	JTEXT	2	(2)
JPT4GET	J	321	(141)
JPT4READ	J	321	(141)
JPT4STAR	J	321	(141)
JPT4WRIT	J	321	(141)
JPUTLPND	JFLEBLK	40	(28)
JRECCHK	J	316	(13C)
JRECIN	J	348	(15C)
JRECLIB	J	352	(160)
JRECPCH	J	356	(164)
JRECPRT	J	360	(168)
JREENTR	J	315	(13B)
JREF	JTEXT	2	(2)
JRENT	J	309	(135)
JREQOP	JTEXT	2	(2)
JRL	JFLEBLK	36	(24)
JRLD	J	308	(134)
JRLDCHK	J	316	(13C)
JSAFE	J	1160	(488)
JSAVE	J	0	(0)
JSAVETBL	J	384	(180)
JSEQCL	J	760	(2F8)
JSEQLN	J	764	(2FC)

*POINTER.

FIELD	DSECT	DISPLACEMENT	
		DECIMAL	(HEX)
JSEVER	J	844	(34C)
*JSLEN	J	364	(16C)
JSRCLN	J	748	(2EC)
JSTMT	J	310	(136)
JSUBNAME	JTEXT	3	(3)
JSUBOPCD	JTEXT	3	(3)
JSUBOPND	JTEXT	3	(3)
*JSYSCLOS	J	716	(2CC)
JSYSDATE	J	280	(118)
JSYSGEN	J	846	(34E)
*JSYSLNK	JOUTCOM	8	(8)
*JSYSLST	JOUTCOM	0	(0)
JSYSMAC	J	309	(135)
*JSYSOPEN	J	712	(2C8)
*JSYSPARM	J	300	(12C)
*JSYSPCH	JOUTCOM	4	(4)
JSYSTIME	J	274	(112)
JTBLTRT	J	850	(352)
*JTCLOSE	JFLEBLK	20	(14)
JTCML	JTEXTA	1	(1)
JTCOP	JTEXTA	0	(0)
JTCPR	JTEXT	12	(C)
JTERM	J	310	(136)
JTEST	J	308	(134)
JTFLGA	JTEXT	2	(2)
JTFLGA1	JTEXT	2	(2)
JTFLGB	JTEXT	3	(3)
JTIOP	JTEXT	4	(4)
JTIOP1	JTEXT	4	(4)
JTIOP2	JTEXT	5	(5)
JTITLE	X5COM	70	(46)
JTNML	JTEXTA	1	(1)
JTNMO	JTEXTA	0	(0)
JTNMOCD	JTEXTA	0	(0)
JTNMP	JTEXT	6	(6)
JTOCL	JTEXTA	1	(1)
JTOCO	JTEXTA	0	(0)
JTOCOCD	JTEXTA	0	(0)
JTOCP	JTEXT	8	(8)
JTOPL	JTEXTA	1	(1)
JTOPO	JTEXTA	0	(0)
JTOPOCD	JTEXTA	0	(0)
JTOPP	JTEXT	10	(A)
JTRLI	JTEXT	0	(0)
JTRTABLE	J	901	(385)
JTSPR	JTEXT	14	(E)
JTSTC	JTEXTA	0	(0)
JTSTL	JTEXTA	2	(2)
JTSTL2	JTEXTA	1	(1)
JTSTO	JTEXTA	1	(1)
JTSTO2	JTEXTA	0	(0)
JTSYMCNT	JTEXT	16	(10)
JWARNFLG	J	315	(13B)
JXREF	J	308	(134)
JXREFCHK	J	316	(13C)
JYCON	J	315	(13B)
KCOUNT	X5COM	1100	(44C)
KLENGTH	X5COM	1084	(43C)

*POINTER.

FIELD	DSECT	DISPLACEMENT	
		DECIMAL	(HEX)
KONSTRT	X5COM	1096	(448)
LATTRIB	OSRDNTRY	1	(1)
LCHAIN	LCLNTRY	0	(0)
LCNTRL	RPRINT	0	(0)
LDICTR	LCLNTRY	1	(1)
LDIMEN	LCLNTRY	4	(4)
LEAVE	X5COM	64	(40)
LEFTHF	X5COM	24	(18)
LFLAGS	LCLNTRY	3	(3)
LHDPTR	RPRINT	43	(2B)
LHFLGS	X5COM	44	(2C)
LHIMD	X5COM	29	(1D)
LHLNG	X5COM	44	(2C)
LHOPCD	X5COM	28	(1C)
LHWORK	X5COM	940	(3AC)
LISTSW	X5COM	66	(42)
LITDTL	PPIN	19	(13)
LITESDID	PPIN	10	(A)
LITLOCTR	PPIN	6	(6)
LITPID	X5COM	1016	(3F8)
LITPOLID	PPIN	12	(C)
LITRLC	X5COM	1012	(3F4)
LITRSD	X5COM	1018	(3FA)
LITRSW	X5COM	56	(38)
LLNGTH	LCLNTRY	4	(4)
LMODSW	X5COM	1102	(44E)
LNCNT	X5COM	20	(14)
LOCATN	X5COM	24	(18)
LOCLEN	X5COM	1000	(3E8)
LQ1	X5COM	234	(EA)
LSTSYSMS	EDSECT	1022	(3FE)
LTDECV	X5COM	57	(39)
LTFVAL	LCLNTRY	0	(0)
MCALL	EDSECT	1061	(425)
MCHAIN	MDDNTRY	0	(0)
MCLA	EDSECT	1061	(425)
MCLC	EDSECT	1061	(425)
MCMPLX	EDSECT	1061	(425)
MDDCHN	EDSECT	988	(3DC)
MDDCNT	EDSECT	992	(3E0)
MDDSLOT	EDSECT	968	(3C8)
MDDSTRT	EDSECT	952	(3B8)
METSW	EDSECT	10	(A)
MEZZOPTR	EDSECT	116	(74)
MFLAGS	MDDNTRY	3	(3)
MGBLSZ	MDDNTRY	31	(1F)
MINDIF	EDSECT	310	(136)
MINPADJ	EDSECT	328	(148)
MINPSTD	EDSECT	320	(140)
MINPUT	EDSECT	316	(13C)
MIOPNDSV	EDSECT	128	(80)
MLCLSZ	MDDNTRY	37	(25)
MNL1	MDDNTRY	3	(3)
MNOPRT	X5COM	65	(41)
MNPSD	MDVNTRY	8	(8)
MNPTXT	MDVNTRY	0	(0)
MPOPSV	EDSECT	912	(390)
MREGSV	EDSECT	344	(158)

*POINTER.

FIELD	DSECT	DISPLACEMENT	
		DECIMAL	(HEX)
MSDL	MDVNTRY	16	(10)
MSEQSZ	MDDNTRY	34	(22)
MSERR	EDSECT	14	(E)
MSLST	EDSECT	1061	(425)
MSYMBL	MDDNTRY	4	(4)
MTSDNP	MDDNTRY	23	(17)
MTSW	X5COM	1106	(452)
MTXTNP	MDDNTRY	15	(F)
MTXTP	EDSECT	312	(138)
MVECTR	MDDNTRY	12	(C)
MXRPRN	EDSECT	14	(E)
MXVS	EDSECT	14	(E)
NAMBYT	EDSECT	12	(C)
NAML	EDSECT	136	(88)
NAML1	EDSECT	144	(90)
NAMPRS	X5COM	55	(37)
NCNCAT	EDSECT	12	(C)
*NCSTK	EDSECT	628	(274)
NEXPSV	EDSECT	916	(394)
NMPURE	EDSECT	12	(C)
NNALFA	EDSECT	12	(C)
NNTGER	EDSECT	12	(C)
NOSEQ	JOUTCOM	30	(1E)
NOSYM	EDSECT	12	(C)
NOTEFIL2	EDSECT	1022	(3FE)
NOTEHS	X5COM	56	(38)
NOTESAVE	EDSECT	1063	(427)
NOTESV1	EDSECT	148	(94)
NOTESV2	EDSECT	152	(98)
NOTEVAL	X5COM	1004	(3EC)
NOTEWL	X5COM	56	(38)
NQTSTG	EDSECT	12	(C)
*NRSTK	EDSECT	368	(170)
NSSYM	EDSECT	12	(C)
NUMERR	ERRIN	3	(3)
NVSYM	EDSECT	12	(C)
OBITS	X5COM	1122	(462)
OCHAIN	OPNTRY	0	(0)
OCPTRSV	EDSECT	108	(6C)
OCSAVE	EDSECT	300	(12C)
OCTS	MDDNTRY	3	(3)
ODEL	OPSYNTRY	3	(3)
OFLAGA	OPNTRY	0	(0)
OFLAGS	OPNTRY	2	(2)
OFPTRSV	EDSECT	356	(164)
OINTCD	OPNTRY	1	(1)
OMAC	OPSYNTRY	3	(3)
OMASK	OPNTRY	2	(2)
ONAME	OPSYNTRY	8	(8)
ONAMEL	OPSYNTRY	7	(7)
OPCDPTR	EDSECT	336	(150)
OPNADR	X5COM	196	(C4)
OPNDCT	X5COM	1120	(460)
OPNDCTR	EDSECT	308	(134)
OPNDPTR	EDSECT	120	(78)
OPNEND	X5COM	1152	(480)
OPNPRS	X5COM	55	(37)
OPREV	OPSYNTRY	3	(3)

*POINTER.

FIELD	DSECT	DISPLACEMENT DECIMAL	(HEX)
OPRNS	X5COM	1321	(529)
OPSCHN	EDSECT	996	(3E4)
OPSFLGS	OPSTBL	0	(0)
OPSTATTS	OPSTBL	1	(1)
OPSTNAM	OPSTBL	5	(5)
OPSTNL	OPSTBL	4	(4)
OPSYNCH	OPSYNTRY	0	(0)
OPSYNCHN	OPSYNTRY	4	(4)
OPSYNFLG	OPSYNTRY	3	(3)
OREFDP	OSREF	3	(3)
OREFRL	OSREF	0	(0)
OREFRT	OSREF	2	(2)
OREFSL	OSREF	6	(6)
OREFTYPE	EDSECT	1041	(411)
OSDLNGTH	EDSECT	1038	(40E)
OSFLGVAL	EDSECT	1040	(410)
OSPAD	OSDIR	12	(C)
OSRAPDIS	EDSECT	1028	(404)
OSRDP	OSDIR	9	(9)
OSRDSTRT	EDSECT	964	(3C4)
OSRTCP	OSRTNTRY	0	(0)
OSRTDP	OSRTNTRY	3	(3)
OSRTSL	OSRTNTRY	6	(6)
OSSYM	OSDIR	0	(0)
OTFVAL	OSDIR	8	(8)
*OUTADR	EDSECT	44	(2C)
OUTSTART	X5COM	1088	(440)
PARENCNT	X5COM	1224	(4C8)
PARMSTAT	EDSECT	11	(B)
PBGLEN	EDSECT	268	(10C)
PCHAIN	PRMNTRY	0	(0)
PCOMM	X5COM	62	(3E)
PDATA	X5COM	60	(3C)
PERR	X5COM	57	(39)
PFLAGS	PRMNTRY	3	(3)
PGEN	X5COM	60	(3C)
PIOPARMA	EDSECT	1052	(41C)
PIOPARMB	EDSECT	1048	(418)
PIOPARMC	EDSECT	1056	(420)
PLEN	P	0	(0)
PLNGTH	PRMNTRY	4	(4)
PNAME	X5COM	62	(3E)
PNDLEN	EDSECT	272	(110)
POPER	X5COM	62	(3E)
POPND	X5COM	62	(3E)
POSID	PPIN	6	(6)
POSID	RLDIN	6	(6)
POSOUT	RPRINT	2	(2)
POSSUBL	EDSECT	5	(5)
PPAD	PRMNTRY	4	(4)
PPFLG	PPIN	2	(2)
PPIOC	PPIN	4	(4)
PPRLI	PPIN	0	(0)
PREGSV	X5COM	1064	(428)
PRINTSW	X5COM	60	(3C)
PRIORDEF	RSYMRCD	5	(5)
PRNLVL	EDSECT	302	(12E)
PRNSAVE	X5COM	204	(CC)

*POINTER.

FIELD	DSECT	DISPLACEMENT DECIMAL	(HEX)
PRNSV1	X5COM	984	(3D8)
PRNTSV	X5COM	1052	(41C)
PROTOCAL	EDSECT	5	(5)
PRPP	X5COM	57	(39)
PRPU	X5COM	918	(396)
PSTMT	X5COM	60	(3C)
PSYSGO	P	82	(52)
PSYSIN	P	34	(22)
PSYSLIB	P	26	(1A)
PSYSPRIN	P	42	(2A)
PSYSPUNC	P	50	(32)
PSYSUT1	P	58	(3A)
PSYSUT2	P	66	(42)
PSYSUT3	P	74	(4A)
PTFVAL	PRMNTRY	0	(0)
PVECTR	PRMNTRY	1	(1)
*RAVSP	EDSECT	364	(16C)
RCNCAT	EDSECT	13	(D)
RCNTRL	RPRINT	0	(0)
REGSAVE1	EDSECT	976	(3D0)
REGSAVE2	EDSECT	1044	(414)
REGSAVE3	EDSECT	928	(3A0)
REGSTACK	EDSECT	1080	(438)
RELID	PPIN	8	(8)
RELID	RLDIN	8	(8)
RELOCTR	X5COM	1208	(4B8)
RELOUT	RPRINT	11	(B)
REPCARD	X5COM	65	(41)
RESDC	RSYMRCD	6	(6)
RESDI	RSYMRCD	6	(6)
RFIELDN	RSYMRCD	5	(5)
RFIELDX	RSYMRCD	5	(5)
RFLAG	PPIN	2	(2)
RFLAG	RLDIN	2	(2)
RFLDI	RSYMRCD	5	(5)
RFLGA	RSYMRCD	2	(2)
RFLGB	RSYMRCD	3	(3)
RIDEC	FARENT	1	(1)
RIST	FARENT	1	(1)
RITEM	RSYMRCD	0	(0)
RLCTR	RSYMRCD	8	(8)
RLDBYT	RCARD	10	(A)
RLDFLD	RCARD	16	(10)
RLDFLG	PPIN	13	(D)
RLDFLG	RLDIN	13	(D)
RLDLEN	PPIN	0	(0)
RLDLEN	RLDIN	0	(0)
RLDNAM	RCARD	1	(1)
RLDVAL	PPIN	10	(A)
RLDVAL	RLDIN	10	(A)
RLIST	X5COM	1160	(488)
RLNGA	RSYMRCD	12	(C)
RLNGB	RSYMRCD	22	(16)
RLNGQ	RSYMRCD	20	(14)
RMPURE	EDSECT	13	(D)
RNALFA	EDSECT	13	(D)
RNAME	RSYMRCD	12	(C)
RNTGER	EDSECT	13	(D)

*POINTER.

FIELD	DSECT	DISPLACEMENT DECIMAL	(HEX)
ROPCDE	PPIN	4	(4)
ROPCDE	RLDIN	4	(4)
ROSYM	EDSECT	13	(D)
RPSOP	RSYMRCD	2	(2)
RQTSTG	EDSECT	13	(D)
RRCDL	RSYMRCD	0	(0)
RSALW	FARENT	1	(1)
RSMOD	FARENT	1	(1)
RSST	FARENT	1	(1)
RSSYM	EDSECT	13	(D)
RSTACK	EDSECT	372	(174)
RSWTS	RSYMRCD	6	(6)
RSYMC1	RSYMRCD	12	(C)
RSYMC2	RSYMRCD	6	(6)
RTNSV	EDSECT	124	(7C)
RTYPE	RSYMRCD	4	(4)
RVSYM	EDSECT	13	(D)
SABORT	EDSECT	3	(3)
SALLCT	EDSECT	1	(1)
SALOC	X5COM	938	(3AA)
SASTCMT	EDSECT	6	(6)
SATTRIB	OSRDNTRY	3	(3)
SAVENOTE	EDSECT	1071	(42F)
SAVMALL	EDSECT	640	(280)
SBDPROTO	EDSECT	6	(6)
SBYCNT	EDSECT	1	(1)
SBYONE	EDSECT	1	(1)
SCMTCT	EDSECT	2	(2)
SCNCAT	EDSECT	5	(5)
SCOPY	EDSECT	4	(4)
SCTLRTN	EDSECT	1	(1)
SDEFNP	SSDEF	3	(3)
SDEFRL	SSDEF	0	(0)
SDEFRT	SSDEF	2	(2)
SDEFSL	SSDEF	11	(B)
SDENT	EDSECT	4	(4)
SDENTR	EDSECT	15	(F)
SDENTR1	EDSECT	16	(10)
SDINIT	EDSECT	4	(4)
SDTCMT	EDSECT	6	(6)
SELFDEFN	X5COM	1128	(468)
SENAME	EDSECT	7	(7)
SENDST	EDSECT	5	(5)
SEOPCD	EDSECT	7	(7)
SEOPND	EDSECT	7	(7)
SEQNUM	RCARD	76	(4C)
SEQSV	EDSECT	156	(9C)
SEQSVT	EDSECT	858	(35A)
SFSTCD	EDSECT	4	(4)
SGBLCL	EDSECT	2	(2)
SHIFTN	X5COM	1232	(4D0)
SICTL	EDSECT	3	(3)
SIGNSW	X5COM	1105	(451)
SINCPY	EDSECT	8	(8)
SINEOF	EDSECT	2	(2)
SISEQ	EDSECT	8	(8)
SKACTRV	SKDCTHDR	27	(1B)
SKADNLD	SKDCTHDR	15	(F)

*POINTER.

FIELD	DSECT	DISPLACEMENT DECIMAL	(HEX)
SKKVADR	SKDCTHDR	12	(C)
SKLDADR	SKDCTHDR	3	(3)
SKLDLNG	SKDCTHDR	6	(6)
SKLOG	X5COM	1103	(44F)
SKMPADR	SKDCTHDR	9	(9)
SKNOFSLS	SKDCTHDR	31	(1F)
SKNPNLT	SKDCTHDR	19	(13)
SKPEND	EDSECT	3	(3)
SKPMND	EDSECT	3	(3)
SKPNAME	EDSECT	5	(5)
SKSRDPT	SKDCTHDR	0	(0)
SKWPRM	EDSECT	5	(5)
SLSTCD	EDSECT	2	(2)
SMAC	EDSECT	8	(8)
SMACNAM	EDSECT	280	(118)
SMDDENTR	EDSECT	4	(4)
SMDEF	EDSECT	0	(0)
SMI	EDSECT	2	(2)
SMISCN	EDSECT	0	(0)
SNMFND	EDSECT	6	(6)
SNOACTR	EDSECT	3	(3)
SNOCNT	EDSECT	8	(8)
SNOFND	EDSECT	6	(6)
SNOPND	EDSECT	1	(1)
SNOPSYN	EDSECT	0	(0)
SNOSMCRO	EDSECT	6	(6)
SNOSYSMD	EDSECT	6	(6)
SNXTCT	EDSECT	2	(2)
SONECD	EDSECT	1	(1)
SONECT	EDSECT	1	(1)
SOPNCD	EDSECT	3	(3)
SPGRMD	EDSECT	3	(3)
SPRMER	EDSECT	5	(5)
SPRVCT	EDSECT	2	(2)
SREFDP	SSREF	3	(3)
SREFRL	SSREF	0	(0)
SREFRT	SSREF	2	(2)
SREFSL	SSREF	6	(6)
SREFTYPE	EDSECT	1033	(409)
SSDLNGTH	EDSECT	1030	(406)
SSDTCP	SSDTNTRY	0	(0)
SSDTNP	SSDTNTRY	3	(3)
SSDTSL	SSDTNTRY	11	(B)
SSFLGVAL	EDSECT	1032	(408)
SSPAD	SSDIR	12	(C)
SSRAPDIS	EDSECT	1020	(3FC)
SSRDP	SSDIR	9	(9)
SSRDSTRT	EDSECT	956	(3BC)
SSSYM	SSDIR	0	(0)
SSYSMD	EDSECT	3	(3)
STACK	EDSECT	782	(30E)
STATUS	X5COM	1201	(4B1)
STFVAL	SSDIR	8	(8)
STGCNT	EDSECT	856	(358)
STGNDX	EDSECT	76	(4C)
STMTN	X5COM	16	(10)
STNPADJ	EDSECT	332	(14C)
STNPSTD	EDSECT	324	(144)

*POINTER.

FIELD	DSECT	DISPLACEMENT DECIMAL	(HEX)
STRADR	X5COM	200	(C8)
STRCMT	EDSECT	6	(6)
STRTLC	X5COM	1080	(438)
SUBLST	EDSECT	5	(5)
SUBSAVE	EDSECT	920	(398)
SUBSOP	EDSECT	2	(2)
SUPDNT	EDSECT	4	(4)
SVENDWKA	EDSECT	348	(15C)
SVLAST	EDSECT	772	(304)
SVMINDIF	EDSECT	360	(168)
SWITCHA	EDSECT	1022	(3FE)
SWITCHES	X5COM	60	(3C)
SWITCH1	EDSECT	0	(0)
SWITCH2	EDSECT	1	(1)
SWITCH3	EDSECT	2	(2)
SWITCH4	EDSECT	3	(3)
SWITCH5	EDSECT	4	(4)
SWITCH6	EDSECT	5	(5)
SWITCH7	EDSECT	6	(6)
SWITCH8	EDSECT	7	(7)
SWITCH9	EDSECT	8	(8)
SXMCRO	EDSECT	4	(4)
SXPRTO	EDSECT	0	(0)
SYMCNT	X5COM	54	(36)
SYMDEF	X5COM	48	(30)
SYMXRF	X5COM	52	(34)
TATTRIB	OSRDNTRY	0	(0)
TBGLN	EDSECT	256	(100)
TCNTLN	EDSECT	264	(108)
TEMPBIND	EDSECT	781	(30D)
TEMPLC	X5COM	1108	(454)
TEMPOP	EDSECT	780	(30C)
TERMS	X5COM	1200	(4B0)
TITLE	RPRINT	1	(1)
TOOMANY	X5COM	64	(40)
TPTEXT	X5COM	57	(39)
TSEDIT	MDDNTRY	3	(3)
TSRCLN	EDSECT	260	(104)
TUBEOP	X5COM	1127	(467)
TWASLC	X5COM	57	(39)
TXTPTR	X5COM	12	(C)
UADR1	X5COM	964	(3C4)
UADR2	X5COM	972	(3CC)
UBASD1	X5COM	952	(3B8)
UBASD2	X5COM	956	(3BC)
UESD	UDSECT	2	(2)
UGARB	X5COM	960	(3C0)
ULOCO	X5COM	940	(3AC)
UOPCOD	X5COM	948	(3B4)
UREG	UDSECT	8	(8)
USINGT	X5COM	238	(EE)
USPHL	X5COM	58	(3A)
USSRT	X5COM	55	(37)
UVAL	UDSECT	4	(4)
VALOUT	RPRINT	28	(1C)
VCHAIN	VSDENTRY	0	(0)
VCOMPLEX	X5COM	1217	(4C1)
VDIM	EDSECT	1062	(426)

*POINTER.

FIELD	DSECT	DISPLACEMENT DECIMAL	(HEX)
VECPTR	EDSECT	48	(30)
VEJBYTE	RPRINT	0	(0)
VENDPARN	X5COM	1320	(528)
VEOP	FARENT	0	(0)
VFLAGS	VSDENTRY	3	(3)
VGDIMEN	VSDENTRY	4	(4)
VGVECTR	VSDENTRY	1	(1)
VHIDEFOP	JTEXT	4	(4)
VHIGENOP	JTEXT	4	(4)
VHIREFOP	JTEXT	4	(4)
VJEEOF	JERRCD	4	(4)
VJEOPCOD	JERRCD	4	(4)
VJTACTR	JTEXT	4	(4)
VJTADJII	JTEXT	4	(4)
VJTAGO	JTEXT	4	(4)
VJTAGOB	JTEXT	4	(4)
VJTAIF	JTEXT	4	(4)
VJTAIFB	JTEXT	4	(4)
VJTANOP	JTEXT	4	(4)
VJTCALL	JTEXT	4	(4)
VJTCCW	JTEXT	4	(4)
VJTCMNT	JTEXT	4	(4)
VJTCNOP	JTEXT	4	(4)
VJTCOM	JTEXT	4	(4)
VJTCOPY	JTEXT	4	(4)
VJTCPKEY	JTEXT	4	(4)
VJTCPPOS	JTEXT	4	(4)
VJTCSECT	JTEXT	4	(4)
VJTCXD	JTEXT	4	(4)
VJTDC	JTEXT	4	(4)
VJTDROP	JTEXT	4	(4)
VJTDS	JTEXT	4	(4)
VJTDSECT	JTEXT	4	(4)
VJTDXD	JTEXT	4	(4)
VJTEEOF	JTEXT	4	(4)
VJTEJECT	JTEXT	4	(4)
VJTEND	JTEXT	4	(4)
VJTENTRY	JTEXT	4	(4)
VJTEOF	JTEXT	4	(4)
VJTEOFII	JTEXT	4	(4)
VJTEQU	JTEXT	4	(4)
VJTERROR	JTEXT	4	(4)
VJTEXTRN	JTEXT	4	(4)
VJTGBLA	JTEXT	4	(4)
VJTGBLB	JTEXT	4	(4)
VJTGBLC	JTEXT	4	(4)
VJTHCMNT	JTEXT	4	(4)
VJTICTL	JTEXT	4	(4)
VJTINPC	JTEXT	4	(4)
VJTISEQ	JTEXT	4	(4)
VJTLCLA	JTEXT	4	(4)
VJTLCLB	JTEXT	4	(4)
VJTLCLC	JTEXT	4	(4)
VJTLITII	JTEXT	4	(4)
VJTLITR	JTEXT	4	(4)
VJTLTDC	JTEXT	4	(4)
VJTLTEND	JTEXT	4	(4)
VJTLTLC	JTEXT	4	(4)

*POINTER.

FIELD	DSECT	DISPLACEMENT DECIMAL	(HEX)
VJTLTND	JTEXT	4	(4)
VJTLTORG	JTEXT	4	(4)
VJTMACRO	JTEXT	4	(4)
VJTMEND	JTEXT	4	(4)
VJTMEXIT	JTEXT	4	(4)
VJTMNOTE	JTEXT	4	(4)
VJTOPSYN	JTEXT	4	(4)
VJTORG	JTEXT	4	(4)
VJTPASS	JTEXT	4	(4)
VJTPEND	JTEXT	4	(4)
VJTPMOP	JTEXT	4	(4)
VJTPOP	JTEXT	4	(4)
VJTPPCH	JTEXT	4	(4)
VJTPPKEY	JTEXT	4	(4)
VJTPPPOS	JTEXT	4	(4)
VJTPREP	JTEXT	4	(4)
VJTPRINT	JTEXT	4	(4)
VJTPROTO	JTEXT	4	(4)
VJTPUNCH	JTEXT	4	(4)
VJTPUSH	JTEXT	4	(4)
VJTREPRO	JTEXT	4	(4)
VJTSETA	JTEXT	4	(4)
VJTSETB	JTEXT	4	(4)
VJTSETC	JTEXT	4	(4)
VJTSICTL	JTEXT	4	(4)
VJTSPACE	JTEXT	4	(4)
VJTSTART	JTEXT	4	(4)
VJTSYMBL	JTEXT	4	(4)
VJTSYMII	JTEXT	4	(4)
VJTTITLE	JTEXT	4	(4)
VJTUSING	JTEXT	4	(4)
VJTWXTRN	JTEXT	4	(4)
VLDICTR	VSDENTRY	1	(1)
VLDIMEN	VSDENTRY	4	(4)
VLIT	X5COM	63	(3F)
VLNGTH	VSDENTRY	4	(4)
VLODEFOP	JTEXT	4	(4)
VLOGENOP	JTEXT	4	(4)
VLONOPRN	JTEXT	4	(4)
VLOREFOP	JTEXT	4	(4)
VMAXBIT	X5COM	1233	(4D1)
VMAXCHAR	X5COM	1233	(4D1)
VMAXDEC	X5COM	1233	(4D1)
VMAXHEX	X5COM	1233	(4D1)
VMAXPARN	X5COM	1225	(4C9)
VNOPAREN	X5COM	1225	(4C9)
VNORELOC	X5COM	1208	(4B8)
VPPAD	VSDENTRY	4	(4)
VPTYP	EDSECT	1062	(426)
VPVECTR	VSDENTRY	1	(1)
VSDSLOT	EDSECT	960	(3C0)
VSDSTRT	EDSECT	948	(3B4)
VSELFDEF	X5COM	1128	(468)
VSFLG	EDSECT	854	(356)

*POINTER.

FIELD	DSECT	DISPLACEMENT DECIMAL	(HEX)
VSHIFTB	X5COM	1232	(4D0)
VSHIFTC	X5COM	1232	(4D0)
VSHIFTD	X5COM	1232	(4D0)
VSHIFTH	X5COM	1232	(4D0)
VSLS	EDSECT	1062	(426)
VSNS	EDSECT	1062	(426)
VSPACE1	RPRINT	0	(0)
VSPACE2	RPRINT	0	(0)
VSPACE3	RPRINT	0	(0)
VSRSV	EDSECT	904	(388)
VSRSV1	EDSECT	908	(38C)
VSTATUS1	X5COM	1201	(4B1)
VSTATUS2	X5COM	1201	(4B1)
VSTP1	EDSECT	1062	(426)
VSTP2	EDSECT	1062	(426)
VTFVAL	VSDENTRY	0	(0)
VTYP1	EDSECT	1062	(426)
VTYP2	EDSECT	1062	(426)
WORKAREA	X5COM	1020	(3FC)
WRPFLG	X5COM	57	(39)
XCNTRL	RPRINT	0	(0)
XDE	RPRINT	30	(1E)
XDEFOUT	RPRINT	25	(19)
XFLAG	PPIN	2	(2)
XFLAG	XRFIN	2	(2)
XLENOUT	RPRINT	10	(A)
XOPCDE	PPIN	4	(4)
XOPCDE	XRFIN	4	(4)
XRECLN	PPIN	0	(0)
XRECLN	XRFIN	0	(0)
XREFYES	X5COM	1116	(45C)
XRFENT	RPRINT	32	(20)
XRFFLG	PPIN	14	(E)
XRFFLG	XRFIN	14	(E)
XRFLEN	PPIN	17	(11)
XRFLEN	XRFIN	17	(11)
XRFNO	X5COM	56	(38)
XRFREF	RPRINT	32	(20)
XRFSTM	PPIN	15	(F)
XRFSTM	XRFIN	15	(F)
XRFSYM	PPIN	6	(6)
XRFSYM	XRFIN	6	(6)
XRFVAL	PPIN	19	(13)
XRFVAL	XRFIN	19	(13)
XSSAV	X5COM	1350	(546)
XSYMOUT	RPRINT	1	(1)
XVALOUT	RPRINT	16	(10)
X5ASW	X5COM	65	(41)
X5LSAV	X5COM	212	(D4)
X5MSW	X5COM	64	(40)
X5SW1	X5COM	57	(39)
X5VSW	X5COM	63	(3F)
ZAPIT	X5COM	63	(3F)
ZDUPSW	X5COM	1104	(450)

*POINTER.

Directory

This section serves as a cross-reference between the items in the "Method of Operation" section and the microfiche listings.

SYMBOLIC NAME	DESCRIPTION: NAME AND USE	PLM REF**	CSECT/ DSECT	MODULE/ MCROFCH
ALIGN	SAVE REGISTERS, ASSEMBLY PHASE; ALIGNMENT ROUTINE	25	IFNX5A20	IFNX5A
AOP350	IS THIS AN END STATEMENT, ASSEMBLER OPCODE PROCESSOR; ASSEMBLY PHASE; BRANCH TABLE AND EXIT ROUTINES	21	IFNX5A00	IFNX5A
AYKON	SET TEXT POINTER, ASSEMBLY PHASE; DC EVALUATION	23	IFNX5D00	IFNX5D
BKON	CLEAR BIT-LENGTH, ASSEMBLY PHASE; DC EVALUATION; PROCESS B-TYPE CONSTANTS	23	IFNX5D00	IFNX5D
BLDESD	SAVE REGISTERS IN STACK, EXTERNAL SYMBOL DICTIONARY SUBROUTINES	17	IFNX4E00	IFNX4E
BRONTYP	ENTRY POINT; DICTIONARY INTERLUDE PHASE	9	IFNX2A02	IFNX2A
CALLEND	ENTRY POINT; GENERATE PHASE DICTIONARY ROUTINES	11,12	IFNX3N00	IFNX3N
CCW100	GET ALIGNMENT CHECK BITS, ASSEMBLER OPCODE PROCESSOR; ASSEMBLY PHASE; 'CCW' STATEMENT PROCESSOR	23	IFNX5A00	IFNX5A
CKON	SET STEPPER, ASSEMBLY PHASE; DC EVALUATION; PROCESS C-TYPE CONSTANT	23	IFNX5D00	IFNX5D
COMNEND	ENTRY POINT; EDIT PHASE DICTIONARY ROUTINES	6	IFNX1J00	IFNX1J
COMSTRT	PTR TO PHASE COMMON AREA, DICTIONARY INTERLUDE PHASE	F5	INTRCOM	IFNX2A
COPY	ENTRY POINT; EDIT PHASE	3,4	IFNX1A00	IFNX1A
CRDSCT	DSECT NAME; DSECT FOR TXT CARDS, ASSEMBLY PHASE; PRINT ROUTINE		CRDSCT	IFNX5P
CSTKENT	ENTRY POINT; EDIT PHASE	4	IFNX1A10	IFNX1A
CSTKEXT	ENTRY POINT; EDIT PHASE	4	IFNX1A10	IFNX1A
CURFLE	FILE 1, POST PROCESSOR PHASE	F11	X6ACOMM	IFNX6A
CURFLE2	ALIAS FOR CFL1VV+2. FILE 2, POST PROCESSOR PHASE	F11	X6ACOMM	IFNX6A
CURFLE3	ALIAS FOR CFL2VV+2. FILE 3, POST PROCESSOR PHASE	F11	X6ACOMM	IFNX6A
CURRDICT	PTR TO HEADER OF CURRENT DICT, XKE MACRO GENERATOR	F8	GENCOM	IFNX3A
CURRDICT	PTR TO HEADER OF CURRENT DICT, GENERATE PHASE DICTIONARY ROUTINES	F8,12	GENCOM	IFNX3N
CURRGLBL	PTR TO CURRENT GLOBAL VECTOR, XKE MACRO GENERATOR	13	GENCOM	IFNX3A
CURRGLBL	PTR TO CURRENT GLOBAL VECTOR, GENERATE PHASE DICTIONARY ROUTINES	13	GENCOM	IFNX3N
CURRKEYD	PTR TO CURRENT KEYWD PARAM VCTR, XKE MACRO GENERATOR	13	GENCOM	IFNX3A
CURRKEYD	PTR TO CURRENT KEYWD PARAM VCTR, GENERATE PHASE DICTIONARY ROUTINES	12,13	GENCOM	IFNX3N

*DATA AREA. SEE DATA AREA SECTION FOR DETAILED LAYOUT.

**EXPLANATION OF PLM NUMBERED REFERENCES:
A SINGLE NUMERAL REFERS TO AN OPERATIONS DIAGRAM IN THE METHOD OF OPERATIONS SECTION.
'F', FOLLOWED BY A NUMERAL, REFERS TO A FIGURE IN THE PROGRAM ORGANIZATION SECTION.

SYMBOLIC NAME	DESCRIPTION: NAME AND USE	PLM REF**	CSECT/ DSECT	MODULE/ MCROFCH
CURRLOCL	PTR TO CURRENT LOCAL DICTIONARY, XKE MACRO GENERATOR	13	GENCOM	IFNX3A
CURRLOCL	PTR TO CURRENT LOCAL DICTIONARY, GENERATE PHASE DICTIONARY ROUTINES	12,13	GENCOM	IFNX3N
CURRPARM	PTR TO CURRENT PARAM TABLE, XKE MACRO GENERATOR	F8,13	GENCOM	IFNX3A
CURRPARM	PTR TO CURRENT PARAM TABLE, SYMBOL RESOLUTION PREPROCESSOR	F8	GENCOM	IFNX3B
CURRPARM	PTR TO CURRENT PARAM TABLE, GENERATE PHASE DICTIONARY ROUTINES	F8,12	GENCOM	IFNX3N
CURRPOST	PTR TO CURRENT POSIT PARAM VCTR, XKE MACRO GENERATOR	13	GENCOM	IFNX3A
CURRPOST	PTR TO CURRENT POSIT PARAM VCTR, GENERATE PHASE DICTIONARY ROUTINES	12	GENCOM	IFNX3N
CURRSEQS	PTR TO CURRENT SEQ SYMB REF DCT, XKE MACRO GENERATOR	12	GENCOM	IFNX3A
CURRSEQS	PTR TO CURRENT SEQ SYMB REF DCT, GENERATE PHASE DICTIONARY ROUTINES	12	GENCOM	IFNX3N
DATAPTR	DATA AREA POINTER, POST PROCESSOR PHASE	F11	X6ACOMM	IFNX6A
DCEVAL	SAVE ENTRY REGISTERS, ASSEMBLY PHASE; DC EVALUATION; INITIALIZATION	23	IFNX5D00	IFNX5D
DCSTK	DSECT NAME; EDIT PHASE		DCSTK	IFNX1A
DC0100	GO GET GOOD OPERAND COUNT, ASSEMBLER OPCODE PROCESSOR; ASSEMBLY PHASE; 'DC & DS' STATEMENT; CALL DC	23	IFNX5A00	IFNX5A
DKON	SET PARAMETER POINTER, ASSEMBLY PHASE; DC EVALUATION; PROCESS L-, D-, E-, F-, H-TYPE CONSTANTS	23	IFNX5D00	IFNX5D
DMSASM	VM/370 INTERFACE ROUTINE	27	DMSASM	DMSASM
DRIVER	EXIT IF UNRECOVERABLE ERROR, MACHINE INSTRUCTION PROCESSOR	22	IFNX5M00	IFNX5M
DROP00	GET OPERAND POINTER, ASSEMBLER OPCODE PROCESSOR; ASSEMBLY PHASE; 'DROP' STATEMENT PROCESSOR	24	IFNX5A00	IFNX5A
DRSTK	DSECT NAME; EDIT PHASE		DRSTK	IFNX1A
DSECT10	DSECT NAME; ADDRESS, SYMBOL RESOLUTION PHASE DC/DS EVALUATION		DSECT10	IFNX4D
DSECT10	DSECT NAME; ,, EXTERNAL SYMBOL DICTIONARY SUBROUTINES		DSECT10	IFNX4E
DSECT10	DSECT NAME; ADDRESS, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL		DSECT10	IFNX4M
DSECT10	DSECT NAME; ADDRESS, SYMBOL RESOLUTION PHASE DC/DS EVALUATION		DSECT10	IFNX4N
DSECT10	DSECT NAME; ADDRESS, SYMBOL TABLE SUBROUTINES		DSECT10	IFNX4S

*DATA AREA. SEE DATA AREA SECTION FOR DETAILED LAYOUT.
**EXPLANATION OF PLM NUMBERED REFERENCES:
A SINGLE NUMERAL REFERS TO AN OPERATIONS DIAGRAM IN THE METHOD OF OPERATIONS SECTION.
'F', FOLLOWED BY A NUMERAL, REFERS TO A FIGURE IN THE PROGRAM ORGANIZATION SECTION.

SYMBOLIC NAME	DESCRIPTION: NAME AND USE	PLM REF**	CSECT/ DSECT	MODULE/ MCROFCH
DSECT10	DSECT NAME; ADDRESS, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL		DSECT10	IFNX4T
DSECT10	DSECT NAME; ADDRESS, EXPRESSION EVALUATION SUBROUTINE		DSECT10	IFNX4V
DSECT10	DSECT NAME; DEFINE RESOLUTION DATA DSECT, ASSEMBLER OPCODE PROCESSOR; ASSEMBLY PHASE; COPY CODE	21,23,24	DSECT10	IFNX5A
DSECT10	DSECT NAME; ASSEMBLY PHASE; MAINLINE CONTROL; DSECT10 DEFINITION (RSYMRCD)	21	DSECT10	IFNX5C
DSECT10	DSECT NAME; EXPRESSION EVALUATION SUBROUTINE		DSECT10	IFNX5V
DSECT11	DSECT NAME; ADDRESS, SYMBOL RESOLUTION PHASE DC/DS EVALUATION		DSECT11	IFNX4D
DSECT11	DSECT NAME; ADDRESS, EXTERNAL SYMBOL DICTIONARY SUBROUTINES		DSECT11	IFNX4E
DSECT11	DSECT NAME; ADDRESS, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL		DSECT11	IFNX4M
DSECT11	DSECT NAME; ADDRESS, SYMBOL RESOLUTION PHASE DC/DS EVALUATION		DSECT11	IFNX4N
DSECT11	DSECT NAME; ADDRESS, SYMBOL TABLE SUBROUTINES		DSECT11	IFNX4S
DSECT11	DSECT NAME; ADDRESS, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL		DSECT11	IFNX4T
DSECT11	DSECT NAME; ADDRESS, EXPRESSION EVALUATION SUBROUTINE		DSECT11	IFNX4V
DSECT12	DSECT NAME; ADDRESS, SYMBOL RESOLUTION PHASE DC/DS EVALUATION		DSECT12	IFNX4D
DSECT12	DSECT NAME; ADDRESS, EXTERNAL SYMBOL DICTIONARY SUBROUTINES		DSECT12	IFNX4E
DSECT12	DSECT NAME; ADDRESS, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL		DSECT12	IFNX4M
DSECT12	DSECT NAME; ADDRESS, SYMBOL RESOLUTION PHASE DC/DS EVALUATION		DSECT12	IFNX4N
DSECT12	DSECT NAME; ADDRESS, SYMBOL TABLE SUBROUTINES		DSECT12	IFNX4S
DSECT12	DSECT NAME; ADDRESS, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL		DSECT12	IFNX4T
DSECT12	DSECT NAME; ADDRESS, EXPRESSION EVALUATION SUBROUTINE		DSECT12	IFNX4V
DSECT14	DSECT NAME; ADDRESS, SYMBOL RESOLUTION PHASE DC/DS EVALUATION	17	DSECT14	IFNX4D
DSECT14	DSECT NAME; ADDRESS, EXTERNAL SYMBOL DICTIONARY SUBROUTINES	17	DSECT14	IFNX4E

*DATA AREA. SEE DATA AREA SECTION FOR DETAILED LAYOUT.
**EXPLANATION OF PLM NUMBERED REFERENCES:
A SINGLE NUMERAL REFERS TO AN OPERATIONS DIAGRAM IN THE METHOD OF OPERATIONS SECTION.
'F', FOLLOWED BY A NUMERAL, REFERS TO A FIGURE IN THE PROGRAM ORGANIZATION SECTION.

SYMBOLIC NAME	DESCRIPTION: NAME AND USE	PLM REF**	CSECT/ DSECT	MODULE/ MCROFCH
DSECT14	DSECT NAME; ADDRESS, EXTERNAL SYMBOL DICTIONARY SUBROUTINES			
DSECT14	DSECT NAME; ADDRESS, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL	17,19	DSECT14	IFNX4M
DSECT14	DSECT NAME; ADDRESS, SYMBOL RESOLUTION PHASE DC/DS EVALUATION		DSECT14	IFNX4N
DSECT14	DSECT NAME; ADDRESS, SYMBOL TABLE SUBROUTINES		DSECT14	IFNX4S
DSECT14	DSECT NAME; ADDRESS, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL		DSECT14	IFNX4T
DSECT14	DSECT NAME; ADDRESS, EXPRESSION EVALUATION SUBROUTINE		DSECT14	IFNX4V
DSECT14	DSECT NAME; INPUT POINTER, ASSEMBLY PHASE; MAINLINE CONTROL; X5COM COPY CODE		DSECT14	IFNX5C
DSECT14	DSECT NAME; INPUT POINTER, DC FIXED-FLOATING POINT CONVERSION		DSECT14	IFNX5F
DSECT14	DSECT NAME; INPUT POINTER, EXPRESSION EVALUATION SUBROUTINE		DSECT14	IFNX5V
DSECT15	DSECT NAME; ADDRESS, SYMBOL RESOLUTION PHASE DC/DS EVALUATION		DSECT15	IFNX4D
DSECT15	DSECT NAME; ADDRESS, EXTERNAL SYMBOL DICTIONARY SUBROUTINES		DSECT15	IFNX4E
DSECT15	DSECT NAME; ADDRESS, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL		DSECT15	IFNX4M
DSECT15	DSECT NAME; ADDRESS, SYMBOL RESOLUTION PHASE DC/DS EVALUATION		DSECT15	IFNX4N
DSECT15	DSECT NAME; ADDRESS, SYMBOL TABLE SUBROUTINES		DSECT15	IFNX4S
DSECT15	DSECT NAME; ADDRESS, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL		DSECT15	IFNX4T
DSECT15	DSECT NAME; ADDRESS, EXPRESSION EVALUATION SUBROUTINE		DSECT15	IFNX4V
DSECT2	DSECT NAME; ADDRESS, SYMBOL RESOLUTION PHASE DC/DS EVALUATION		DSECT2	IFNX4D
DSECT2	DSECT NAME; ADDRESS, EXTERNAL SYMBOL DICTIONARY SUBROUTINES		DSECT2	IFNX4E
DSECT2	DSECT NAME; TERM STACK POINTER, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL		DSECT2	IFNX4M
DSECT2	DSECT NAME; ADDRESS, SYMBOL RESOLUTION PHASE DC/DS EVALUATION		DSECT2	IFNX4N
DSECT2	DSECT NAME; TERM STACK POINTER, SYMBOL TABLE SUBROUTINES		DSECT2	IFNX4S

*DATA AREA. SEE DATA AREA SECTION FOR DETAILED LAYOUT.
**EXPLANATION OF PLM NUMBERED REFERENCES:
A SINGLE NUMERAL REFERS TO AN OPERATIONS DIAGRAM IN THE METHOD OF OPERATIONS SECTION.
'F', FOLLOWED BY A NUMERAL, REFERS TO A FIGURE IN THE PROGRAM ORGANIZATION SECTION.

SYMBOLIC NAME	DESCRIPTION: NAME AND USE	PLM REF**	CSECT/ DSECT	MODULE/ MCROFCH
DSECT2	DSECT NAME; ADDRESS, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL		DSECT2	IFNX4T
DSECT2	DSECT NAME; TERM STACK POINTER, EXPRESSION EVALUATION SUBROUTINE		DSECT2	IFNX4V
DSECT2	DSECT NAME; TERM STACK POINTER, ASSEMBLY PHASE; MAINLINE CONTROL; X5COM COPY CODE		DSECT2	IFNX5C
DSECT2	DSECT NAME; TERM STACK POINTER, DC FIXED-FLOATING POINT CONVERSION		DSECT2	IFNX5F
DSECT2	DSECT NAME; TERM STACK POINTER, EXPRESSION EVALUATION SUBROUTINE		DSECT2	IFNX5V
DSECT3	DSECT NAME; RELOCATION LIST POINTER, SYMBOL RESOLUTION PHASE DC/DS EVALUATION		DSECT3	IFNX4D
DSECT3	DSECT NAME; RELOCATION LIST POINTER, EXTERNAL SYMBOL DICTIONARY SUBROUTINES		DSECT3	IFNX4E
DSECT3	DSECT NAME; ADDRESS, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL		DSECT3	IFNX4M
DSECT3	DSECT NAME; RELOCATION LIST POINTER, SYMBOL RESOLUTION PHASE DC/DS EVALUATION		DSECT3	IFNX4N
DSECT3	DSECT NAME; RELOCATION LIST POINTER, SYMBOL TABLE SUBROUTINES		DSECT3	IFNX4S
DSECT3	DSECT NAME; RELOCATION LIST POINTER, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL		DSECT3	IFNX4T
DSECT3	DSECT NAME; RELOCATION LIST POINTER, EXPRESSION EVALUATION SUBROUTINE		DSECT3	IFNX4V
DSECT3	DSECT NAME; RELOCATION LIST POINTER, ASSEMBLY PHASE; MAINLINE CONTROL; X5COM COPY CODE		DSECT3	IFNX5C
DSECT3	DSECT NAME; RELOCATION LIST POINTER, DC FIXED-FLOATING POINT CONVERSION		DSECT3	IFNX5F
DSECT3	DSECT NAME; RELOCATION LIST POINTER, EXPRESSION EVALUATION SUBROUTINE		DSECT3	IFNX5V
DSECT5	DSECT NAME; ADDRESS, SYMBOL RESOLUTION PHASE DC/DS EVALUATION		DSECT5	IFNX4D
DSECT5	DSECT NAME; ADDRESS, EXTERNAL SYMBOL DICTIONARY SUBROUTINES		DSECT5	IFNX4E
DSECT5	DSECT NAME; INPUT POINTER, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL		DSECT5	IFNX4M
DSECT5	DSECT NAME; ADDRESS, SYMBOL RESOLUTION PHASE DC/DS EVALUATION		DSECT5	IFNX4N
DSECT5	DSECT NAME; INPUT POINTER, SYMBOL TABLE SUBROUTINES		DSECT5	IFNX4S
DSECT5	DSECT NAME; ADDRESS, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL		DSECT5	IFNX4T
DSECT5	DSECT NAME; INPUT POINTER, EXPRESSION EVALUATION SUBROUTINE		DSECT5	IFNX4V
DSECT5	DSECT NAME; INPUT POINTER, ASSEMBLY PHASE; MAINLINE CONTROL; X5COM COPY CODE		DSECT5	IFNX5C

*DATA AREA. SEE DATA AREA SECTION FOR DETAILED LAYOUT.

**EXPLANATION OF PLM NUMBERED REFERENCES:
A SINGLE NUMERAL REFERS TO AN OPERATIONS DIAGRAM IN THE METHOD OF OPERATIONS SECTION.
'F', FOLLOWED BY A NUMERAL, REFERS TO A FIGURE IN THE PROGRAM ORGANIZATION SECTION.

SYMBOLIC NAME	DESCRIPTION: NAME AND USE	PLM REF**	CSECT/ DSECT	MODULE/ MCROFCH
DSECT5	DSECT NAME; INPUT POINTER, DC FIXED-FLOATING POINT CONVERSION		DSECT5	IFNX5F
DSECT5	DSECT NAME; INPUT POINTER, EXPRESSION EVALUATION SUBROUTINE		DSECT5	IFNX5V
DSECT6	DSECT NAME; ADDRESS, SYMBOL RESOLUTION PHASE DC/DS EVALUATION		DSECT6	IFNX4D
DSECT6	DSECT NAME; ADDRESS, EXTERNAL SYMBOL DICTIONARY SUBROUTINES	17,18	DSECT6	IFNX4E
DSECT6	DSECT NAME; ADDRESS, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL		DSECT6	IFNX4M
DSECT6	DSECT NAME; ADDRESS, SYMBOL RESOLUTION PHASE DC/DS EVALUATION		DSECT6	IFNX4N
DSECT6	DSECT NAME; ADDRESS, SYMBOL TABLE SUBROUTINES		DSECT6	IFNX4S
DSECT6	DSECT NAME; ADDRESS, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL		DSECT6	IFNX4T
DSECT6	DSECT NAME; ADDRESS, EXPRESSION EVALUATION SUBROUTINE		DSECT6	IFNX4V
DSECT7	DSECT NAME; ADDRESS, SYMBOL RESOLUTION PHASE DC/DS EVALUATION		DSECT7	IFNX4D
DSECT7	DSECT NAME; ADDRESS, EXTERNAL SYMBOL DICTIONARY SUBROUTINES		DSECT7	IFNX4E
DSECT7	DSECT NAME; RESUME DSECT, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL		DSECT7	IFNX4M
DSECT7	DSECT NAME; ADDRESS, SYMBOL RESOLUTION PHASE DC/DS EVALUATION		DSECT7	IFNX4N
DSECT7	DSECT NAME; RESUME DSECT, SYMBOL TABLE SUBROUTINES		DSECT7	IFNX4S
DSECT7	DSECT NAME; ADDRESS, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL		DSECT7	IFNX4T
DSECT7	DSECT NAME; RESUME DSECT, EXPRESSION EVALUATION SUBROUTINE		DSECT7	IFNX4V
DS0100	INDICATE ENTRY IS A DS, ASSEMBLER OPCODE PROCESSOR; ASSEMBLY PHASE; 'DC & DS' STATEMENT; CALL DC EV	23	IFNX5A00	IFNX5A
DXD100	INDICATE DXD FOR DCEVAL, ASSEMBLER OPCODE PROCESSOR; ASSEMBLY PHASE; 'DXD' STATEMENT PROCESSOR	23	IFNX5A00	IFNX5A
EDITSYSM	SKIP IT ALL IF OVERLAP OCCURRED, EDIT PHASE DICTIONARY ROUTINES	6	IFNX1J00	IFNX1J
*EDSECT	DSECT NAME; EDIT PHASE COMMON AREA, EDIT PHASE		EDSECT	IFNX1A
*EDSECT	DSECT NAME; EDIT PHASE COMMON AREA, EDIT PHASE DICTIONARY ROUTINES		EDSECT	IFNX1J

*DATA AREA. SEE DATA AREA SECTION FOR DETAILED LAYOUT.
**EXPLANATION OF PLM NUMBERED REFERENCES:
A SINGLE NUMERAL REFERS TO AN OPERATIONS DIAGRAM IN THE METHOD OF OPERATIONS SECTION.
'F', FOLLOWED BY A NUMERAL, REFERS TO A FIGURE IN THE PROGRAM ORGANIZATION SECTION.

SYMBOLIC NAME	DESCRIPTION: NAME AND USE	PLM REF**	CSECT/ DSECT	MODULE/ MCROFCH
*EDSECT	DSECT NAME; EDIT PHASE COMMON AREA, CONDITIONAL ASSEMBLY POSTFIX ROUTINE		EDSECT	IFNX1S
EJECT0	LOAD NEGATIVE VALUE IN REGISTER, ASSEMBLER OPCODE PROCESSOR; ASSEMBLY PHASE; 'SPACE' AND 'EJECT' STA	24	IFNX5A00	IFNX5A
*ENDFIL	DSECT NAME; EDIT PHASE DICTIONARY ROUTINES		ENDFIL	IFNX1J
*ENDFIL	DSECT NAME; DICTIONARY INTERLUDE PHASE		ENDFIL	IFNX2A
*ENDSEG	DSECT NAME; EDIT PHASE DICTIONARY ROUTINES	5	ENDSEG	IFNX1J
*ENDSEG	DSECT NAME; DICTIONARY INTERLUDE PHASE		ENDSEG	IFNX2A
ENDSEGB	ENTRY POINT; DICTIONARY INTERLUDE PHASE	10,8	IFNX2A00	IFNX2A
END100	GET OPERAND POINTER, ASSEMBLER OPCODE PROCESSOR; ASSEMBLY PHASE; 'END' STATEMENT PROCESSOR	24	IFNX5A00	IFNX5A
ENTER	SAVE REGISTERS IN STACK, SYMBOL TABLE SUBROUTINES	17,20,27	IFNX4S00	IFNX4S
ENTRY	SAVE REGISTERS IN STACK, EXTERNAL SYMBOL DICTIONARY SUBROUTINES	17	IFNX4E00	IFNX4E
ENTRY	GET ADDRESS OF PIVOT, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL	20	IFNX4M00	IFNX4M
ENTRY0	IS OPERAND BLANK, ASSEMBLER OPCODE PROCESSOR; ASSEMBLY PHASE; 'ENTRY & EXTRN' STATEMENT PROCESSOR	24	IFNX5A00	IFNX5A
EOFIIS	SYMBOL RESOLUTION PHASE MAIN LINE CONTROL	20,27	IFNX4M00	IFNX4M
EQU100	GET NAME RECORD PTR, ASSEMBLER OPCODE PROCESSOR; ASSEMBLY PHASE; 'EQU' STATEMENT PROCESSOR	24	IFNX5A00	IFNX5A
ERRBLK	PTR TO ERROR RECORD BUFFER, DICTIONARY INTERLUDE PHASE	F5	INTRCOM	IFNX2A
*ERRIN	DSECT NAME; ERROR INPUT RECORD FORMAT, ASSEMBLY PHASE; MAINLINE CONTROL; POST PROCESSOR RECORD DEFNS (BMDSECTS)		ERRIN	IFNX5C
*ERRIN	DSECT NAME; ERROR INPUT RECORD FORMAT, EXPRESSION EVALUATION SUBROUTINE		ERRIN	IFNX5V
*ERRMESS	DSECT NAME; DSECT DEFINED FOR ERRSTK, EDIT PHASE		ERRMESS	IFNX1A
*ERRMESS	DSECT NAME; DSECT DEFINED FOR ERRSTK, EDIT PHASE DICTIONARY ROUTINES		ERRMESS	IFNX1J
*ERRMESS	DSECT NAME; DSECT DEFINED FOR ERRSTK, CONDITIONAL ASSEMBLY POSTFIX ROUTINE		ERRMESS	IFNX1S
*ERRMESS	DSECT NAME; XKE MACRO GENERATOR		ERRMESS	IFNX3A
*ERRMESS	DSECT NAME; GENERATE PHASE DICTIONARY ROUTINES		ERRMESS	IFNX3N
ERROR0	SET ERROR RECORDS PRESENT, ASSEMBLER OPCODE PROCESSOR; ASSEMBLY PHASE; ERROR RECORD PROCESSING	21	IFNX5A00	IFNX5A

*DATA AREA. SEE DATA AREA SECTION FOR DETAILED LAYOUT.
**EXPLANATION OF PLM NUMBERED REFERENCES:
A SINGLE NUMERAL REFERS TO AN OPERATIONS DIAGRAM IN THE METHOD OF OPERATIONS SECTION.
'F', FOLLOWED BY A NUMERAL, REFERS TO A FIGURE IN THE PROGRAM ORGANIZATION SECTION.

SYMBOLIC NAME	DESCRIPTION: NAME AND USE	PLM REF**	CSECT/ DSECT	MODULE/ MCROFCH
ESDBLK1	ESD BLOCK 1, SYMBOL RESOLUTION PHASE DC/DS EVALUATION	17	DSECT7	IFNX4D
ESDBLK1	ESD BLOCK 1, EXTERNAL SYMBOL DICTIONARY SUBROUTINES	17,18	DSECT7	IFNX4E
ESDBLK1	ESD BLOCK 1, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL	17	DSECT7	IFNX4M
ESDBLK1	ESD BLOCK 1, SYMBOL TABLE SUBROUTINES	17	DSECT7	IFNX4S
ESDBLK2	ESD BLOCK 2, SYMBOL RESOLUTION PHASE DC/DS EVALUATION	17	DSECT7	IFNX4D
ESDBLK2	ESD BLOCK 2, EXTERNAL SYMBOL DICTIONARY SUBROUTINES	17,18	DSECT7	IFNX4E
ESDBLK2	ESD BLOCK 2, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL	17	DSECT7	IFNX4M
ESDBLK2	ESD BLOCK 2, SYMBOL TABLE SUBROUTINES	17	DSECT7	IFNX4S
ESYSMAC	BRANCH IF SYS MACROS ALL EDITED, EDIT PHASE	6	IFNX1A30	IFNX1A
EVAL	ENTRY POINT; XKE MACRO GENERATOR	13	IFNX3A03	IFNX3A
EXTRN	SAVE REGISTERS IN STACK, EXTERNAL SYMBOL DICTIONARY SUBROUTINES	17,20,27	IFNX4E00	IFNX4E
EXTRN	TYPE, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL	20,27	IFNX4M00	IFNX4M
EXTRN0	ALIAS FOR ENTRY0. DEFINE EXTRN ERTRY POINT, ASSEMBLER OPCODE PROCESSOR; ASSEMBLY PHASE; 'ENTRY & EXTRN' STATEMENT PROC	24	IFNX5A00	IFNX5A
*FARENT	DSECT NAME; MACHINE INSTRUCTION PROCESSOR		FARENT	IFNX5M
FGTXE	DSECT NAME; ENTRY IN FGT FOR EXTENDED OPS, MACHINE INSTRUCTION PROCESSOR		FGTXE	IFNX5M
FILE1	ALIAS FOR JFLEBLK1-JCOMMON. FILE 1 CODE, EDIT PHASE DICTIONARY ROUTINES	3	J	IFNX1J
FILE1	ALIAS FOR JFLEBLK1-JCOMMON. FILE 1 CODE, DICTIONARY INTERLUDE PHASE	10	J	IFNX2A
FILE1	ALIAS FOR JFLEBLK1-JCOMMON. FILE 1 CODE, GENERATE PHASE DICTIONARY ROUTINES	11,12	J	IFNX3N
FILE2	ALIAS FOR JFLEBLK2-JCOMMON. FILE 2 CODE, EDIT PHASE DICTIONARY ROUTINES	3	J	IFNX1J
FILE2	ALIAS FOR JFLEBLK2-JCOMMON. FILE 2 CODE, DICTIONARY INTERLUDE PHASE	10,9	J	IFNX2A
FILE2	ALIAS FOR JFLEBLK2-JCOMMON. FILE 2 CODE, XKE MACRO GENERATOR	13	J	IFNX3A
FILE2	ALIAS FOR JFLEBLK2-JCOMMON. FILE 2 CODE, GENERATE PHASE DICTIONARY ROUTINES	11,13	J	IFNX3N
FILE3	ALIAS FOR JFLEBLK3-JCOMMON. FILE 3 CODE, EDIT PHASE DICTIONARY ROUTINES	3,5,6	J	IFNX1J
FILE3	ALIAS FOR JFLEBLK3-JCOMMON. FILE 3 CODE, GENERATE PHASE DICTIONARY ROUTINES	11	J	IFNX3N

*DATA AREA. SEE DATA AREA SECTION FOR DETAILED LAYOUT.
**EXPLANATION OF PLM NUMBERED REFERENCES:
A SINGLE NUMERAL REFERS TO AN OPERATIONS DIAGRAM IN THE METHOD OF OPERATIONS SECTION.
'F', FOLLOWED BY A NUMERAL, REFERS TO A FIGURE IN THE PROGRAM ORGANIZATION SECTION.

SYMBOLIC NAME	DESCRIPTION: NAME AND USE	PLM REF**	CSECT/ DSECT	MODULE/ MCROFCH
FIND	GET NAME, SYMBOL TABLE SUBROUTINES	17,19,20	IFNX4S00	IFNX4S
FREEEND	PTR TO HIGH END OF WORK AREA, DICTIONARY INTERLUDE PHASE	F5	INTRCOM	IFNX2A
FREESTRT	PTR TO LOW END OF WORK AREA, DICTIONARY INTERLUDE PHASE	F5	INTRCOM	IFNX2A
*GBLDEF	DSECT NAME; EDIT PHASE DICTIONARY ROUTINES	5	GBLDEF	IFNX1J
GBLDEF	ENTRY POINT; DICTIONARY INTERLUDE PHASE	10,8	IFNX2A00	IFNX2A
GBLDEFN	DSECT NAME; DICTIONARY INTERLUDE PHASE		GBLDEFN	IFNX2A
GBLDICTR	ENTRY POINT; GENERATE PHASE DICTIONARY ROUTINES	13	IFNX3N00	IFNX3N
GBLDICTS	ENTRY POINT; GENERATE PHASE DICTIONARY ROUTINES	13	IFNX3N00	IFNX3N
*GBLNTRY	DSECT NAME; EDIT PHASE DICTIONARY ROUTINES	5	GBLNTRY	IFNX1J
GBLSK	PTR TO START OF GLOBAL VECTOR, DICTIONARY INTERLUDE PHASE	F5,10	INTRCOM	IFNX2A
GDEND	PTR TO CURRENT END OF GBL DIR, DICTIONARY INTERLUDE PHASE	F5	INTRCOM	IFNX2A
*GDNTRY	DSECT NAME; DICTIONARY INTERLUDE PHASE	10	GDNTRY	IFNX2A
GDSTRT	PTR TO START OF GBL DIRECTORY, DICTIONARY INTERLUDE PHASE	F5,10	INTRCOM	IFNX2A
GDSTRT	PTR TO START OF GBL DICTIONARY, XKE MACRO GENERATOR	F8,13	GENCOM	IFNX3A
GDSTRT	PTR TO START OF GBL DICTIONARY, SYMBOL RESOLUTION PREPROCESSOR	F8	GENCOM	IFNX3B
GDSTRT	PTR TO START OF GBL DICTIONARY, GENERATE PHASE DICTIONARY ROUTINES	F8	GENCOM	IFNX3N
GENCOM	DSECT NAME; XKE MACRO GENERATOR		GENCOM	IFNX3A
GENCOM	DSECT NAME; SYMBOL RESOLUTION PREPROCESSOR		GENCOM	IFNX3B
GENCOM	DSECT NAME; GENERATE PHASE DICTIONARY ROUTINES	13	GENCOM	IFNX3N
GENFLD	ENTRY POINT; XKE MACRO GENERATOR	13	IFNX3A00	IFNX3A
GENSTRNG	ENTRY POINT; XKE MACRO GENERATOR	13	IFNX3A03	IFNX3A
GSCAN	ENTRY POINT; EDIT PHASE	3	IFNX1A10	IFNX1A
GSHASHER	CLEAR WORK AREA FOR VARB NAME, DICTIONARY INTERLUDE PHASE	10	IFNX2A00	IFNX2A
GSYSVALS	DSECT NAME; GENERATE PHASE DICTIONARY ROUTINES		GSYSVALS	IFNX3N
GTRGTR	READ RECORDS, POST PROCESSOR PHASE	26	IFNX6A00	IFNX6A
HASH	ENTRY POINT; DICTIONARY INTERLUDE PHASE	9	IFNX2A00	IFNX2A
IASGN	IS IT 4-BIT OR 8-BIT FIELD, MACHINE INSTRUCTION PROCESSOR	22	IFNX5M00	IFNX5M

*DATA AREA. SEE DATA AREA SECTION FOR DETAILED LAYOUT.
**EXPLANATION OF PLM NUMBERED REFERENCES:
A SINGLE NUMERAL REFERS TO AN OPERATIONS DIAGRAM IN THE METHOD OF OPERATIONS SECTION.
'F', FOLLOWED BY A NUMERAL, REFERS TO A FIGURE IN THE PROGRAM ORGANIZATION SECTION.

SYMBOLIC NAME	DESCRIPTION: NAME AND USE	PLM REF**	CSECT/ DSECT	MODULE/ MCROFCH
ICTL	ENTRY POINT; EDIT PHASE	3,4	IFNX1A00	IFNX1A
IDR	DSECT NAME; ASSEMBLER INITILIZATION; DCBD MACRO		IDR	IFOX0D
IFNX1A00	CSECT NAME; EDIT PHASE	2	IFNX1A00	IFNX1A
IFNX1A10	CSECT NAME; EDIT PHASE		IFNX1A10	IFNX1A
IFNX1A20	CSECT NAME; EDIT PHASE		IFNX1A20	IFNX1A
IFNX1A30	CSECT NAME; EDIT PHASE		IFNX1A30	IFNX1A
IFNX1J00	CSECT NAME; EDIT PHASE DICTIONARY ROUTINES	2,5	IFNX1J00	IFNX1J
IFNX1KUN	CSECT NAME; OP CODE TABLE	3	IFNX1KUN	IFNX1K
IFNX1S00	CSECT NAME; CONDITIONAL ASSEMBLY POSTFIX ROUTINE	2,3	IFNX1S00	IFNX1S
IFNX2A00	CSECT NAME; DICTIONARY INTERLUDE PHASE	2	IFNX2A00	IFNX2A
IFNX2A02	CSECT NAME; DICTIONARY INTERLUDE PHASE		IFNX2A02	IFNX2A
IFNX3A00	CSECT NAME; XKE MACRO GENERATOR	11,2	IFNX3A00	IFNX3A
IFNX3A03	CSECT NAME; XKE MACRO GENERATOR		IFNX3A03	IFNX3A
IFNX3B00	CSECT NAME; SYMBOL RESOLUTION PREPROCESSOR	14,15,17	IFNX3B00	IFNX3B
IFNX3KUN	CSECT NAME; OP CODE TABLE		IFNX3KUN	IFNX3K
IFNX3N00	CSECT NAME; GENERATE PHASE DICTIONARY ROUTINES		IFNX3N00	IFNX3N
IFNX4D00	CSECT NAME; SYMBOL RESOLUTION PHASE DC/DS EVALUATION	15,20	IFNX4D00	IFNX4D
IFNX4D00	CSECT NAME; SYMBOL RESOLUTION PHASE DC/DS EVALUATION	15,20	IFNX4D00	IFNX4D
IFNX4E00	CSECT NAME; EXTERNAL SYMBOL DICTIONARY SUBROUTINES	15	IFNX4E00	IFNX4E
IFNX4E00	CSECT NAME; EXTERNAL SYMBOL DICTIONARY SUBROUTINES	15	IFNX4E00	IFNX4E
IFNX4M00	CSECT NAME; SYMBOL RESOLUTION PHASE MAIN LINE CONTROL	15,20	IFNX4M00	IFNX4M
IFNX4M00	CSECT NAME; SYMBOL RESOLUTION PHASE MAIN LINE CONTROL	15,20	IFNX4M00	IFNX4M
IFNX4N00	CSECT NAME; SYMBOL RESOLUTION PHASE DC/DS EVALUATION		IFNX4N00	IFNX4N
IFNX4S00	CSECT NAME; SYMBOL TABLE SUBROUTINES	15	IFNX4S00	IFNX4S

*DATA AREA. SEE DATA AREA SECTION FOR DETAILED LAYOUT.
**EXPLANATION OF PLM NUMBERED REFERENCES:
A SINGLE NUMERAL REFERS TO AN OPERATIONS DIAGRAM IN THE METHOD OF OPERATIONS SECTION.
'F', FOLLOWED BY A NUMERAL, REFERS TO A FIGURE IN THE PROGRAM ORGANIZATION SECTION.

SYMBOLIC NAME	DESCRIPTION: NAME AND USE	PLM REF**	CSECT/ DSECT	MODULE/ MCROFCH
IFNX4S00	CSECT NAME; SYMBOL TABLE SUBROUTINES	15	IFNX4S00	IFNX4S
IFNX4T00	CSECT NAME; SYMBOL RESOLUTION PHASE MAIN LINE CONTROL		IFNX4T00	IFNX4T
IFNX4V00	CSECT NAME; EXPRESSION EVALUATION SUBROUTINE	15	IFNX4V00	IFNX4V
IFNX5A00	CSECT NAME; ASSEMBLER OPCODE PROCESSOR; ASSEMBLY PHASE; INITIALIZATION	21	IFNX5A00	IFNX5A
IFNX5A00	CSECT NAME; ASSEMBLY PHASE; PATCH AREA	21	IFNX5A00	IFNX5A
IFNX5A20	CSECT NAME; ASSEMBLY PHASE; ALIGNMENT ROUTINE		IFNX5A20	IFNX5A
IFNX5A30	CSECT NAME; ASSEMBLY PHASE; LOCATION COUNTER UPDATE ROUTINE		IFNX5A30	IFNX5A
IFNX5A40	CSECT NAME; ASSEMBLY PHASE; RLD OUTPUT ROUTINE		IFNX5A40	IFNX5A
IFNX5A50	CSECT NAME; ASSEMBLY PHASE; XREF OUTPUT ROUTINE		IFNX5A50	IFNX5A
IFNX5C00	CSECT NAME; ASSEMBLY PHASE; MAINLINE CONTROL; CONSTANTS AND PATCH AREA	21	IFNX5C00	IFNX5C
IFNX5D00	CSECT NAME; ASSEMBLY PHASE; DC EVALUATION; INITIALIZATION		IFNX5D00	IFNX5D
IFNX5F00	CSECT NAME; DC FIXED-FLOATING POINT CONVERSION	21	IFNX5F00	IFNX5F
IFNX5L00	CSECT NAME; ASSEMBLY PHASE; ERROR LOGGING ROUTINE		IFNX5L00	IFNX5L
IFNX5M00	CSECT NAME; MACHINE INSTRUCTION PROCESSOR	21,22	IFNX5M00	IFNX5M
IFNX5P00	CSECT NAME; ASSEMBLY PHASE; PRINT ROUTINE	21	IFNX5P00	IFNX5P
IFNX5V00	CSECT NAME; EXPRESSION EVALUATION SUBROUTINE		IFNX5V00	IFNX5V
IFNX6A00	CSECT NAME; POST PROCESSOR PHASE	21	IFNX6A00	IFNX6A
IFNX6B00	CSECT NAME; DIAGNOSTIC PHASE	21	IFNX6B00	IFNX6B
IFNX6B20	CSECT NAME; DIAGNOSTIC PHASE		IFNX6B20	IFNX6B
IFNX6C00	CSECT NAME; DIAGNOSTIC MESSAGE PHASE; ERROR MESSAGES		IFNX6C00	IFNX6C
IFOX0A00	CSECT NAME; ASSEMBLER DRIVER	27	IFOX0A00	IFOX0A

*DATA AREA. SEE DATA AREA SECTION FOR DETAILED LAYOUT.
**EXPLANATION OF PLM NUMBERED REFERENCES:
A SINGLE NUMERAL REFERS TO AN OPERATIONS DIAGRAM IN THE METHOD OF OPERATIONS SECTION.
'F', FOLLOWED BY A NUMERAL, REFERS TO A FIGURE IN THE PROGRAM ORGANIZATION SECTION.

SYMBOLIC NAME	DESCRIPTION: NAME AND USE	PLM REF**	CSECT/ DSECT	MODULE/ MCROFCH
IFOX0A00	CSECT NAME; ASSEMBLER DRIVER; CONSTANTS	27	IFOX0A00	IFOX0A
IFOX0B00	CSECT NAME; WORKFILE I/O MODULE; MAINLINE CONTROL		IFOX0B00	IFOX0B
IFOX0C00	CSECT NAME; ASSEMBLER COMMON LOAD MODULE; PATCH AREA		IFOX0C00	IFOX0C
IFOX0D00	CSECT NAME; ASSEMBLER INITILIZATION; CONSTANTS AND PATCH AREA	27	IFOX0D00	IFOX0D
IFOX0E00	CSECT NAME; INPUT COMMON LOAD MODULE; DECB		IFOX0E00	IFOX0E
IFOX0F00	CSECT NAME; INPUT I/O MODULE; DCB OVERRIDE AND DCB EXIT ROUTINE FOR 'SYSIN' AND 'SYSLIB' DCB'S	27	IFOX0F00	IFOX0F
IFOX0G00	CSECT NAME; OUTPUT COMMON LOAD MODULE; POINTER LIST		IFOX0G00	IFOX0G
IFOX0H00	CSECT NAME; OUTPUT I/O MODULE; FREEPOOL ROUTINE, CONSTANTS AND PATCH AREA	27	IFOX0H00	IFOX0H
IFOX0I00	CSECT NAME; ABORT ROUTINE		IFOX0I00	IFOX0I
IFOX0J00	CSECT NAME; ASSEMBLER XF PARAMETER OPTIONS		IFOX0J00	IFOX0J
IHADCB	DSECT NAME; ASSEMBLER INITILIZATION; DCBD MACRO		IHADCB	IFOX0D
IHADCB	DSECT NAME; INPUT I/O MODULE; DCBD MACRO		IHADCB	IFOX0F
IHADCB	DSECT NAME; OUTPUT I/O MODULE; DCBD MACRO		IHADCB	IFOX0H
IHADCB	DSECT NAME; ABORT ROUTINE; DCBD MACRO		IHADCB	IFOX0I
INPFMT	DSECT NAME; ADDRESSED BY REG. 'INPTR', EDIT PHASE		INPFMT	IFNX1A
INTRCOM	DSECT NAME; DICTIONARY INTERLUDE PHASE		INTRCOM	IFNX2A
*J	DSECT NAME; EDIT PHASE DICTIONARY ROUTINES		J	IFNX1J
*J	DSECT NAME; CONDITIONAL ASSEMBLY POSTFIX ROUTINE		J	IFNX1S
*J	DSECT NAME; DICTIONARY INTERLUDE PHASE		J	IFNX2A
*J	DSECT NAME; XKE MACRO GENERATOR		J	IFNX3A
*J	DSECT NAME; SYMBOL RESOLUTION PREPROCESSOR		J	IFNX3B
*J	DSECT NAME; GENERATE PHASE DICTIONARY ROUTINES		J	IFNX3N
*J	DSECT NAME; SYMBOL RESOLUTION PHASE DC/DS EVALUATION		J	IFNX4D
*J	DSECT NAME; EXTERNAL SYMBOL DICTIONARY SUBROUTINES		J	IFNX4E

*DATA AREA. SEE DATA AREA SECTION FOR DETAILED LAYOUT.
**EXPLANATION OF PLM NUMBERED REFERENCES:
A SINGLE NUMERAL REFERS TO AN OPERATIONS DIAGRAM IN THE METHOD OF OPERATIONS SECTION.
'F', FOLLOWED BY A NUMERAL, REFERS TO A FIGURE IN THE PROGRAM ORGANIZATION SECTION.

SYMBOLIC NAME	DESCRIPTION: NAME AND USE	PLM REF**	CSECT/ DSECT	MODULE/ MCROFCH
*J	DSECT NAME; SYMBOL RESOLUTION PHASE MAIN LINE CONTROL		J	IFNX4M
*J	DSECT NAME; SYMBOL RESOLUTION PHASE DC/DS EVALUATION		J	IFNX4N
*J	DSECT NAME; SYMBOL TABLE SUBROUTINES		J	IFNX4S
*J	DSECT NAME; SYMBOL RESOLUTION PHASE MAIN LINE CONTROL		J	IFNX4T
*J	DSECT NAME; EXPRESSION EVALUATION SUBROUTINE		J	IFNX4V
*J	DSECT NAME; ASSEMBLY PHASE; MAINLINE CONTROL; JCOMMON COPY CODE		J	IFNX5C
*J	DSECT NAME; DC FIXED-FLOATING POINT CONVERSION		J	IFNX5F
*J	DSECT NAME; EXPRESSION EVALUATION SUBROUTINE		J	IFNX5V
*J	DSECT NAME; POST PROCESSOR PHASE		J	IFNX6A
*J	DSECT NAME; DIAGNOSTIC MESSAGE PHASE; JCOMMON COPY CODE		J	IFNX6C
*J	DSECT NAME; ASSEMBLER DRIVER; JCOMMON COPY CODE		J	IFOX0A
*J	DSECT NAME; ABORT ROUTINE; JCOMMON COPY CODE		J	IFOX0I
JERMSGCD	DSECT NAME; EDIT PHASE DICTIONARY ROUTINES		JERMSGCD	IFNX1J
JERMSGCD	DSECT NAME; DICTIONARY INTERLUDE PHASE		JERMSGCD	IFNX2A
JERMSGCD	DSECT NAME; XKE MACRO GENERATOR		JERMSGCD	IFNX3A
JERMSGCD	DSECT NAME; GENERATE PHASE DICTIONARY ROUTINES		JERMSGCD	IFNX3N
JERMSGCD	DSECT NAME; ASSEMBLY PHASE; MAINLINE CONTROL; JERMSGCD COPY CODE		JERMSGCD	IFNX5C
JERMSGCD	DSECT NAME; EXPRESSION EVALUATION SUBROUTINE		JERMSGCD	IFNX5V
JERMSGCD	DSECT NAME; ABORT ROUTINE; JERMSGCD COPY CODE		JERMSGCD	IFOX0I
*JERRCD	DSECT NAME; EDIT PHASE		JERRCD	IFNX1A
*JERRCD	DSECT NAME; DICTIONARY INTERLUDE PHASE		JERRCD	IFNX2A
*JERRCD	DSECT NAME; XKE MACRO GENERATOR		JERRCD	IFNX3A
*JFLEBLK	DSECT NAME; FILE BLOCK DSECT, ASSEMBLER DRIVER; JFLEBLK COPY CODE		JFLEBLK	IFOX0A
*JFLEBLK	DSECT NAME; FILE BLOCK DSECT, ASSEMBLER INITILIZATION; FILE BLOCK DSECT (JFLEBLK)		JFLEBLK	IFOX0D
*JINCOM	DSECT NAME; INPUT I/O MODULE; JINCOM COPY CODE		JINCOM	IFOX0F
*JINCOM	DSECT NAME; ABORT ROUTINE; JINCOM COPY CODE		JINCOM	IFOX0I
*JOUTCOM	DSECT NAME; OUTPUT COMMON DSECT, OUTPUT COMMON LOAD MODULE; JOUTCOM DSECT		JOUTCOM	IFOX0G

*DATA AREA. SEE DATA AREA SECTION FOR DETAILED LAYOUT.
**EXPLANATION OF PLM NUMBERED REFERENCES:
A SINGLE NUMERAL REFERS TO AN OPERATIONS DIAGRAM IN THE METHOD OF OPERATIONS SECTION.
'F', FOLLOWED BY A NUMERAL, REFERS TO A FIGURE IN THE PROGRAM ORGANIZATION SECTION.

SYMBOLIC NAME	DESCRIPTION: NAME AND USE	PLM REF**	CSECT/ DSECT	MODULE/ MCROFCH
*JOUTCOM	DSECT NAME; OUTPUT COMMON DSECT, OUTPUT I/O MODULE; JOUTCOM COPY CODE		JOUTCOM	IFOX0H
*JOUTCOM	DSECT NAME; OUTPUT COMMON DSECT, ABORT ROUTINE; JOUTCOM COPY CODE		JOUTCOM	IFOX0I
*JTEXT	DSECT NAME; TEXT RECORD DSECT, EDIT PHASE DICTIONARY ROUTINES	3,4	JTEXT	IFNX1J
*JTEXT	DSECT NAME; TEXT RECORD DSECT, DICTIONARY INTERLUDE PHASE		JTEXT	IFNX2A
*JTEXT	DSECT NAME; TEXT RECORD DSECT, XKE MACRO GENERATOR		JTEXT	IFNX3A
*JTEXT	DSECT NAME; TEXT RECORD DSECT, SYMBOL RESOLUTION PREPROCESSOR	16	JTEXT	IFNX3B
*JTEXT	DSECT NAME; TEXT RECORD DSECT, GENERATE PHASE DICTIONARY ROUTINES		JTEXT	IFNX3N
*JTEXT	DSECT NAME; SYMBOL RESOLUTION PHASE DC/DS EVALUATION	17	JTEXT	IFNX4D
*JTEXT	DSECT NAME; EXTERNAL SYMBOL DICTIONARY SUBROUTINES		JTEXT	IFNX4E
*JTEXT	DSECT NAME; TEXT RECORD DSECT, EXTERNAL SYMBOL DICTIONARY SUBROUTINES			
*JTEXT	DSECT NAME; OPERATOR STACK POINTER, EXTERNAL SYMBOL DICTIONARY SUBROUTINES			
*JTEXT	DSECT NAME; TEXT RECORD DSECT, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL	17	JTEXT	IFNX4M
	DSECT NAME; SYMBOL RESOLUTION PHASE MAIN LINE CONTROL	17		
	DSECT NAME; OPERATOR STACK POINTER, SYMBOL RESOLUTION PHASE DC/DS EVALUATION		JTEXT	IFNX4N
*JTEXT	DSECT NAME; OPERATOR STACK POINTER, SYMBOL TABLE SUBROUTINES	17	JTEXT	IFNX4S
*JTEXT	DSECT NAME; TEXT RECORD DSECT, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL		JTEXT	IFNX4T

*DATA AREA. SEE DATA AREA SECTION FOR DETAILED LAYOUT.
**EXPLANATION OF PLM NUMBERED REFERENCES:
A SINGLE NUMERAL REFERS TO AN OPERATIONS DIAGRAM IN THE METHOD OF OPERATIONS SECTION.
'F', FOLLOWED BY A NUMERAL, REFERS TO A FIGURE IN THE PROGRAM ORGANIZATION SECTION.

SYMBOLIC NAME	DESCRIPTION: NAME AND USE	PLM REF**	CSECT/ DSECT	MODULE/ MCROFCH
*JTEXT	DSECT NAME; OPERATOR STACK POINTER, EXPRESSION EVALUATION SUBROUTINE		JTEXT	IFNX4V
*JTEXT	DSECT NAME; TEXT RECORD DSECT, ASSEMBLY PHASE; MAINLINE CONTROL; JTEXT COPY CODE	21	JTEXT	IFNX5C
*JTEXT	DSECT NAME; OPERATOR STACK POINTER, DC FIXED-FLOATING POINT CONVERSION	21	JTEXT	IFNX5F
*JTEXT	DSECT NAME; OPERATOR STACK POINTER, EXPRESSION EVALUATION SUBROUTINE		JTEXT	IFNX5V
*JTEXTA	DSECT NAME; DSECT FOR VARIABLE TEXT PORTION, EDIT PHASE DICTIONARY ROUTINES		JTEXTA	IFNX1J
*JTEXTA	DSECT NAME; DSECT FOR VARIABLE TEXT PORTION, DICTIONARY INTERLUDE PHASE		JTEXTA	IFNX2A
*JTEXTA	DSECT NAME; DSECT FOR VARIABLE TEXT PORTION, XKE MACRO GENERATOR		JTEXTA	IFNX3A
*JTEXTA	DSECT NAME; DSECT FOR VARIABLE TEXT PORTION, SYMBOL RESOLUTION PREPROCESSOR		JTEXTA	IFNX3B
*JTEXTA	DSECT NAME; DSECT FOR VARIABLE TEXT PORTION, GENERATE PHASE DICTIONARY ROUTINES		JTEXTA	IFNX3N
*JTEXTA	DSECT NAME; DSECT FOR VARIABLE TEXT PORTION, SYMBOL RESOLUTION PHASE DC/DS EVALUATION		JTEXTA	IFNX4D
*JTEXTA	DSECT NAME; DSECT FOR VARIABLE TEXT PORTION, EXTERNAL SYMBOL DICTIONARY SUBROUTINES		JTEXTA	IFNX4E
*JTEXTA	DSECT NAME; DSECT FOR VARIABLE TEXT PORTION, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL		JTEXTA	IFNX4M
*JTEXTA	DSECT NAME; DSECT FOR VARIABLE TEXT PORTION, SYMBOL RESOLUTION PHASE DC/DS EVALUATION		JTEXTA	IFNX4N
*JTEXTA	DSECT NAME; DSECT FOR VARIABLE TEXT PORTION, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL		JTEXTA	IFNX4T
*JTEXTA	DSECT NAME; DSECT FOR VARIABLE TEXT PORTION, ASSEMBLY PHASE; MAINLINE CONTROL; JTEXT COP		JTEXTA	IFNX5C
*JTEXTA	DSECT NAME; RESUME VARIABLE PART OF TEXT, MACHINE INSTRUCTION PROCESSOR	22	JTEXTA	IFNX5M
*JTEXTA	DSECT NAME; ASSEMBLY PHASE; PRINT ROUTINE		JTEXTA	IFNX5P
*JTEXTA	DSECT NAME; JTEXTA DSECT, DIAGNOSTIC PHASE; TERMINAL BUFFER DSECT AND JTEXTA DSECT		JTEXTA	IFNX6B
JTMTXT	DSECT NAME; EDIT PHASE		JTMTXT	IFNX1A
JTMTXT	DSECT NAME; EDIT PHASE DICTIONARY ROUTINES		JTMTXT	IFNX1J

*DATA AREA. SEE DATA AREA SECTION FOR DETAILED LAYOUT.
**EXPLANATION OF PLM NUMBERED REFERENCES:
A SINGLE NUMERAL REFERS TO AN OPERATIONS DIAGRAM IN THE METHOD OF OPERATIONS SECTION.
'F', FOLLOWED BY A NUMERAL, REFERS TO A FIGURE IN THE PROGRAM ORGANIZATION SECTION.

SYMBOLIC NAME	DESCRIPTION: NAME AND USE	PLM REF**	CSECT/ DSECT	MODULE/ MCROFCH
JTMTXT	DSECT NAME; CONDITIONAL ASSEMBLY POSTFIX ROUTINE		JTMTXT	IFNX1S
JTMTXT	DSECT NAME; XKE MACRO GENERATOR		JTMTXT	IFNX3A
JTMTXT	DSECT NAME; GENERATE PHASE DICTIONARY ROUTINES		JTMTXT	IFNX3N
LATADD	FIRST LITERAL ENTRY ADDRESS, SYMBOL RESOLUTION PHASE DC/DS EVALUATION	F9,17	DSECT7	IFNX4D
LATADD	FIRST LITERAL ENTRY ADDRESS, EXTERNAL SYMBOL DICTIONARY SUBROUTINES	F9,17	DSECT7	IFNX4E
LATADD	FIRST LITERAL ENTRY ADDRESS, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL	F9,19	DSECT7	IFNX4M
LATADD	FIRST LITERAL ENTRY ADDRESS, SYMBOL TABLE SUBROUTINES	F9,17,19	DSECT7	IFNX4S
LATADD	FIRST LITERAL ENTRY ADDRESS, EXPRESSION EVALUATION SUBROUTINE	F9	DSECT7	IFNX4V
LATEND	NEXT LITERAL ENTRY ADDRESS, EXTERNAL SYMBOL DICTIONARY SUBROUTINES	17	DSECT7	IFNX4E
LCLDICTR	ENTRY POINT; GENERATE PHASE DICTIONARY ROUTINES	13	IFNX3N00	IFNX3N
LCLDICTS	ENTRY POINT; GENERATE PHASE DICTIONARY ROUTINES	13	IFNX3N00	IFNX3N
*LCLNTRY	DSECT NAME; EDIT PHASE DICTIONARY ROUTINES	5	LCLNTRY	IFNX1J
LITERAL	PASS PARAMETERS, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL	17	IFNX4M00	IFNX4M
LITRII	ADJUSTMENT INDEX, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL	19	IFNX4M00	IFNX4M
LOCUPD	SAVE REGISTERS, ASSEMBLY PHASE; LOCATION COUNTER UPDATE ROUTINE	21,25	IFNX5A30	IFNX5A
LSYSVALS	DSECT NAME; GENERATE PHASE DICTIONARY ROUTINES		LSYSVALS	IFNX3N
LTDUMP	SAVE REGISTERS, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL	17,20	IFNX4M00	IFNX4M
LTORG	ALIGN TO DOUBLEWORD BOUNDARY, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL	17	IFNX4M00	IFNX4M
LVTMDT	DSECT NAME; LEVEL TIME AND DATE DSECT, ASSEMBLER INITILIZATION; DCBD MACRO		LVTMDT	IFOX0D
MACENTRY	ENTRY POINT; EDIT PHASE DICTIONARY ROUTINES	6	IFNX1J00	IFNX1J
MACRCALL	ENTRY POINT; GENERATE PHASE DICTIONARY ROUTINES	11	IFNX3N00	IFNX3N
MACREND	ENTRY POINT; EDIT PHASE DICTIONARY ROUTINES	3,6	IFNX1J00	IFNX1J
MACRENT	ENTRY POINT; EDIT PHASE DICTIONARY ROUTINES	3,6	IFNX1J00	IFNX1J
MACRKWRD	ENTRY POINT; GENERATE PHASE DICTIONARY ROUTINES	11,12	IFNX3N00	IFNX3N
MACRNAME	ENTRY POINT; EDIT PHASE DICTIONARY ROUTINES	6	IFNX1J00	IFNX1J

*DATA AREA. SEE DATA AREA SECTION FOR DETAILED LAYOUT.
**EXPLANATION OF PLM NUMBERED REFERENCES:
A SINGLE NUMERAL REFERS TO AN OPERATIONS DIAGRAM IN THE METHOD OF OPERATIONS SECTION.
'F', FOLLOWED BY A NUMERAL, REFERS TO A FIGURE IN THE PROGRAM ORGANIZATION SECTION.

SYMBOLIC NAME	DESCRIPTION: NAME AND USE	PLM REF**	CSECT/ DSECT	MODULE/ MCROFCH
MACRO	ENTRY POINT; EDIT PHASE	3	IFNX1A00	IFNX1A
MACRPOST	ENTRY POINT; GENERATE PHASE DICTIONARY ROUTINES	11,12	IFNX3N00	IFNX3N
MACTR	ENTRY POINT; ACTR ENTRY, XKE MACRO GENERATOR	13	IFNX3A00	IFNX3A
MAIF	ENTRY POINT; AIF ENTRY, XKE MACRO GENERATOR	13	IFNX3A00	IFNX3A
MAKESD	SAVE REGISTERS IN STACK, EXTERNAL SYMBOL DICTIONARY SUBROUTINES	15,18	IFNX4E00	IFNX4E
MAKGET	GET NEXT SEQUENTIAL ESD ENTRY, EXTERNAL SYMBOL DICTIONARY SUBROUTINES	18	IFNX4E00	IFNX4E
MBRANCH1	ENTRY POINT; AGO ENTRY POINT, XKE MACRO GENERATOR	13	IFNX3A00	IFNX3A
MCALLIN	ENTRY POINT; EDIT PHASE	3	IFNX1A20	IFNX1A
MCRCAL	DSECT NAME; GENERATE PHASE DICTIONARY ROUTINES		MCRCAL	IFNX3N
MDDND	PTR TO END OF MACRO DEF DIRECT, DICTIONARY INTERLUDE PHASE	F5,F7	INTRCOM	IFNX2A
*MDDNTRY	DSECT NAME; EDIT PHASE DICTIONARY ROUTINES	3,6	MDDNTRY	IFNX1J
*MDDNTRY	DSECT NAME; DICTIONARY INTERLUDE PHASE	10,8	MDDNTRY	IFNX2A
MDDSTRT	PTR TO START OF MACR DEFN DIR, EDIT PHASE	F4,4	EDSECT	IFNX1A
MDDSTRT	PTR TO START OF MACR DEFN DIR, EDIT PHASE DICTIONARY ROUTINES	F4,3,4	EDSECT	IFNX1J
MDDSTRT	PTR TO START OF MACR DEFN DIR, CONDITIONAL ASSEMBLY POSTFIX ROUTINE	F4	EDSECT	IFNX1S
MDDSTRT	PTR TO START OF MACRO DEF DIRCT, DICTIONARY INTERLUDE PHASE	F5	INTRCOM	IFNX2A
*MDVNTRY	DSECT NAME; DICTIONARY INTERLUDE PHASE	10,8	MDVNTRY	IFNX2A
*MDVNTRY	DSECT NAME; GENERATE PHASE DICTIONARY ROUTINES		MDVNTRY	IFNX3N
MDVSTRT	PTR TO START OF MACRO DEF VECTR, DICTIONARY INTERLUDE PHASE	F5	INTRCOM	IFNX2A
MDVSTRT	PTR TO START OF MDV, XKE MACRO GENERATOR	F8	GENCOM	IFNX3A
MDVSTRT	PTR TO START OF MDV, SYMBOL RESOLUTION PREPROCESSOR	F8	GENCOM	IFNX3B
MDVSTRT	PTR TO START OF MDV, GENERATE PHASE DICTIONARY ROUTINES	F8,12	GENCOM	IFNX3N
MEND	ENTRY POINT; EDIT PHASE	3	IFNX1A00	IFNX1A
MERGE	ENTRY POINT; POST PROCESSOR PHASE	26	IFNX6A00	IFNX6A
METASCAN	ENTRY POINT; EDIT PHASE	7	IFNX1A10	IFNX1A
MNOTEO	CHECK IF OPERAND PRESENT, ASSEMBLER OPCODE PROCESSOR; ASSEMBLY PHASE; 'MNOTE' STATEMENT PROCESSOR	24	IFNX5A00	IFNX5A

*DATA AREA. SEE DATA AREA SECTION FOR DETAILED LAYOUT.
**EXPLANATION OF PLM NUMBERED REFERENCES:
A SINGLE NUMERAL REFERS TO AN OPERATIONS DIAGRAM IN THE METHOD OF OPERATIONS SECTION.
'F', FOLLOWED BY A NUMERAL, REFERS TO A FIGURE IN THE PROGRAM ORGANIZATION SECTION.

SYMBOLIC NAME	DESCRIPTION: NAME AND USE	PLM REF**	CSECT/ DSECT	MODULE/ MCROFCH
MPMT	DSECT NAME; EDIT PHASE		MPMT	IFNX1A
MPROTO	DSECT NAME; EDIT PHASE		MPROTO	IFNX1A
MSCANA	SEE IF PRGMR MACRO PROTOTYPE, EDIT PHASE DICTIONARY ROUTINES	6	IFNX1J00	IFNX1J
MSETA	ENTRY POINT; SETA ENTRY, XKE MACRO GENERATOR	13	IFNX3A00	IFNX3A
MSETB	ENTRY POINT; SETB ENTRY, XKE MACRO GENERATOR	13	IFNX3A00	IFNX3A
MSETC	ENTRY POINT; SETC ENTRY, XKE MACRO GENERATOR	13	IFNX3A00	IFNX3A
MTBLE	DSECT NAME; DIAGNOSTIC PHASE; PRINT BUFFER DSECT AND PUNCH BUFFER DSECT		MTBLE	IFNX6B
MTEXT	DSECT NAME; COPY OF JTEXT FOR OUTPUT, XKE MACRO GENERATOR		MTEXT	IFNX3A
MTEXTA	DSECT NAME; VARIABLE PORTION--OUTPUT, XKE MACRO GENERATOR		MTEXTA	IFNX3A
NEOFRTN	RESTORE EOF SWITCH, EDIT PHASE	6	IFNX1A30	IFNX1A
NEXTPARM	ENTRY POINT; EDIT NEXT OPERAND, EDIT PHASE	3	IFNX1A20	IFNX1A
NEXTPM	ENTRY POINT; EDIT PHASE	3	IFNX1A20	IFNX1A
OPENEND	ENTRY POINT; EDIT PHASE DICTIONARY ROUTINES	6	IFNX1J00	IFNX1J
OPENENT	ENTRY POINT; EDIT PHASE DICTIONARY ROUTINES	3	IFNX1J00	IFNX1J
OPERCODE	ENTRY POINT; EDIT PHASE DICTIONARY ROUTINES	3	IFNX1J00	IFNX1J
*OPNTRY	DSECT NAME; EDIT PHASE DICTIONARY ROUTINES		OPNTRY	IFNX1J
*OPNTRY	DSECT NAME; XKE MACRO GENERATOR		OPNTRY	IFNX3A
OPSNSTRT	PTR TO START OF OPSYN TABLE, XKE MACRO GENERATOR	F8	GENCOM	IFNX3A
OPSNSTRT	PTR TO START OF OPSYN TABLE, SYMBOL RESOLUTION PREPROCESSOR	F8	GENCOM	IFNX3B
OPSNSTRT	PTR TO START OF OPSYN TABLE, GENERATE PHASE DICTIONARY ROUTINES	F8	GENCOM	IFNX3N
*OPSTBL	DSECT NAME; DICTIONARY INTERLUDE PHASE	8	OPSTBL	IFNX2A
OPSYN	ENTRY POINT; EDIT PHASE	3,4	IFNX1A00	IFNX1A
OPSYNBLD	ENTRY POINT; EDIT PHASE DICTIONARY ROUTINES	4	IFNX1J00	IFNX1J
OPSYNBLD	ENTRY POINT; DICTIONARY INTERLUDE PHASE	8	IFNX2A02	IFNX2A
*OPSYNTRY	DSECT NAME; EDIT PHASE DICTIONARY ROUTINES	3,4	OPSYNTRY	IFNX1J
*OPSYNTRY	DSECT NAME; DICTIONARY INTERLUDE PHASE	8	OPSYNTRY	IFNX2A
*OPSYNTRY	DSECT NAME; XKE MACRO GENERATOR		OPSYNTRY	IFNX3A
ORDREF	ENTRY POINT; DICTIONARY INTERLUDE PHASE	8,9	IFNX2A00	IFNX2A
ORDREF	DSECT NAME; GENERATE PHASE DICTIONARY ROUTINES		ORDREF	IFNX3N
ORDSYMBR	ENTRY POINT; EDIT PHASE DICTIONARY ROUTINES	3,5	IFNX1J00	IFNX1J

*DATA AREA. SEE DATA AREA SECTION FOR DETAILED LAYOUT.
**EXPLANATION OF PLM NUMBERED REFERENCES:
A SINGLE NUMERAL REFERS TO AN OPERATIONS DIAGRAM IN THE METHOD OF OPERATIONS SECTION.
'F', FOLLOWED BY A NUMERAL, REFERS TO A FIGURE IN THE PROGRAM ORGANIZATION SECTION.

SYMBOLIC NAME	DESCRIPTION: NAME AND USE	PLM REF**	CSECT/ DSECT	MODULE/ MCROFCH
ORDSYMBR	ENTRY POINT; DICTIONARY INTERLUDE PHASE	8,9	IFNX2A02	IFNX2A
ORDSYMBR	ENTRY POINT; GENERATE PHASE DICTIONARY ROUTINES	13	IFNX3N00	IFNX3N
ORG100	GET SYMBOL DEFINITION POINTER, ASSEMBLER OPCODE PROCESSOR; ASSEMBLY PHASE; 'ORG' STATEMENT PROCESSOR	25	IFNX5A00	IFNX5A
*OSDIR	DSECT NAME; EDIT PHASE DICTIONARY ROUTINES	5	OSDIR	IFNX1J
OSLUKUP	CHECK FOR END OF CHAIN, DICTIONARY INTERLUDE PHASE	9	IFNX2A02	IFNX2A
*OSRDNTRY	DSECT NAME; DICTIONARY INTERLUDE PHASE	8,9	OSRDNTRY	IFNX2A
OSRDSTRT	PTR TO START OF ORD SYMB REF DT, EDIT PHASE	F4	EDSECT	IFNX1A
OSRDSTRT	PTR TO START OF ORD SYMB REF DT, EDIT PHASE DICTIONARY ROUTINES	F4,5	EDSECT	IFNX1J
OSRDSTRT	PTR TO START OF ORD SYMB REF DT, CONDITIONAL ASSEMBLY POSTFIX ROUTINE	F4	EDSECT	IFNX1S
OSRDSTRT	ALIAS FOR OSRTEND. PTR TO START OF ORD SYMB REF DI, DICTIONARY INTERLUDE PHASE	F6	INTRCOM	IFNX2A
OSRDSTRT	PTR TO START OF ORD SYMB REF DT, XKE MACRO GENERATOR	F8,13	GENCOM	IFNX3A
OSRDSTRT	PTR TO START OF ORD SYMB REF DT, SYMBOL RESOLUTION PREPROCESSOR	F8	GENCOM	IFNX3B
OSRDSTRT	PTR TO START OF ORD SYMB REF DT, GENERATE PHASE DICTIONARY ROUTINES	F8	GENCOM	IFNX3N
*OSREF	DSECT NAME; EDIT PHASE DICTIONARY ROUTINES	5	OSREF	IFNX1J
*OSREF	DSECT NAME; DICTIONARY INTERLUDE PHASE		OSREF	IFNX2A
OSRTEND	ALIAS FOR GDEND. PTR TO CURRENT END OF ORD SYMB, DICTIONARY INTERLUDE PHASE	F5	INTRCOM	IFNX2A
*OSRTNTRY	DSECT NAME; DICTIONARY INTERLUDE PHASE	8,9	OSRTNTRY	IFNX2A
OSRTSTRT	ALIAS FOR GDSTRT. PTR TO START OF ORD SYMB REF TB, DICTIONARY INTERLUDE PHASE	F5	INTRCOM	IFNX2A
OUTPUTS	HI LO COMPARE, POST PROCESSOR PHASE	26	IFNX6A00	IFNX6A
*P	DSECT NAME; DDNAME OVERRIDE LIST, ASSEMBLER INITILIZATION; DCBD MACRO		P	IFOX0D
*P	DSECT NAME; DDNAME OVERRIDE LIST, INPUT I/O MODULE; DDNAME OVERRIDE LIST		P	IFOX0F
*P	DSECT NAME; DDNAME OVERRIDE LIST, OUTPUT I/O MODULE; PRINT IMAGE, PUNCH IMAGE AND DDNAME OVERRIDE DDNAME LIST DSE		P	IFOX0H
*P	DSECT NAME; DDNAME OVERRIDE LIST, ABORT ROUTINE; DDNAME OVERRIDE DSECT		P	IFOX0I
PARMREF	DSECT NAME; GENERATE PHASE DICTIONARY ROUTINES		PARMREF	IFNX3N
PARMTAB	DSECT NAME; MAP OF PARAMETER TABLE ENTRIES, XKE MACRO GENERATOR		PARMTAB	IFNX3A

*DATA AREA. SEE DATA AREA SECTION FOR DETAILED LAYOUT.
**EXPLANATION OF PLM NUMBERED REFERENCES:
A SINGLE NUMERAL REFERS TO AN OPERATIONS DIAGRAM IN THE METHOD OF OPERATIONS SECTION.
'F', FOLLOWED BY A NUMERAL, REFERS TO A FIGURE IN THE PROGRAM ORGANIZATION SECTION.

SYMBOLIC NAME	DESCRIPTION: NAME AND USE	PLM REF**	CSECT/ DSECT	MODULE/ MCROFCH
PARMTBLR	XKE MACRO GENERATOR	13	X3000	IFNX3A
PARMTBLR	ENTRY POINT; GENERATE PHASE DICTIONARY ROUTINES	13	IFNX3N00	IFNX3N
PCHBUF	DSECT NAME; DIAGNOSTIC PHASE; PRINT BUFFER DSECT AND PUNCH BUFFER DSECT		PCHBUF	IFNX6B
PCHLINE	DSECT NAME; PUNCHLINE DSECT, OUTPUT I/O MODULE; PRINT IMAGE, PUNCH IMAGE AND DDNAME OVERRIDE DDNAME LIST DSECT'S		PCHLINE	IFOX0H
PHASENTR	ENTRY POINT; GENERATE PHASE DICTIONARY ROUTINES	11	IFNX3N00	IFNX3N
PKON	CLEAR CHARACTER REGISTER, ASSEMBLY PHASE; DC EVALUATION; PROCESS P-TYPE CONSTANTS	23	IFNX5D00	IFNX5D
POP100	DOES POP HAVE AN OPERAND, ASSEMBLER OPCODE PROCESSOR; ASSEMBLY PHASE; 'POP' STATEMENT PROCESSOR	24	IFNX5A00	IFNX5A
*PPIN	DSECT NAME; MACHINE INSTRUCTION PROCESSOR		PPIN	IFNX5M
*PPIN	DSECT NAME; POST PROCESSOR PHASE	26	PPIN	IFNX6A
PRINTSW	PRINT STATEMENT, ASSEMBLY PHASE; MAINLINE CONTROL; X5COM COPY CODE	21	X5COM	IFNX5C
PRINTSW	PRINT STATEMENT, DC FIXED-FLOATING POINT CONVERSION	21	X5COM	IFNX5F
PRINT0	SAVE PRINT SWITCH, ASSEMBLER OPCODE PROCESSOR; ASSEMBLY PHASE; 'PRINT' STATEMENT PROCESSOR	24	IFNX5A00	IFNX5A
*PRMNTRY	DSECT NAME; EDIT PHASE DICTIONARY ROUTINES	5	PRMNTRY	IFNX1J
PRNIMG	DSECT NAME; ASSEMBLY PHASE; PRINT ROUTINE		PRNIMG	IFNX5P
PROTOEND	ENTRY POINT; GENERATE PHASE DICTIONARY ROUTINES	11,12	IFNX3N00	IFNX3N
PROTOIN	ENTRY POINT; PROTOTYPE PROCESSOR, EDIT PHASE	3	IFNX1A20	IFNX1A
PROTOKWD	ENTRY POINT; GENERATE PHASE DICTIONARY ROUTINES	11,12	IFNX3N00	IFNX3N
PRTBUF	DSECT NAME; DIAGNOSTIC PHASE; PRINT BUFFER DSECT AND PUNCH BUFFER DSECT		PRTBUF	IFNX6B
PRTLINE	DSECT NAME; PRINTLINE DSECT, OUTPUT I/O MODULE; PRINT IMAGE, PUNCH IMAGE AND DDNAME OVERRIDE DDNAME LIST DSECT'S		PRTLINE	IFOX0H
PUNCH0	GO PRINT PUNCH STATEMENT, ASSEMBLER OPCODE PROCESSOR; ASSEMBLY PHASE; 'PUNCH' & 'REPRO' STATEMENT PR	23	IFNX5A00	IFNX5A
PUSH00	ERROR NO OPERAND, ASSEMBLER OPCODE PROCESSOR; ASSEMBLY PHASE; 'PUSH' STATEMENT PROCESSOR	24	IFNX5A00	IFNX5A
PUTOPSYN	SEE IF ANY TABLE TO PUT, DICTIONARY INTERLUDE PHASE	8	IFNX2A02	IFNX2A
PVECTPTR	ALIAS FOR SPACEAH. PTR TO CURRENT PARAM VECTOR, XKE MACRO GENERATOR	F8	GENCOM	IFNX3A

*DATA AREA. SEE DATA AREA SECTION FOR DETAILED LAYOUT.

**EXPLANATION OF PLM NUMBERED REFERENCES:
A SINGLE NUMERAL REFERS TO AN OPERATIONS DIAGRAM IN THE METHOD OF OPERATIONS SECTION.
'F', FOLLOWED BY A NUMERAL, REFERS TO A FIGURE IN THE PROGRAM ORGANIZATION SECTION.

SYMBOLIC NAME	DESCRIPTION: NAME AND USE	PLM REF**	CSECT/ DSECT	MODULE/ MCROFCH
PVECTPTR	ALIAS FOR SPACEAH. PTR TO CURRENT PARAM VECTOR, SYMBOL RESOLUTION PREPROCESSOR	F8	GENCOM	IFNX3B
PVECTPTR	ALIAS FOR SPACEAH. PTR TO CURRENT PARAM VECTOR, GENERATE PHASE DICTIONARY ROUTINES	F8	GENCOM	IFNX3N
QCON	SAVE REGISTERS IN STACK, EXTERNAL SYMBOL DICTIONARY SUBROUTINES	17	IFNX4E00	IFNX4E
QKON	SET TEXT POINTER, ASSEMBLY PHASE; DC EVALUATION	23	IFNX5D00	IFNX5D
*RCARD	DSECT NAME; RLD PUNCH RECORD FORMAT, POST PROCESSOR PHASE		RCARD	IFNX6A
RDSRC	ENTRY POINT; EDIT PHASE	3	IFNX1A20	IFNX1A
RDWRPARM	DSECT NAME; READ/WRITE PARAMETER LIST, WORKFILE I/O MODULE; COPY CODE		RDWRPARM	IFOX0B
READNEXT	POINT TO RDSRC, EDIT PHASE	3	IFNX1A00	IFNX1A
REFER	SAVE REGISTERS IN STACK, EXTERNAL SYMBOL DICTIONARY SUBROUTINES	17,20	IFNX4E00	IFNX4E
REHASH	SYMBOL RESOLUTION PHASE MAIN LINE CONTROL	20,27	IFNX4M00	IFNX4M
REPRO0	LOAD INDEX FOR REPRO CARD, ASSEMBLER OPCODE PROCESSOR; ASSEMBLY PHASE; 'PUNCH' & 'REPRO' STATEMENT P	23	IFNX5A00	IFNX5A
RESOLVE	ENTRY POINT; XKE MACRO GENERATOR	13	IFNX3A03	IFNX3A
*RLDIN	DSECT NAME; RLD INPUT RECORD FORMAT, ASSEMBLY PHASE; MAINLINE CONTROL; POST PROCESSOR RECORD DEFNS (BMDSECTS)		RLDIN	IFNX5C
*RLDIN	DSECT NAME; RLD INPUT RECORD FORMAT, EXPRESSION EVALUATION SUBROUTINE		RLDIN	IFNX5V
RLDOUT	SAVE REGISTERS, ASSEMBLY PHASE; RLD OUTPUT ROUTINE	23	IFNX5A40	IFNX5A
*RPRINT	DSECT NAME; RLD PRINT RECORD FORMAT, POST PROCESSOR PHASE		RPRINT	IFNX6A
*RSYMRCD	DSECT NAME; XKE MACRO GENERATOR		RSYMRCD	IFNX3A
*RSYMRCD	DSECT NAME; ASSEMBLY PHASE; ERROR LOGGING ROUTINE; RSYMRCD COPY CODE		RSYMRCD	IFNX5L
SEARCH	SYMBOL RESOLUTION PHASE MAIN LINE CONTROL	20	IFNX4M00	IFNX4M
SEQDEF	ENTRY POINT; DICTIONARY INTERLUDE PHASE	10,8	IFNX2A00	IFNX2A
SEQREF	ENTRY POINT; DICTIONARY INTERLUDE PHASE	8	IFNX2A00	IFNX2A
SEQREF	DSECT NAME; GENERATE PHASE DICTIONARY ROUTINES		SEQREF	IFNX3N
SEQSK	PTR TO START OF SEQ,SYM REF DIC, DICTIONARY INTERLUDE PHASE	F5	INTRCOM	IFNX2A
SEQSYMBD	ENTRY POINT; EDIT PHASE DICTIONARY ROUTINES	3,5	IFNX1J00	IFNX1J
SEQSYMBR	ENTRY POINT; EDIT PHASE DICTIONARY ROUTINES	3,5	IFNX1J00	IFNX1J
SEQSYMBR	ENTRY POINT; GENERATE PHASE DICTIONARY ROUTINES	13	IFNX3N00	IFNX3N

*DATA AREA. SEE DATA AREA SECTION FOR DETAILED LAYOUT.
**EXPLANATION OF PLM NUMBERED REFERENCES:
A SINGLE NUMERAL REFERS TO AN OPERATIONS DIAGRAM IN THE METHOD OF OPERATIONS SECTION.
'F', FOLLOWED BY A NUMERAL, REFERS TO A FIGURE IN THE PROGRAM ORGANIZATION SECTION.

SYMBOLIC NAME	DESCRIPTION: NAME AND USE	PLM REF**	CSECT/ DSECT	MODULE/ MCROFCH
SETANTRY	DSECT NAME; GENERATE PHASE DICTIONARY ROUTINES		SETANTRY	IFNX3N
SETBNTRY	DSECT NAME; GENERATE PHASE DICTIONARY ROUTINES		SETBNTRY	IFNX3N
SETCNTRY	DSECT NAME; GENERATE PHASE DICTIONARY ROUTINES		SETCNTRY	IFNX3N
SKDCSTRT	PTR TO START OF SKEL DICT, DICTIONARY INTERLUDE PHASE	F5,10	INTRCOM	IFNX2A
*SKDCTHDR	DSECT NAME; DICTIONARY INTERLUDE PHASE	10,8	SKDCTHDR	IFNX2A
*SKDCTHDR	DSECT NAME; GENERATE PHASE DICTIONARY ROUTINES		SKDCTHDR	IFNX3N
SKON	TURN OFF SUB-FIELD FLAG, ASSEMBLY PHASE; DC EVALUATION	23	IFNX5D00	IFNX5D
SORTPTR	SORT AREA POINTER, POST PROCESSOR PHASE	F11	X6ACOMM	IFNX6A
SPACE0	GO SPACE 1 IF NO OPERAND, ASSEMBLER OPCODE PROCESSOR; ASSEMBLY PHASE; 'SPACE' AND 'EJECT' STATEMENT	24	IFNX5A00	IFNX5A
SPART	GO IF NO EXPLICIT BASE, MACHINE INSTRUCTION PROCESSOR	22	IFNX5M00	IFNX5M
SPASGN	GET ZERO+GARBAGE+S PART ALLOC, MACHINE INSTRUCTION PROCESSOR	22	IFNX5M00	IFNX5M
*SSDEF	DSECT NAME; EDIT PHASE DICTIONARY ROUTINES	5	SSDEF	IFNX1J
*SSDEF	DSECT NAME; DICTIONARY INTERLUDE PHASE		SSDEF	IFNX2A
*SSDIR	DSECT NAME; EDIT PHASE DICTIONARY ROUTINES	5	SSDIR	IFNX1J
SSDTEND	PTR TO CURRENT END OF SEQ SYMB, DICTIONARY INTERLUDE PHASE	F5	INTRCOM	IFNX2A
*SSDTNTRY	DSECT NAME; DICTIONARY INTERLUDE PHASE	10	SSDTNTRY	IFNX2A
SSDTSTRT	PTR TO START OF SEQ SYMB DEF TB, DICTIONARY INTERLUDE PHASE	F5,10	INTRCOM	IFNX2A
SSHASHER	CLEAR WORK AREA FOR SEQ SYMB, DICTIONARY INTERLUDE PHASE	10	IFNX2A00	IFNX2A
SSRDSTRT	PTR TO START OF SEQ SYMB REF DT, EDIT PHASE	F4	EDSECT	IFNX1A
SSRDSTRT	PTR TO START OF SEQ SYMB REF DT, EDIT PHASE DICTIONARY ROUTINES	F4	EDSECT	IFNX1J
SSRDSTRT	PTR TO START OF SEQ SYMB REF DT, CONDITIONAL ASSEMBLY POSTFIX ROUTINE	F4	EDSECT	IFNX1S
*SSREF	DSECT NAME; EDIT PHASE DICTIONARY ROUTINES	5	SSREF	IFNX1J
*SSREF	DSECT NAME; DICTIONARY INTERLUDE PHASE		SSREF	IFNX2A
STACKREC	DSECT NAME; EDIT PHASE		STACKREC	IFNX1A
STACKREC	DSECT NAME; STACK ELEMENT SNAPSHOT, CONDITIONAL ASSEMBLY POSTFIX ROUTINE		STACKREC	IFNX1S
START0	GET DATA POINTER, ASSEMBLER OPCODE PROCESSOR- ASSEMBLY PHASE- 'START, CSECT, DSECT & COM' STATEMENT PR	25	IFNX5A00	IFNX5A

*DATA AREA. SEE DATA AREA SECTION FOR DETAILED LAYOUT.
**EXPLANATION OF PLM NUMBERED REFERENCES:
A SINGLE NUMERAL REFERS TO AN OPERATIONS DIAGRAM IN THE METHOD OF OPERATIONS SECTION.
'F', FOLLOWED BY A NUMERAL, REFERS TO A FIGURE IN THE PROGRAM ORGANIZATION SECTION.

SYMBOLIC NAME	DESCRIPTION: NAME AND USE	PLM REF**	CSECT/ DSECT	MODULE/ MCROFCH
STGAREA	DSECT NAME; STRING COLLECTION AREA, EDIT PHASE		STGAREA	IFNX1A
STMTSEQ	ENTRY POINT; EDIT PHASE	3	IFNX1A10	IFNX1A
SUBSET	EXIT IF SUBSETTED THIS ROUND, SYMBOL TABLE SUBROUTINES	20,27	IFNX4S00	IFNX4S
SUMCST	GET CURRENT AND HIGH ADDRESS, EXTERNAL SYMBOL DICTIONARY SUBROUTINES	18	IFNX4E00	IFNX4E
SUMDSD	CHANGE TYPE, EXTERNAL SYMBOL DICTIONARY SUBROUTINES	18	IFNX4E00	IFNX4E
SUMDXD	GET LENGTH AND ALIGNMENT FACTOR, EXTERNAL SYMBOL DICTIONARY SUBROUTINES	18	IFNX4E00	IFNX4E
SUMESD	PUSH DOWN ONE MORE LEVEL, EXTERNAL SYMBOL DICTIONARY SUBROUTINES	18	IFNX4E00	IFNX4E
SUMGET	PASS, EXTERNAL SYMBOL DICTIONARY SUBROUTINES	18	IFNX4E00	IFNX4E
SYM	POINT TO NEXT CHARACTER, EXPRESSION EVALUATION SUBROUTINE	22	IFNX5V00	IFNX5V
SYMBL	GET COUNT, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL	19,20,27	IFNX4M00	IFNX4M
SYMBOL	PARAMETER, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL	17	IFNX4M00	IFNX4M
SYMDIMEN	SYMBOL TABLE POINTERS, SYMBOL RESOLUTION PHASE DC/DS EVALUATION	F9	DSECT7	IFNX4D
SYMDIMEN	SYMBOL TABLE POINTERS, EXTERNAL SYMBOL DICTIONARY SUBROUTINES	F9,17	DSECT7	IFNX4E
SYMDIMEN	SYMBOL TABLE POINTERS, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL	F9	DSECT7	IFNX4M
SYMDIMEN	SYMBOL TABLE POINTERS, SYMBOL TABLE SUBROUTINES	F9	DSECT7	IFNX4S
SYMDIMEN	SYMBOL TABLE POINTERS, EXPRESSION EVALUATION SUBROUTINE	F9	DSECT7	IFNX4V
TBLOPS	ENTRY POINT; EDIT OPERAND FIELD, EDIT PHASE	3	IFNX1A00	IFNX1A
TERMBUF	DSECT NAME; DIAGNOSTIC PHASE; TERMINAL BUFFER DSECT AND JTEXTA DSECT		TERMBUF	IFNX6B
TEXTGET	GET NUMBER OF SYMBOL XREFED, ASSEMBLY PHASE; MAINLINE CONTROL; TEXT RECORD GET ROUTINE	21	IFNX5C00	IFNX5C
TEXTPTR	DSECT NAME; CONDITIONAL ASSEMBLY POSTFIX ROUTINE		TEXTPTR	IFNX1S
TITLE0	GO SQUEEZE OUT QUOTE AND AMPSND, ASSEMBLER OPCODE PROCESSOR; ASSEMBLY PHASE; 'TITLE' STATEMENT PROCE	24	IFNX5A00	IFNX5A
TRANSFER	ADDRESS OF OUTPUT FILE, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL	20,27	IFNX4M00	IFNX4M
*UDSECT	DSECT NAME; USING TABLE ENTRY DSECT, ASSEMBLY PHASE; MAINLINE CONTROL; X5COM COPY CODE		UDSECT	IFNX5C

*DATA AREA. SEE DATA AREA SECTION FOR DETAILED LAYOUT.

**EXPLANATION OF PLM NUMBERED REFERENCES:
A SINGLE NUMERAL REFERS TO AN OPERATIONS DIAGRAM IN THE METHOD OF OPERATIONS SECTION.
'F', FOLLOWED BY A NUMERAL, REFERS TO A FIGURE IN THE PROGRAM ORGANIZATION SECTION.

SYMBOLIC		PLM	CSECT/	MODULE/
NAME	DESCRIPTION: NAME AND USE	REF**	DSECT	MCROFCH
*UDSECT	DSECT NAME; USING TABLE ENTRY DSECT, DC FIXED-FLOATING POINT CONVERSION		UDSECT	IFNX5F
*UDSECT	DSECT NAME; USING TABLE ENTRY DSECT, EXPRESSION EVALUATION SUBROUTINE		UDSECT	IFNX5V
USINGO	OPERAND PRESENT?, ASSEMBLER OPCODE PROCESSOR; ASSEMBLY PHASE; 'USING' STATEMENT PROCESSOR	24	IFNX5A00	IFNX5A
VARSYM	MAKE RECURSION STACK ENTRY, EDIT PHASE	3	IFNX1A10	IFNX1A
VCON	SAVE REGISTERS IN STACK, EXTERNAL SYMBOL DICTIONARY SUBROUTINES	17	IFNX4E00	IFNX4E
VKON	SET TEXT POINTER, ASSEMBLY PHASE; DC EVALUATION	23	IFNX5D00	IFNX5D
VREF	DSECT NAME; GENERATE PHASE DICTIONARY ROUTINES		VREF	IFNX3N
*VSDENTRY	DSECT NAME; EDIT PHASE DICTIONARY ROUTINES	5	VSDENTRY	IFNX1J
VSDSTRT	PTR TO START OF VARB SYMB DIR, EDIT PHASE	F4	EDSECT	IFNX1A
VSDSTRT	PTR TO START OF VARB SYMB DIR, EDIT PHASE DICTIONARY ROUTINES	F4	EDSECT	IFNX1J
VSDSTRT	PTR TO START OF VARB SYMB DIR, CONDITIONAL ASSEMBLY POSTFIX ROUTINE	F4	EDSECT	IFNX1S
VSLOOKUP	ENTRY POINT; EDIT PHASE DICTIONARY ROUTINES	5	IFNX1J00	IFNX1J
WRAPFLD	ENTRY POINT; EDIT PHASE	3	IFNX1A00	IFNX1A
WRITE	DEFINE EXIT, SYMBOL RESOLUTION PHASE MAIN LINE CONTROL	17	IFNX4M00	IFNX4M
WSDSECT	DSECT NAME; DC FIXED-FLOATING POINT CONVERSION		WSDSECT	IFNX5F
XKON	CLEAR CHARACTER REGISTER, ASSEMBLY PHASE; DC EVALUATION; PROCESS X-TYPE CONSTANTS	23	IFNX5D00	IFNX5D
XREF	SAVE REGISTERS, ASSEMBLY PHASE; XREF OUTPUT ROUTINE	23	IFNX5A50	IFNX5A
*XRFIN	DSECT NAME; XREF INPUT RECORD FORMAT, ASSEMBLY PHASE; MAINLINE CONTROL; POST PROCESSOR RECORD DEFNS (BMDSECTS)		XRFIN	IFNX5C
*XRFIN	DSECT NAME; XREF INPUT RECORD FORMAT, EXPRESSION EVALUATION SUBROUTINE		XRFIN	IFNX5V
X3000	DSECT NAME; XKE MACRO GENERATOR		X3000	IFNX3A
*X5COM	DSECT NAME; RESUME DSECT, ASSEMBLY PHASE; MAINLINE CONTROL; X5COM COPY CODE		X5COM	IFNX5C
*X5COM	DSECT NAME; ASSEMBLY PHASE; MAINLINE CONTROL; X5COM COPY CODE		X5COM	IFNX5C
*X5COM	DSECT NAME; RESUME DSECT, DC FIXED-FLOATING POINT CONVERSION		X5COM	IFNX5F
*X5COM	DSECT NAME; DC FIXED-FLOATING POINT CONVERSION		X5COM	IFNX5F

*DATA AREA. SEE DATA AREA SECTION FOR DETAILED LAYOUT.

**EXPLANATION OF PLM NUMBERED REFERENCES:
A SINGLE NUMERAL REFERS TO AN OPERATIONS DIAGRAM IN THE METHOD OF OPERATIONS SECTION.
'F', FOLLOWED BY A NUMERAL, REFERS TO A FIGURE IN THE PROGRAM ORGANIZATION SECTION.

SYMBOLIC NAME	DESCRIPTION: NAME AND USE	PLM REF**	CSECT/ DSECT	MODULE/ MCROFCH
*X5COM	DSECT NAME; RESUME DSECT, EXPRESSION EVALUATION SUBROUTINE		X5COM	IFNX5V
*X5COM	DSECT NAME; EXPRESSION EVALUATION SUBROUTINE		X5COM	IFNX5V
X6ACOMM	DSECT NAME; COMMON FOR X6A, POST PROCESSOR PHASE		X6ACOMM	IFNX6A
X6BCOM	DSECT NAME; X6B COMMON AREA, DIAGNOSTIC PHASE; X6B COMMON AREA (X6BCOM)		X6BCOM	IFNX6B
ZKON	CLEAR CHARACTER REGISTER, ASSEMBLY PHASE; DC EVALUATION; PROCESS Z-TYPE CONSTANTS	23	IFNX5D00	IFNX5D

Diagnostic Aids

This section contains information designed to be helpful in debugging.

Eyecatchers: Object Module and Control Section (CSECT) Identifiers

OBJECT MODULE IDENTIFIER

In a dump, object module identifiers are located at the beginning of each assembler object module. The identifier consists of two items:

- the object module name
- a halfword hexadecimal change level identifier

The following is an example of an object module identifier:

$$\underbrace{\text{C9C6D5E7F5C1}}_{\text{IFNX5A}}\underbrace{\text{0001}}_{\text{HEX0001}}$$

CONTROL SECTION (CSECT) IDENTIFIER

In a dump, CSECT identifiers are located at the end of each assembler CSECT. The CSECT identifier immediately precedes the patch area for the CSECT. The identifier consists of three items, separated by blanks:

- the CSECT name
- the time at which the CSECT was assembled
- the date on which the CSECT was assembled

An example of CSECT identifier is given below:

$$\underbrace{\text{C9C6D5E7F5C1F0F0}}_{\substack{\text{IFNX5A00}\\\text{(CSECT name)}}}\underbrace{\text{40}}_{\text{blank}}\underbrace{\text{F1F44BF2F3}}_{\substack{\text{14.23}\\\text{(time)}}}\underbrace{\text{40}}_{\text{blank}}\underbrace{\text{F0F74BF2F34BF7F2}}_{\substack{\text{07.23.72}\\\text{(date)}}}\downarrow_{\substack{\text{beginning of}\\\text{CSECT patch}\\\text{area}}}$$

Data Set Activity Summary

The following tables show the I/O activity of the assembler phases. The tables cross-reference the type of I/O request to (1) the data set the request is for and (2) the routine which issued the request.

In some cases, a second routine is given in the phase or module in parentheses. This indicates that this routine is called by the first routine listed.

EDIT PHASE (MODULE IFNX1A)

I/O ACTION	DATA SET		
	System Input	System Library	File 1
READ SOURCE (JINPUT)	RDSRC (JINLIB SWITCH OFF)	RDSRC (JINLIB SWITCH ON)	---
LOCATE O/P BUFFER (JPUTL)	---	---	OPUTL
FIND (JFIND)	---	COPY ESYSMAC	---
NOTE (NOTELB)	---	CSTKENT	---
POINT (JPOINTLB)	---	CSTKEXT	---
TRUNCATE (JTRUNC)	---	---	PHSEND

EDIT PHASE (MODULE IFNX1J)

I/O ACTION	DATA SET		
	File 1	File 2	File 3
PUT(Locate)	----	VARBSYMD SEQSYMBR SEQSYMBD COMNEND	PHASEND ORDSYMBD
WRITE	----	----	PHASEND (BUFRITE)
CHECK	----	----	PHASEND (BUFRITE)
NOTE	MACRENT SEQSYMBD	GETNPF2	PHASEND (BUFRITE)
POINT	----	PHASEND	----
TRUNCate	----	PHASEND	PHASEND
PUT(Move)	----	----	ORDSYMBD

DICTIONARY INTERLUDE PHASE (MODULE IFNX2A)

I/O ACTION	DATA SET		
	File 1	File 2	File 3
GET (Locate)	----	GETNXT	ORDSYMBR
READ	----	----	INTRENTR*
WRITE	ENDSEGB* ERLOGER* INTREXIT*	INITOSR* OSRDFINI* OPSYNBLD*	----
CHECK	ENDSEGB* ERLOGER* INTREXIT*	INITOSR* OSRDFINI* OPSYNBLD*	INTRENTR*
NOTE	ENDSEGB* ERLOGER* INTREXIT*	INITOSR* OSRDFINI* OPSYNBLD*	----
POINT	INITOSR	RESCAN INITOSR	INTRENTR OPSYNBLD

*(BUFRITE)

GENERATE PHASE (MODULE IFNX3A)

I/O ACTION	DATA SET		
	File 1	File 2	File 3
GET (Locate)	MINPUT MCALLEND	----	----
POINT	MEXIT10	----	MEXIT10
PUT (Locate)	----	FEVAL25 ERRDUMP PRINT90	----
PUT (Move)	----	MEXIT10 MINPUT12 DMYENDRT MCALLEND	----
TRUNC(ate)	----	MEXIT10	----

GENERATE PHASE (MODULE IFNX3B)

I/O ACTION	DATA SET		
	File 1	File 2	File 3
PUT (Locate)	----	----	IFNX3B
PUT (Move)	----	----	IFNX3B

GENERATE PHASE (MODULE IFNX3N)

I/O ACTION	DATA SET		
	File 1	File 2	File 3
READ	DCFETCH* PHASENTR*	PHASENTR*	----
GET (Locate)	PROTOEND	----	----
PUT (Move)	----	PHASENTR	----
CHECK	PHASENTR* DCTFETCH*	PHASENTR*	----
NOTE	CALLEND PROTOEND	----	----
POINT	PHASENTR MACRFINI CALLEND PROTOEND SEQSYMBR DCTFETCH	PHASENTR	----

*(BUFREAD)

SYMBOL RESOLUTION PHASE (ALL MODULES)

I/O ACTION	DATA SET		
	File 1	File 2	File 3
GET (Locate)	GETNEXT	----	GETNEXT
PUT (Move)	TRANSFER	----	TRANSFER
READ	----	GETESD GETLAT	----
WRITE	----	GETESD GETLAT	----
CHECK	----	GETESD GETLAT	----
NOTE	GETREF	GETESD GETLAT	GETREF
POINT	ENDOFILE GETREF	ENDOFILE GETESD	ENDOFILE GETREF

ASSEMBLY PHASE (MODULES IFNX5A, IFNX5C, IFNX5D, IFNX5M, IFNX5V)

I/O ACTION	DATA SET	
	File 2	JINFILE (File 1 or File 3)
GET (Locate)	EDITED TEXT	RESOLVED SYMBOL DATA

ASSEMBLY PHASE (MODULES IFNX5A, IFNX5L, IFNX5P, IFNX5V)

I/O ACTION		DATA SET		
	JOUTFILE*	SYSPRINT	SYSLINK	SYSPCH
PUT (Locate)	ERROR XREF RLD	LISTING	PUNCH REPRO TXT	PUNCH REPRO TXT

* Opposite of JINFILE

POST-PROCESSOR PHASE (MODULE IFNX6A)

I/O ACTION	DATA SET		
	CURFLE	CURFLE2	CURFLE3
READ FROM JINFILE (JREAD)	BUFIN	----	----
CHECK (JCHECK)	BUFIN SPILL READFL1	READFL2	EEREC WRITE
JNCTE (JINFILE)	XGARDX	----	----
JPOINT	XGARDX EEREC	XGARDX EEREC	EEREC
JWRITE	PADDING SPILL	----	WRITE
READ (JREAD)	READFL1	READ FL2	----

Register Usage Tables

DMSASM VM/370 INITIALIZATION ROUTINE

Register	Register Usage
0	Base Address for NUCON
1	Pointer to the ASSEMBLE option list; pointer to CMS plist
2	Work Register
3	Work Register
4	Work Register
5	Work Register
6	Pointer to the File Control Block
7	Pointer to the File Status Table
8	Work Register
9	Work Register
10	Return Linkage Register
11	Second Base Register for DMSASM
12	Base Register for DMSASM
13	Pointer to Save Area
14	Return Linkage Register
15	Address of Linkage Routine

IFOX0A DRIVER ROUTINES

Register	Register Usage
0	Work Register
1	" "
2	" "
3	" "
4	Not Used
5	Pointer to Phase Name
6	Address of Load Routine
7	Address of Delete Routine
8	Base for IFOX0A
9	Return Linkage
10	Work Register
11	" "
12	Target Linkage
13	Common Base
14	Work Register
15	" "

IFOX0B WORKFILE I/O AND STORAGE MANAGEMENT ROUTINES

Register	Register Usage
0	Not Used
1	Work Register
2	" "
3	Pointer to Physical Buffer
4	Work Register
5	Pointer to Logical Record
6	Not Used
7	Pointer to JFLEBLK
8	Buffer Address
9	Return Linkage
10	Work Register
11	" "
12	Base for IFOX0B
13	Common Base
14	Return Linkage
15	Work Register

Deviations - PUTM Routine

Register	Register Usage
3	From Address

Deviations - GETCORE Routine

Register	Register Usage
3	Work Register

IFOXOD MASTER COMMON AREA INITIALIZATION ROUTINES

Register	Register Usage
0	Work Register
1	Parm Field Pointer
2	Work Register
3	Pointer to Cefault Options
4	Work Register
5	Parm Field Pointer
6	Remaining Length of Parm Field
7	Work Register
8	Base for IFOXOD
9	Return Linkage
10	Work Register
11	Input Pointer
12	Target Linkage
13	Common Ease
14	Work Register
15	Work Register

IFOX0F INPUT ROUTINES

Register	Register Usage
0	Work Register
1	DCB Pointer
2	DD Name Pointer
3	Work Register
4	" "
5	" "
6	" "
7	Base for Input Common
8	Work Register
9	Return Linkage
10	Work Register
11	Pointer to Logical Record
12	Base for IFOX0F
13	Common Base
14	Work Register
15	" "

Deviations - FIND Routine

Register	Register Usage
10	Points to Member Name

Deviations - POINTLB Routine

Register	Register Usage
10	Points to Note/Point Value

Deviations - DCB EXIT Routine

Register	Register Usage
10	Base Register

Deviations - ININIT Routine

Register	Register Usage
11	Parm Field Pointer

IFOX0H OUTPUT ROUTINES

Register	Register Usage
0	Work Register
1	DCB Pointer
2	DDname Pointer
3	Open Parm List SYSPUNCH
4	" " " SYSGO
5	Close Parm List SYSPUNCH
6	" " " SYSGO
7	Output Common
8	Level Pointer for Saved Registers
9	Return Linkage
10	Addr of SYSPRINT DCB Exit
11	Parm Field Pointer
12	Base for IFOX0H
13	Common Base
14	Work Register
15	" "

Deviations - PRINT, PUNCH, TSO PRINT Routines

Register	Register Usage
1	Output Record Pointer
2	Work Register
11	Buffer Address

Deviations - DCB EXIT Routine

Register	Register Usage
3	Work Register
4	" "
5	" "
11	Address of SYSPUNCH and SYSGO DCB Exit
15	Base for Exit Routine

IFOX0I ABORT ROUTINE

Register	Register Usage
0	Work Register
1	" "
2	" "
3	Not Used
4	" "
5	DDname Pointer
6	Work Register
7	Input or Output Common Pointer
8	Base for IFOX0I
9	Return Linkage
10	Contains Error Code
11	Parm Field Pointer
12	Target Linkage
13	Common Base
14	Not Used
15	Work Register

IFNX1A   EDIT PHASE (MAINLINE)

Register	Register Usage
0	Work Register
1	Work Register/Return Linkage
2	Base for Phase Common
3	Return Linkage
4	Pointer to position in variable part of Edit Record
5	Base for Edit Record Header
6	Pointer to Position in Source Record (Input)
7	IFNX1A10 Base for IFNX1A20 IFNX1A30
8	Base for IFNX1A00
9	Return Linkage
10	Pass symbol pointers to and from IFNX1J
11	Pass symbol pointers to and from IFNX1J
12	Target Linkage
13	Base for Common
14	Work Register
15	Work Register

Deviations - METASCAN Routine

Register	Register Usage
3	Exit Code
5	Mtable Index
9	Return Linkage
12	Pointer to current entry in Mtable
14	Work Register

Deviations - RDSRC Routine

Register	Register Usage
3	Source data move length
4	Source begin column
5	Source continue character column
9	Return Linkage Sequence field begin
10	Source record pointer Sequence field length
11	INPTR Pointer
12	Continue field begin column Source record end
13	string count card "

Deviations - TRTEST Routine

Register	Register Usage
1	Terminating character addr (at exit)
2	Work Register
3	Catagory number (at exit)
10	Search type (at entry) Type Number (at exit)
11	Work Register
12	Translate table pointer
15	String move length (At Exit)

IFNX1J EDIT DICTIONARY ROUTINES

Register	Register Usage
0	Work Register
1	" "
2	Base for Phase Common
3	Symbol length register
4	Symbol pointer register
5	Base for Variable Symbol Dictionary Sequence Symbol Reference Opcode table Opsyn synonym
6	Base for Current Variable Symbol Directory entry Opsyn Table entry
7	Base for Current Macro Definition Directory entry Opsyn synonym entry
8	Base for IFNX1J00
9	Return Link
10	Pass symbol pointers to and from IFNX1A
11	Pass symbol pointers to and from IFNX1A
12	Base for IFNX1J00 at entry
13	Base for Common
14	Work Register
15	Work Register

IFNX1S   POSTFIX

Register	Register Usage
0	Unused
1	Unused
2	Base for Phase Common
3	Pointer to last stack element
4	Pointer to current position in Edit Record (output)
5	Unused
6	Unused
7	Unused
8	Unused
9	Return linkage
10	Input Operator
11	Unused
12	Base for IFNX1S00
13	Unused
14	Unused
15	Binding factor work register

IFNX2A DICTIONARY INTERLUDE

Register	Register Usage
0	Work Register
1	Work Register
2	Base for Phase Common
3	Base for current macro definition entry
4	Base for Skeleton Dictionary Ordinary Symbol Attribute Reference Dictionary
5	Branch Table Base
6	Ordinary Symbol Attribute Reference Table Base for Sequence Symbol Definition Table start Global Definition Directory
7	Pointer to Sequence Symbol Definition table entry Operand being scanned
8	Base for IFNX2A00
9	Return link
10	Work Register
1	Work Register
12	Base for IFNX2A00 at entry Symbol length register
13	Base for Common
14	Work Register
15	Work Register

IFNX3A GENERATE PHASE (MAINLINE)

Register	Register Usage
0	Work Register
1	Work Register
2	Work Register
3	Work Register
4	Pointer to Output Field
5	Input Text Record
6	Pointer to input field
7	Base for Generate common
8	Base Register
9	Return linkage
10	Work Register
11	Output text record
12	Target Linkage
13	Base for Common
14	Work Register
15	Work Register

Deviations - RESOLVE Routine

Register	Register Usage
10	Pointer to term
14	Pointer to parameter entry

Deviations - EVAL Routine

Register	Register Usage
5	Pointer to stack
6	Meta text pointer

Deviations - GENSTRNG Routine

Register	Register Usage
4	Pointer to length field of current string
6	Meta text pointer
10	Next available output Position
11	Next Meta flag

IFNX3B GENERATE PHASE (SYMBOL RESOLUTION PREPROCESSOR)

Register	Register Usage
0	Work Register
1	Work Register
2	Work Register
3	Work Register, Operand Pointer
4	Points to beginning of operand
5	Points to input text record
6	Work Register
7	Base for GENCOM
8	Base for IFNX3B
9	Return Linkage
10	Work Register
11	Output record pointer
12	Target linkage
13	Base for Common
14	Not Used
15	Not Used

IFNX3N GENERATE PHASE DICTIONARY ROUTINES

Register	Register Usage
0	Work Register
1	Work Register
2	Pointer to next Allocation Address
3	Pointer to Error Record
4	Not Used
5	Maximum Record Length for File
6	File Pointer
7	Base for Generate Common
8	Base for IFNX3N
9	Return Linkage
10	Work Register
11	Work Register
12	Target Linkage
13	Base for Common
14	Work Register
15	Work Register

Deviations - MACRCALL Routine

Register	Register Usage
2	Pointer to MDVENTRY
7	Base for Generate Common
8	Base for IFNX3N
10	Meta text pointer
11	Return Code

Deviations - MACRPOST Routine

Register	Register Usage
0	Work Register
1	Work Register
2	Pointer to next parameter table entry
3	Pointer to parameter vector entry
4	Overlay check pointer
5	Parameter value length
6	Pointer to parameter record header
7	Base for generate common
8	Base for IFNX3N
10	Pointer to parameter record
11	Pointer to Parameter Value Length and Value
12	Target linkage
13	Base for Common
14	Work Register
15	Work Register

Deviations - MACRKWRD Routine

Register	Register Usage
2	Pointer to parameter table entry
3	Pointer to parameter vector entry
4	Overlay check pointer
5	Keyword value length
6	Operand Pointer

Deviations - PROTOKWD Routine

Register	Register Usage
1	Work Register
2	Pointer to keyword in parameter table
3	Keyword length
4	Chain Pointer
7	Base for Generate Common
8	Base for IFOX3N
9	Return Linkage
10	Pointer to parameter record
11	Pointer to parameter value length and value
12	Target linkage
13	Base for Common
14	Work Register
15	Work Register

Deviations - PROTOEND Routine

Register	Register Usage
0	Work Register
1	Work Register
2	Chain pointer Skeleton dictionary pointer
10	Pointer to MDV entry
14	Work Register
15	Work Register

Deviations - GBLDICTR/S, LCLDICTR/S Routines

Register	Register Usage
1	Work Register
3	Dictionary pointer
10	Meta text pointer
11	Value of value pointer
14	Work Register
15	Work Register

Deviations - PARMTBLR Routine

Register	Register Usage
1	Work Register
2	Parameter vector
10	Meta text pointer
11	Parameter table entry
14	Work Register

Deviations - SEQSYMBR Routine

Register	Register Usage
2	Sequence symbol
10	Meta text pointer
11	Pointer to note/point address

Deviations - ORDSYMBR Routine

Register	Register Usage
10	Meta text pointer
11	Pointer to entry in ordinary symbol reference dictionary

IFNX4D SYMBOL RESOLUTION PHASE (DC/DC EVALUATION ROUTINES)

Register	Register Usage
0	Work Register
1	Work Register
2	Work Register
3	Return linkage
4	Text record pointer
5	Operand pointer
6	ESD entry pointer
7	Common base for phase
8	IFNX4D base
9	Return linkage
10	Work Register
11	Work Register
12	Target linkage
13	Common Base
14	Pointer to symbol table entry
15	Work Register

IFNX4E SYMBOL RESOLUTION (ESD ROUTINES)

Register	Register Usage
0	Work Register
1	" "
2	" "
3	" "
4	Text record pointer
5	Work Register
6	" "
7	Phase Common Base
8	INFX4E Base
9	Return linkage
10	Input and output pointer
11	Work Register
12	Target Linkage
13	Common Base
14	Symbol table entry pointer
15	Work Register

Deviations - GETESD Routine

Register	Register Usage
4	Note list pointer

Deviations - MAKESD Routine

Register	Register Usage
6	ESD entry pointer
14	Work Register

IFNX4M SYMBOL RESOLUTION (MAINLINE)

Register	Register Usage
0	Work register
1	" "
2	" "
3	" "
4	Input record pointer
5	Operand pointer
6	ESD entry pointer
7	Phase common base
8	IFNX4M
9	Return Linkage
10	Work Register
11	" "
12	Target linkage
13	Common Base
14	Work Register
15	" "

IFNX4S SYMBOL RESOLUTION (SYMBOL TABLE ROUTINES)

Register	Register Usage
0	Work Register
1	Return Linkage
2	Work Register
3	" "
4	" "
5	" "
6	Not used
7	Symbol Resolution Common Base
8	Base for IFNX4S
9	Return linkage
10	Work Register
11	" "
12	Target linkage
13	Common Base
14	Symbol table entry pointer
15	Work Register

IFNX4V SYMBOL RESOLUTION (EXPRESSION EVALUATION)

Register	Register Usage
0	Work Register
1	Work Register
2	Term stack pointer
3	Relocation list pointer
4	Operator stack pointer
5	Expression pointer
6	Pointer to ESD entry
7	Symbol Resolution Common Base
8	Base for IFNX4S
9	Return linkage
10	ESDID of expression
11	Value of expression
12	Target linkage
13	Base for Common
14	Work Register
15	Work Register

IFNX5A ASSEMBLER OPCODE PROCESSOR

Register	Register Usage
0	Not Used
1	Work Register
2	Work Register
3	Base Register for over 4K USING
4	Variable text pointer
5	Fixed test pointer
6	Return register for BAL
7	Base register for Phase Common Area
8	Base register for first 4K USING
9	Return register for BALR
10	Print index, File index
11	Resolved symbol data pointer, RLD and XREF record pointer
12	Branch register BALR
13	Base register for Common
14	Work operand pointer
15	Work Register

IFNX5C   ASSEMBLER INITIALIZATION

Register	Register Usage
0	Not Used
1	Work Register
2	Work Register
3	Not Used
4	Variable text pointer, symbol data record base register
5	Fixed text pointer
6	Return register in BAL
7	Phase Common Base Register
8	IFNX51 Mainline Base Register
9	Return Register
10	File index
11	Symbol data record base, XREF record base
12	Branch register BALR
13	Base register for Common
14	Work Register
15	Work Register

IFNX5D DC EVALUATION ROUTINE

Register	Register Usage
0	Work Register
1	" "
2	" "
3	DC table pointer
4	Work Register
5	" "
6	Operand pointer
7	Phase Common Base
8	Base Register for routine
9	Return register for BALR
10	ESDID of evaluated expression
11	Value of evaluated expression, symbol record pointer
12	Branch Register in BALR
13	Base Register for Common
14	Work Register
15	Save area pointer

IFNX5F FLOATING POINT CONVERSION ROUTINE

Register	Register Usage
0	Scale factor
1	Pointer to scan character
2	Pointer to array
3	Work Register
4	Working storage base
5	Address of last constant
6	Work Register
7	Phase common low limit scale
8	Phase common high limit scale
9	Low word of decimal
10	Position of decimal point
11	End pointer
12	Exponent modifier
13	Base for Common
14	Adjective exponent, Return Register
15	Binary result

IFNX5L   ERROR LOGGING ROUTINE

Register	Register Usage
0	Work Register
1	Work Register
2	Work Register
3	Work Register
4	Not Used
5	Text record pointer
6	Not Used
7	Base for Phase Common
8	Base Register
9	Return register BALR
10	File index
11	Base for error record
12	Branch register BALR
13	Base for Common
14	Work Register
15	Work Register

IFNX5M MACHINE OP PROCESSOR

Register	Register Usage
0	Not Used
1	" "
2	Work Register
3	Entry pointer
4	Value of expression
5	Fixed part pointer
6	ESDID of expression
7	Base for Phase Common
8	Routine Base
9	Return register BALR
10	Variable text pointer, file index
11	XREF record base
12	Branch register for BALR
13	Base for Common
14	Using fcr extended opcodes
15	Using table pointer

IFNX5P PRINT ROUTINE

Register	Register Usage
0	Not Used
1	Work Register
2	Print buffer using, Punch buffer using
3	Work Register
4	Variable text part Using
5	Fixed text part Using
6	Return register BAL
7	Phase Common Base
8	Return Register BALR
9	Subroutine Base
10	Print index, File index
11	Work Register
12	Branch Register BALR
13	Common Base
14	Field length
15	Work Register

IFNX5V   EVALUATION ROUTINE

Register	Register Usage
0	Not Used
1	Work Register
2	Term stack pointer
3	Relocation list pointer
4	Operator stack pointer
5	Input character pointer (DSECTS)
6	XREF Using
7	Base Phase Common
8	Work Register
9	Branch Register BALR
10	File index, Work Register
11	Symbol definition record Using
12	Return Register BALR
13	Base Common
14	
15	Work Register

IFNX6A POSTPROCESSOR

Register	Register Usage
0	End of Buffer, string count file 1
1	String count file 2
2	Record pointer file 1
3	Record pointer file 2
4	Work Register
5	Work Register
6	Return Register BAL
7	Phase Common Base
8	Subroutine Base
9	Branch register
10	File index
11	Buffer pointer
12	Return register
13	Common Base
14	Work Register, RLD byte count
15	Work Register

IFNX6B   DIAGNOSTIC PHASE

Register	Register Usage
0	Work Register
1	Work Register
2	Message index
3	Message table pointer
4	Counting register
5	Return register BAL
6	Input pointer JGETL buffer
7	Base for Phase Common
8	Base Register
9	Return Register BALR
10	File index, message length
11	Buffer pointer
12	Branch Register BALR
13	Common Base
14	Work Register
15	Work Register

This page intentionally left blank

Appendixes

This section contains reference information about error message origin, macro and copy code usage, metatext flags, internal operation codes, entry points and EXTRN symbols, record formats, and the internal character set.

Appendix A: Error Message/Module Cross-reference

ERROR MESSAGE NUMBER	ISSUING MODULE
DMSASM001E	DMSASM
DMSASM002E	DMSASM
DMSASM003E	DMSASM
DMSASM006E	DMSASM
DMSASM007E	DMSASM
DMSASM038E	DMSASM
DMSASM052E	DMSASM
DMSASM070E	DMSASM
DMSASM074E	DMSASM
DMSASM075E	DMSASM
IFO000	IFNX6B
IFO001	IFNX1J
IFO002	IFNX1J
IFO003	IFNX1J
IFO004	IFNX1J
IFO005	IFNX1J
IFO006	IFNX1J
IFO007	IFNX1J, IFNX1A
IFO008	IFNX1J
IFO009	IFNX1A, IFNX1J
IFO010	IFNX1A, IFNX1J
IFO011	IFNX1J
IFO012	IFNX1A
IFO013	IFNX1A
IFO014	IFNX1A, IFNX1J
IFO015	-
IFO016	IFNX1A, IFNX3A
IFO017	IFNX1A
IFO018	IFNX1A
IFO019	IFNX1A
IFO020	-
IFO021	IFNX1A
IFO022	IFNX1A
IFO023	IFNX1A
IFO024	IFNX1A
IFO025	IFNX1A
IFO026	IFNX1A
IFO027	IFNX1A
IFO028	IFNX1A
IFO029	IFNX1A
IFO030	IFNX1A
IFO031	IFNX1A
IFO032	IFNX1A
IFO033	IFNX1A
IFO034	-
IFO035	IFNX1A, IFNX5D
IFO036	IFNX1A
IFO037	IFNX1A
IFO038	IFNX1A
IFO039	IFNX1A
IFO040	IFNX1A
IFO041	-
IFO042	IFNX1A
IFO043	IFNX1A
IFO044	-
IFO045	-
IFO046	IFNX1A

ERROR MESSAGE NUMBER	ISSUING MODULE
IFO047	IFNX1A
IFO048	IFNX1A
IFO049	IFNX1A
IFO050	IFNX1A
IFO051	IFNX1A
IFO052	IFNX1A
IFO053	IFNX1A
IFO054	IFNX1A
IFO055	IFNX1A
IFO056	-
IFO057	IFNX1A
IFO058	IFNX1A
IFO059	IFNX1A
IFO060	IFNX1A, IFNX3A, IFNX5A
IFO061	IFNX1A
IFO062	IFNX1A
IFO063	-
IFO064	IFNX2A
IFO065	IFNX2A
IFO066	IFNX2A
IFO067	IFNX2A
IFO068	IFNX1A
IFO069	IFNX1A
IFO070	IFNX1A
IFO071	-
IFO072	-
IFO073	IFNX1A
IFO074	IFNX2A
IFO075	-
IFO076	IFNX2A
IFO077	-
IFO078	IFNX3A
IFO079	-
IFO080	IFNX3A
IFO081	IFNX2A
IFO082	-
IFO083	-
IFO084	-
IFO085	IFNX3A
IFO086	-
IFO087	IFNX1A, IFNX3A
IFO088	IFNX3A
IFO089	IFNX3N
IFO090	IFNX3A
IFO091	IFNX3N
IFO092	IFNX3N
IFO093	-
IFO094	-
IFO095	-
IFO096	-
IFO097	-
IFO098	-
IFO099	-
IFO100	IFNX3A
IFO101	IFNX3A
IFO102	IFNX3A
IFO103	-
IFO104	IFNX3A
IFO105	IFNX3A
IFO106	-
IFO107	IFNX3A
IFO108	IFNX3A
IFO109	IFNX3A

ERROR MESSAGE NUMBER	ISSUING MODULE
IFO110	IFNX3A
IFO111	IFNX3A
IFO112	IFNX3A
IFO113	IFNX3A
IFO114	IFNX3A
IFO115	IFNX3A
IFO116	IFNX3A
IFO117	IFNX3A
IFO118	IFNX3A
IFO119	IFNX3A
IFO120	IFNX3A, IFNX5V
IFO121	IFNX3A
IFO122	IFNX3A
IFO123	IFNX3A
IFO124	IFNX3A
IFO125	IFNX3A
IFO126	IFNX3A
IFO127	IFNX3A
IFO128	IFNX3A
IFO129	IFNX3A
IFO130	IFNX3A
IFO131	IFNX3N
IFO132	IFNX3N
IFO133	IFNX3N
IFO134	-
IFO135	-
IFO136	-
IFO137	-
IFO138	-
IFO139	-
IFO140	-
IFO141	-
IFO142	-
IFO143	-
IFO144	-
IFO145	-
IFO146	-
IFO147	-
IFO148	-
IFO149	-
IFO150	-
IFO151	-
IFO152	-
IFO153	-
IFO154	-
IFO155	-
IFO156	-
IFO157	IFNX5D
IFO158	IFNX5A
IFO159	IFNX5D
IFO160	-
IFO161	IFNX5M
IFO162	IFNX5M
IFO163	IFNX5A
IFO164	IFNX5A
IFO165	IFNX5A
IFO166	-
IFO167	IFNX5A, IFNX5C, IFNX5M
IFO168	IFNX5V
IFO169	IFNX5V
IFO170	IFNX5V
IFO171	IFNX5A
IFO172	IFNX5A

ERROR MESSAGE NUMBER	ISSUING MODULE
IFO173	IFNX5A
IFO174	IFNX5A
IFO175	IFNX5A
IFO176	IFNX5A, IFNX5D
IFO177	IFNX5A
IFO178	IFNX5A, IFNX5D, IFNX5M
IFO179	IFNX5A, IFNX5D
IFO180	IFNX5A
IFO181	IFNX5A
IFO182	IFNX5A
IFO183	IFNX5A
IFO184	IFNX5A
IFO185	IFNX5A
IFO186	IFNX5A
IFO187	IFNX5A, IFNX5D, IFNX5V
IFO188	IFNX5V
IFO189	IFNX5A
IFO190	IFNX5A
IFO191	IFNX5A
IFO192	IFNX5A
IFO193	IFNX5A
IFO194	IFNX5A
IFO195	IFNX5A
IFO196	IFNX5A, IFNX5C
IFO197	IFNX5A
IFO198	IFNX5D
IFO199	IFNX5D
IFO200	IFNX5D
IFO201	IFNX5D
IFO202	IFNX5D
IFO203	IFNX5D
IFO203	IFNX5D
IFO204	IFNX5D
IFO205	IFNX5D
IFO206	IFNX5D
IFO207	IFNX5D
IDO208	IFNX5D, IFNX5M
IFO209	IFNX5D, IFNX5M
IFO210	IFNX5A, IFNX5M
IFO211	IFNX5A, IFNX5M
IFO212	IFNX5M
IFO213	IFNX5A, IFNX5D, IFNX5M
IFO214	IFNX5M
IFO215	IFNX5M
IFO216	IFNX5A, IFNX5M
IFO217	IFNX5A, IFNX5M, IFNX5V
IFO218	IFNX5M
IFO219	IFNX5M
IFO220	IFNX5M, IFNX5A
IFO221	IFNX5M
IFO222	IFNX5M
IFO223	IFNX5M
IFO224	IFNX5D, IFNX5M
IFO225	IFNX5M
IFO226	IFNX5M
IFO227	IFNX5M
IFO228	IFNX5M
IFO229	IFNX5M
IFO230	IFNX5D, IFNX5M
IFO231	IFNX5D, IFNX5V
IFO232	-
IFO233	IFNX5V
IFO234	IFNX5V

ERROR MESSAGE NUMBER	ISSUING MODULE
IFO235	IFNX5V
IFO236	IFNX5D, IFNX5V
IFO237	IFNX5A
IFO238	IFNX5V
IFO239	IFNX5D
IFO240	IFNX5A, IFNX5V
IFO241	IFNX5A
IFO242	IFNX5A
IFO243	IFNX5A
IFO244	IFNX5A
IFO245	-
IFO246	IFNX5A, IFNX5M
IFO247	-
IFO248	-
IFO249	-
IFO250	-
IFO251	-
IFO252	-
IFO253	-
IFO254	IFNX5A
IFO255	IFNX5D
IFO256	IFNX6B
IFO257	IFNX6B
IFO258	IFNX6B
IFO259	IFNX6B
IFO260	IFOX0D, IFOX0I
IFO261	IFOX0C, IFOX0E, IFOX0G, IFOX0I
IFO262	IFOX0B, IFOX0I
IFO263	IFOX0I
IFO264	IFNX6B
IFO265	IFNX6B
IFO266	IFOX0I
IFO267	IFNX6B
IFO268	IFNX6B
IFO269	IFNX6B

Appendix B: Macro & Copy Code/Module Cross-reference

Macro Name	Used in Object Modules	Description of Macro
ADT	DMSASM	Maps the Active Displacement Table
CHECK	IFOX0B, IFOX0F	See OS/VS Data Management Macro Instructions.
CLOSE	IFOX0A, IFOX0B, IFOX0C. IFOX0E, IFOX0F, IFOX0G, IFOX0H, IFOX0I	See OS/VS Data Management Macro Instructions.
CMSCB	DMSASM	Simulates OS Control Block
CONTAINS	IFNX4D, IFNX4E, IFNX4M, IFNX4N, IFNX4S, IFNX4T, IFNX4V	Inner macro to ICOMMON used to create external routine name array
CONTENTS	IFNX4E, IFNX4S	Generates a branch table to routines in IFNX4E
DBV	IFNX1A, IFNX1J, IFNX1S, IFNX2A, IFNX3A, IFNX3B, IFNX3N, IFNX4D, IFNX4E, IFNX4M, IFNX4N, IFNX4S, IFNX4T, IFNX4V, IFNX5A, IFNX5C, IFNX5D, IFNX5L, IFNX5M, IFNX5P, IFNX5V, IFNX6B	Defines byte values by using equates and DS 0X.
DCB	IFOX0C, IFOX0E, IFOX0G	See OS/VS Data Management Macro Instructions.
DCBD	IFOX0D, IFOX0F, IFOX0I, IFOX0H	See OS/VS Data Management Macro Instructions.
DCDSWORK	IFNX4D, IFNX4N	Generates a work area for DC/DS Evaluation Routine.
DELETE	IFOX0A, IFOX0I	See OS/VS Supervisor Services and Macro Instructions.
DEVTYPE	IOFX0D, IOFX0I	See OS/VS Data Management for the System Programmer.
DMSERR	DMSASM	Generates Error Messages
DMSKEY	DMSASM	Resets Nucleus or User Protect Key
DSW	IFOX0A, IFNX1A, IFNX1J, IFNX1S, IFNX2A, IFNX3A, IFNX3B, IFNX3N, IFNX4D, IFNX4E, IFNX4M, IFNX4N, IFNX4S, IFNX4T, IFNX4V, IFNX5A, IFNX5C, IFNX5D, IFNX5F, IFNX5L, IFNX5M, IFNX5P, IFNX5V, IFNX6A, IFNX6B, IFNX6C	Defines a switch byte and names the bits in the byte.

Macro Name	Used in Object Modules	Description of Macro
EVALWORK	IFNX4V, IFNX5A, IFNX5C, IFNX5D, IFNX5F, IFNX5V	Generates an evaluation routine work area in the common area of the phase which calls EVALWORK.
FIND	IFOX0F	See OS/VS Data Management Macro Instructions.
FREEMAIN	DMSASM, IFOX0A, IFOX0D, IFOX0F, IFOX0I	See OS/VS Supervisor Services and Macro Instructions.
FREEPOOL	IFOX0F, IFOX0H, IFOX0I	See OS/VS Data Management Macro Instructions.
FSTB	DMSASM	Maps the File Status Table
GENERR	IFNX6C	Generates error messages and a branch table.
GENOP	IFNX1K, IFNX3K	Generates the two op code table modules according to the value of the operands in the call. The macro hashes the op codes into the table and prints the hash chains.
GENOPS	IFNX3A	Programmer macro.
GENTAB	IFNX6C	Programmer macro.
GET	IFOX0F	See OS/VS Data Management Macro Instructions.
GETMAIN	DMSASM, IFOX0A, IFOX0D, IFOX0F	See OS/VS Supervisor Services and Macro Instructions.
GOIF	IFOX0A, IFOX0B, IFOX0D, IFOX0F, IFOX0H, IFOX0I, IFNX1A, IFNX1J, IFNX1S, IFNX2A, IFNX3A, IFNX3B, IFNX4D, IFNX4E, IFNX4M, IFNX4N, IFNX4S, IFNX4T, IFNX4V, IFNX5A, IFNX5C, IFNX5D, IFNX5F, IFNX5L, IFNX5M, IFNX5P, IFNX5V, IFNX6A, IFNX6B	Generates instructions to test a given condition and branch if the condition is satisfied.
GOIF1	IFOX0A, IFOX0B, IFOX0D, IFOX0F, IFOX0H, IFOX0I, IFNX1A, IFNX1J, IFNX1S, IFNX2A, IFNX3A, IFNX3B, IFNX4D, IFNX4E, IFNX4M, IFNX4N, IFNX4S, IFNX4T, IFNX4V, IFNX5A, IFNX5C, IFNX5D, IFNX5E, IFNX5L, IFNX5M, IFNX5P, IFNX5V, IFNX6A, IFNX6B	Inner macro to GOIF. Generates instructions if a switch is to be tested.

Macro Name	Used in Object Modules	Description of Macro
GOIF3	IFOX0A, IFOX0B, IFOX0D, IFOX0F, IFOX0H, IFOX0I, IFNX1A, IFNX1J, IFNX1S, IFNX2A, IFNX3A, IFNX3B, IFNX4D, IFNX4E, IFNX4M, IFNX4N, IFNX4S, IFNX4T, IFNX4V, IFNX5A, IFNX5C, IFNX5D, IFNX5F, IFNX5L, IFNX5M, IFNX5P, IFNX5V, IFNX6A, IFNX6B	Inner macro to GOIF. Generates instructions to test a field other than a switch.
GOTO	IFNX4D, IFNX4E, IFNX4M, IFNX4N, IFNX4S, INFX4T, IFNX4V, IFNX5V	Generates a branch and link to a specified subroutine. The subroutine specified must be a symbol defined in the global array built by the CONTAINS macro.
JCALL	IFOX0D, IFNX3A, IFNX5A, IFNX5C, IFNX5D, IFNX5M, IFNX6B	Generates a branch and link to a subroutine.
JCHECK	IFBX1J, IFNX2A, IFNX3N, IFNX4E, IFNX6A	Generates a call to the Workfile I/O Module Check Routine. This routine checks for a start I/O operation.
JCSECT	IFOX0A, IFOX0B, IFOX0C, IFOX0D, IFOX0E, IFOX0F, IFOX0G, IFOX0H, IFOX0I, IFNX1A, IFNX1J, IFNX1S, IFNX2A, IFNX3A, IFNX3B, IFNX3N, IFNX5A, IFNX5C, IFNX5D, IFNX5F, IFNX5L, IFNX5M, IFNX5P, IFNX6A, IFNX6B, IFNX6C	Generates a CSECT with a CSECT name from the macro instruction operands. If desired, an EQU to the CSECT name will be generated.
JENTRY	IFOX0A, IFOX0B, IFOX0C, IFOX0D, IFOX0F, IFOX0E, IFOX0H, IFOX0I, IFNX1A, IFNX1J, IFNX1S, IFNX2A, IFNX3A, IFNX3B, IFNX3N, IFNX5A, IFNX5C, IFNX5D, IFNX5F, IFNX5L, IFNX5M, IFNX5P, IFNX6A, IFNX6B, IFNX6C	Generates an entry statement and, if desired, an EQU to the entry point.
JEXTRN	IFOX0A, IFOX0D, IFNX1J, IFNX3A, IFNX3N, IFNX4V, IFNX5A, IFNX5C, IFNX5D, IFNX5M, IFNX5V, IFNX6B	Generates an EXTRN statement. An EQU to the external symbol is generated if specified in the macro call.
JFIND	IFNX1A	Generates a call to the FIND routine of the Input I/O Module (IFOX04). The FIND routine locates a macro or a copy code member.
JFRECORE	IFNX1A, IFNX1J, IFNX2A, IFNX3A, IFNX4M, IFNX4T, IFNX5C, IFNX6A, IFNX6B	Generates a call to the Workfile I/O Module (IFOX00) to free a block of storage.
JGEN	IFOX0C	Generates an ORG to a specified address and a DC of specified type and value.

Macro Name	Used in Object Modules	Description of Macro
JGENERR	IFOX0B, IFOX0C, IFOX0D, IFOX0E, IFOX0G, IFOX0I, IFNX1A, IFNX1J, IFNX2A, IFNX3A, IFNX3N, IFNX5A, IFNX5C, IFNX5D, IFNX5M, IFNX5V, IFNX6B	Copies copy code (ERMS) into JERMSGCD.
JGENIN	IFOX0C, IFOX0D, IFNX4E, IFNX5A, IFNX5P, IFNX6A, IFNX6B, IFNX6C	Generates internally coded character strings. It accepts alphanumeric characters, and all special characters except ampersands and quotes.
JGETCORE	IFOX0A, IFNX1A, IFNX1J, IFNX2A, IFNX3A, IFNX3N, IFNX4M, IFNX4T, IFNX5C, IFNX6A, IFNX6B	Generates a call to the I/O Interface Modules to obtain main storage.
JGETL	IFNX3A, IFNX3N, IFNX4V, IFNX5A, IFNX5C, IFNX5D, IFNX5M, IFNX5V, IFNX6B	Generates a call to the Workfile I/O Module (IFOX00) to get the address of the next logical record.
JHEAD	IFOX0A, IFOX0B, IFOX0C, IFOX0D, IFOX0E, IFOX0F, IFOX0G, IFOX0H, IFOX0I, IFNX1J, IFNX1S, IFNX2A, IFNX3A, IFNX3N, IFNX4D, IFNX4E, IFNX4M, IFNX4N, IFNX4S, IFNX4T, IFNX4V, IFNX5A, IFNX5C, IFNX5D, IFNX5F, IFNX5L, IFNX5M, IFNX5P, IFNX5V, IFNX6A, IFNX6B, IFNX6C	Generates a TITLE statement and a status MNOTE in the prolog of a module.
JINPUT	IFNX1A	Generates a call to the Input I/O Module (IFOX04) to get the next record from the input file.
JINST	IFNX1A	Generates machine instructions according to macro call operand values.
JMODID	IFOX0A, IFOX0B, IFOX0C, IFOX0D, IFOX0E, IFOX0F, IFOX0G, IFOX0H, IFOX0I, IFNX1A, IFNX1J, IFNX1S, IFNX2A, IFNX3A, IFNX3B, IFNX3N, IFNX4D, IFNX4E, IFNX4M, IFNX4N, IFNX4S, IFNX4T, IFNX5A, IFNX5C, IFNX5D, IFNX5F, IFNX5L, IFNX5M, IFNX5P, IFNX5V, IFNX6A, IFNX6B, IFNX6C	Generates an embedded identifier which consists of a six character module name identifier and a half word change level identifier.
JNOTE	IFNX1J, IFNX2A, IFNX3N, IFNX4E, IFNX4V, IFNX5D, IFNX5V, IFNX6A	Generates a call to the Workfile I/O Module (IFOX00) to note the position of the last READ or WRITE on a work file.

Macro Name	Used in Object Modules	Description of Macro
JNOTELB	IFNX1A	Generates a call to the Input I/O Module (IFOX04) to note a position in the macro library.
JPARM	IFOX0J	Generates code in IFOX0J which contains bit strings representing the options specified in the PARM field.
JPATCH	IFOX0A, IFOX0B, IFOX0C, IFOX0D, IFOX0E, IFOX0F, IFOX0G, IFOX0H, IFOX0I, IFNX1J, IFNX1S, IFNX2A, IFNX3A, IFNX3B, IFNX3N, IFNX4D, IFNX4E, IFNX4M, IFNX4N, IFNX4S, IFNX4T, IFNX4V, IFNX5A, IFNX5C, IFNX5D, IFNX5F, IFNX5L, IFNX5M, IFNX5P, IFNX5V, IFNX6B, IFNX6C	Calculates the size of a patch area that is originally 5% of the CSECT size, then allocates that patch area.
JPHASE	IFOX0A	Programmer macro
JPOINT	IFNX1J, IFNX2A, IFNX3A, IFNX3N, IFNX4E, IFNX4M, IFNX4T, IFNX4V, IFNX5C, IFNX5D, IFNX5V, IFNX6A	Generates a call to the Workfile I/O Module (IFOX00) to locate a specified position in the work file.
JPOINTLB	IFNX1A	Generates a call to the Input I/O Module (IFOX04) to position the library file in order to get the record after the one noted.
JPRINT	IFNX4E, IFNX5P, IFNX6A, IFNX6B	Generates a call to the Output I/O Module (IFOX06) to print a line on SYSPRINT and to obtain the address of the next buffer.
JPUNCH	IFNX4D, IFNX4E, IFNX4M, IFNX4N, IFNX4T, IFNX5P, IFNX6A, IFNX6B	Generates a call to the Output I/O Module (IFOX06) to output an 80 byte record on SYSPUNCH and SYSGO, and to obtain the address of the next buffer.
JPUTL	IFNX1A, IFNX1J, IFNX3A, IFNX3B, IFNX4M, IFNX4T, IFNX4V, IFNX5A, IFNX5C, IFNX5L, IFNX5M, IFNX5V	Generates a call to the Workfile I/O Module (IFOX00) to obtain the address of the next record in the buffer.
JPUTM	IFNX1J, IFNX3A, IFNX3B, IFNX3N, IFNX4E, IFNX4M, IFNX4T, IFNX5A, IFNX5C, IFNX5L	Generates a call to the Workfile I/O Module (IFOX00) to copy a record into the output buffer.
JREAD	IFNX2A, IFNX3N, IFNX4E, IFNX6A	Generates a call to the Workfile I/O Module (IFOX00) to read a physical record. A JCHECK macro call must be issued before any additional operations on the workfile are attempted.

Macro Name	Used in Object Modules	Description of Macro
JRETURN	IFOX0A, IFOX0B, IFOX0D, IFOX0F, IFOX0H, IFOX0I, IFNX1A, IFNX2A, IFNX3A, IFNX4M, IFNX4T, IGNX5A, IFNX5C, IFNX5D, IFNX5F, IFNX5L, IFNX5M, IFNX5P, IFNX6A, IFNX6B	Restores registers R2 through R9 of the calling program from a push down save area and then returns to the caller via R9.
JSAVE	IFOX0A, IFOX0B, IFOX0D, IFOX0F, IFOX0H, IFOX0I, IFNX1A, IFNX2A, IFNX3A, IFNX4M, IFNX4T, IFNX4V, IFNX5A, IFNX5C, IFNX5F, IFNX5L, IFNX5M, IFNX5P, IFNX5V, IFNX6A, IFNX6B	Saves registers R2 through R9 of the calling program in a push down save area. Unless overridden, the macro will load the base register, R8, from R12 and generate a USING statement.
JPRINT	IFNX6B	Generates a call to the Output I/O Module (IFOX06) to output a record on the system data set.
JTRUNC	IFNX1A, IFNX1J, IFNX3A	Generates a call to the Workfile I/O Module (IFOX00) to truncate an output buffer. This causes the current output buffer to be regarded as full and it is written out on the file. The next logical record will be put in the next physical buffer.
JWRITE	IFNX1J, IFNX2A, IFNX4E, IFNX6A	Generates a call to the Workfile I/O Module (IFOX00) to write a pysical record. This operation must be checked (JCHECK) for completion before any additional operations on the file are attempted.
LOAD	IFOX0A, DMSASM	See Data Management Macro Instructions.
MIEND	IFNX5M	Programmer macro.
MITAB	IFNX5M	Programmer macro.
NOTE	IFOX0B, IFOX0F	See OS/VS Data Management Macro Instructions.
NUCON	DMSASM	Maps the Nucleus Constant Area
OP	IFNX1K, IFNX3K	Inner macro to GENOP. OP is called each time as an op code is to be added to the op code table.
OPCD	IFNX5M	Programmer macro.
OPEN	IFOX0A, IFOX0C, IFOX0E, IFOX0F, IFOX0G, IFOX0H	See OS/VS Data Management Macro Instructions.
OPND	IFNX5M	Programmer macro.
OPS	IFNX3A	Programmer macro.

Macro Name	Used in Object Modules	Description of Macro
OPTDEF	DMSASM	Inner Macro Used to Generate Assembler Options
POINT	IFOX0B, IFOX0F	See OS/VS Data Management Macro Instructions.
PUT	IFOX0H	See OS/VS Data Management Macro Instructions.
READ	IFOX0B, IFOX0E, IFOX0F	See OS/VS Data Management Macro Instructions.
REGEQU	DMSASM	Used to Name Registers
RETURN	IFOX0A	See OS/VS Supervisor Services and Macro Instructions.
SAVE	IFOX0A	See OS/VS Supervisor Services and Macro Instructions.
SET	IFOX0A, IFOX0B, IFOX0D, IFOX0F, IFOX0H, IFNX1A, IFNX1J, IFNX2A, IFNX3A, IFNX3B, IFNX4D, IFNX4E, IFNX4M, IFNX4N, IFNX4S, IFNX4T, IFNX4V, IFNX5A, IFNX5C, IFNX5D, IFNX5L, IFNX5M, IFNX5P, IFNX5V, IFNX6A, IFNX6B	Sets a specified bit on or off.
SYNADAF	IFOX0I	See OS/VS Data Management Macro Instructions.
SYNADRLS	IFOX0I	See OS/VS Data Management Macro Instructions.
TBLGEN	IFNX1A	Generates two different tables: one table of displacements and one table of constants.
TEXT	IFNX5M	Programmer macro.
TIME	IFOX0D	See OS/VS Supervisor Services and Macro Instructions.
TPUT	IFOX0H	See OS/VS Data Management Macro Instructions.
WRITE	IFOX0B, IFOX0C	See OS/VS Data Management Macro Instructions.
WTO	IFOX0D, IFOX0I	See OS/VS Supervisor Services and Macro Instructions.
XDCDS	IFNX4D, IFNX4N	Depending on the call, generates assembler Symbol Resolution DC/DS Evaluation Routines.
XDICT	IFNX4E	Generates module IFNX4E.

Macro Name	Used in Object Modules	Description of Macro
XEVAL	IFNX4V, IFNX5V	Generates evaluation routines IFNX4V, IFNX5V.
XFOUR	IFNX4M, IFNX4T	Depending on the call, generates IFNX4M and IFNX4T.
XSTBL	IFNX4S	Generates IFNX4S.
X5ERRL	IFNX4V, IFNX5A, IFNX5C, IFNX5D, IFNX5M, IFNX5V	Generates a call to IFOX51 Error Logging Routine (IFNX5L) with the error number as a parameter.

Copy Code Name	Used by Object Modules	Description of Copy Code
BMDSECTS	IFNX4V, IFNX5C, IFNX5V	DSECT mapping RLD, XREF, and error records.
EDSECT	IFNX1A, IFNX1J, IFNX1S	DSECT mapping the Edit Phase (IFOX11) Common Area.
ERMS	IFOX0B, IFOX0C, IFOX0D, IFOX0E, IFOX0G, IFOX0I, IFNX1A, IFNX1J, IFNX2A, IFNX3A, IFNX3N, IFNX5A, IFNX5D, IFNX5L, IFNX5M, IFNX5V, IFNX6B	Contains the symbolic names and associated text of all error messages.
GENCOM	IFNX3A, IFNX3N	DSECT mapping generate phase (IFOX31) Common Area.
ICOMMON	IFNX4D, IFNX4E, IFNX4M, IFNX4N, IFNX4S, IFNX4T, IFNX4V, IFNX5V	Contains the code for modules IFNX4D and IFNX4N.
JCOMMON	IFOX0A, IFOX0C, IFOX0D, IFOX0E, IFOX0F, IFOX0G, IFOX0H, IFOX0I, IFNX1A, IFNX1J, IFNX1S, IFNX2A, IFNX3A, IFNX3B, IFNX3N, IFNX4D, IFNX4E, IFNX4M, IFNX4N, IFNX4S, IFNX4V, IFNX5A, IFNX5C, IFNX5D, IFNX5F, IFNX5L, IFNX5M, IFNX5P, IFNX5V, IFNX6A, IFNX6B, IFNX6C	Common DSECT which defines register equates and displacements, bit equates, file equates, internal character code equates, it also issues a USING statement for register 13.
JERMSGCD	IFOX0C, IFOX0D, IFOX0E, IFOX0G, IFOX0I, IFNX1J, IFNX2A, IFNX3A, IFNX3N, IFNX4V, IFNX5A, IFNX5C, IFNX5D, IFNX5M, IFNX5V	DSECT providing symbolic names for error messages and their severity codes.
JERRCD	IFNX1A, IFNX2A, IFNX5A, IFNX5L, IFNX6B	DSECT mapping the error record passed to assembly phase.
JFLEBLK	IFOX0A, IFOX0C, IFOX0D	DSECT mapping the information for a workfile in Master Common.
JINCOM	IFOX0E, IFOX0F, IFOX0I	DSECT mapping Input Common Area.
JOUTCOM	IFOX0G, IFOX0H, IFOX0I	DSECT mapping Output Common Area.
JTEXT	IFNX2A, IFNX3A, IFNX3B, IFNX3N, IFNX4D, IFNX4E, IFNX4M, IFNX4N, IFNX4T, IFNX5A, IFNX5C, IFNX5D, IFNX5L, IFNX5M, IFNX5P, IFNX6B	DSECT mapping the header of edited text records.

Copy Code Name	Used by Object Modules	Description of Copy Code
JTMTXT	IFNX1A, IFNX1J, IFNX3N	DSECT mapping the meta text operators and identifiers.
RSYMRCD	IFNX3A, IFNX3B, IFNX4D, IFNX4E, IFNX4M, IFNX4N, IFNX4T, IFNX4V, IFNX5A, IFNX5L, IFNX5V	Copy code mapping the symbol file records.
RXLFMTS	IFNX5A, IFNX5M, IFNX6A	DSECT mapping RLD, XREF, and literal XREF records.
X5COM	IFNX4V, IFNX5A, IFNX5C, IFNX5D, IFNX5F, IFNX5L, IFNX5M, IFNX5P, IFNX5V	DSECT mapping IFOX51 Common Area.

Appendix C: Internal Operation Codes

The internal operation codes used by the assembler are listed below. These internal codes define the record type of the internal text format for assembler statements. For an internal operation code to exist within an edited text record, the flag JPSOP must be on.

HEX	SYMBOL	DESCRIPTION
00	JTICTL	ICTL instruction
01	JTISEQ	ISEQ instruction
02	JTOPSYN	OPSYN instruction
03	JTCOPY	COPY instruction
04	JTANOP	ANOP instruction
05	JTGBLA	GBLA instruction
06	JTGBLB	GBLB instruction
07	JTGBLC	GBLC instruction
08	JTLCLA	LCLA instruction
09	JTLCLB	LCLB instruction
0A	JTLCLC	LCLC instruction
0B	JTMACRO	MACRO instruction
0C	JTACTR	ACTR instruction
0D	JTAGO	AGO instruction
0D	JTAGB	Same as AGO. (Provided for compatibility.)
0E	JTAIF	AIF instruction
0E	JTAIFB	Same as AIF. (Provided for compatibility.)
0F	JTSETA	SETA instruction
10	JTSETB	SETB instruction
11	JTSET	SETC instruction
12	JTMEXIT	MEXIT instruction
13	JTMEND	MEND instruction
14	JTCALL	CALL macro instruction
15	JTCPKEY	Keyword call parameter
16	JTCPPOS	Positional call parameter
17	JTPROTO	Prototype statement
18	JTPPKEY	Keyword prototype parameter
19	JTPPPOS	Positional prototype parameter
1A	JTPEND	Indicates end of all parameters record for all macro or prototype statements.
1B	JTEND	END instruction
1C	JTDXD	DXD instruction
1D	JTEQU	EQU instruction
1E	JTORG	ORG instruction
1F	JTCNOP	CNOP instruction
20	JTCCW	CCW instruction
21	JTDC	DC instruction
22	JTDS	DS instruction
23	JTSTART	START instruction
24	JTCSECT	CSECT instruction

HEX	SYMBOL	DESCRIPTION
25	JTDSECT	DSECT instruction
26	JTCOM	COM instruction
27	JTENTRY	ENTRY instruction
28	JTEXTRN	EXTRN instruction
29	JTWXTRN	WXTRN instruction
2A	JTCXD	CXD instruction
2B	JTLTORG	LTORG instruction
2C	JTLITR	Literal definition
2D	JTSYMBL	Symbol reference
2E	JTPUNCH	PUNCH instruction
	JTADJII	EST adjustment record
2F	JTREPRO	REPRO instruction
	JTLITII	Literal adjustment record
30	JTPUSH	PUSH instruction
	JTPMOP	Symbol definition in machine operation instructions.
	JTLTEND	End of literal pool
31	JTPOP	POP instruction
	JTEOFII	End of file for symbol interlude phase
32	JTPRINT	PRINT instruction
	JTINPC	Initiate private code
	JTSYMII	Symbol table entry
33	JTUSING	USING instruction
34	JTDROP	DROP instruction
35	JTCMNT	Comment card (* in column 1)
36	JTHCMNT	Hidden comment card (.* in columns 1 and 2)
37	JTERROR	Internal error record
38	JTSPACE	SPACE instruction
39	JTEJECT	EJECT instruction
3A	JTTITLE	TITLE instruction
3B	JTMNOTE	MNOTE conditional assembly instruction
FF	JTEOF	End of text file

Appendix D: Meta Text Flags

HEX	SYMBOL	DESCRIPTION
00	JTMSCM	start character mode
01	JTMECM	end character mode
02	JTMCOM	comma
03	JTMPER	period
04	JTMLPAR	left parenthesis
05	JTMRPAR	right parenthesis
06	JTMPLUS	prefix plus
07	JTMMIN	prefix minus
08	JTMMULT	multiply
09	JTMDIV	divide
0A	JTMADD	add
0B	JTMSUB	subtract
0C	JTMGT	greater than
0D	JTMGE	greater than or equal to
0E	JTMEQ	equal
0F	JTMLE	less than or equal to
10	JTMLT	less than
11	JTMNE	not equal
12	JTMNOT	logical not
13	JTMAND	logical and
14	JTMOR	logical or
15	JTMSTR	string operator
16	JTMDUP	duplication operator
17	JTMDIM	dimension operator
18	JTMDIM2	SYSLIST(n,m) first dimension
19	JTMDIM3	SYSLIST(n,m) second dimension
1A	JTMSTRM	statement termination
1B	JTMTAT	type attribute
1C	JTMLAT	length attribute
1D	JTMSAT	scale attribute
1E	JTMIAT	integer attribute
1F	JTMKAT	count attribute
20	JTMNAT	number attribute
20	JTMHIOP	highest operator
22	JTMSVA	SETA symbol
24	JTMSVB	SETB symbol
26	JTMSVC	SETC symbol
28	JTMOSA	ordinary symbol attribute
2A	JTMSEQ	sequence symbol
2C	JTMSDT	self defining term
2E	JTMCS	character string
30	JTMLSTD	SYSLIST
32	JTMKPAR	keyword parameter
34	JTMPPAR	positional parameter

Appendix E: Entry Point & EXTRN Symbol/Module Cross-reference

Module	Entry Point	EXTRN
ASSEMBLE	DMSASM	DMSASD
IFOX0A	IFOX0A01	IFOX0B01
IFOX0B	IFOX0B01	
IFOX0C	IFOX0C01	
IFOX0D	IFOX0D01	IFOX0J00
IFOX0E	IFOX0E01	
IFOX0F	IFOX0F01	
IFOX0G	IFOX0G01	
IFOX0H	IFOX0H01	
IFOX0I	IFOX0I01	
IFOX0J	IFOX0J00	
IFNX1A	IFNX1A01	IFNX1J01 IFNX1S01
IFNX1J	IFNX1J01	IFNX1K01
IFNX1K	IFNX1K01	
IFNX1S	IFNX1S01	
IFNX2A	IFNX2A01	
IFNX3A	IFNX3A01 IFNX3A02	IFNX3K01 IFNX3B01 IFNX3N01
IFNX3B	IFNX3B01	
IFNX3K	IFNX3K01	
IFNX3N	IFNX3N01	IFNX3N02
IFNX4D	IFNX4D01	
IFNX4E	IFNX4E01	
IFNX4M	IFNX4M01	IFNX4D01 IFNX4E01 IFNX4S01 IFNX4V01

Module	Entry Point	EXTRN
IFNX4N	IFNX4N01	
IFNX4S	IFNX501	
IFNX4T	IFNX4T01	
IFNX4V	IFNX4V01	IFNX4N01 IFNX4E01 IFNX4S01 IFNX4V01
IFNX5A	IFNX5A01 IFNX5A21 IFNX5A31 IFNX5A41 IFNX5A51	IFNX5P01 IFNX5L01 IFNX5V01 IFNX5D01
IFNX5C	IFNX5C01	IFNX5M01 IFNX5A01 IFNX5P01 IFNX5L01
IFNX5D	IFNX5D01	IFNX5V01 IFNX5A21 IFNX5A31 IFNX5F01 IFNX5A41 IFNX5A51 IFNX5L01 IFNX5P01
IFNX5F	IFNX5F01	
IFNX5L	IFNX5L01	
IFNX5M	IFNX5M01	IFNX5P01 IFNX5L01 IFNX5V01
IFNX5P	IFNX5P01	
IFNX5V	IFNX5V01	IFNX5L01
IFNX6A	IFNX6A01 IFNX6A21	
IFNX6B	IFNX6B01 IFNX6B21	IFNX6C01 IFNX6C02
IFNX6C	IFNX6C01 IFNX6C02	

Appendix F: Internal Character Set

Character	Internal	External	Punch
0	00	F0	0
1	01	F1	1
2	02	F2	2
3	03	F3	3
4	04	F4	4
5	05	F5	5
6	06	F6	6
7	07	F7	7
8	08	F8	8
9	09	F9	9
A	0A	C1	12.1
B	0B	C2	12.2
C	0C	C3	12.3
D	0D	C4	12.4
E	0E	C5	12.5
F	0F	C6	12.6
G	10	C7	12.7
H	11	C8	12.8
I	12	C9	12.9
J	13	D1	11.1
K	14	D2	11.2
L	15	D3	11.3
M	16	D4	11.4
N	17	D5	11.5
O	18	D6	11.6
P	19	D7	11.7
Q	1A	D8	11.8
R	1B	D9	11.9
S	1C	E2	0.2
T	1D	E3	0.3
U	1E	E4	0.4
V	1F	E5	0.5
W	20	E6	0.6
X	21	E7	0.7
Y	22	E8	0.8
Z	23	E9	0.9
$	24	5B	11.3.8
#	25	7B	3.8
@	26	7C	4.8
=	27	7E	6.8
(	28	4D	12.5.8
+	29	4E	12.6.8
-	2A	60	11
*	2B	5C	11.4.8
/	2C	61	0.1
)	2D	5D	11.5.8
,	2E	6B	0.3.8
ƀ	2F	40	
'	30	7D	5.8
&	31	50	12
.	32	4B	12.3.8

Appendix G: ESD, TXT, RLD, SYM Record Format

ESD RECORD FORMAT

Columns	Contents
1	12-2-9 punch
2-4	ESD
5-10	Blank
11-12	Variable field count -- number of bytes of information in variable field (columns 17-64)
13-14	Blank
15-16	ESDID of first SD, XD, CM, PC, ER, or WX in variable field
17-64	Variable field. One to three 16 byte items of the following format: 8 bytes -- Name, padded with blanks 1 byte -- ESD type code The hex value is: 00 SD 01 LD 02 ER 04 PC 05 CM 06 XD (PR) 0A WX 3 bytes -- Address 1 byte -- Alignment if XD; otherwise blank 3 bytes -- Length, LDID, or blank
65-72	Blank
73-80	Deck ID and/or sequence number -- The Deck ID is the name from the first named TITLE statement. The name can be one to eight alphameric characters long. If the name is less than eight characters long or if there is no name, the remaining columns contain a card sequence number. (Columns 73-80 of cards produced by PUNCH or REPRO statements do not contain a deck ID or a sequence number).

TEXT (TXT) RECORD FORMAT

Columns	Contents
1	12-2-9 punch
2-4	TXT
5	Blank
6-8	Relative address of first instruction on card
9-10	Blank
11-12	Byte count -- number of bytes in information field (columns 17-72)
13-14	Blank
15-16	ESDID
17-72	56-byte information field
73-90	Deck ID and/or sequence number -- The deck ID is the name from the first named TITLE statement. The name can be one to eight alphanumeric characters long. If the name is less than eight characters long or if there is no name, the remaining columns contain a card sequence number. (Columns 73-80 of cards produced by PUNCH or REPRO statements do not contain a deck ID or a sequence number.)

RLD RECORD FORMAT

Columns	Contents
1	12-2-9 punch
2-4	RLD
5-10	Blank
11-12	Data field count -- number if bytes of information in data field (columns 17-72)
13-16	Blank
17-72	Data field:
17-18	Relocation ESDID
19-20	Position ESDID
21	Flag byte
22-24	Absolute address to be relocated
25-72	Remaining RLD entries
73-80	Deck ID and/or sequence number -- The deck ID is the name from the first named TITLE statement. The name can be one to eight alphanumeric characters long. If the name is less than eight characters long or if there is no name, the remaining columns contain a card sequence number. (Columns 73-80 of cards produced by the PUNCH or REPRO statements do not contain a deck ID or a sequence number.)

If the rightmost bit of the flag byte is set, the following RLD entry has the same relocation ESDID and position ESDID, and this information will not be repeated; if the rightmost bit of the flag byte is not set, the next RLD entry has a different relocation ESDID and/or position ESDID, and both ESDIDs will be recorded.

For example, if the RLD Entries 1, 2, and 3 of the program listing contain the following information:

	Position	Relocation		
	ESDID	ESDID	Flag	Address
Entry 1	02	04	0C	000100
Entry 2	02	04	0C	000104
Entry 3	03	01	0C	000800

SYM RECORD FORMAT

If you specify the TEST assembler option, the assembler punches out symbolic information concerning the assembled program. This output appears ahead of the object module. The format of the card images for SYM output is as follows:

Columns	Contents
1	12-2-9 punch
2-4	SYM
5-10	Blank
11-12	Variable field count -- number of bytes of text in variable field (columns 17-72)
13-16	Blank
17-72	Variable field (see below)
73-80	Deck ID and/or sequence number -- The deck ID is the name from the first named TITLE statement. The name can be one to eight alphanumeric characters long. If the name is less than eight characters long or if there is no name, the remaining cloumns contain a card sequence number. (Columns 73-80 of cards produced by PUNCH or REPRO statements do not contain a deck ID or a sequence number.)

The variable field (columns 17-72) contains up to fifty-six bytes of SYM text. The items making the text are packed together; consequently, only the last card may contain less than fifty-six bytes of text in the variable field. The formats of a text card and an individual text item are shown in Figure 18. The contents of the fields within an individual entry are as follows:

1. Organization (one byte)

   Bit 0:
   - 0 = non-data type
   - 1 = data type

   Bits 1-3 (if non-data type):
   - 000 = space
   - 001 = control section
   - 010 = dummy control section
   - 011 = common
   - 100 = machine instruction
   - 101 = CCW
   - 110 = simply relocatable EQU, named ETORG, named CNOP, or named ORG

   Bit 1 (if data type):
   - 0 = no multiplicity
   - 1 = multiplicity (indicates presence of M field)

   Bit 2 (if data type):
   - 0 = independent (not a packed or zoned decimal constant)
   - 1 = cluster (packed or zoned decimal constant)

Bit 3 (if data type):
0 = no scaling
1 = scaling (indicates presence of S field)

Bit 4:
0 = name present
1 = name not present

Bits 5-7:
Length of name minus 1

2. Address (three bytes) -- displacement from base of control section

3. Symbol Name (zero to eight bytes) -- symbolic name of particular item

Note: The following fields are present only for data-type items.

4. Data Type (one byte) -- contents in hexadecimal

00 = C-type data
04 = X-type data
08 = B-type data
10 = F-type data
14 = H-type data
18 = E-type data
1C = D-type data
20 = A-type or Q-type data
24 = Y-type data
28 = S-type data
2C = V-type data
30 = P-type data
34 = Z-type data
38 = L-type data

5. Length (two bytes for character, hexadecimal or binary items; one byte for other types) -- length of data item minus 1

6. Multiplicity - M field (three bytes) -- equals 1 if not present

7. Scale - signed integer - S field (two bytes) -- present only for F, H, Z, D, L, P, and Z type data, and only if scale is non-zero.

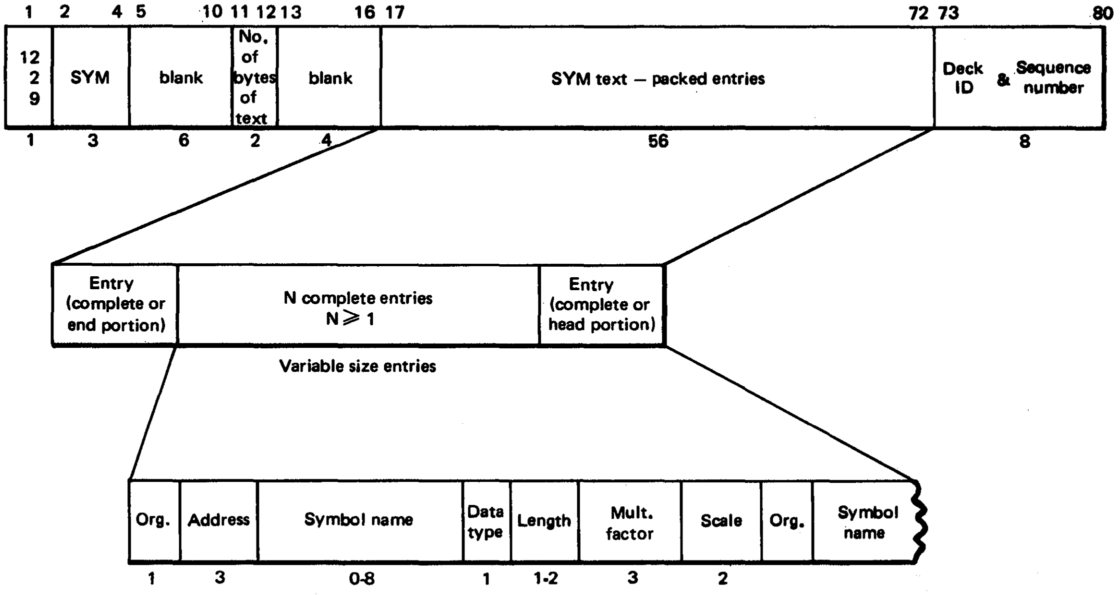


Figure 13. SYM Record Format

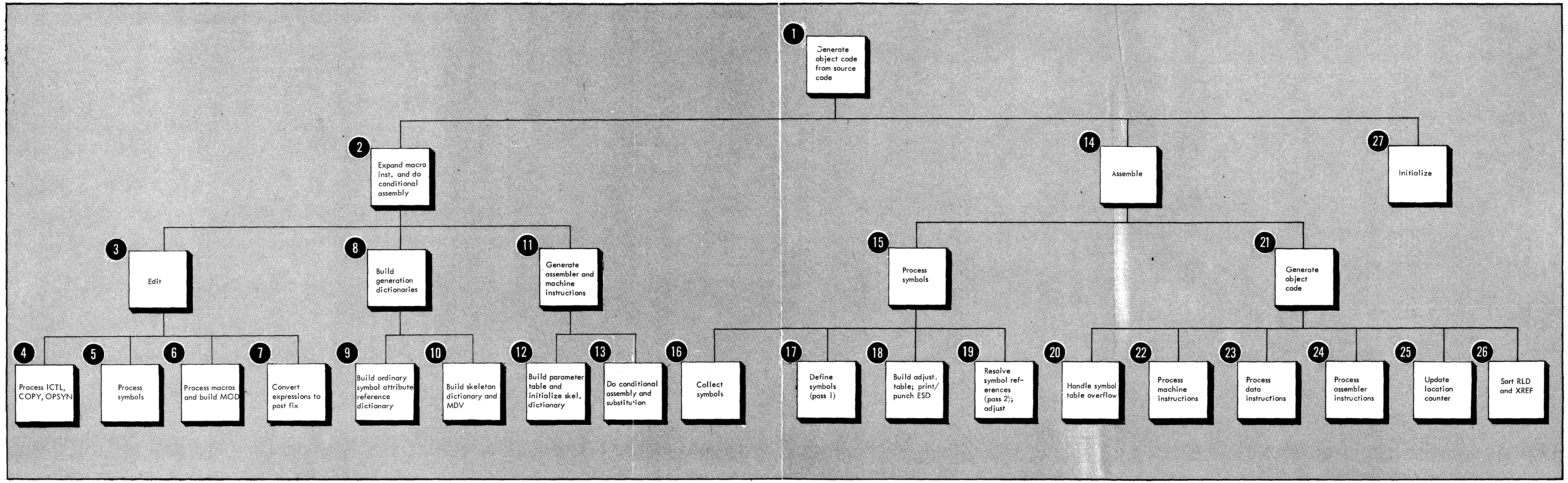

1
Generate object code from source code
2
Expand macro inst. and do conditional assembly
14
Assemble
27
Initialize
3
Edit
8
Build generation dictionaries
11
Generate assembler and machine instructions
15
Process symbols
21
Generate object code
4
Process ICTL, COPY, OPSYN
5
Process symbols
6
Process macros and build MOD
7
Convert expressions to post fix
9
Build ordinary symbol attribute reference dictionary
10
Build skeleton dictionary and MDV
12
Build parameter table and initialize skel. dictionary
13
Do conditional assembly and substitution
16
Collect symbols
17
Define symbols (pass 1)
18
Build adjust. table; print/ punch ESD
19
Resolve symbol references (pass 2); adjust
20
Handle symbol table overflow
22
Process machine instructions
23
Process data instructions
24
Process assembler instructions
25
Update location counter
26
Sort RLD and XREF

Figure 14. Guide to Method of Operation Diagrams

Index

Indexes to program logic manuals are consolidated in the publication OS/VS Master Index for Logic, Order No. GY28-0603. For additional information about any subject listed below, refer to other publications listed for the same subject in the Master Index.

OS/VS - VM/370
Assembler Logic

SY33-8041-1

READER'S COMMENT FORM

Your views about this publication may help improve its usefulness; this form will be sent to the author's department for appropriate action. Using this form to request system assistance or additional publications will delay response, however. *For more direct handling of such request, please contact your IBM representative or the IBM Branch Office serving your locality.*

CUT ALONG DOTTED LINE

Reply requested:
Yes □
No □

Name: ____________________
Job Title: ____________________
Address: ____________________
____________________ Zip ____________________

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)

SY33-8041-1

Your comments, please . . .

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

Fold Fold

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:

International Business Machines Corporation
Department 813 L
1133 Westchester Avenue
White Plains, New York 10604

Fold Fold

IBM®

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601

CUT OR FOLD ALONG LINE

SY33-8041-1

OS/VS – VM/370 Assembler Logic (File No. S370-21 (OS/VS, VM/370) Printed in U.S.A. SY33-8041-1

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601