

LY21-0563-0

File No. S34-32

Licensed Material
Property of IBM

IBM System/34
Utilities Program Product
Program Logic Manual

Program Number 5726-UT1

UTILITIES LOGIC

IBM System/34
Utilities Program Product
Program Logic Manual

Program Number 5726-UT1

Preface

This publication is designed to aid IBM personnel in supporting the IBM System/34 Utilities Program Product. The manual is intended to serve as a recall mechanism and a guide to program listings.

The "How to Use this Manual" section describes the content and organization of this manual.

RELATED PUBLICATIONS

- *IBM System/34 System Support Reference Manual*, SC21-5155
- *IBM System/34 Data File Utility Reference Manual*, SC21-7656
- *IBM System/34 Source Entry Utility Reference Manual*, SC21-7657
- *IBM System/34 Sort Reference Manual*, SC21-7658
- *IBM System/34 Work Station Utility Reference Manual*, SC21-7663
- *IBM System/34 System Data Areas and Diagnostic Aids Handbook*, LY21-0049
- *IBM System/34 Displayed Messages Guide*, SC21-5159
- *IBM System/34 Screen Design Aid Programmer's Guide and Reference Manual*, SC21-7716

First Edition (December 1977)

This edition applies to version 01, modification level 00 of IBM System/34 Utilities Program Product (Program Number 5726-UT1) and to all subsequent versions and modification levels until otherwise indicated in new editions or technical newsletters.

Changes are periodically made to the information herein; before using this publication, refer to the latest *IBM System/34 Bibliography*, GH30-0231, for editions that are applicable and current.

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

A Reader's Comment Form is at the back of this publication. If the form has been removed, address your comments to IBM Corporation, Publications, Department 245, Rochester, Minnesota 55901. Comments become the property of IBM.

CHAPTER 1. WORK STATION UTILITY (WSU)	1-1	Procedures	2-16
INTRODUCTION	1-1	SDA Mainline	2-16
How the Program Works	1-1	SDADROP	2-16
WSU Design	1-4	SDAH	2-16
Physical Characteristics	1-6	SDALOAD	2-16
System Configuration	1-6	SDASAVE	2-17
METHOD OF OPERATION	1-7	SDA1	2-17
PROGRAM ORGANIZATION	1-16	SDA2	2-17
WSU Execution Logic Flow	1-18	SDA#	2-18
Execution Modules and Routines	1-35	SDA5	2-18
#WSX11 (WSU Initialization Phase, Part 1)	1-36	SDA6	2-18
#WSX12 (WSU Initialization Phase, Part 2)	1-37	SDA7	2-19
#WSXP (WSU Execution Processing Phase)	1-37	SDA8	2-19
Resident Processor Descriptions	1-38	SDA9	2-19
Descriptions of Routines Accessed Via Call and Exit	1-39	SDA Index	2-20
Routines Resident in #WSXP	1-39	CHAPTER 3. DATA FILE UTILITY (DFU)	3-1
#WSX00 Routines	1-41	INTRODUCTION	3-1
#WSX01 Routines	1-41	How the Program Works	3-1
#WSX02 Routines	1-42	Physical Characteristics	3-1
#WSX03 Routines	1-43	Nonexecutable Load Modules	3-1
#WSX04 Routines	1-45	Procedures	3-1
#WSX05 Routines	1-45	Input/Output	3-1
#WSX06 Routines	1-49	I/O Storage Requirements	3-2
#WSX07 Routines	1-50	System Configuration	3-2
#WSX08 Routines	1-51	METHOD OF OPERATION	3-3
#WSX09 Routines	1-52	PROGRAM ORGANIZATION	3-10
#WSX10 Routines	1-53	DIRECTORY	3-16
#WSX11 Routines	1-55	DATA AREAS	3-18
#WSX12 Routines	1-55	Job Setup Communications Area (COMMON)	3-18
#WSX13 Routines	1-55	Job Setup Disk Work File (DFUWORKA)	3-18
#WSX14 Routines	1-55	DFU Format Description	3-18
#WSX15 Routines	1-56	Job Execution Communications Area (COMMON)	3-19
#WSX16 Routines	1-56	DIAGNOSTIC AIDS	3-20
#WSX17 Routines	1-58	Messages	3-21
#WSX18 Routines	1-58	Procedures	3-23
#WSX19 Routines	1-59	Update Procedure	3-24
Routine Cross-Reference List	1-63	Inquiry Procedure	3-24
External Calls Made by Execution Routines	1-69	List Procedure	3-25
DIRECTORY	1-72	#DFMP Procedure	3-26
DATA AREAS	1-82	#DFST Procedure	3-26
DIAGNOSTIC AIDS	1-86	DFUDROP Procedure	3-27
WSU Index	1-89	DFULOAD Procedure	3-27
 		DFUSAVE Procedure	3-27
CHAPTER 2. SCREEN DESIGN AID	2-1	CHAPTER 4. SOURCE ENTRY UTILITY	4-1
INTRODUCTION	2-1	INTRODUCTION	4-1
How the Program Works	2-1	Physical Characteristics	4-1
Physical Characteristics	2-1	Load Members	4-1
Load Members	2-1	Procedure Members	4-1
Procedure Members	2-2	Source Members	4-2
Input/Output	2-2	Expansion Buffers	4-2
I/O Storage Requirements	2-2	Input/Output	4-2
System Configuration	2-2	I/O Storage Requirements	4-2
METHOD OF OPERATION	2-3	System Configuration	4-2
PROGRAM ORGANIZATION	2-12	METHOD OF OPERATION	4-3
DIRECTORY	2-13	DIRECTORY	4-11
DATA AREAS	2-14	DATA AREAS	4-13
Local Communications Area	2-14		
DIAGNOSTIC AIDS	2-15		
Messages	2-15		

Communications Area	4-13
Work File	4-13
Work File Record	4-13
Index Area	4-14
Data Area	4-14
DIAGNOSTIC AIDS	4-15
Messages	4-15
Main Storage Status For SEU Members	4-18
Procedures	4-18
SEU	4-18
SEUDROP	4-19
SEULOAD	4-20
SEUSAVE	4-20
CHAPTER 5. SORT	5-1
INTRODUCTION	5-1
How the Program Works	5-1
Physical Characteristics	5-1
Input/Output	5-1
I/O Storage Requirements	5-1
System Configuration	5-2
METHOD OF OPERATION	5-3
PROGRAM ORGANIZATION	5-4
DIRECTORY	5-10
DATA AREAS	5-14
DIAGNOSTIC AIDS	5-16
Messages	5-16
Procedures	5-18
APPENDIX. ACRONYMS AND ABBREVIATIONS	A-1

How to Use this Manual

This manual contains five chapters, one for each portion of the IBM System/34 Utilities Program Product:

- Chapter 1. Work Station Utility
- Chapter 2. Screen Design Aid Utility
- Chapter 3. Data File Utility
- Chapter 4. Source Entry Utility
- Chapter 5. Sort

The following sections are included within each chapter:

1. *Introduction* describes the purpose, function, and operation of the Utilities Program Product.
2. *Method of Operation* describes the functional flow within the program product.
3. *Program Organization* describes module flow and storage maps within the various programs that comprise the Utility Program Product.
4. *Directory* contains information needed for quick reference to the program product module listings.
5. *Data Areas* describes the usage and location of the data areas used by the program product.
6. *Diagnostic Aids* lists the diagnostic messages issued by the various modules within the program product and lists certain applicable program procedure listings.

Additionally, the WSU and SDA chapters contain indexes to help you locate subjects within the chapter.



Introduction

WSU (the work station utility), a program of the Utilities Program Product, provides a way to code data entry, data edit, data correction, and inquiry programs for System/34. Data entry, edit, and correction programs can be front-end entry for RPG II programs that do final editing, processing, updating, and printing (WSU does not provide printed output).

A WSU program can be used by eight operators at once (the program is a multiple-requesting-terminal program). Each program, using input from data entered by operators, from as many as ten master files (direct or indexed files), and from results of operations within the program, creates or maintains one transaction file (a direct, output file). Also, the program can update master files.

HOW THE PROGRAM WORKS

Figure 1-1 is an overview of how WSU works. WSU generation begins when the WSU procedure command or the WSU procedure OCL is entered. The procedure command that begins WSU generation is:

WSU source program name,[library name],[number of blocks],[REPLACE],

LIST
NOLIST
NOLISTW
NOLISTS

source program name: Name of the member that contains the source program.

library name: Name of the library that contains the source program.

number of blocks: Number of blocks allocated for a work file.

REPLACE: Replace existing members with new members that have the same name as those created during generation.

LIST: A complete listing is printed for generation.

NOLIST: Only the WSU generated procedure, job statistics, and message text are listed.

NOLISTW: The WSU generated procedure, job statistics, message text, and \$SFGR output are listed.

NOLISTS: A complete WSU listing is printed, but no \$SFGR listing is printed.

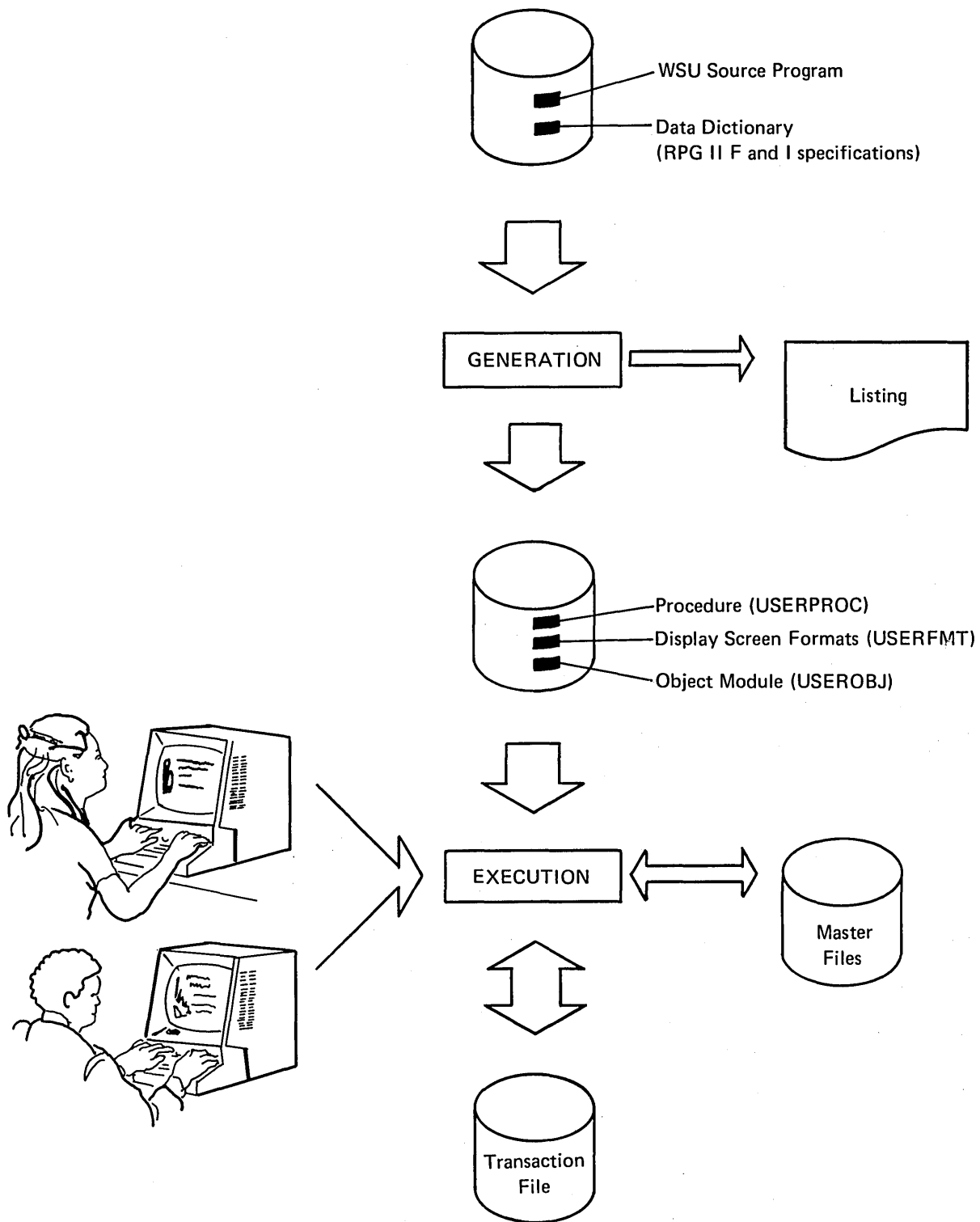


Figure 1-1. WSU Overview

Refer to Chapter 1 of the *WSU Reference Manual* for a description of the OCL that begins WSU generation.

During program generation, WSU uses the last 18 bytes of the display station's local data area for communicating with \$SFGR. Any user data in these 18 bytes will be lost during generation.

Output from generation consists of:

- A WSU program (USEROBJ object member). This object member contains compressed source statements that are used by WSU execution.
- A procedure (USERPROC) that calls the WSU program. This procedure is an MRT procedure. Refer to Chapter 1 of the *WSU Reference Manual* for a description of the OCL that begins WSU generation.
- A format member (USERFMT object member) that contains as many as 32 display screen formats (one for each display in the program).
- A listing of:
 - RPG II F specifications and I specifications in the data dictionary
 - WSU source specifications
 - Generation information (for example, messages)
 - Source statements for the screen format generator program (\$SFGR)

Execution processing begins when USERPROC is invoked by the operator. OCL statements in USERPROC are processed by the system support program (SSP) scheduler, and control is passed to the WSU execution initiator (#WSX11 and #WSX12).

#WSX11 and #WSX12 initialize the execution environment. The main functions performed by #WSX11 are:

- Find the input members (USEROBJ, USERFMT, and USERMSG) specified on the utility control statement in USERPROC.
- Read and process USEROBJ.
- Format the permanent elements and main storage data areas.
- Open the screen format members.

When #WSX11 processing is complete, #WSX12 is loaded over #WSX11 and control is passed to #WSX12.

The main functions performed by #WSX11 are:

- Allocate and open the transaction file and the master files.
- Build a work file (WORKFILE).
- Position the initialization parameter block ahead of the permanent elements.

The last function performed by the execution initiator is to load the WSU execution processor (#WSXP) over #WSX12 and pass control to #WSXP.

#WSXP contains permanently resident data areas and code and completes WSU execution initialization by:

- Securing the IPB
- Formatting the pool area

Execution processing is controlled by #WSXP and the execution transients. The operation involves processing data and specification areas prepared by the execution initiation phase. These areas are located in the main storage region occupied by #WSXP and in the WSU execution work file (see Figure 1-2).

If additional work station operators call USERPROC, the system scheduler and execution initiator processing are bypassed. A work session is started for each associated work station (unless end of job is pending) until the maximum number of work sessions is active. When the maximum number of sessions is active, the system support program prevents additional calls to the execution function until at least one session sign off.

Work session processing is interactive.

As input is received from a work station, processing of the associated work session continues until additional input is requested or the work session is terminated. If there are no suspended gets the session can be restarted and input is then accepted from any display station and the cycle begins again. Input is processed in the order that the input is received.

Work station input is received by way of the SSP work station data management accept input function. Accept input suspends WSU processing until work station input is available. Processing continues from the point of suspension when work station input is again available. When all display stations have been released, the accept is rejected (NEP) and the job ends (non-NEP), or the job is suspended until the procedure is reinvoked. Input from special function keys (command function keys, Roll Down, or Roll Up) is handled by the appropriate routine. If input is not from a special function key, the normal function processor is dispatched and continues processing data until the interaction is complete.

The processing of each work session may be considered serially, without regard to other sessions, when considering the WSU interactive processing cycle. Overall work session processing logic is defined by WSU. The detailed program logic is defined by the account programmer; and the actual processing is under operator control.

For a complete description of WSU's interactive cycle, refer to *WSU Execution Logic Flow in Program Organization*.

At any given time, a work session is in one of three WSU defined modes:

- Enter mode
- Insert mode
- Review mode

In enter mode, records may be logically added to the chain of transaction file records associated with the work session. In insert mode, records may be logically inserted between records in the transaction file chain. In review mode, records may be updated in the transaction file chain. A detailed description of enter mode, insert mode, and review mode is given in the *WSU Reference Manual*.

The ability of the operator to control execution processing is dependent, along with programmer definition, on WSU defined functions. The operator controls execution processing by using the special function keys. A discussion of how the operator may use the special function keys while in enter mode, insert mode, or review mode is also given in the *WSU Reference Manual*.

Error messages and informational messages are issued by the WSU execution processor as they are needed. The messages have the prefix WSU followed by a four digit number (WSU-0XXX). An explanation of the WSU messages and the action required is given in *Displayed Messages Guide*.

The WSU session may be terminated by requesting the menu display and specifying EW on the menu, or by setting on the EW or EJ indicator in the WSU program. The last operator to end a work session ends the WSU program.

WSU DESIGN

In the following explanation of WSU design, letters **A** through **I** refer to Figure 1-2.

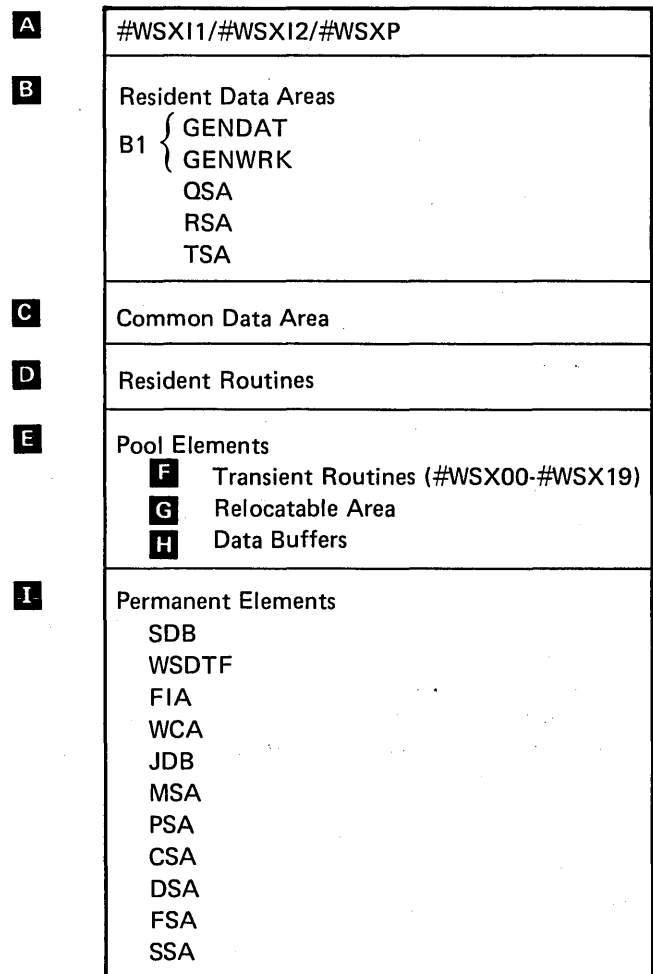


Figure 1-2. WSU Area Layout Execution

The design of WSU execution minimizes storage requirements yet at the same time fully utilizes available region space and reduces execution time.

All initialization functions are performed by code that is executed once and then overlaid **A**.

Except for a small number of services performed by resident routines **D**, post-initialization functions are performed by transient routines. These routines are packaged in load modules (#WSX00-#WSX19) **F** accessed via call/exit linkage subroutines, which handle module loading. Since transients are not loaded at predetermined locations, entry point addresses must be resolved dynamically. Address contents for individual routines are in a table at the end of the module to aid resolution.

To reduce the frequency of transient loading, many inter-module calls expect no return from the called routine. Control does not return (even when expected) if an abort situation is encountered in the called routine. This sequence of calls cycles at natural breakpoints in a WSU/user interaction: when a session must be interrupted for intervention, suspended due to data file record contention, or ended.

COMMON (see the *Data Areas Handbook* for a description of COMMON **C** and other data areas **I**) contains, among other things, 16 work area bytes, the contents of which are saved outside COMMON in the QSA when a transient routine is called. QSBs for this purpose are assigned in order as needed, and also contain linkage information. The work area bytes are restored from the QSB when the transient routine exits, and the called routine QSB is released. The QSA contains 10 QSBs, the maximum call level for a cycle. The QSA is initialized at interaction cycle restart, completion, or end. COMMON also contains 16 work area bytes that are not restored upon return from a called routine.

A large relocation work area B1 is also used as a work area by transient routines. Care must be taken to avoid dependencies on relocation work area integrity across subroutine calls.

Regardless of the number of user formats defined or work stations supported, WSU uses only one work station DTF and screen data buffer.

Frequently-accessed routines and data are permanently resident in #WSXP B, C, D **I**. All other region space is maintained as a pool of other data areas free and temporarily-allocated elements **E**. Other data areas are built as permanent elements at the end of the region **I**. These elements are chained together and have variable sizes.

Each pool element has a permanent priority and (re)assignable attributes. These attributes determine if the element may be relocated or deallocated.

Elements are dynamically relocated to consolidate free space and preserve ordering by priority. Elements are deallocated only when space is required for a new element and no lower priority deallocatable elements are in the pool. Only one element of a particular type (the current one) is unable to be deallocated at a given time.

Data file buffers **H** are allocated from the storage pool, with different priorities given to the transaction file buffer, update-capable master file buffers, and read-only master file buffers.

Nonresident data other than data file records, and transients when not pooled, is kept in a work file. At initialization time, data blocks (message and C-spec) are transferred to the work file then read into the storage pool when needed. Since work file CDBs are padded with zeros and MDBs are blocked three to a sector, CDB and MDB pool elements are reallocated to delete the padding after being read in.

The work file and pool are also used for data blocks composed of work station level pointers and indicators from COMMON and from the job data block.

When the operating mode switches from enter to review, a copy of the current WDB is saved as an alternate WDB. The fields and indicators can then be restored when the operating mode goes back to enter.

Both normal and alternate work station data blocks are built in the storage pool and placed in the work file only when space is required for some other element.

The work file is also used for enter and review mode display save/restore areas for each work station.

Transient modules are loaded as nonrelocatable pool elements. Priorities of modules and other pool elements may be altered to achieve improved performance for a given program, MRTMAX, and region size available for execution.

Transaction file records contain control information in trailers that chain the data records associated with each work session in the proper sequence. The current control information is kept in COMMON. This information is used to format a trailer for each transaction file record added or inserted.

PHYSICAL CHARACTERISTICS

WSU generation is done by 35 modules. WSU execution is done by the WSU execution initiation processors (#WSX11 and #WSX12), the WSU execution mainline processor (#WSXP), and 20 WSU execution transients (#WSX00 through #WSX19). Refer to the *Directory* for a list of WSU modules and a brief description of each.

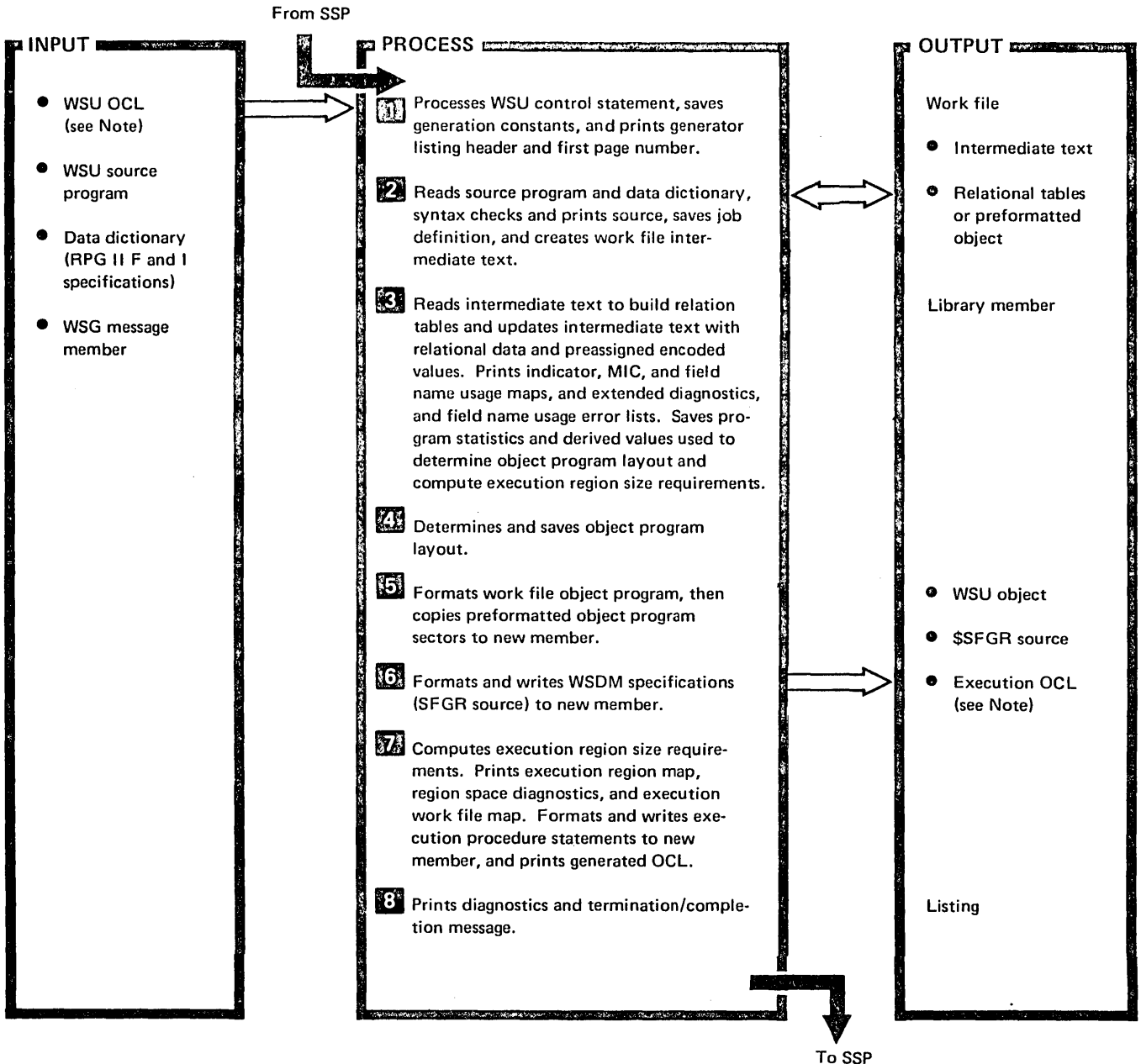
SYSTEM CONFIGURATION

WSU runs on all models of the System/34.

Method of Operation

This section contains two HIPO diagrams:

- Diagram 1-1 describes the input, processing, and output that occurs during WSU generation.
- Diagram 1-2 describes the input, processing, and output that occurs during WSU execution initialization.



Note: Chapter 1 of the *WSU Reference Manual* describes the OCL that is contained in:

- The procedure that begins WSU generation
- The procedure (execution OCL) created by WSU generation

Diagram 1-1 (Part 1 of 6). WSU Generation

DESCRIPTION	MODULE/ ROUTINE
<p>1 Utility control statements are read and processed. The blocks parameter is used to open and allocate the work file.</p> <p>The LIBNAME and MEMNAME parameters are used to find the source program.</p> <p>The system date, generator processor level, screen size, and user (source program) library name are saved in the generator common data area as generation constants.</p> <p>The printer is allocated and opened and a heading and page number are printed to start the listing.</p> <p>2 Source program and data dictionary specifications are read and processed sequentially.</p> <p>Comments are printed and ignored.</p> <p>Unrecognized specifications (invalid form type, J after first specification, or T or M after first S, D, or C specification) are printed with an error note number and ignored.</p> <p>Recognized specifications are printed with statement numbers (except for F, out-of-sequence I, D continuation, C continuation, or C PRTY specifications), then syntax checked and converted to intermediate text records and/or generator common data.</p> <p>Syntax checking involves testing for valid specification sequence, fields required blank/nonblank, and field content (for example data type and ranges), as well as for conflicts with other entries on the same specification or information saved about previously processed source.</p> <p>Library and member entries on T and M specifications are used to locate the data dictionary source.</p> <p>All specifications in the indicated library member are ignored except for properly sequenced F and record-type I specifications with matching filenames and associated AND/OR and field type I specifications.</p> <p>These data dictionary specifications are read, printed, and processed following the corresponding T or M specification. Intermediate text records are formatted and information is saved in the common data area as syntax checking progresses.</p> <p>This information includes the job definition (J specification entries and T and M specification file names) and any data required to implement syntax checks involving previously processed source. Defaults are provided where appropriate to allow generation, or at least syntax checking, to complete.</p> <p>Intermediate text records reflecting errors so severe that extended relational checking for the record fields cannot be done are flagged as dropped.</p> <p>S intermediate text records are formed to provide process segments for special processing levels not represented in the source program.</p> <p>C tag records are formed to provide linkages between process segments and associated command groups.</p> <p>Any syntax errors encountered are printed in a list of note numbers following the specification to which they apply.</p> <p>The intermediate text records are placed in the generator work file. If source program end-of-file is encountered when unexpected (program contains comments only, no T, M, S, D or C after first specification; no S, D, or C after last T or M specification), steps 3, 4, 5, and 6 are bypassed and the generation abnormally ends.</p> <p>3 Multiple passes are made through the intermediate text records for the purpose of collecting relational information required to assign internal values to defined and/or referenced program elements, complete program validation, and make appropriate defaults.</p> <p>Intermediate text records reflecting errors so severe that additional relational checking for the record fields cannot be done are flagged as dropped.</p>	<p>#WSG0 #WSG1 #WSG2 #WSG3A #WSG3B #WSG4A #WSG4B #WSG4C #WSG4D</p>

Diagram 1-1 (Part 2 of 6). WSU Generation

DESCRIPTION	MODULE/ ROUTINE
<p>In addition, statistics about the WSU source are collected and object program and execution region size and work file space requirements for individual control blocks/areas (<i>file names, record types</i>) are computed.</p> <p><i>First pass:</i> File names and record identifying indicators are collected in in-core tables and assigned relative numbers according to the order in which they are encountered in the source program.</p> <p><i>Next 2 passes:</i> Generator common data area fields and T, S, and C records referencing these names or indicators are updated to contain the assigned relative numbers, which can be used to develop interblock pointers in the generated object.</p> <p>Format identifiers are converted to relative processing levels. Information about C record references to file name is saved in the file table and used in the next pass to update T and M records to reflect file usage.</p> <p>During these passes, additional syntax checking is done to detect relational errors involving the usage of file names and record identifying indicators, the definition of screen input/output fields, and other miscellaneous syntax errors.</p> <p>An extended diagnostics list of error note numbers cross-referenced to statement numbers is printed if any of these errors is encountered, and the appropriate defaults are made.</p>	#WSG5
<p><i>(Format identifiers, format names)</i></p> <p><i>First pass:</i> Format names are collected in an in-main storage table and assigned relative numbers according to the order in which they are entered in the source program.</p> <p><i>Next 2 passes:</i> C records referencing these names are updated to contain the assigned relative numbers, which can be used to develop interblock pointers in the generated object. During these passes, additional syntax checking is performed to detect relational errors involving the usage of format identifiers and format names, as well as miscellaneous processing sequence syntax errors.</p> <p>An extended diagnostics list of error note numbers cross-referenced to statement numbers is printed under translatable headings if any of these errors is encountered, and the appropriate defaults are made.</p>	#WSG6
<p><i>(Indicators)</i></p> <p><i>Next pass:</i> Indicator references in all intermediate text records are replaced with mask/displacement equivalents, which act as locators in the generated object.</p> <p>The table of indicators used in the conversion is also used to save information about intermediate text record references to individual indicators.</p> <p>Upon completion of this pass, indicator usage maps are printed. Indicators set and tested are packed into lines in indicator table order and printed. Indicators set but not tested are packed into lines in indicator table order and printed. Indicators tested but not set are packed into lines in indicator order and printed.</p>	#WSGAI
<p><i>(Message texts and message identification codes)</i></p> <p><i>Next 2 passes:</i> Relative numbers are assigned to literal messages in the order in which they are encountered in C records.</p> <p>Once the total number of literal messages is known, another intermediate text updating pass is made to assign the next available relative message numbers to MIC messages in the order in which they are first referenced in the program.</p> <p>The assigned relative numbers are used to develop interblock pointers in the generated object.</p> <p>A table of binary MIC equivalents is built to handle duplicate references, which must be assigned the same relative message number. The MIC table is placed in the workfile after the intermediate text if it cannot be accommodated in main storage.</p>	#WSGAM

Diagram 1-1 (Part 3 of 6). WSU Generation

DESCRIPTION	MODULE/ ROUTINE
<p>Multiple MIC table passes:</p> <p>The table must be scanned each time a MIC is encountered.</p> <p>Additional MIC table pass:</p> <p>Upon completion of this pass, a MIC usage map is printed. Referenced codes are packed in lines in MIC table order and printed.</p> <p><i>(Data field and program label names)</i></p> <p><i>Next pass:</i> The next pass through the intermediate text results in the building of a field name table from field data blocks embedded in the intermediate text. Each data field name or label used in the program is represented by an entry in the table. The first definition of a field name overrides any subsequently encountered definitions and is saved in the appropriate table entry.</p> <p>The intermediate text field data blocks are updated with the appropriate field name table indices.</p> <p>The field name table is placed in the work file after the intermediate text if it cannot fit in storage.</p> <p>Multiple field name table passes:</p> <p>The table must be scanned each time a field data block is encountered.</p> <p>Additional field name table pass:</p> <p>Relative field numbers and JDB or WDB displacements are assigned in table order, ignoring labels and unreferenced or undefined field names, and saved in the corresponding table entries. The assigned relative numbers are used to develop interblock pointers in the generated object.</p> <p>Information concerning the presence or absence of field name definition/reference errors is saved in the common data area to trigger subsequent listing sections.</p> <p>Three additional field name table passes:</p> <p>Field name usage maps are printed, each under translatable headings, for work session level data fields, job level data fields, and program labels.</p> <p>Each name is cross-referenced to the statement number of the specification in which it was defined and listed with its defined length, decimal positions, and assigned data block location (data fields), or with its defined type (program labels).</p> <p><i>Next pass:</i> Commands are built from C records and blocking algorithms are applied to anticipate the location of program labels. These assignments are saved in the field name table.</p> <p>The relative numbers and offsets are used to develop interblock and intrablock pointers in the generated object.</p> <p><i>Next pass:</i> Concurrent multiple direct access to field name table . . .</p> <p>Field name definitions and assignments are copied from the corresponding field name table entries to the field data blocks imbedded in the intermediate text.</p> <p><i>(MIC format field lengths)</i></p> <p><i>Next 2 passes:</i> Tables are built from S and D records for the purpose of computing screen field lengths for MIC message texts.</p> <p>The computed field lengths, combined with specified screen locations for MIC message texts, are used to assign MIC message text lengths. The lengths are checked for WSDM compatibility and defaulted if necessary.</p>	<p>#WSGBF</p> <p>#WSGAF</p> <p>#WSGAL</p> <p>#WSGRF</p> <p>#WSGML</p>

Diagram 1-1 (Part 4 of 6). WSU Generation

DESCRIPTION	MODULE/ ROUTINE
<p>The corresponding D records are updated with the assigned lengths in a separate pass.</p> <p>Any errors encountered are listed as note numbers cross-referenced to the statement numbers of the specifications to which they apply.</p> <p><i>(Operation codes)</i></p> <p><i>Next pass:</i> Any defined but unreferenced field names are listed, cross-referenced to the statement numbers of the specifications in which they were defined.</p> <p><i>Next pass:</i> Any multiply-defined field names are listed, cross-referenced to the statement numbers of the specifications in which they were defined or referenced.</p> <p><i>Next pass:</i> Any referenced but undefined field names are listed, cross-referenced to the statement numbers of the specifications in which they were referenced.</p> <p><i>Next pass:</i> Syntax checks and final command code assignments dependent upon the resolution of embedded field data blocks are performed and appropriate defaults made.</p> <p>Any errors encountered are listed as note numbers, cross-referenced to the statement numbers of the specifications to which they apply.</p> <p>If any terminal diagnostic note numbers have been listed, steps 4 through 7 are bypassed.</p>	<p>#WSG8</p> <p>#WSGFO</p>
<p>4 Program statistics and derived values saved in the common data area and encoded intermediate text records are used to anticipate the object program layout.</p> <p>Individual specification block area displacements are saved for use in converting relative numbers to block locators in generated code. Total specification and data area size requirements are saved for the object control header and used to assign relative sector addresses for the object areas.</p> <p>The specification and data area relative sector addresses are also saved for the control header.</p>	
<p>5 Multiple passes are made through the intermediate text records to format specification and data areas and place them in the work file after the intermediate text.</p> <p>Also referenced for formatting purposes are common data area fields initialized in preceding steps and the tables of specification block displacements.</p> <p>The formatting buffer used occupies the space between the end of the executing module and the start of the intermediate text buffer used. Formatting is done in place at the end of the buffered object stream, except for individual commands, which must be prebuilt in a separate formatting buffer before being blocked at the stream end. Additional statistics are collected during these formatting passes. The control header is built last from generation constants, the job definition, statistics and derived values, and object layout data. Once built, it is placed in the first relative work file object sector. WSDM module \$MAPGS is loaded after the executing WSU processor and all space to the end of the region made available for buffering. Preformatted object program sectors are retrieved from the work file to the assigned buffer and put (via a call to \$MAPGS each time the buffer is filled or after all object sectors have been buffered) into a new object member in the specified or defaulted library.</p>	<p>#WSGFS #WSGDS #WSGCS #WSGMS #WSGPS #WSGSS #WSGMD #WSGCD #WSGEH #WSGOM</p>
<p>6 WSDM module \$MAPUR is loaded after the executing WSU processor and all space to the start of intermediate text buffers made available for \$SFGR source buffering.</p> <p>\$SFGR source specifications are formatted from intermediate text and nontranslatable character constants and put (via a call to \$MAPUR for each statement) into a new source member in the specified or defaulted library.</p>	<p>#WSG9</p>

Diagram 1-1 (Part 5 of 6). WSU Generation

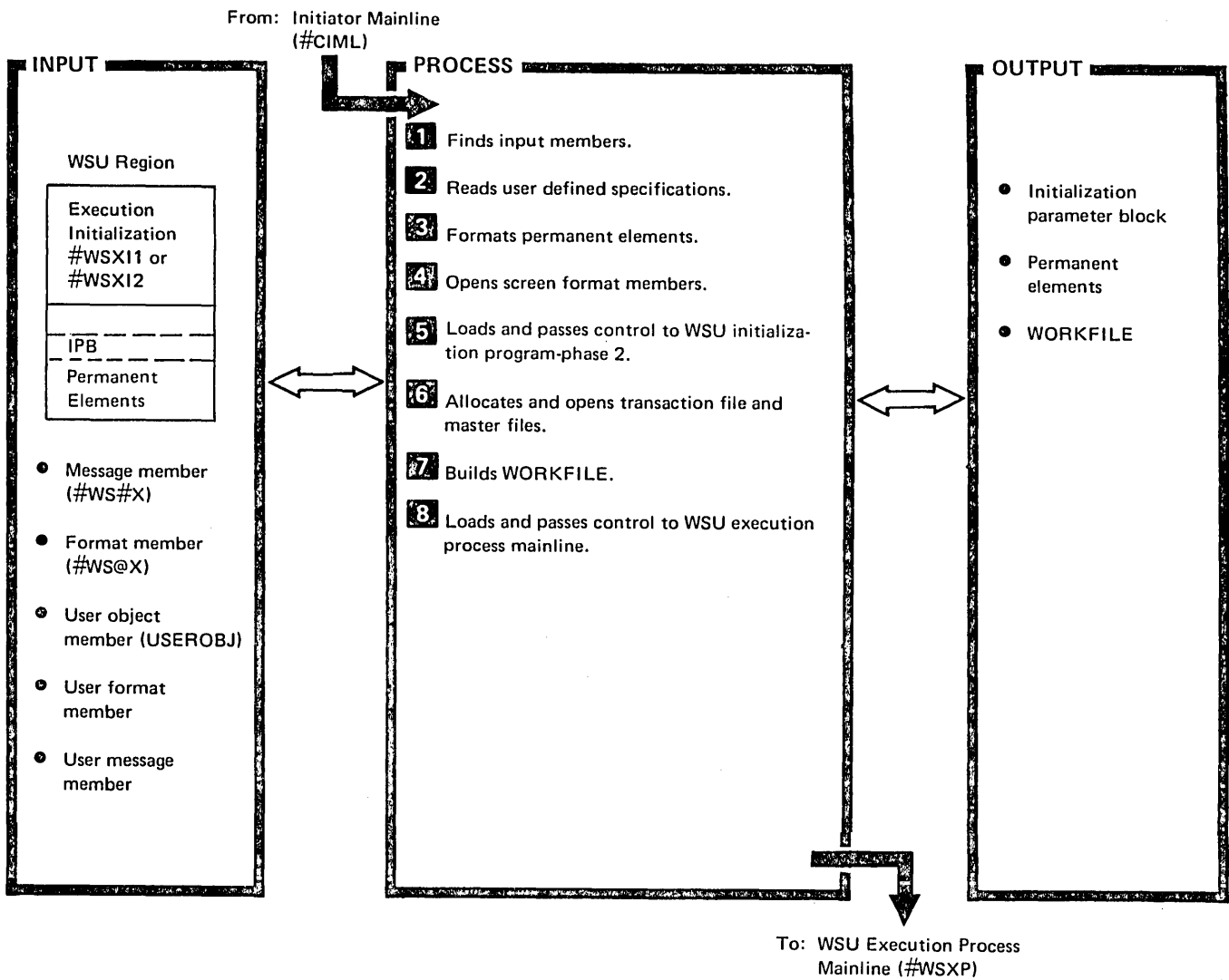


Diagram 1-2 (Part 1 of 3). WSU Execution Initialization (#WSX11 and #WSX12)

DESCRIPTION	MODULE/ ROUTINE
1 <ul style="list-style-type: none"> ● Finds WSU message member. 	#MASFN
<p style="padding-left: 20px;">If message member not found, issue abort via SYSLOG.</p>	#WSX11
<p style="padding-left: 20px;">Overlay prog-1 message member in job control block (JCB).</p>	
<ul style="list-style-type: none"> ● Reads utility control statements. 	#CLSS
<ul style="list-style-type: none"> ● Finds user object member (USEROBJ). 	#MASFN
<ul style="list-style-type: none"> ● Puts USEROBJ address in IOS table. 	#WSX11
<ul style="list-style-type: none"> ● Finds user message member. 	#MASFN
<ul style="list-style-type: none"> ● Puts format and relative address of user message member in JCB. 	#WSX11
2 <ul style="list-style-type: none"> ● Uses information retrieval transient to get region, NEP attribute and MRTMAX value. 	
<ul style="list-style-type: none"> ● Reads USEROBJ control sector from disk. 	Disk IOS
<ul style="list-style-type: none"> ● Calculates required region size. 	#WSX11
3 <ul style="list-style-type: none"> ● Sets up and loads screen specification area (SSA). 	
<ul style="list-style-type: none"> ● Sets up and loads file specification area (FSA). 	
<ul style="list-style-type: none"> ● Loads data specification area (DSA). 	
<ul style="list-style-type: none"> ● Loads C-spec specification area (CSA). 	
<ul style="list-style-type: none"> ● Sets up and loads process specification area (PSA). 	
<ul style="list-style-type: none"> ● Calculates WSU message specification area (MSA) size and set to X'00'. 	
<ul style="list-style-type: none"> ● Loads message specification area (MSA). 	
<ul style="list-style-type: none"> ● Creates job data block (JDB) and WS data block (WDB) work areas. 	
<ul style="list-style-type: none"> ● Creates work station control area (WCA). 	
<ul style="list-style-type: none"> ● If required, creates master track index area and format index area. 	#WSX11
4 <ul style="list-style-type: none"> ● If no user format library name specified, inserts WSU format library name in DTF. 	
<ul style="list-style-type: none"> ● Allocates user/WSU format member. 	#CAML
<ul style="list-style-type: none"> ● Opens format member. 	#DMOP
<ul style="list-style-type: none"> ● Builds screen data block (SDB). 	#WSX11
5 <ul style="list-style-type: none"> ● Finds #WSX12. 	#MASFN
<ul style="list-style-type: none"> ● Load and pass control to #WSX12. 	#WSX11

Diagram 1-2 (Part 2 of 3). WSU Execution Initialization (#WSX11 and #WSX12)

DESCRIPTION	MODULE/ ROUTINE
6 <ul style="list-style-type: none"> Moves IOB build area address into the DTF in the file specification block (FSB). 	#WSX12
<ul style="list-style-type: none"> Allocates transaction or master file. 	#CAML
<ul style="list-style-type: none"> For a transaction file, does deallocate and special allocate functions. 	#WSX12
<ul style="list-style-type: none"> Opens file. 	#DMDP
<ul style="list-style-type: none"> Updates FSB and copies IOBs. 	#WSX12
7 <ul style="list-style-type: none"> Calculates WORKFILE size. 	#CAS1
<ul style="list-style-type: none"> Allocates WORKFILE as scratch file. 	#DMOP
<ul style="list-style-type: none"> Opens WORKFILE (dummy open). 	Disk IOS
<ul style="list-style-type: none"> Retrieves USEROBJ literal messages and places in WORKFILE. 	#MGRET
<ul style="list-style-type: none"> Retrieves messages from user message member and places in WORKFILE. 	#WSX12
<ul style="list-style-type: none"> Retrieves messages from WSU execution message member and places in WORKFILE. 	Disk IOS
<ul style="list-style-type: none"> Processes C-spec: <ul style="list-style-type: none"> Reads from USEROBJ. Updates CSA, CSB, DSA, and CSP. Writes C-spec to WORKFILE. 	#WSX12
<p>Initializes work session control block (WCB) normal and alternate WDB pointers and enter and review screen save area pointer.</p>	#WSX12
<p>Initializes WDB key entries with keyfield length-1 from DTFs for update-capable fields.</p>	#WSX12
8 <ul style="list-style-type: none"> Positions IPB ahead of permanent elements. 	#MASFN
<ul style="list-style-type: none"> Finds #WSXP. 	#WSX12
<ul style="list-style-type: none"> Loads and passes control to #WSXP startup routine. 	#WSX12

Diagram 1-2 (Part 3 of 3). WSU Execution Initialization (#WSX11 and #WSX12)

Program Organization

This section contains:

- Generation logic flow figure
- Execution logic flow figure
- Execution modules and routines descriptions
- Execution routines cross reference list
- Execution routines external calls list

WSU Generation Logic Flow
 Figure 1-3 shows the generation logic flow.

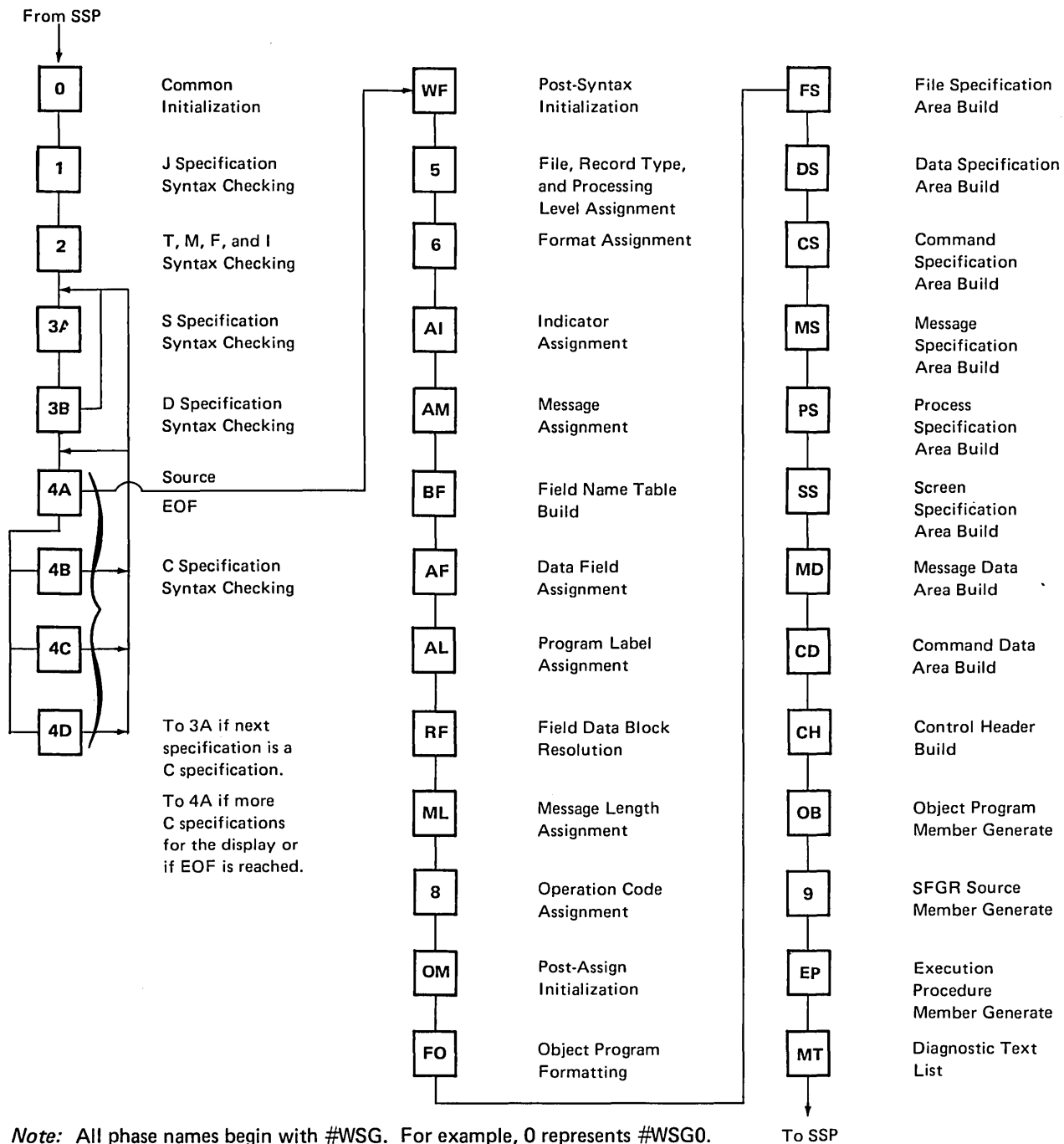
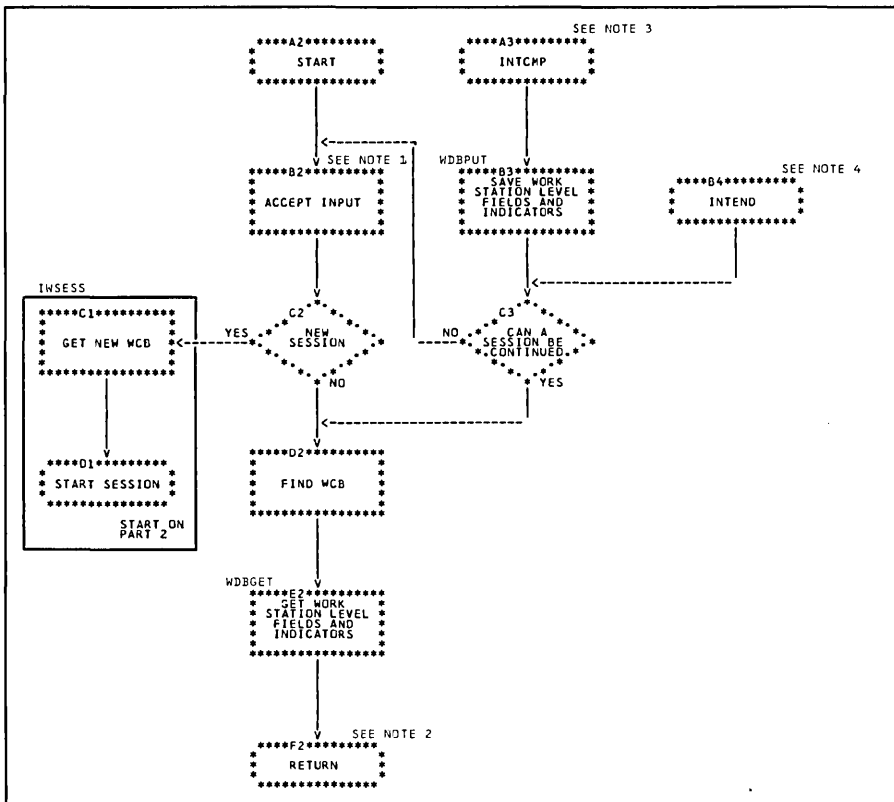


Figure 1-3. WSU Generation Module Flow

WSU EXECUTION LOGIC FLOW

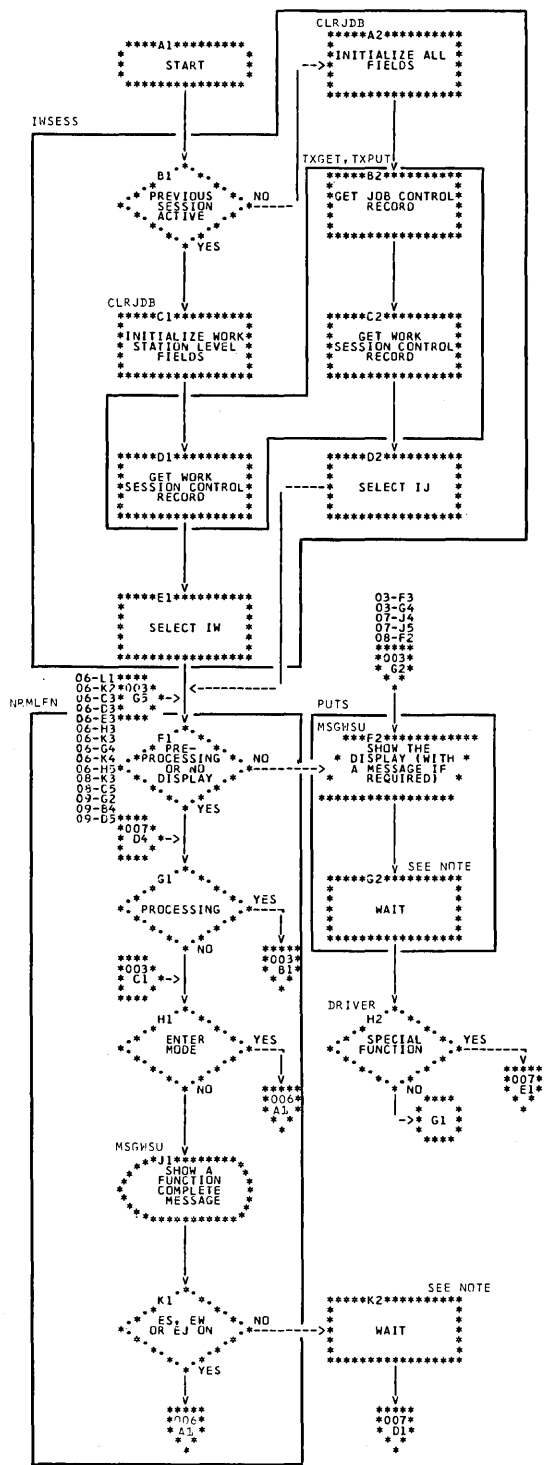
Work session processing occurs interactively under control of the cycle driver routine (DRIVER in #WSX10). Part 1 of Figure 1-4 shows the logic flow *between* work stations using a WSU program. Parts 2 through 9 of this figure show the logic flow for an individual work station.

Note: Explanations of routines that are specified in Figure 1-4 can be found following the figure (Execution Modules and Routines).

**Notes:**

1. Work station input is received via the SSP work station data management accept input function. Accept input suspends WSU processing (produces a wait state) until work station input is available.
2. Return to the point in the cycle at which the wait occurred.
3. INTCMP is branched to each time a wait occurs.
4. INTEND is branched to each time a work session ends.

Figure 1-4 (Part 1 of 9). Work Station Logic Flow



Note: Control is given to INTCMP (on Part 1 of this figure) to determine if processing can occur for another session.

Figure 1-4 (Part 2 of 9). Work Station Logic Flow

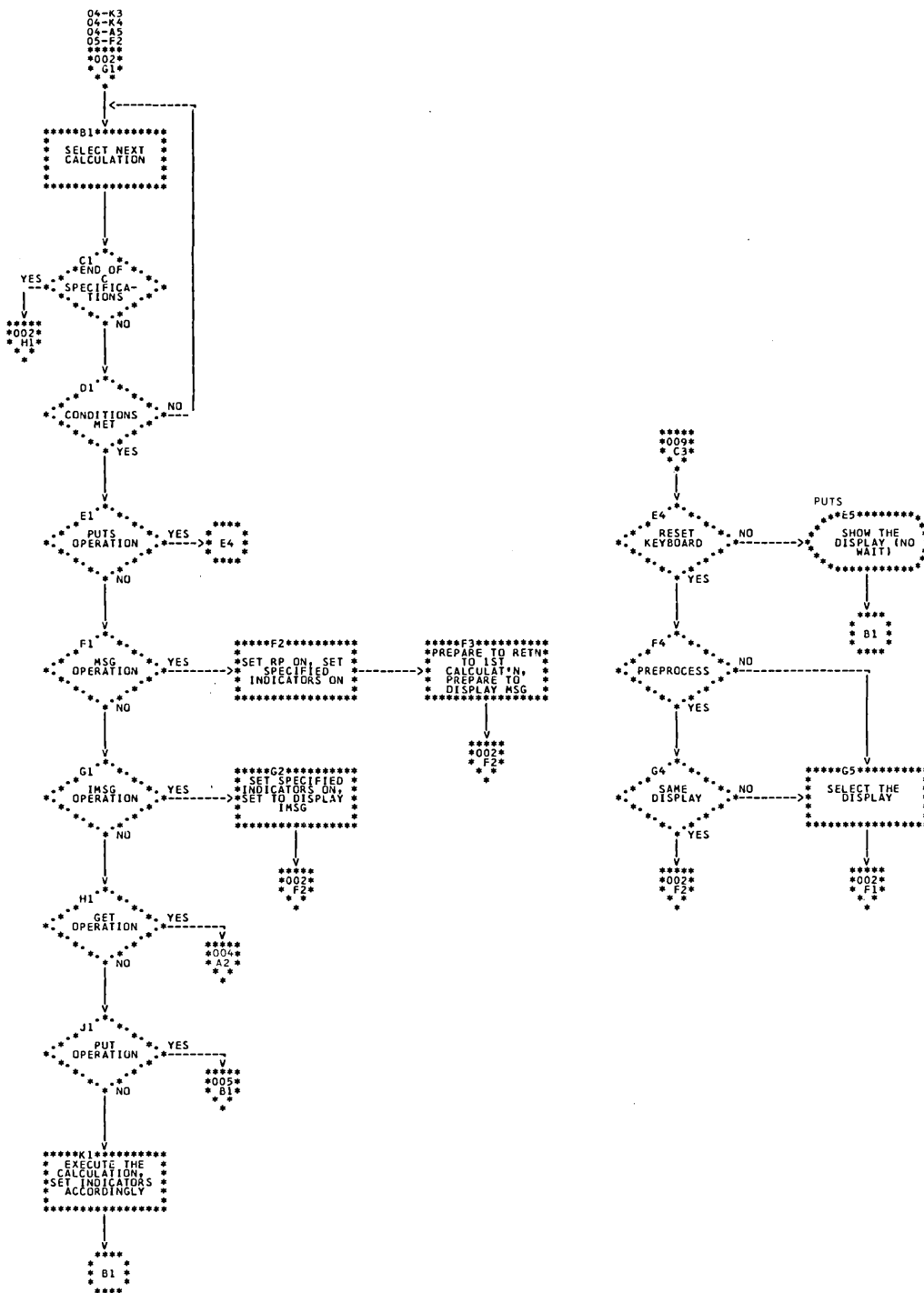
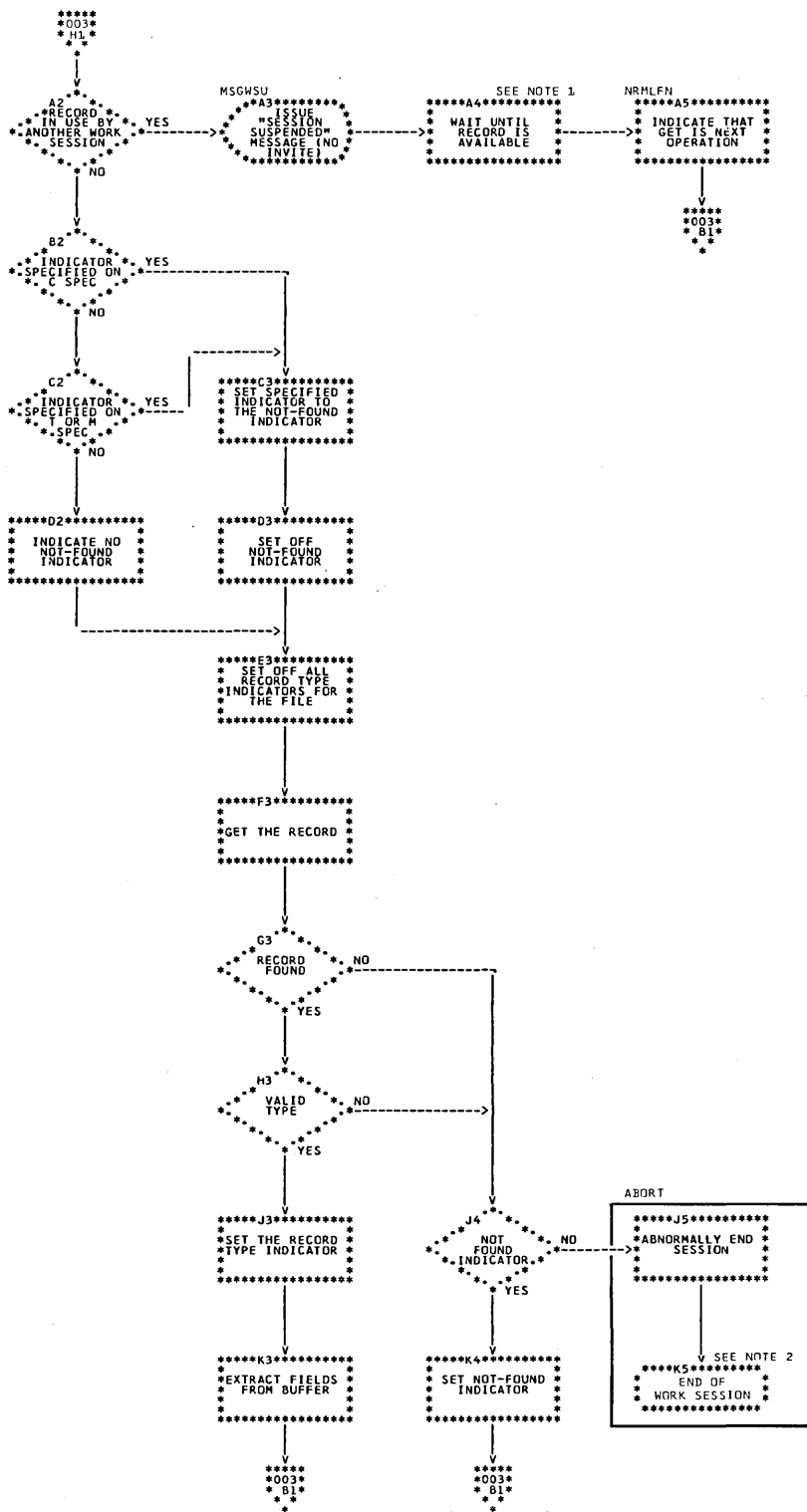


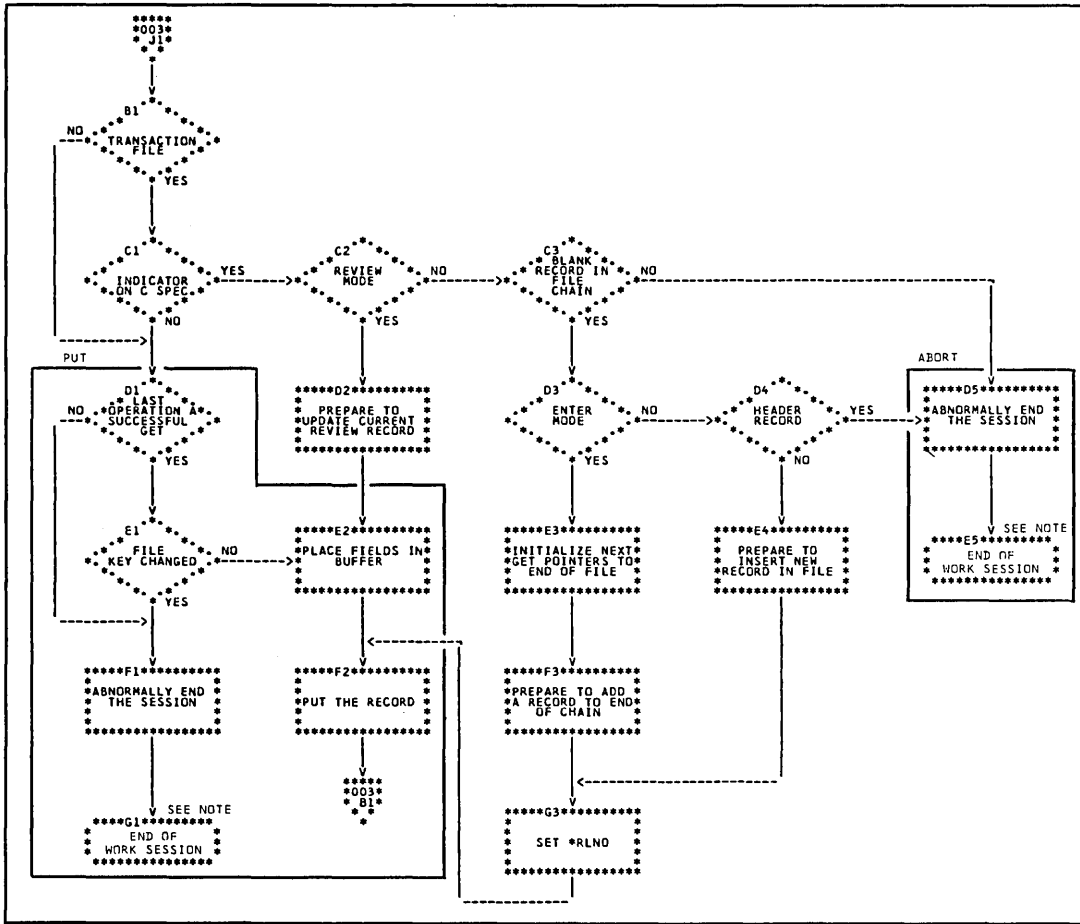
Figure 1-4 (Part 3 of 9). Work Station Logic Flow



Notes:

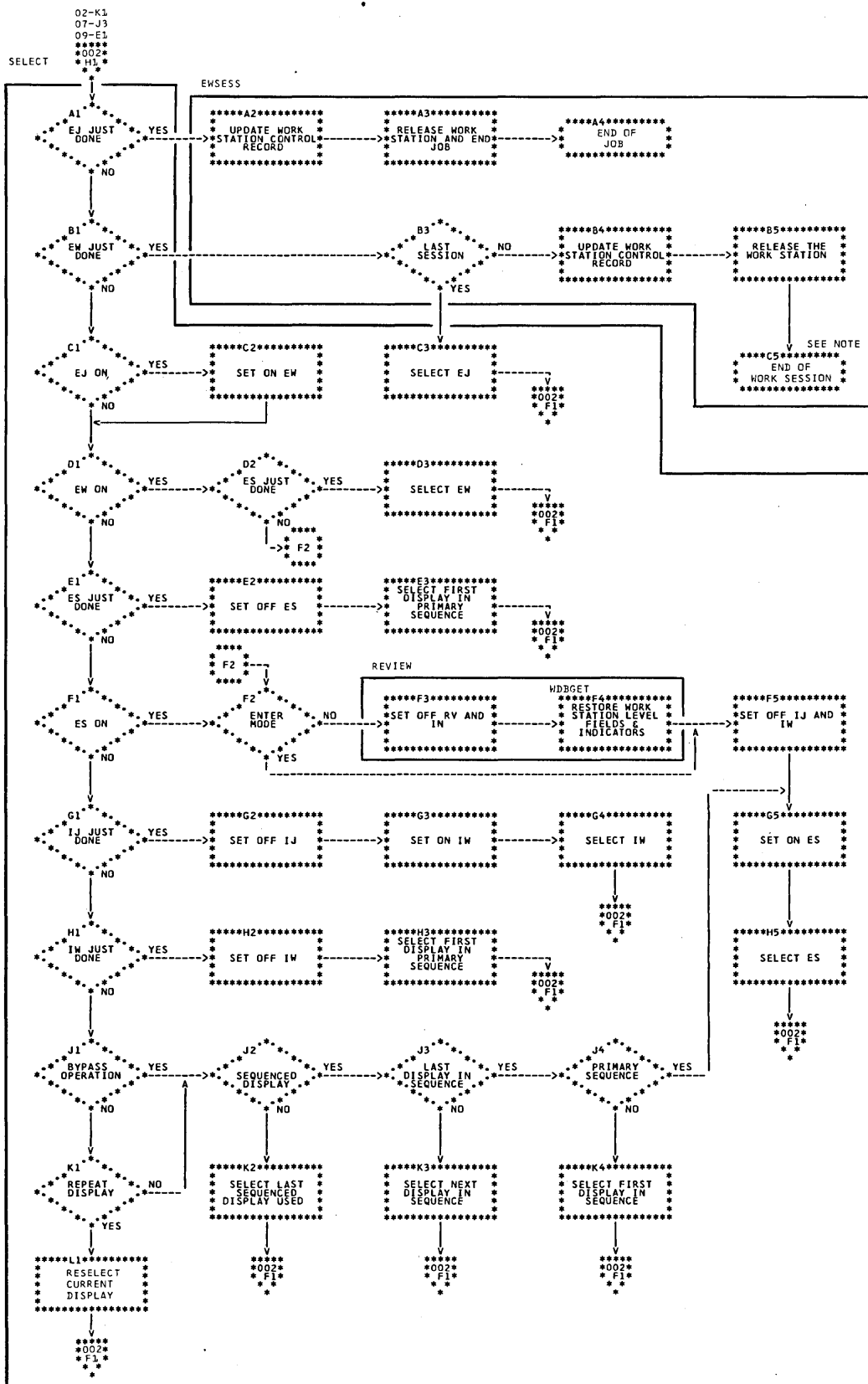
1. Control is given to INTCMP (on Part 1 of this figure) to determine if processing can occur for another session.
2. Processing begins at INTEND on Part 1 of this figure.

Figure 1-4 (Part 4 of 9). Work Station Logic Flow



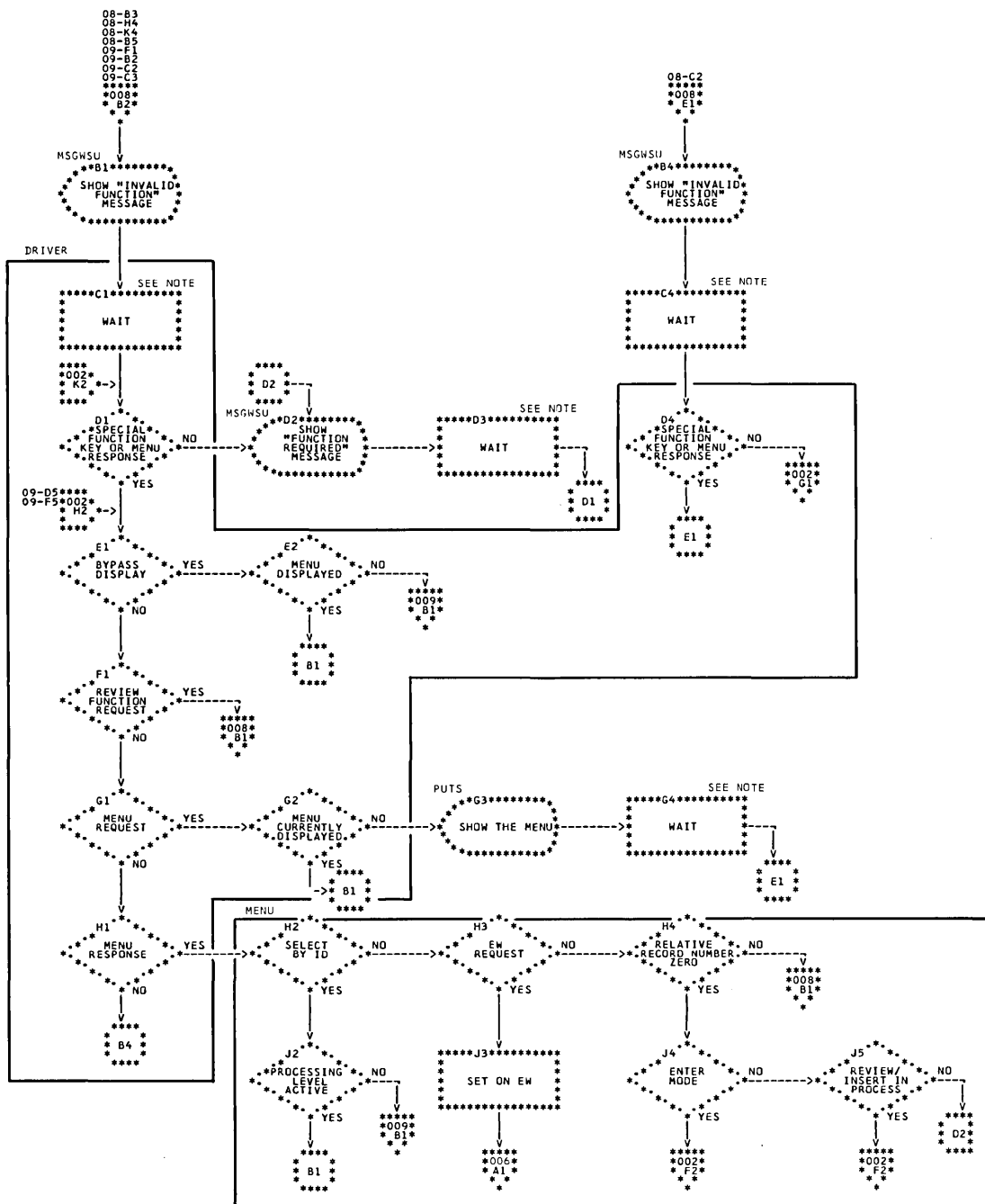
Note: Processing begins at INTEND on Part 1 of this figure.

Figure 1-4 (Part 5 of 9). Work Station Logic Flow



Note: Processing begins at INTEND on Part 1 of this figure.

Figure 1-4 (Part 6 of 9). Work Station Logic Flow



Note: Control is given to INTCMP (on Part 1 of this figure) to determine if processing can occur for another session.

Figure 1-4 (Part 7 of 9). Work Station Logic Flow

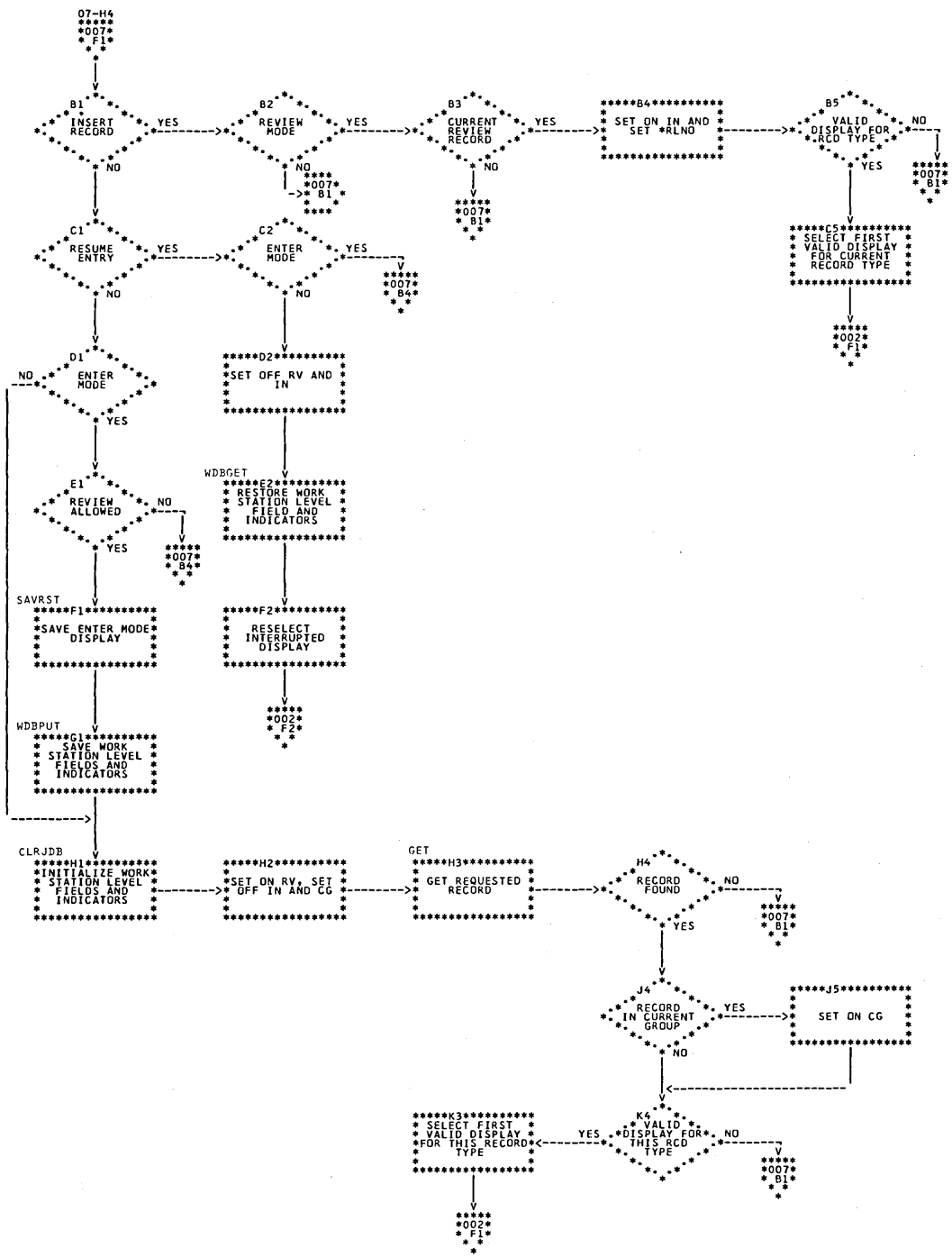
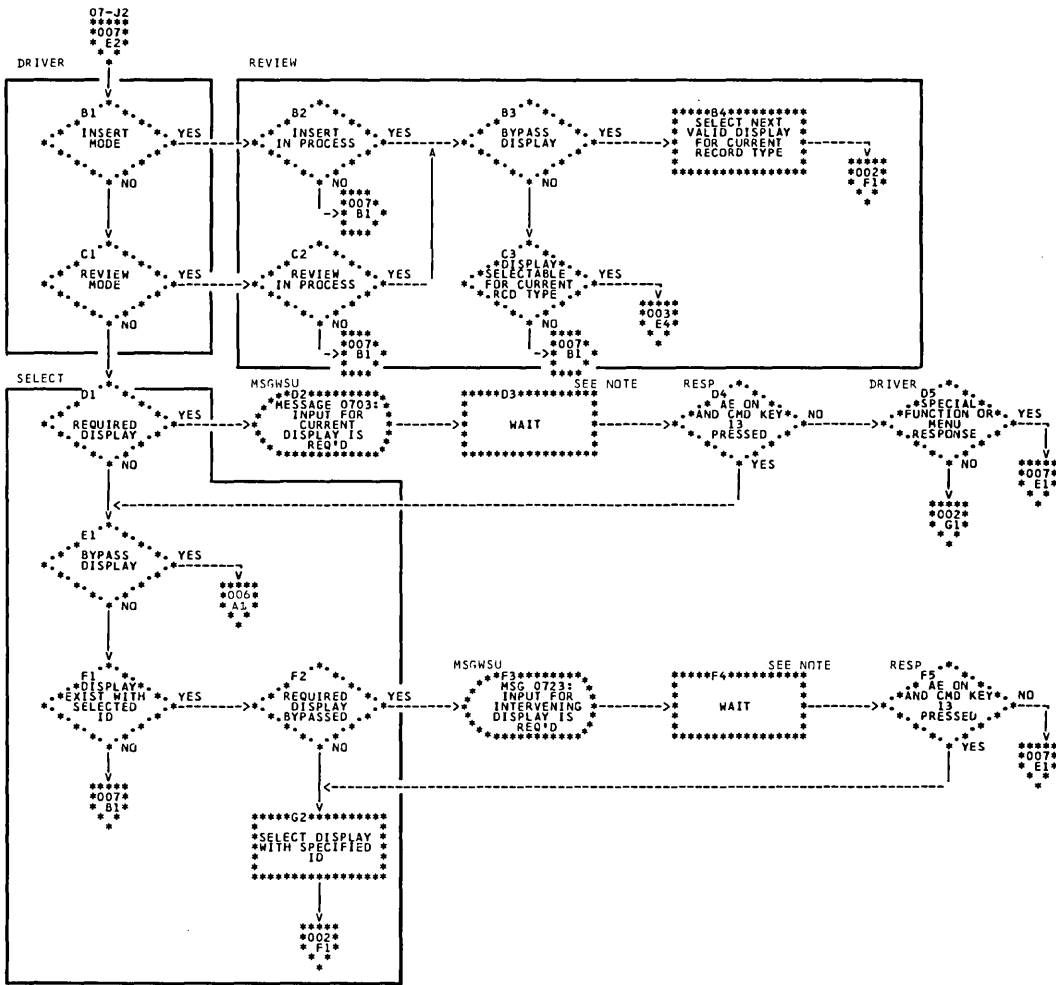


Figure 1-4 (Part 8 of 9). Work Station Logic Flow



Note: Control is given to INTCMP (on Part 1 of this figure) to determine if processing can occur for another session.

Figure 1-4 (Part 9 of 9). Work Station Logic Flow

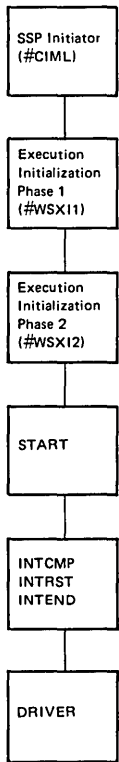


Figure 1-5 (Part 1 of 7).WSU Execution Hierarchy

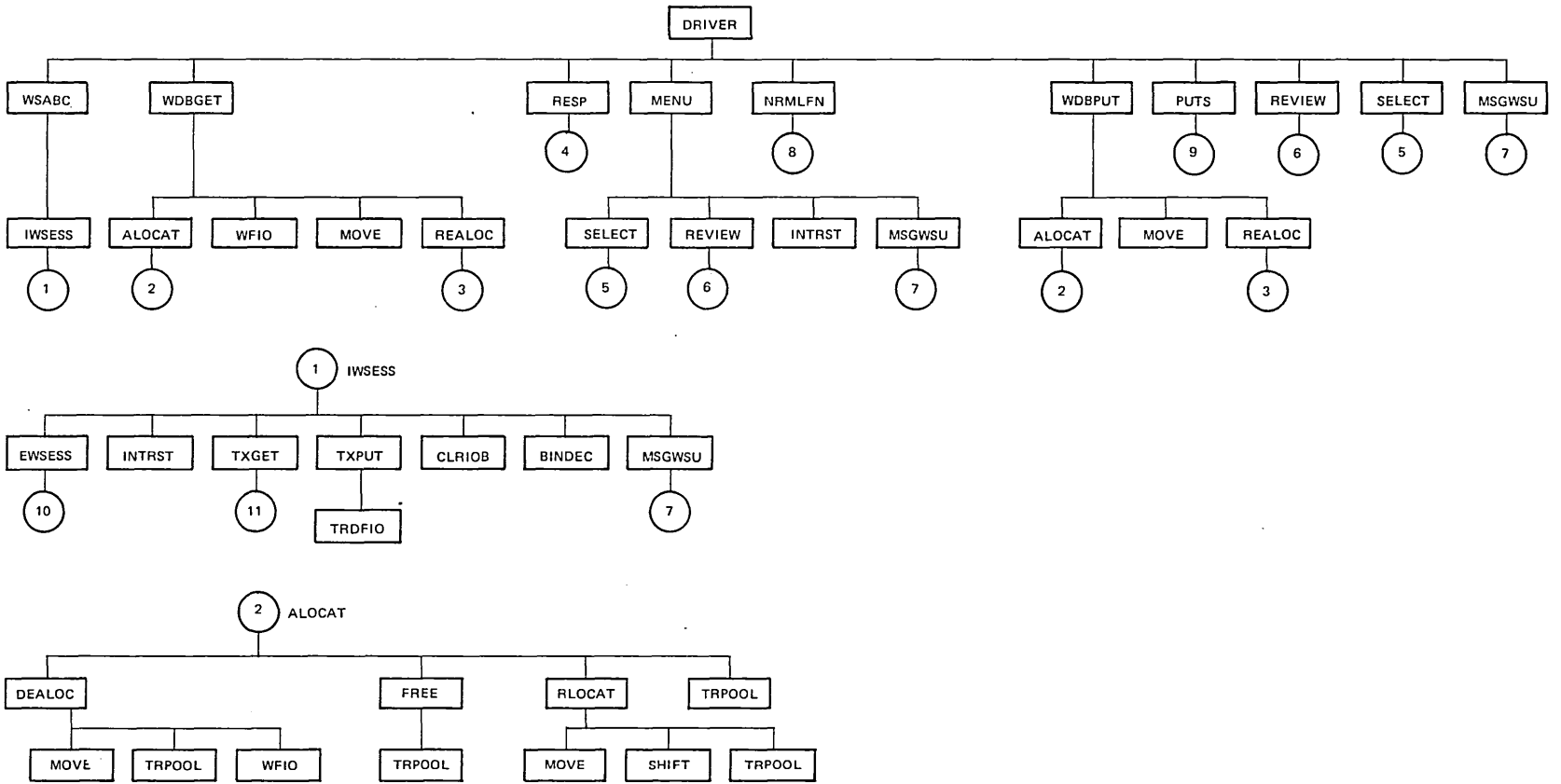


Figure 1-5 (Part 2 of 7). WSU Execution Hierarchy

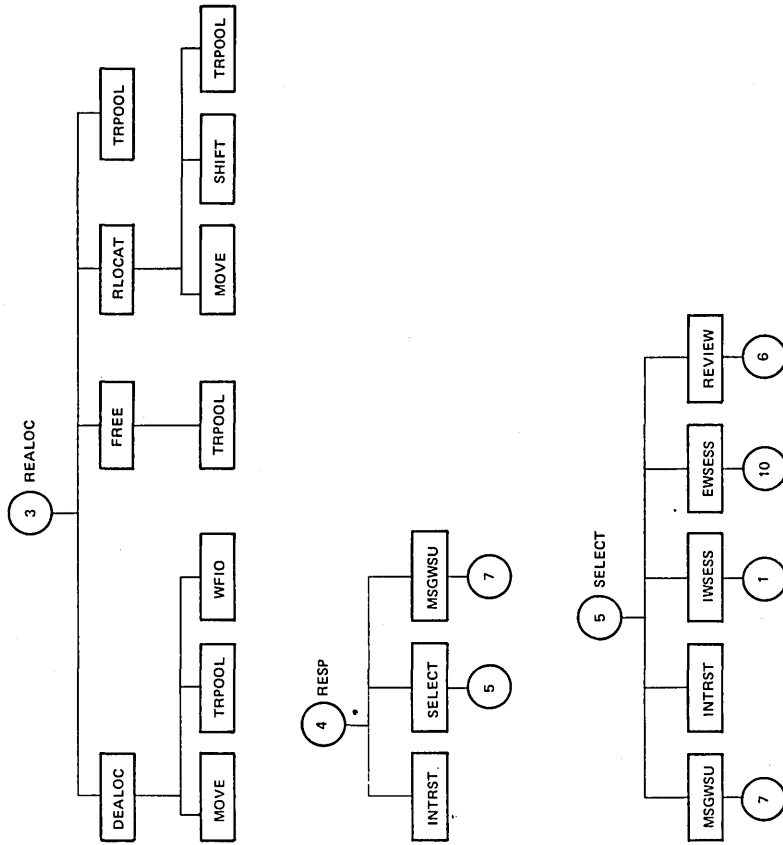


Figure 1-5 (Part 3 of 7). WSU Execution Hierarchy

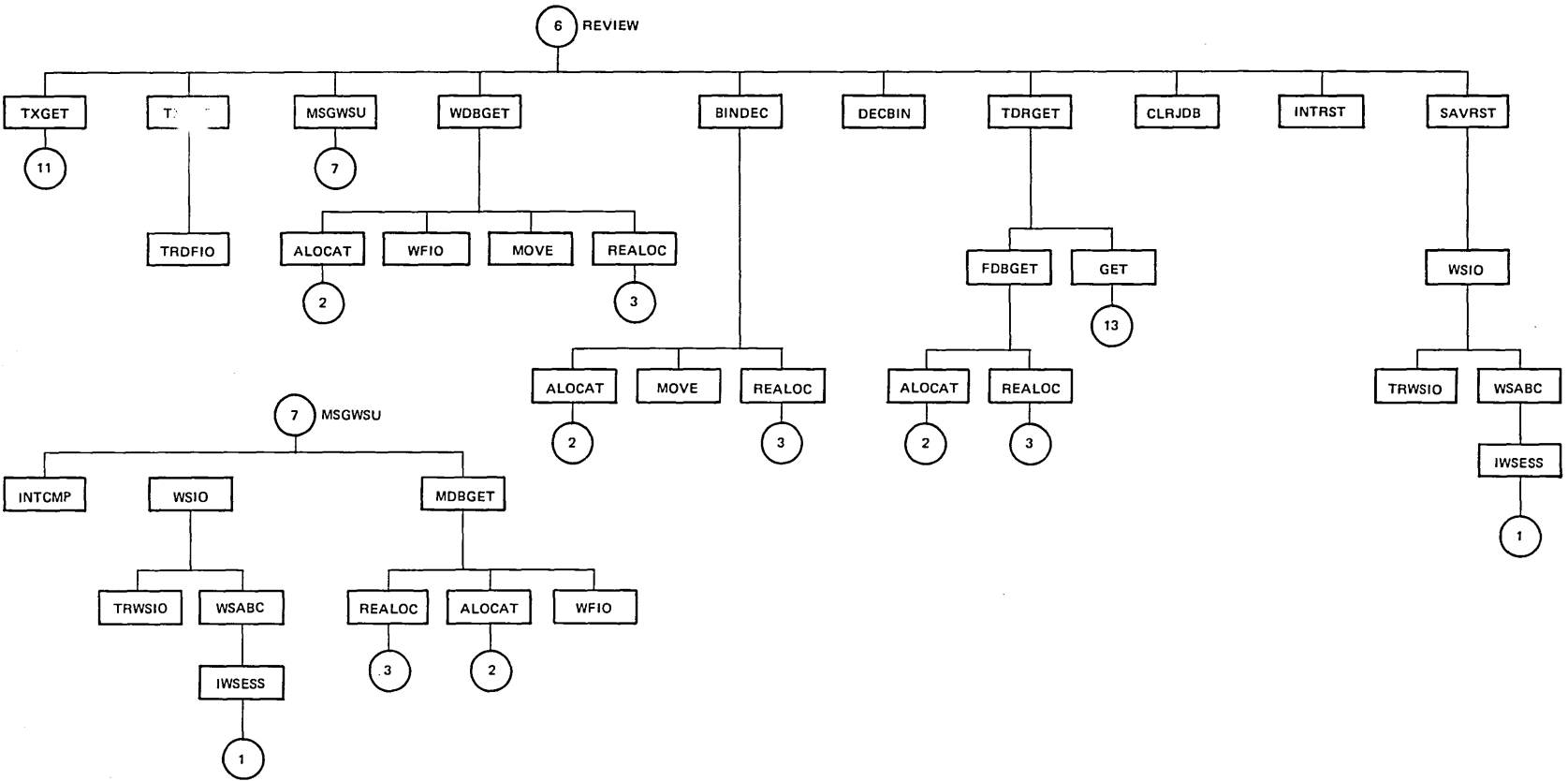


Figure 1-5 (Part 4 of 7). WSU Execution Hierarchy

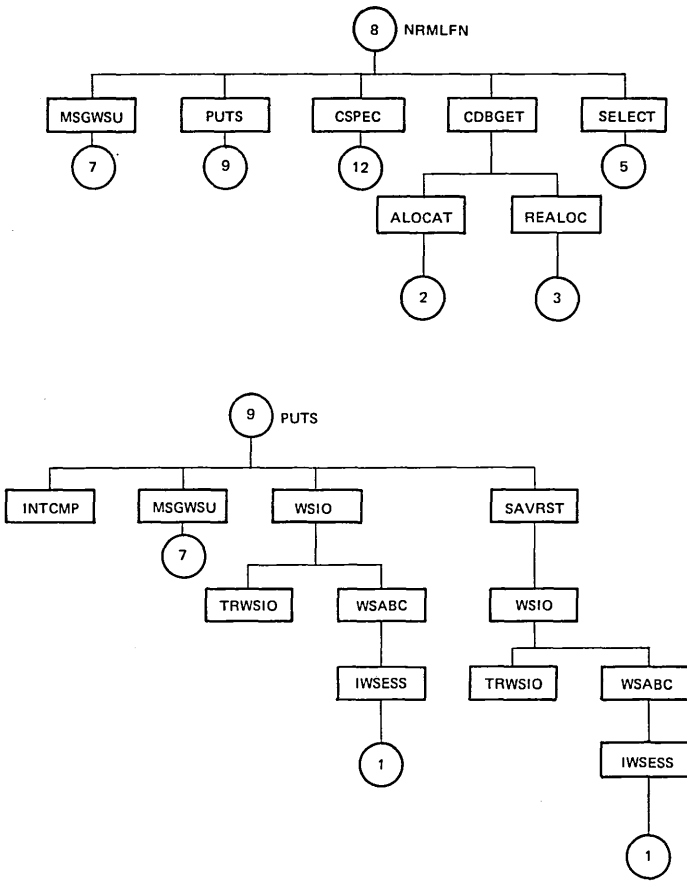


Figure 1-5 (Part 5 of 7). WSU Execution Hierarchy

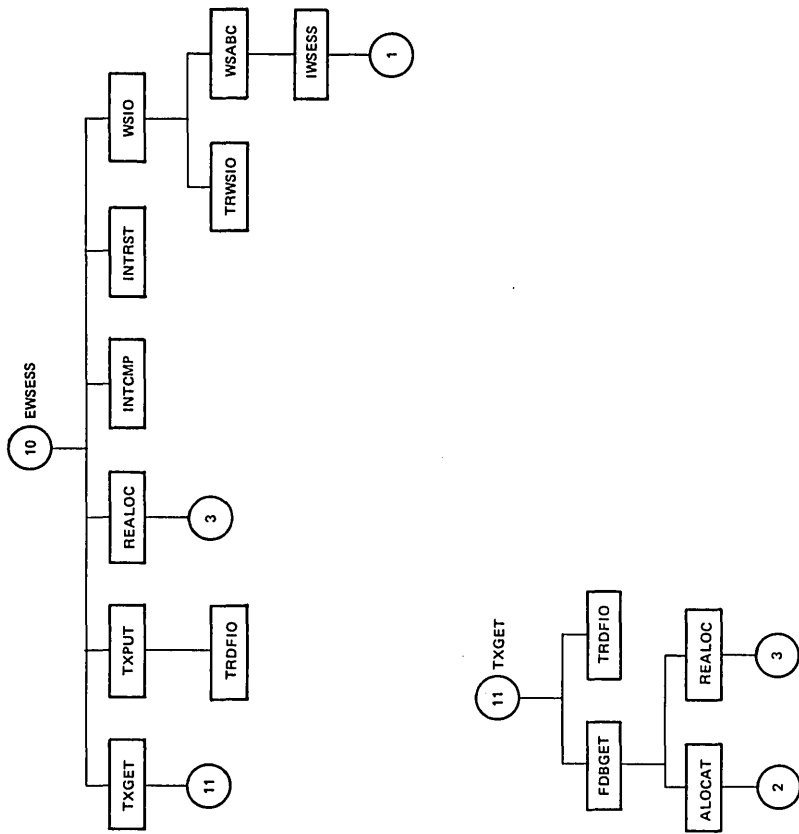


Figure 1-5 (Part 6 of 7). WSU Execution Hierarchy

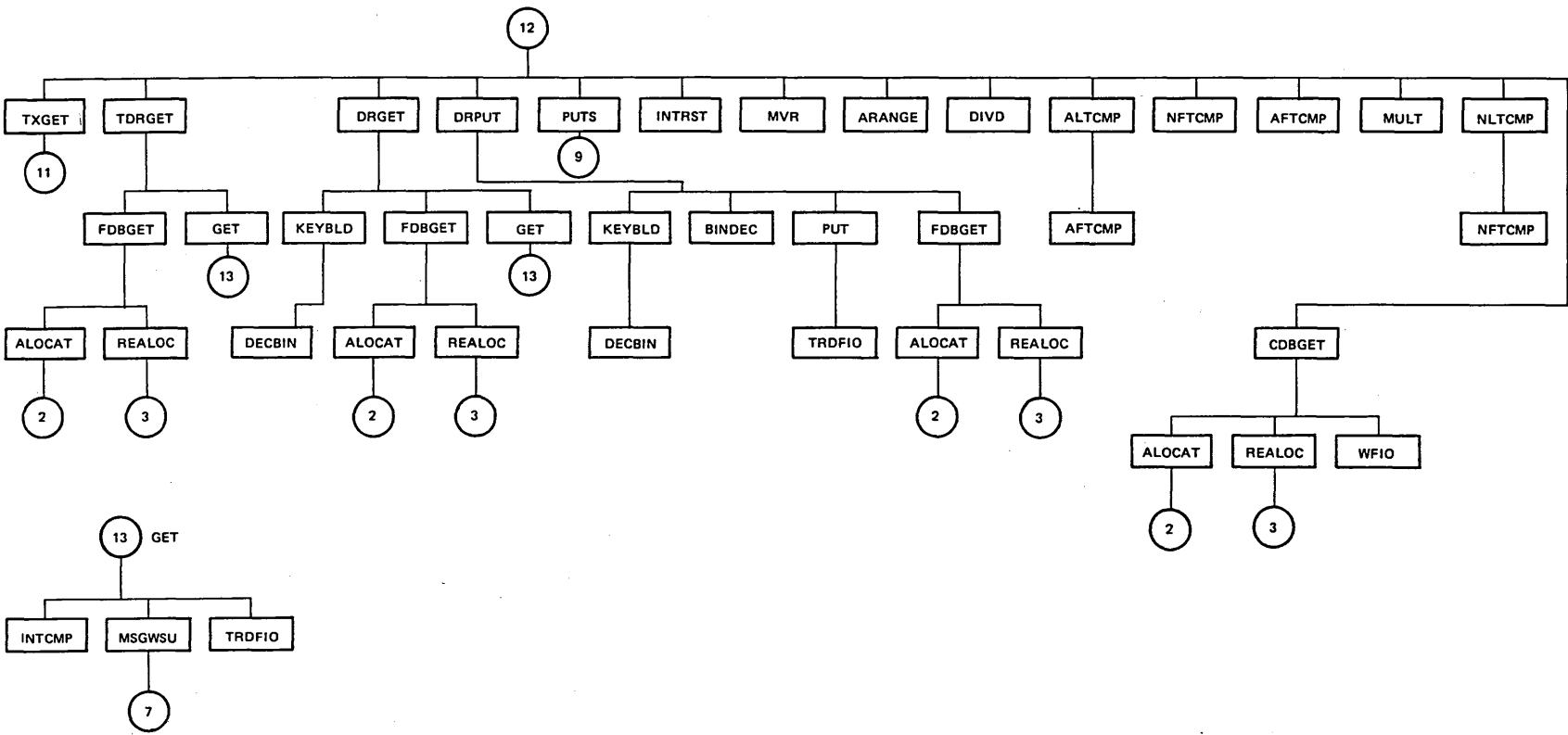


Figure 1-5 (Part 7 of 7). WSU Execution Hierarchy

EXECUTION MODULES AND ROUTINES

This section explains the functions of each of the WSU execution modules. The modules are described in phase order (#WSX11, #WSX12, #WSXP, and #WSX00 through #WSX19) and by routine (alphabetically) within the phase.

Routines and services called are listed to the left. Parentheses indicate that return from the called routine or service is expected unless an abnormal situation is encountered.

#WSX11 (WSU Initialization Phase, Part 1)

Routines and Services Called

Description

(\$FIND)	Finds WSU message member.
(\$INFO)	Updates job control box (JCB) with format-1 address and SS address.
(\$LOG)	If WSU message member not found, aborts job.
(MSG)	Retrieves OCL translation template message.
(SYSIN)	Using hardwired translation table, reads and interprets WSX control statements. Saves trace parameter.
(\$XFER)	If OBJLIB— parameter specified, finds user object program library (RFINDLIB).
(\$FIND)	Finds and saves object program sector address.
(\$XFER)	If MSGLIB— parameter specified, finds user message library (RFINDLIB).
(\$FIND)	Finds user message member.
(\$INFO)	If not in library with WSU message member, updates JCB with format-1 address and SS address.
(GETP)	Extends partition.
(\$INFO)	Retrieves and saves NEP attribute.
(\$INFO)	Gets MRTMAX value.
(FDIO)	Reads control header from object program and calculates region size requirements.
(FDIO)	Starting at region end, reads preinitialized SSA, FSA, DSA, CSA, PSA, and MSA specification blocks from object program into high storage and saves area addresses. Clears MSA space allowed for WSU messages, and saves address of first WSU MSB for #WSXP. Assigns JDB address. Saves release level from control header. Blanks JDB, then processes DSA to initialize arithmetic JDB fields to character zeros and change DSA DB offsets to addresses. Saves World Trade editing attribute from control header.
(\$INFO)	Retrieves system date and saves converted values in reserved date name JDB fields. Changes SSB (screen output/input fields), PSB (display screen and C-spec group) and FSB (file record fields) SB offsets to addresses. Calculates WCA size, saves WCA address, and clears WCA to binary zeros. Calculates normal and alternate WDB sizes. Assigns master index area address, if required, and initializes MTI and key area pointers in each FSB DTF. Assigns format index area address, and initializes. FIA pointer in work station STF.
(\$XFER)	If FMTLIB—parameter specified, finds user format library format 1 address (RFINDLIB).
(\$ALOC)	Builds work station DTF and attempts allocate, open, and close (except for last) for every format,
(\$OPEN)	and for menu format.
(\$CLOS)	
	Moves open DTF to high storage and saves address. Assigns screen data block address, and saves address and maximum input data length in work station DTF.
(\$FIND)	Finds second job initialization module, #WSX12.
(\$LOAD)	Loads #WSX12 at region start.
#WSX12	Shared work areas are left in higher storage, and the work file and data files are prepared.

#WSX12 (WSU Initialization Phase, Part 2)

Routines and Services Called	Description
	Initializes each FSB in the following manner:
(\$ALOC)	Allocates DTF, using local IOB(s).
(AFARW)	Performs special processing to prevent loss of data in new transaction file should WSU abnormally end.
(DALOC)	Gets transaction file format 1.
(SALOC)	If retain code not S or J and file new, deallocates DTF.
(\$OPEN)	Performs special allocate to force disposition old.
	Opens DTF and destroys backward chain.
	Copies local index and/or data IOB(s) to FSA, and adjusts DTF pointer(s) accordingly.
	Initializes DTF pointer to index data area.
	Changes key save area WCA offset and chain field DSA offsets to addresses.
	After all files handled, changes header record ID FSA offset to address and saves for #WSXP.
	Saves review supported attribute from control header.
(SALOC)	Calculates size requirement for, then allocates and opens, work file.
(\$OPEN)	
(SVC)	Creates message data area (MDA) in work file by:
(FDIO)	— Reading and reformatting MDA from object program.
(MSG)	— Retrieving texts from user message member for all MICs encountered in MSA.
(MSG)	— Retrieving texts from WSU message member for all WSU messages.
	Adjusts MSA MDA offsets and sector addresses accordingly.
(SVC)	Creates command data area (CDA) in work file by:
(FDIO)	— Copying CDA from object program,
	— Changing specification area offsets to addresses, and adjusting CSA CDB sector addresses accordingly.
	Computes and saves sector addresses for normal and alternate work station data areas.
	Computes and saves sector addresses for screen save areas.
	Determines and saves key field lengths for key save area format.
	Repositions initialization parameter block (IPB) at pool end, just ahead of permanent elements.
(\$FIND)	Finds resident processing phase, #WSXP.
(\$LOAD)	Loads #WSXP at region start, places pool end address in XR1.
#WSXP	Exits to set up low storage resident data/code and pool.

#WSXP (WSU Execution Processing Phase)

Routines and Services Called	Description
	Copies initialization parameter block to COMIPB.
	Clears pool (except for start code) and initializes pool element storage control blocks (fixed non-relocatable first, maximum relocatable FREE1, fixed relocatable last) and associated control fields in GENDAT.
(\$FIND)	Resolves TSA, and changes transient load module names to \$LOAD parameters.
ABORT	If any transient module cannot be found, aborts job (03).
	If requested (TRACE— parameter), activates special call/exit trace.
	If review not supported, lowers priority of review handler transient.
INTBEG	Begins first WSU cycle (by branching into interaction begin processor to accept sign-on for job-initiating work station).
	Cycles through low storage processors and pool resident transient routines, serially processes interactions (for up to a maximum of MRTMAX attached work stations), and accepts requests from work station data manager queue until sign-off completed for last work station (if non-NEP) or WSU job ended, aborted, or canceled by operator.

Resident Processor Descriptions

Routine	External Call	Description
(#WSXP) CALL00		<i>Call to Subroutine Processor.</i> Secures called routine RSB address (from ARR). Steps forward to next QSA level. Saves work area and registers in QSA.
CALLXX	(TRNGET) (LOGTR) *	If called routine not resident, loads transient module that contains the routine. If trace is active, logs subroutine call trace entry. Places address of common in XR1 and loads IAR to enter called routine.
(#WSXP) EXIT00	(TRNGET) (LOGTR) *	<i>Exit from Subroutine Processor.</i> Secures calling routine RSB address (from ARR). If calling routine not resident, loads transient module that contains the routine. Restores work area and registers from QSA. Steps back to previous QSA level. If trace is active, logs subroutine exit trace entry. Places address of common in XR1, reconstructs exit IAR, and loads IAR to return to calling routine.
(#WSXP) TRNGET	(ALOCAT) (\$LOAD) (REALOC)	<i>Transient Load Module Get Processor Subroutine.</i> Locates TSB for transient module that contains routine specified via RSB address. Allocates pool element of requisite size and priority. Constructs load parameter block from TSB fields. Loads transient module in new pool element. Updates each RSB for transient module to show resident status and entry point address. (Address constants are expected in RSB order at module end.) Reallocates unused space at module end to free pool element.
(#WSXP) INTCMP INTRST INTBEG INTEND	 CALLXX	<i>Interaction Complete/Restart/Begin/End Processor.</i> Signals interaction completing. Signals interaction restarting. Sets XR2 to point to RSB address for the driver routine. (For entry at INTEND, called routine RSB address must be in XR2 and should be that of AWSESS.) Steps back to QSB length before QSA start. Calls to work session driver (or AWSESS).
(#WSXP) ABORT	(LOGTR) (\$LOG) (\$LOG) INTEND	<i>Abort Session/Job Processor.</i> Gets abort ID (from ARR). Formats and logs abort trace entry. If abort already in progress or if no current WCB exists, or if IJ or EJ time logs WSU-0802 to console, with a 3 only option. Otherwise, signals abort in progress. Identifies aborted work station in log record and logs WSU-0801 to console with option 0 or 3. Selects AWSESS RSB address for called routine and exits to End Interaction Cycle.

Routine	External Call	Description
(#WSXP) WFIO		<i>Workfile I/O Processor.</i> (Expects number of sectors and sector start in WRKIOB.) Completes initialization of work file IOB.
	(FDIO)	Performs disk I/O, and transfers specified number of sectors between work file and storage pool element.
	ABORT	If work file I/O completed abnormally, aborts session, or else returns to caller.
(#WSXP) SHIFT		<i>Shift Data Block Processor.</i> Shifts block of specified length from old starting address to new starting address (in segments up to 256 bytes, beginning with leftmost byte). Returns to caller.
(#WSXP) MOVE		<i>Move Data Block Processor.</i> Moves block of specified length from old ending address to new ending address (in segments up to 256 bytes, beginning with rightmost byte). Returns to caller.

DESCRIPTIONS OF ROUTINES ACCESSED VIA CALL AND EXIT

Routines Resident in #WSXP

Routine	External Call	Description
ALOC00		<i>Pool Element Allocation Routine.</i>
	(TRPOOL)	If active, logs storage request trace entry.
	ABORT	If SRB size exceeds total free and deallocatable relocatable storage, aborts session (31).
	ABORT	If invalid SRB attribute, priority, or size requested, or relocatable storage request and relocation required (still pending for last request), aborts session (01). If relocatable storage requested, adjusts maximum available relocatable free space. Signals relocation required, and steps past nonrelocatable pool elements. If nonrelocatable storage requested, steps past higher priority nonrelocatable pool elements.
	(DEALLOC)	Deallocates all subsequent deallocatable pool elements of lower priority than element following or element to be allocated until large enough free element available.
	(FREE)	Allocates element, moving pool attributes/priority to SCB. Frees unused space at element end.
	(RLOCAT)	If relocation required, relocates element. Places address of assigned storage in specified pointer. Clears SRB. Returns to caller.

Routine	External Call	Description
REAL00	(TRPOOL) (FREE) (RLOCAT) (DEALLOC) ABORT	<p><i>Pool Element Reallocation Routine.</i></p> <p>If active, logs storage request trace entry.</p> <p>As requested via SRB:</p> <ul style="list-style-type: none"> – Sets nondeallocatable attribute in SCb. – Sets deallocatable attribute in SCB. – Frees unused space at end of element and adjusts maximum available relocatable free space. – Replaces data pointer, copies address from old pointer to new pointer and removes old pointer. – Relocates pool element. – Deallocates pool element. – Clears SRB. – If invalid request encountered, aborts session (02). <p>Returns to caller.</p>
FREE00	(TRPOOL)	<p><i>Free Pool Element Creation/Expansion Routine.</i></p> <p>If next element is a free element, combines area to be freed with this element, or else creates a new free element in area to be freed.</p> <p>Adjusts data pointer in new/expanded free element SCB.</p> <p>Adjusts size in shortened element SCB.</p> <p>If active, logs storage request trace entry.</p> <p>Returns to caller.</p>
RLOC00	(MOVE) (SHIFT) (SHIFT) (MOVE) TRPOOL	<p><i>Pool Element Relocation Routine.</i></p> <p>Determines END limits of relocation.</p> <p>If any other allocated elements to be repositioned, then if no relocatable free element, or if relocatable free element smaller than element incrementally (in blocks up to 520 bytes, from rightmost) moves any portion of element to be overlaid to work area, shifts other affected elements toward vacated space, and moves saved portion after repositioned elements.</p> <p>If relocatable free element not smaller than element, shifts any other affected elements toward free element.</p> <p>Moves any unrelocated portion of element to new position.</p> <p>Updates every affected SCB.</p> <p>Signals relocation not required.</p> <p>If active, logs storage request trace entry.</p> <p>Returns to caller.</p>
DEAL00	(WFIO) (MOVE) TRPOOL	<p><i>Pool Element Deallocation Routine</i></p> <p>If special deallocation for normal or alternate work station data block, formats work file and saves WDB in work file.</p> <p>If special deallocation for transient module, updates each RSB for transient to show non resident status.</p> <p>Clears data pointer.</p> <p>Moves any preceding relocatable elements to deallocated space.</p> <p>Creates new free element or expands any preceding and/or following free elements.</p> <p>If active, logs storage request trace entry.</p> <p>Returns to caller.</p>
TRPOOL	(LOGTR)	<p><i>Pool Storage Request Trace Writing Subroutine.</i></p> <p>Formats and logs trace entry for pool storage request.</p> <p>Returns to caller.</p>

#WSX00 Routines

Routine	External Call	Description
TRDR00	(\$TOD) (LOGTR)	<i>Driver Trace Writing Routine.</i> Gets system time. Formats and logs time-stamped trace entry for driver-interaction cycle. Returns to caller.
TRWS00	(LOGTR)	<i>Work Station I/O Trace Writing Routine.</i> Formats and logs trace entry for work station access. Returns to caller.
TRDF00	(LOGTR) (LOGTR) (LOGTR)	<i>Data File I/O Trace Writing Routine.</i> Formats and logs trace entry for data file access. If transaction file, formats and logs trailer trace entry. If master file, formats and logs key trace entry. Returns to caller.

#WSX01 Routines

Routine	External Call	Description
GET000	(MSGWSU) INTCMP (\$GETD) GET905 ABORT ABORT GET905 GET905	<i>Data File Record Read Routine.</i> (Expects key in key area and file buffer to be allocated and current.) If C-spec get for update capable file, saves key in WCB key entry, flags entry for no PUT pending, and flags entry for GET before PUT not required. If update capable file flags corresponding key entry in every other active WCB flagged for PUT pending for GET before PUT required, and if GET contention found (PUT pending for same key in some other active WCB), suspends session: <ul style="list-style-type: none">– Flags WCB suspended.– Saves offset of contending WCB key entry.– Issues WSU-0709.– Exits to complete interaction.– Signals record found. If C-spec GET: <ul style="list-style-type: none">– Sets off all record identifying indicators for file.– Sets off not-found indicator specified for operation or defaulted for file.– Adjusts buffer pointers in DTF and IOB.– Gets record into FDB.– Saves logical record displacement.– If no record found, signals record not found.– If key rejected, exits to Aborts Session (06).– If any other I/O error, exits to Aborts Session (04).– If transaction file, copies trailer from FDB to common work area then if not data or header record for current work station, signals record not found and goes to handle not-found condition.– Determines and saves record type RID address.– If record type not identified, signals record found but clears record type RID address and goes to handle not-found condition.– Sets on record identifying indicator for record type.

Routine	External Call	Description
GET000 (continued)		If update capable file and C-spec get: <ul style="list-style-type: none"> – Saves record type RID address in WCB key entry, flags entry for PUT pending, and signals GET update performed. – Moves fields from FDB to JDB, and expands packed fields.
GET490	(TRDFIO)	If active, logs data file trace entry. Returns to caller.
GET905		Handles not-found condition:
	GET490	If not C-spec get, returns.
	ABORT	If no not-found indicator specified, Aborts Session (05). Sets on not-found indicator.
	GET490	Returns to caller.

#WSX02 Routines

Routine	External Call	Description
PUT000		<i>Data File Record Write Routine.</i> (Expects key in key area and file buffer to be allocated and current.) Updates buffer addresses in DTF and IOB. If no RID in C-spec (record type not specified), defaults RID saved in WCB key entry for file. If transaction file and RID in C-spec: <ul style="list-style-type: none"> – Signals GET before PUT required. – If add pending and relative record number overflow, Aborts Session (30).
	ABORT	If master file or no RID in C-spec and WCB key entry not marked for PUT pending, Aborts Session (08).
	ABORT	Flags WCB key entry for no PUT pending and GET before PUT not required. If WCB key entry was marked for GET before PUT required, signals GET before PUT required.
	ABORT	If key differs from PUT pending key in WCB key entry, Aborts Session (33). If GET before PUT required or if transaction file and RID in C-spec, flags corresponding key entry in every other active WCB flagged for PUT pending for GET before PUT required.
	(\$GETD)	If first time GET, or GET before PUT required, gets record into FDB.
	ABORT	If transaction file and past extent, Aborts Session (29).
	ABORT	If any other I/O error, Aborts Session (07). Saves logical record displacement. If transaction file and add pending, signals no add pending, and copies trailer from common work area to FDB.
	(\$PUTD)	Moves fields from JDB to FDB, and compresses packed fields. Puts record into data file.
	ABORT	If update error (key not last get), Aborts Session (33).
	ABORT	If any other I/O error, Aborts Session (09).
	(TRDFIO)	If active, logs data file I/O trace entry. Returns to caller.

#WSX03 Routines

Routine	External Call	Description
PUTS00	(SAVRST)	<p><i>Screen Display Puts Routine.</i> (Expects address of selected PSB in WRK0X-WRKX1.) Performs save/restore for enter or review screen if required. If user or menu response pending, selects any timely WSU message:</p> <ul style="list-style-type: none"> – If stop required, selects WSU-0710. <p>If mode change message pending, signals mode message not pending then selects WSU-0713 (enter mode active), WSU-0708 (review mode active), or WSU-0724 (insert mode active). If menu response pending:</p> <ul style="list-style-type: none"> – Defaults relative record number to review from *RLRN. – Signals session suspended for menu. – Signals menu display up. – Formats work station DTF for PUT, and inserts WSU menu name. – Goes to display menu.
	PUTS310	<p>If MSG or IMSG response pending:</p> <ul style="list-style-type: none"> – Requests invite input. – Signals MSG or IMSG operation.
	MSGWSU	<p>– If IMSG response pending, requests erase format and exits to display message. Prepares domestic or World Trade editing mask.</p>
	MSGWSU	<p>If no display defined for selected PSB, requests erase format and exits to display any message(s).</p> <ul style="list-style-type: none"> – If display has no input fields but has keyboard reset feature, requests erase format. – Extracts output fields to SDB and edits fields as required. – Inserts user format name into work station DTF. – If PUT override allowed and if selected PSB current, formats work station DTF for PUT override.
	PUTS310	<ul style="list-style-type: none"> – Displays user format.
PUTS310		<p>Formats work station DTF for PUT then invite. If no user or menu response pending, or message selected, formats work station DTF for PUT, no invite.</p>
	(WSIO)	<p>Puts display on screen. If no user or menu response pending, returns to caller. If menu response not pending, signals PUT override allowed.</p>
	INTCMP MSGWSU	<p>If no message selected, exits to complete interaction cycle. Exits to display message(s).</p>
MSGWSU	PUTS410	<p><i>Message Put Routine.</i> If WSU message input requested and menu is displayed, signals menu response pending and proceeds to issue message. If erase requested:</p> <ul style="list-style-type: none"> – Signals PUT override not OK. – Formats work station DTF and SDB for erase format table. – Erases format table. – Reformats work station DTF.
	(WSIO)	

Routine	External Call	Description
PUTS410		Formats work station DTF for write error.
		If message selected:
		– Formats work station DTF for PUT.
		– Converts selected message to MSB address.
	(MDBGET)	– If selected MSB not current, gets corresponding MDB.
		– Formats SDB from MDB message text.
		– If no MSG or IMSG response pending and if invite input requested, formats work station DTF for PUT then invite.
	(WSIO)	– Puts message on screen.
		– If no MSG or IMSG response pending:
		– If invite input not requested, returns to caller.
INTCMP	– If invite input requested, exits to complete interaction cycle.	
	If MSG or IMSG operation:	
	– Formats work station DTF for PUT then invite.	
(MDBGET)	– If requested MSB not current, gets corresponding MDB.	
	– Formats SDB from MDB message text.	
(WSIO)	– Puts message on screen.	
	– Signals no MSG or IMSG operation.	
INTCMP	– Exits to complete interaction cycle.	
	If no message selected and no MSG or IMSG response pending:	
	– Requests reset then invite.	
(WSIO)	– Invites input from work station.	
	– Signals no MSG or IMSG operation.	
INTCMP	– Exits to complete interaction cycle.	
SAVR00		<i>Save/Restore Screen Routine.</i>
		If review mode, and if enter mode screen not saved, signals enter mode screen saved and sets to save screen.
		If menu request in enter mode, and if enter mode screen not saved, signals enter mode screen saved and sets to save screen.
		If menu request in review mode and if review mode screen not saved, signals review mode screen saved and sets to save screen.
		If enter mode screen saved and if enter mode:
		– Signals enter mode screen not saved.
		– Signals menu display not up.
		– Sets to restore screen.
		If review mode screen saved and if review mode and menu response not pending:
		– Signals review mode screen not saved.
	– Signals menu display not up, and sets to restore screen.	
	If no save/restore required, returns to caller.	
(WSIO)	Formats work station DTF for save/restore and exits to Save/Restore Screen.	
WSIO00		<i>Work Station I/O Routine.</i>
	(\$WSIO)	Relays preformatted work station DTF request to work station data management (WSDM).
	(TRWSIO)	If active, logs work station I/O trace entry.
	WSABC	If WSDM completion abnormal, exits to process work station I/O completion code. (Control may return.) Returns to caller.

Routine	External Call	Description
MDBG00		<i>Message Data Block Get Routine.</i> (Expects address of selected MSB in WRK10-WRK11.)
	(REALOC)	Reallocates any current MDB as deallocatable, and transfers pointer from common to MSB. Makes selected MSB current.
	(ALOCAT) (REALOC)	If MDB for selected MSB not allocated, allocates relocatable, non-deallocatable pool element of requisite size and priority. Otherwise, reallocates MDB as non-deallocatable, and transfers pointer from MSB to common.
	(WFIO) (REALOC)	If new MDB element allocated, formats work file IOB, fetches MDB from work file, shifts message text to MDB start, and reallocates pool element to free unused space. Returns to caller.

#WSX04 Routines

Routine	External Call	Description
ARAN00		<i>Alphameric Range Routine.</i> Processes RANGE C-spec operation for alphameric factors. Returns to caller.

#WSX05 Routines

Routine	External Call	Description
CSP040		<i>C-Specification(C-Spec) Processing Routine.</i> Adjusts IAR past current C-spec.
CSP000	(SP070) (CDBGGET) CSP000	If more C-specs in group, goes to process next C-spec. If not last CSB of group, gets CDB for next CSB and goes to process first C-spec.
	(CDBGGET)	If CSB in subroutine group, restores CSB and C-spec IAR; gets CDB for resumed CSB, goes to process next instruction. Returns to caller.
CSP070	CSP040 (LOGTR) ABORT CSP	Saves length of current C-spec. Branches to bypass C-spec as dictated by conditioning indicator status. If active, formats and logs C-spec trace entry. Performs operation setup, by building C-spec description in COMQSB. If operation unrecognized, Aborts Session (10). Selects and routes control to subprocessor for specified operation.
CSP0200		<i>GETNH Subprocessor.</i> Adjusts next record to <i>last GET</i> next record. Selects not-found indicator from C-spec.
	(TXGET)	Reads next records until header record or chain end found. Selects located header record.
	(TDRGET) CSP040	Gets selected transaction file data record. Returns for next C-spec.

Routine	External Call	Description
CSP0400	(TDRGET) CSP040	<i>GETNR Subprocessor.</i> Selects not-found indicator from C-spec. Selects <i>last GET</i> next record. Gets selected transaction file data record. Returns for next C-spec.
CSP0600	(TDRGET) CSP040	<i>GETPR Subprocessor.</i> Selects not-found indicator from C-spec. Selects <i>last GET</i> previous record. Gets selected transaction file data record. Returns for next C-spec.
CSP0800	(TDRGET) CSP040	<i>GETPH Subprocessor.</i> Selects not-found indicator from C-spec. Selects <i>last GET</i> previous header record. Gets selected transaction file data record. Returns for next C-spec.
CSP0A00	(DRGET) CSP040	<i>GET Subprocessor.</i> Selects FSB from C-spec. Selects not found indicator from C-spec. Gets master file data record. (Session may be suspended for GET contention.) Returns for next C-spec.
CSP0C00	(DRPUT) CSP040	<i>PUT Subprocessor.</i> Selects FSB from C-spec. Selects RID from C-spec. Puts master/transaction file data record. Returns for next C-spec.
CSP0E00	(PUTS) CSP040 INTRST PUTS	<i>PUTS Subprocessor.</i> If reset keyboard not requested, displays user format and returns for next C-spec. If PSB auto selected or not current, selects specified PSB as next to process and exits to restart interaction. Signals preprocessing response pending. Signals override not allowed. Exits to display user format.
CSP1000	PUTS	<i>MSG Subprocessor.</i> Signals MSG response pending. Sets on RP indicator. Sets on any resulting indicators. Selects MSB from C-spec. Exits to display user format with message line.
CSP1200	PUTS	<i>IMSG Subprocessor.</i> Signals IMSG response pending. Sets on any resulting indicators. Selects MSB from C-spec. Exits to display message line.

Routine	External Call	Description
CSP1400	(CDBGET) CSP000	<i>EXSR Subprocessor.</i> Saves IAR. Sets up IAR for subroutine CSB. Get CDB for subroutine CSB to process first subroutine C-spec.
CSP1600	CSP040	<i>SETON Subprocessor.</i> Sets on specified indicators. Returns for next C-spec.
CSP1800	CSP040	<i>SETOF Subprocessor.</i> Sets off specified indicators. Returns for next C-spec.
CSP1A00	(CDBGET) CSP000	<i>GOTO Subprocessor.</i> Sets up IAR for target C-spec. If not current, gets CDB for target CSB. Goes to process target C-spec.
CSP4000	CSP940	<i>COMP (Alpha) Subprocessor.</i> Processes alpha compare C-spec operation. Goes to set resulting indicators.
CSP4200	(ARANGE) CSP940	<i>RANGE (Alpha) Subprocessor.</i> Processes alpha RANGE C-spec operation. Goes to set resulting indicators.
CSP4400	CSP040	<i>Move Subprocessor.</i> Processes MOVE C-spec operation, and updates JDB field. Returns for next C-spec.
CSP4600	CSP040	<i>MOVEL Subprocessor.</i> Processes move left C-spec operation, and updates JDB field. Returns for next C-spec.
CSP4800	CSP040	<i>MOVE *BLANK Subprocessor.</i> Processes MOVE *BLANK C-spec operation, and updates JDB field. Returns for next C-spec.
CSP4A00	(ALTCMP) CSP940	<i>COMP (alpha, literal table) Subprocessor.</i> Processes alpha literal table compare C-spec operation. Goes to set resulting indicators.
CSP4C00	(AFTCMP) CSP940	<i>COMP (alpha, field table) Subprocessor.</i> Processes alpha field table compare C-spec operation. Goes to set resulting indicators.
CSP6A00	(NLTCMP) CSP940	<i>COMP (numeric, literal table) Subprocessor.</i> Processes numeric literal table compare C-spec operation. Goes to set resulting indicators.

Routine	External Call	Description
CSP6C00	(NFTCMP) CSP940	<i>COMP (numeric, literal table) Subprocessor.</i> Processes numeric field table compare C-sped operation. Goes to set resulting indicators.
CSP6E00	(MULT) CSP940	<i>MULT Subprocessor.</i> Processes multiply C-spec operation. Goes to set resulting indicators.
CSP7000	(DIVD) CSP940	<i>DIV Subprocessor.</i> Processes divide C-spec operation. Goes to set resulting indicators.
CSP7200	(MVR) CSP940	<i>MVR Subprocessors.</i> Processes move remainder C-spec operation. Goes to set resulting indicators.
CSP7400	CSP900	<i>Z-ADD Subprocessor.</i> Processes zero and add C-spec operation. Goes to complete decimal operation.
CSP7600	CSP900	<i>Z-SUB Subprocessor.</i> Processes zero and subtract C-spec operation. Goes to complete decimal operation.
CSP7800	CSP900	<i>ADD Subprocessor.</i> Processes add C-spec operation. Goes to complete decimal operation.
CSP7A00	CSP900	<i>SUB Subprocessor.</i> Processes subtract C-spec operation. Goes to complete decimal operation.
CSP7C00	CSP940	<i>RANGE (Numeric) Subprocessor.</i> Processes numeric range C-spec operation. Goes to set resulting indicators.
CSP7E00	CSP940	<i>COMP (Numeric) Subprocessor.</i> Processes numeric compare C-spec operation. Goes to set resulting indicators.
CSP900	CSP940	Half adjusts result if necessary. Moves result to JDB. Goes to set resulting indicators.
CSP940	CSP040	Sets resulting indicators based on condition register. Returns for next C-spec.

#WSX06 Routines

Routine	External Call	Description
IWS000		<i>Initiate Work Session Routine.</i>
		If IJ processing segment not previously selected, performs job initialization (startup):
	(TXGET)	– Creates or updates job control record in transaction file, and tags job control record (JCR) for job ABEND.
	(TXPUT)	– Processes work station control records, determines last reserved record, and searches for aborted sessions.
	(TXGET)	– If aborted session discovered, reads to end of data record chain for session, updates corresponding work station control record, validates chain fields and tags work station control record (WCR) as recovered.
	(TXPUT)	
		If no current WCB exists, activates session:
		– Releases session if EJ time.
		– Assigns work station ID to first available inactive WCB.
	ABORT	– If no inactive WCB available, aborts job (16).
		– Makes selected WCB current and flags that WCB as active.
		– Clears work station level pointers in common and sets off work station level indicators.
	(CLRJDB)	– Zeros or blanks work station level JDB fields.
	(\$INFO)	– Retrieves UPSI switches to set indicators U1-U8.
	(TXGET)	– Performs work station initialization (sign-on): reprocesses work station control record chain.
		– If old work station control record with matching work station ID found, sets on RS indicator.
		– If previous session recovered, sets on RC indicator.
	ABORT	If previous session aborted but not yet recovered, aborts session (17).
		– If no matching work station control record found, allocates work station control record.
	(TXPUT)	– Work station control record for session ABEND.
		– Saves work station control record number in WCB.
	(TXGET)	– Initializes transaction record pointers in COMMON.
		If IJ processing segment not previously selected,
		– Selects IJ segment.
		– Signals IJ time.
		– Sets on IJ indicator.
		– Flags WCB job-initiating.
	(BINDEC)	– Initializes *RLNO to last reserved record.
	INTRST	– Exits to restart interaction cycle.
		If not IW time,
		– Signals IW time.
		– Sets on IW indicator.
		– Selects IW processing segment.
	(BINDEC)	– Initializes *RLNO to add next record.
	INTRST	– If not IJ time, exits to restart interaction cycle.
	MSGWSU	– If not job-initiating WCB, signals WSU response pending and exits to erase format table and issue WSU-0701.
		– Signals not IJ time.
		– Sets off IJ indicator.
	INTRST	– Exits to restart interaction.
		If IW time, initiates first user sequence processing segment:
		– Signals not IW time.
		– Sets off IW indicator.
		– Selects first PSB in primary sequence.
	INTRST	– Exits to Restart Interaction.

#WSX07 Routines

Routine	External Call	Description
SELECT		<i>Select Next Enter Mode Processing Segment Routine.</i>
		If select display by ID and not command key 13, find PSB with selected ID.
	MSGWSU	If there is no PSB with that ID, or the PSB is not enter selectable, exits to issue WSU-0707.
		Saves PSB address.
	MSGWSU	If IJ, IW, ES, EW, or EJ indicator on or IJ, IW, ES, EW, or EJ time, or EJ forced, exits to issue WSU-0720.
		If identified PSB current, reinitiates identified (current) processing segment: <ul style="list-style-type: none"> – Selects identified PSB. – Signals no user response pending. – Exits to restart interaction.
	INTRST MSGWSU	If not select next display and not command key 13, and required input pending, signals WSU response pending and exits to issue WSU-0703.
		If not select display by ID, initiates <i>next</i> enter mode processing segment: <ul style="list-style-type: none"> – If EJ indicator on, signals EOJ forced. – If EOJ forced, sets on EJ indicator. – If not EW time, sets on EW indicator. – If EW time or EJ time, exits to terminate work session. – If ES, EW, EJ indicator on: If review mode, requests resume entry (command key 3) and invokes review mode handler. – Signals not IJ time or IW time. – Sets off IJ and IW indicators.
	EWSESS (REVIEW)	– If ES time and EW or EJ indicator on, exits to end session.
		– Signals ES time.
		– Sets on ES indicator.
	INTRST	– Selects ES PSB and exits to restart interaction.
	IWSESS	– If IJ time or IW time, exits to initiate work session.
	INTRST	– If ES time, signals not ES time, sets off ES indicator, selects first PSB in primary sequence and exits to restart interaction.
	INTRST	– If current PSB looping and bypass display not keyed, selects current PSB and exits to restart interaction.
	INTRST	– If last PSB in primary sequence, signals ES time, sets on ES indicator, selects ES PSB and exits to restart interaction.
	INTRST	– If PSB not sequenced, selects previous sequenced PSB and exits to restart interaction.
	INTRST	– If PSB sequenced and not sequence end, selects next logical PSB and exits to restart interaction.
		– If PSB sequenced and sequence end, finds and selects first PSB in current sequence and exits to restart interaction.
		If select display by ID, initiates identified validated processing segment: <ul style="list-style-type: none"> – Unless identified PSB not sequenced, scans intervening logical sequence. – If primary sequence member, checks all PSBs between most recently current primary sequence PSB and identified PSB, and cycles back to sequence start when appropriate. – If preceded in own sequence by current PSB, checks all PSBs between current PSB and identified PSB. Otherwise checks all PSBs from start of own sequence to identified PSB.
	MSGWSU	– If any PSB in logical sequence required has required input or is ES PSB with C-specs, signals WSU response pending and exits to issue WSU-0723.
	INTRST	– Selects identified PSB, and exits to restart interaction.

#WSX08 Routines

Routine	External Call	Description
NRM000		<p><i>Normal Function Handling Routine.</i></p> <p>If session suspended, restores session:</p> <ul style="list-style-type: none"> - Signals session not suspended. - If session suspended for GET:
	NRM130	<ul style="list-style-type: none"> - Restart processing at GET C-spec.
	PUTS	<ul style="list-style-type: none"> - If user response pending, exits to display PSB-associated user format.
	MSGWSU	<p>If no user response pending:</p> <ul style="list-style-type: none"> - Sets off RP indicator. - If next PSB unselected (user intervention required), exits to issue WSU-0722. - Makes next PSB current and new next PSB unselected. - Signals PUT override not allowed. - If enter mode and input required for PSB, signals required input pending. - If in sequence, adjusts last sequenced PSB pointer. - If in primary sequence, adjusts last primary sequence PSB pointer.
	PUTS	<ul style="list-style-type: none"> - If auto display indicated, signals auto PUTS response pending and exits to display PSB-associated user format.
		<p>If user response pending:</p> <ul style="list-style-type: none"> - Signals required input not pending. - If input data present, extracts input fields from SDB to JDB. <p>If C-spec group associated with current PSB, (re)starts C-spec processing:</p> <ul style="list-style-type: none"> - If preprocessing PUTS or IMSG response pending, sets IAR past last C-spec processed. - If auto PUTS, MSG, or no response pending, clears IAR and ARR and zeros length of current C-spec. - Initializes IAR CSB to start of PSB-associated group.
NRM130		<p>Signals no user response pending.</p> <ul style="list-style-type: none"> - Signals C-spec in process.
	(CDBGET)	<ul style="list-style-type: none"> - If not current, gets CDB for IAR CSB.
	CSPEC	<ul style="list-style-type: none"> - Exits to start processing at first C-spec in group. (Control may return.) - Signals C-spec not in process.
NRM140		<p>Signals no user response pending.</p> <p>If RV time:</p>
	MSGWSU	<ul style="list-style-type: none"> - Makes current PSB undetermined. - If ES, EW, or indicator not on and EJ not forced, erases format table and issues WSU-0717/0718 with invite input.
	(MSGWSU)	<ul style="list-style-type: none"> - Erases format table and issues WSU-0717/0718 with no invite input.
	SELECT	<p>Signals select next display requested and exits to select next enter mode segment.</p>
CDBG00		<p><i>Command Data Block Get Routine.</i></p>
	(REALOC)	<p>Reallocates any current CDB as deallocatable, and transfers pointer from COMMON to CSB.</p>
	(ALOCAT)	<p>Makes IAR CSB current.</p>
	(WFIO)	<p>If CDB for IAR CSB not allowed, allocates relocatable, non-deallocatable pool element of requisite size and priority, formats work file IOB, fetches CDB from work file, and reallocates pool element to free unused space and relocate.</p>
	(REALOC)	<p>If CDB is allocated, reallocates CDB as non-deallocatable, and transfers pointer from CSB to COMMON.</p>
	(REALOC)	<p>Returns to caller.</p>

#WSX09 Routines

Routine	External Call	Description
EWS000		<i>End Work Session Routine.</i>
	EWS070	If abort in progress or work station release requested, performs abnormal session termination.
	EWS100	If EJ time: <ul style="list-style-type: none"> – Performs normal session termination for last work station. If EW time: <ul style="list-style-type: none"> – Signals not EW time. – Selects WSU-0712. – If no other active session and either non-NEP job or EOJ forced, initiates EJ processing: <ul style="list-style-type: none"> – Signals EJ time, – Sets ON EJ indicator, – Selects EJ processing, – Signals no user response pending, and exits to restart interaction. – If some other session active or NEP job and EOJ not forced, performs normal session termination.
	INTRST EWS100	
EWS100	(TXGET) (TXPUT)	Performs work station termination (sign-off): <ul style="list-style-type: none"> – Gets work station control record, and updates its trailer for normal session end. – Signals work station release requested.
EWS070	(\$INFO) (REALOC) (REALOC) (TXGET) (TXPUT) (\$CLOS) (\$EOJ) (WSIO) INTCMP	Deactivates session: <ul style="list-style-type: none"> – Replaces UPSI switches with indicators U1-U8. – Clears processing flags in WCB (flags WCB not active). – Deallocates any normal WDB. – Deallocates any alternate WDB. If EJ has been done by this work session: <ul style="list-style-type: none"> – Performs job termination (shutdown): <ul style="list-style-type: none"> – Gets job control record and updates IJS trailer. – Closes all data files. – Terminates job. – Clears processing flags in every WCB key entry. – If work station release requested, signals work station release not requested, formats work station DTF, and releases work station. – Signals no current WCB exits. – Exits to complete interaction. If not EW time: <ul style="list-style-type: none"> – Signals no user response pending. – Sets on EW indicator. – Signals EW time. – Selects EW processing. – Exits to restart interaction.
	INTRST	

#WSX10 Routines

Routine	External Call	Description
DRIV00	(TRDRVR)	<i>Execution Driver Routine.</i> If active, logs driver interaction cycle trace entry.
DRIV010		If interaction cycle beginning:
	(\$WSIO)	<ul style="list-style-type: none"> - Formats work station DTF. - Accepts input from WSDM. - Saves current work station ID and interaction aid. - If input data not null, signals input data present.
	(TRWSIO)	<ul style="list-style-type: none"> - If active, logs work station I/O trace entry. - Locates active WCB for old requestor and makes current.
	WSABC	<ul style="list-style-type: none"> - If WCB not found (assumes new requestor), exits to process work station I/O completion code.
	(WDBGGET)	<ul style="list-style-type: none"> - If current work station ID other than that last tended, retrieves normal WDB to restore session environment. - Sets off command key indicators.
	WSABC	<ul style="list-style-type: none"> - If WDSM completion abnormal, exits to process work station I/O completion code (control may return).
		If interaction cycle (re)starting:
	RESP	<ul style="list-style-type: none"> - Signals no interaction cycle restarting. - If WSU response pending, exits to process message response. - If enter entered:
	MENU NRMLFN	<ul style="list-style-type: none"> - If menu response pending, exits to process menu response, or else exits to process normal function. - If roll down entered:
	MSGWSU REVIEW	<ul style="list-style-type: none"> - If review not supported, exits to issue WSU-0702, or else exits to review mode handler. - If roll up entered:
	MSGWSU REVIEW	<ul style="list-style-type: none"> - Signals roll up requested. If review not supported, exits to issue WSU-0702, or else exits to review mode handler. - Sets command key indicator. - If review function entered (command keys 3-6):
	MSGWSU REVIEW NRMLFN MSGWSU	<ul style="list-style-type: none"> - If review not supported, exits to issue WSU-0702, or else exits to review mode handler. (Control returns for command key 3.) Exits to process normal function. - If menu response pending, exits to issue WSU-0702. - If menu request (command key 1):
	PUTS	<ul style="list-style-type: none"> - Signals menu response pending and exits to display menu. - If bypass display request (command key 2):
	SELECT REVIEW	<ul style="list-style-type: none"> - If not RV time, exits to select next enter mode segment, or else exits to Review Mode Handler. - If reserved function entered (command keys 14-15):
	MSGWSU NRMLFN	<ul style="list-style-type: none"> - Exits to issue WSU-0702. - If user-defined function entered (other command key), exits to process normal function.

Routine	External Call	Description	
DRIV010 (continued)		If interaction cycle completing:	
	(WDBPUT)	<ul style="list-style-type: none"> – If current WCB exits, if GET update performed, flags all WCB key entries with PUT pending (for shared update-capable files). – If GET before PUT required, prepares normal WDB to save session environment, and clears processing flags 1 and 2 (signals no current WCB exists). Determines status of sessions. 	
	DRIV010	<ul style="list-style-type: none"> – If no session suspended, begins new cycle. – Searches for first active, suspended, resumable (no other WCB has put pending and matching key in contending WCB key entry) session. 	
	DRIV010	<ul style="list-style-type: none"> – If no session resumable, but some session active and unsuspended, begins cycle. 	
	ABORT	<ul style="list-style-type: none"> – If all active sessions suspended (deadlock) Aborts Job (25). 	
		Resumes session:	
	(WDBGET)	<ul style="list-style-type: none"> – Makes first resumable WCB current. – Retrieves WDB to restore session environment. – Signals current WCB exists. – Signals GET suspension. – Flags WCB not suspended. – Restores current work station ID from WCB. 	
	NRMLFN	<ul style="list-style-type: none"> – Exits to process normal function. 	
	WDBG00		<i>Work Station Data Block Get Routine.</i> (Expects normal/alternate indication in WRK11.) Sets up to process normal or alternate WDB.
		(ALOCAT) (WFIO)	If requested WDB not allocated, allocates relocatable, pool element of requisite size and priority, formats work file IOB, and reads requested work file WDB to new pool element.
(MOVE)		Restores work station level JDB fields, common work station level indicators, and common work station level control data fields from WDB.	
(REALOC)		If allocation was done, reallocates pooled WDB to free unused space at end, relocate shortened WDB, and set deallocatable attribute. Returns to caller.	
WDBP00			<i>Work Station Data Block Put Routine.</i> (Expects normal/alternate indication in WRK11.) Sets up to process normal or alternate WDB.
	(ALOCAT)	If requested WDB not allocated, allocates relocatable pool element of requisite size and priority.	
	(MOVE)	Formats WDB in pool, saving work station level JDB fields, common work station level indicators, and common work station level control data fields in WDB.	
	(REALOC)	If allocation was done, reallocates pooled WDB to set deallocatable attribute. Returns to caller.	

#WSX11 Routines

Routine	External Call	Description
MULT00		<i>Multiply Routine.</i> Processes multiply C-spec operation and saves result in JDB. Returns to caller.

#WSX12 Routines

Routine	External Call	Description
DIVD00		<i>Divide Routine.</i> Processes divide C-spec operation, and saves result in JDB and remainder in CSPWRK.
	ABORT	If divide by zero, Aborts Job (32). Returns to caller.
MVR000		<i>Move Remainder Routine.</i> (Expects remainder in CSPWRK.) Processes MVR C-spec operation and saves result in JDB. Returns to caller.

#WSX13 Routines

Routine	External Call	Description
ALTC00		<i>Alphameric Literal Table Compare Routine.</i> Processes table compare C-spec operation for table of alphameric literals. Returns to caller.
AFTC00		<i>Alphameric Field Table Compare Routine.</i> Processes table compare C-spec operation for table of alphameric fields. Returns to caller.

#WSX14 Routines

Routine	External Call	Description
NLTC00		<i>Numeric Literal Table Compare Routine.</i> Processes table compare C-spec operation for table of numeric literals. Returns to caller.
NFTC00		<i>Numeric Field Table Compare Routine.</i> Processes table compare C-spec operation for table of numeric fields. Returns to caller.

#WSX15 Routines

Routine	External Call	Description
BIND00		<i>Binary to Decimal Conversion Routine.</i> (Expects address of specified DSB in WRK01-WRK11 and binary value in WRK5-6.) Converts specified binary value to decimal character equivalent in JDB position indicated in specified DSB. Returns to caller.
DECB00		<i>Decimal to Binary Conversion Routine.</i> (Expects address of specified DSB in WRK01-WRK11). Converts decimal character value in JDB position indicated in specified DSB to binary equivalent in key area. Returns to caller.

#WSX16 Routines

Routine	External Call	Description
RESP00		<i>WSU Message Response Handling Routine.</i> Signals WSU response not pending. If response to WSU-0701:
	MSGWSU	– If IJ time, signals WSU response pending and exits to erase format table and reissue WSU-0701. – Forces enter aid.
	INTRST	– Exits to restart interaction.
		If response to WSU-0703:
	INTRST	– If not command key 13 (accept with error), exits to restart interaction.
	MSGWSU	– If AE indicator off, signals WSU response pending and exits to reissue WSU-0703. – If menu display up, signals select display by ID requested and signals no menu response pending. – If menu display not up, sets on command key 2 (bypass display).
	SELECT	– Exits to select next enter mode segment.
		If response to WSU-0723:
	INTRST	– If not command key 13 (accept error aid), exits to restart interaction. – Forces enter aid.
	MSGWSU	– If AE indicator off, signals WSU response pending and exits to reissue WSU-0723. – Initiates error-accepted identified processing: Signals no user response pending. Signals no menu response pending. Makes selected processing segment next to process.
	INTRST	– Exits to restart interaction.
	ABORT	If response unrecognized, Aborts Job (27).

Routine	External Call	Description
MENU00		<p><i>Menu Response Handling Routine.</i></p> <p>Signals menu response not pending. Signals no input data present. If display ID entered:</p> <ul style="list-style-type: none"> – Signals select display by ID requested.
	SELECT REVIEW	<ul style="list-style-type: none"> – If not RV time, saves display ID and exits to select enter mode segment. – If RV time, exits to review mode handler.
	MSGWSU	<p>If EW entered:</p> <ul style="list-style-type: none"> – If EW time, exits to issue WSU-0702. – Signals not ES time. – Sets on EW indicator. – Signals select next logical display requested.
	SELECT	<ul style="list-style-type: none"> – Exits to select enter mode segment.
	INTRST	<p>If TR entered:</p> <ul style="list-style-type: none"> – Resets trace control from menu-entered template. – Exits to restart interaction.
	MSGWSU	<p>If invalid data entered for EW/TR:</p> <ul style="list-style-type: none"> – Signals menu response pending. – Exits to issue WSU-0719.
	INTRST REVIEW	<p>If relative record number to review entered (assumed):</p> <ul style="list-style-type: none"> – Sets *RLRN to relative record number to review. – If *RLRN zero, exits to restart interaction. – If review supported, requests review by relative record number and exits to review mode handler.
	MSGWSU	<ul style="list-style-type: none"> – Exits to issue WSU-0702.
AWS000		<p><i>Work Session Abort Routine.</i></p> <p>Signals session aborted.</p>
	(MSGWSU)	<p>If not work station I/O error abort ID, signals work station release requested and erases format table and issues WSU-0711.</p>
	EWSESS	<p>Exits to end work session.</p>
WSAB00		<p><i>Work Station I/O Abnormal Completion Handling Routine.</i></p> <p>If no current WCB exists:</p> <ul style="list-style-type: none"> – If sign-on attempt, exits to initiate work session. – If stop requested, Aborts Job (36). – If invite input failed, Aborts Job (11). – If any other error, Aborts Job (26).
	IWSESS ABORT ABORT ABORT	<p>If current WCB exists:</p> <ul style="list-style-type: none"> – If release of SRT station, returns to caller (ignores error). – If stop requested, signals stop required and returns to caller.
	ABORT ABORT ABORT ABORT	<ul style="list-style-type: none"> – If permanent error, Aborts Job (13). – If work station released by operator, Aborts Job (14). – If work station offline, Aborts Job (14). – If any other error, Aborts Job (15).

#WSX17 Routines

Routine	External Call	Description
TXG000		<i>Transaction File Control Record Get Routine.</i> Copies relative record number to key area. Flags any active WCB with a transaction file key entry flagged for PUT pending for GET before PUT required.
	(FDBGGET)	If not current, allocates FDB for transaction file FSB. Adjusts DTF and IOB buffer pointers.
	(\$GETD)	Gets record. Saves logical record displacement. Moves trailer information from FDB to common work area. Clears saved record type.
	(TRDFIO)	If active, logs data file I/O trace entry.
	ABORT	If get out of extent, if not review record GET, Aborts Session (28).
	ABORT	If GET unsuccessful, Aborts Session (22).
		If review record GET or end of data record chain, returns to caller.
	ABORT	If trailer ID invalid for operation type, Aborts Session (19, 20, 21).
	ABORT	If data record not marked with current work station ID, Aborts Session (21). Returns to caller.
TXP000		<i>Transaction File Control Record PUT Routine.</i> (Expects key in key area.) Adjusts DTF and IOB buffer pointers. Copies trailer information from common work area to FDB.
	(\$PUTD)	Puts record. Clears saved record type.
	(TRDFIO)	If active, logs data file I/O trace entry.
	ABORT	If PUT unsuccessful, Aborts Session (23).
		Returns to caller.

#WSX18 Routines

Routine	External Call	Description
TDRG00		<i>Transaction Data File Record GET Routine.</i>
	(FDBGGET)	If not current, allocates FDB for transaction file. Copies relative record number to key area.
	(GET)	Gets transaction file data record. If C-spec GET updates transaction, <i>Last GET</i> pointers in common <i>Last GET</i> template from trailer chain fields. Returns to caller.

Routine	External Call	Description
FDBG00		<i>File Data Block GET Routine.</i> (Expects address of selected FSB in WRK01-WRK11.)
	(REALOC)	Reallocates any current FDB as deallocatable, and transfers pointer from COMMON to FSB.
	(ALOCAT)	Makes selected FSB current. If FDB for selected FSB not allocated, signals first time GET, flags DTF for first time GET, primes buffer pointers, then allocates relocatable, non-deallocatable pool element of requisite size and priority.
	(REALOC)	If FDB allocated, reallocates FDB as non-deallocatable, and transfers pointer from FSB to COMMON. Returns to caller.
KEY000		<i>Master File Key Building Routine.</i>
	(DECBIN)	Builds master file key in key area, extracts chain fields from JDB, concatenates indexed unpacked keys, packs indexed packed keys, or converts direct keys to binary as indicated in the current FSB. Returns to caller.

#WSX19 Routines

Routine	External Call	Description
REV000		<i>Review Mode Handling Routine.</i> (Expects ID of selected PSB in WRK10-WRK11.)
		If command key 2 (bypass display):
	MSGWSU	<ul style="list-style-type: none"> – If review or insert not in progress, exits to issue WSU-0702. – Initiates next review/insert mode processing segment: Selects next PSB which supports review-of or insert-after current review record type. – Signals no user response pending. – Signals no menu response pending.
	INTRST	– Exits to restart interaction.
		If select by ID:
	MSGWSU	– If review or insert not in progress, exits to issue WSU-0702.
	MSGWSU	– If no PSB exists with the selected ID or if PSB does not support review-of or insert-after current review record type, exits to issue WSU-0707.
		– Selects identified PSB.
		– Signals no user response pending.
		– Signals no menu response pending.
	INTRST	– Exits to restart interaction.

Routine	External Call	Description
DRG000	(FDBGET) (KEYBLD) (GET)	<p><i>Master Data File Record GET Routine.</i> (Expects FSB address in WRK0-WRK1.) If not current, allocates FDB for specified master file. Builds key as specified in FSB. Gets master file data record. Returns to caller.</p>
DRP000	(FDBGET) (KEYBLD) (PUT) (PUT) ABORT (BINDEC) (PUT) (PUT) (BINDEC) (PUT)	<p><i>Transaction/Master Data File Record PUT Routine.</i> (Expects FSB address in WRK0-1.) Signals GET before PUT not required. Signals no PUT pending.</p> <p>If master file record update:</p> <ul style="list-style-type: none"> – Builds key as specified in FSB. – Puts master file data record. – Returns to caller. <p>If record identifying indicator not saved (transaction file record last GET update):</p> <ul style="list-style-type: none"> – Places <i>Last GET</i> relative record number in key area. – Puts transaction file data record. – Returns to caller. <p>If insert mode (transaction file record insert):</p> <ul style="list-style-type: none"> – If header record insert attempted, Aborts Session (12). – Signals insert has been done. – Reserves new work station data record. – Updates transaction pointers in COMMON. – Initializes trailer information. – Signals add pending. <p>(BINDEC)</p> <ul style="list-style-type: none"> – Changes *RLNO to reserved next record. – Places relative record number in key area. <p>(PUT)</p> <ul style="list-style-type: none"> – Puts transaction file data record. – Returns to caller. <p>If review mode (transaction file record review update):</p> <ul style="list-style-type: none"> – Places review relative record number in key area. – Puts transaction file data record. – Returns to caller. <p>(PUT)</p> <p>If enter mode (transaction file record add):</p> <ul style="list-style-type: none"> – Reserves new work station data record if an extra was not reserved already. – Initializes trailer information. – If appropriate, marks trailer as header data record. – Updates transaction pointers in COMMON. – Adjusts <i>Last GET</i> pointers to end of file. – Signals add pending. <p>(BINDEC)</p> <ul style="list-style-type: none"> – Changes *RLNO to reserved next record. – Places relative record number in key area. <p>(PUT)</p> <ul style="list-style-type: none"> – Puts transaction file data record. – Returns to caller.

Routine	External Call	Description
REV000 (continued)	MSGWSU REV900 (BINDEC) REV900 INTRST (TXGET) (TXPUT) (TXGET) (TXPUT) (TXGET) (TXPUT)	<p>If command key 4 (begin insert):</p> <ul style="list-style-type: none"> – If IN time or not RV time, exits to issue WSU-0702. – If no current review record RID address or insert attempted at end of chain, goes to erase format table and issue WSU-0704. – Updates transaction pointers in COMMON to prepare for insert. – Sets *RLNO to last reserved work station record. – Sets on IN indicator. – Signals IN time. – Signals mode message pending. – If no PSB supports insert after current review record type, goes to erase format table and issues WSU-0704. – Selects first PSB which supports insert after current review record type. – Signals no user response pending. – Signals no menu response pending. – Exits to restart interaction. <p>If insert mode:</p> <ul style="list-style-type: none"> – If insert was done, completes insert: Updates transaction file trailer in last inserted record (Next record = next review record), – Updates transaction file trailer in next review record (Previous record = last inserted record), – Updates transaction file trailer in current review record (Next record = first insert record). – Signals no insert was done. – Signals not IN time.
	MSGWSU (WDBGGET)	<p>If command key 3 (resume entry):</p> <ul style="list-style-type: none"> – If not review mode, exits to issue WSU-0702. – Resumes enter mode: Restores alternate WDB (enter mode environment). (Signals not RV time and sets off CG, IN, and RV indicators.) – Signals mode message pending. – Signals review screen not saved. – Clears processing flags in all WCB key entries. – Returns to caller.
	MSGWSU (WDBPUT)	<p>Assumes review request:</p> <ul style="list-style-type: none"> – If IJ, IW, ES, EW, or EJ time, exits to issue WSU-0702. – If not review mode, suspends enter mode: Signals session suspended for review mode. – Saves alternate WDB (enter mode environment). – Clears indicator bytes 13-16. – Clears common work station level pointers. – If only one work station record reserved head, reserves new work station record. – Initializes transaction pointers in common for review. – Signals RV time. – Signals mode message pending.
	(SAVRST)	<ul style="list-style-type: none"> – Saves enter mode screen. – Clears processing flags in all WCB key entries. – Clears indicator bytes 00-12.
	(CLRJDB)	<ul style="list-style-type: none"> – Zeros or blanks alternate work station level JDB fields. – Sets on RV indicator.

Routine	External Call	Description
REV000 (continued)	MSGWSU	If command key 5 (page backward group): <ul style="list-style-type: none"> – If no header records, exits to issue WSU-0702. – Selects review previous header.
	MSGWSU (TXGET)	If command key 6 (page forward group): <ul style="list-style-type: none"> – If no header records, exits to issue WSU-0702. – Reads transaction file records, starting at review next record, until header record found. – Selects found header record.
	(DECBIN)	If roll down function key (page backward record), selects review previous record. If roll up function key (page forward record), selects review next record.
	(TDRGET)	If review by record number (menu entry assumed), converts *RLRN to binary and selects that record.
	REV900	Gets selected data record. If record not found, initializes review pointers to end of chain. Clears current review record RID address, and goes to erase format table and issue WSU-0706. <ul style="list-style-type: none"> – Updates review record pointers. Updates <i>Last GET</i> record pointers. <ul style="list-style-type: none"> – If review record current group header or in current chain, sets on CG indicator; otherwise, sets off CG indicator.
	REV900	<ul style="list-style-type: none"> – If record type not identified, clears current review record RID address and goes to erase format table and issue WSU-0705. – Finds and saves current review record RID address.
	REV900	<ul style="list-style-type: none"> – If record type not reviewable, clears current review record RID address and goes to erase format table and issue WSU-0705. – Selects first PSB which supports review of record type. – Signals no user response pending. – Signals no menu response pending.
	INTRST	<ul style="list-style-type: none"> – Exits to restart interaction.
REV900		Makes next PSB unselected. Makes current PSB undetermined. Signals no user response pending. Signals no menu response pending.
	MSGWSU	Exits to erase format table and issue WSU-0704/0705/0706.
CLR000		<i>Job Data Block Clear Routine.</i> (Expects normal/alternate indication in WRK11.) Initializes normal or alternate work station level JDB, blanks all fields, then places character zeros in individual arithmetic fields. Returns to caller.

ROUTINE CROSS-REFERENCE LIST

Figure 1-6 shows how WSU execution routines call one another. For each routine, the module that contains it is listed in parentheses.

Routine Name	Calls	Called by
ABORT (#WSXP)	INTEND	ALOCAT CSPEC DIVD DRIVER DRPUT GET IWSESS PUT REALOC RESP START TXGET TXPUT WFIO WSABC
AFTCMP (#WSX13)	EXIT	ALTCMP* CSPEC
ALOCAT (#WSXP)	ABORT CALL DEALLOC EXIT FREE RLOCAT TRPOOL	CDBGET FDBGET MDBGET TRNGET WDBGET WDBPUT
ALTCMP (#WSX13)	AFTCMP*	CSPEC
AWSESS (#WSX16)	CALL EWSESS MSGWSU	INTEND
BINDEC (#WSX15)	EXIT	DRPUT IWSESS REVIEW

Routine Name	Calls	Called by
CALL (#WSXP)	TRNGET	ALOCAT AWSESS CSPEC CDBGET DRGET DRPUT DRIVER EWSESS FDBGET GET INTBEG INTCMP INTEND INTRST IWSESS KEYBLD MDBGET MENU MSGWSU NRMLFN PUT PUTS REALOC RESP REVIEW SELECT TDRGET TRNGET TXGET TXPUT WDBGET WDBPUT WSABC WSID

*This routine is not specifically called, but its code is shared.

Figure 1-6 (Part 1 of 5). ROUTINE CROSS-REFERENCE CHART

Routine Name	Calls	Called by
CLRJDB (#WSX19)	EXIT	IWSESS REVIEW
CSPEC (#WSX05)	ABORT AFTCMP ALTCMP ARRANGE CALL CDBGET DIVD DRGET DRPUT EXIT INTRST MULT MVR NFTCMP NLTCMP PUTS TDRGET TXGET	NRMLFN
DEALLOC (#WSXP)	MOVE TRPOOL WFIO	ALOCAT REALOC
DECBIN (#WSX15)	EXIT	KEYBLD REVIEW
DIVD (#WSX12)	EXIT ABORT	CSPEC
DRGET (#WSX18)	CALL EXIT FDBGET GET KEYBLD	CSPEC

Routine Name	Calls	Called by
DRIVER (#WSX10)	ABORT CALL MENU MSGWSU NRMLFN PUTS RESP REVIEW SELECT TRDRVR TRWSIO WDBGET WDBPUT WSABC	INTBEG INTCMP
DRPUT (#WSX18)	ABORT BINDEC CALL EXIT KEYBLD PUT	CSPEC
EWSESS (#WSX09)	CALL INTCMP INTRST REALOC TXGET TXPUT WSIO	AWSESS (SELECT)
MVR (#WSX12)	EXIT	CSPEC
ARRANGE (#WSX04)	EXIT	CSPEC
CDBGET (#WSX08)	ALOCAT CALL EXIT REALOC WFIO	CSPEC NRMLFN

Figure 1-6 (Part 2 of 5). ROUTINE CROSS-REFERENCE CHART

Routine Name	Calls	Called by
EXIT (#WSX8)	TRNGET	ALOCAT AFTCMP ARRANGE BINDEC CDBGET CLRJDB CSPEC DECBIN DIVD DRGET DRPUT FDBGET GET KEYBLD MDBGET MSGWSU MULT MVR NFTCMP PUT PUTS REALOC REVIEW SAVRST TDRGET TRDFIO TRDRVR TRPOOL TRWSIO TXGET TXPUT WDBGET WDBPUT WSABC WSIO
FREE (#WSXP)	TRPOOL	ALOCAT REALOC
GET (#WSX01)	ABORT CALL EXIT INTCMP MSGWSU TRDFIO	DRGET TDRGET
INTBEG (#WSXP)	CALL DRIVER	START

Routine Name	Calls	Called by
INTCMP (#WSX7)	CALL DRIVER	EWSESS GET MSGWSU PUTS
INTEND (#WSXP)	CALL AWSESS	ABORT
INTRST (#WSXP)	CALL DRIVER	CSPEC EWSESS IWSESS MENU RESP REVIEW SELECT
IWSESS (#WSX06)	ABORT BINDEC CALL CLRJDB EWESS INTRST MSGWSU TXGET TXPUT	SELECT WSABC
KEYBLD (#WSX18)	CALL DECBIN EXIT	DRGET DRPUT
MDBGET (#WSX03)	ALOCAT CALL EXIT REALOC WFIO	MSGWSU
MENU (#WSX16)	CALL INTRST MSGWSU REVIEW SELECT	DRIVER
MOVE (#WSXP)	RETURN	DEALLOC RLOCAT WDBGET WDBPUT

Figure 1-6 (Part 3 of 5). ROUTINE CROSS-REFERENCE CHART

Routine Name	Calls	Called by
MSGWSU (#WSX03)	CALL EXIT INTCMP MDBGET WSIO	AWSESS DRIVER GET IWSESS MENU NRMLFN PUTS* RESP REVIEW SELECT
MULT (#WSX11)	EXIT	CSPEC
NFTCMP (#WSX14)	EXIT	CSPEC NLTCMP*
NLTCMP (#WSX14)	NFTCMP*	CSPEC
NRMLFN (#WSX08)	CALL CDBGET CSPEC MSGWSU PUTS SELECT	DRIVER
PUT (#WSX02)	ABORT CALL EXIT TRDFIO	DRPUT
PUTS (#WSX03)	CALL EXIT INTCMP MSGWSU* SAVRST WSIO	CSPEC DRIVER NRMLFN
REALOC (#WSXP)	ABORT CALL DEALOC EXIT FREE RLOCAT TRPOOL	CDBGET EWSSESS FDBGET GET MDBGET TRNGET WDBGET WDBPUT

Routine Name	Calls	Called by
FDBGET (#WSX18)	ALOCAT EXIT CALL REALOC	DRGET DRPUT TDRGET TXGET
RESP (#WSX16)	ABORT CALL INTRST MSGWSU SELECT	DRIVER
REVIEW (#WSX19)	BINDEC CALL CLRJDB DECBIN EXIT INTRST MSGWSU SAVRST TDRGET TXGET TXPUT WDBGET WDBPUT	DRIVER MENU SELECT
RLOCAT (#WSXP)	MOVE SHIFT TRPOOL	ALOCAT REALOC
SELECT (#WSX07)	CALL EWSSESS INTRST IWSESS MSGWSU REVIEW	DRIVER MENU NRMLFN RESP
SHIFT (#WSXP)	RETURN	RLOCAT
TDRGET (#WSX18)	CALL EXIT FDBGET GET	CSPEC REVIEW
TRDFIO (#WSX00)	EXIT	GET PUT TXGET TXPUT

*This routine is not specifically called, but its code is shared.

Figure 1-6 (Part 4 of 5). ROUTINE CROSS-REFERENCE CHART

Routine Name	Calls	Called by
TRDRVR (#WSX00)	EXIT	DRIVER
TRNGET (#WSXP)	ALOCAT CALL REALOC	CALL EXIT
TRPOOL (#WSXP)	EXIT RETURN	ALOCAT DEALOC FREE REALOC RLOCAT
TRWSIO (#WSX00)	EXIT	DRIVER WSIO
TXGET (#WSX17)	ABORT CALL EXIT FDBGGET TRDFIO	CSPEC EWSSESS IWSESS REVIEW
SAVRST (#WSX03)	EXIT WSIO*	PUTS REVIEW
TXPUT (#WSX17)	ABORT CALL EXIT TRDFIO	EWSSESS IWSESS REVIEW
WDBGGET (#WSX10)	ALOCAT CALL EXIT MOVE REALOC WFIO	DRIVER REVIEW
WDBPUT (#WSX10)	ALOCAT CALL EXIT MOVE REALOC	DRIVER REVIEW
WFIO (#WSXP)	ABORT RETURN	CDBGGET DEALOC MDBGET WDBGGET

Routine Name	Calls	Called by
WSABC (#WSX16)	ABORT CALL EXIT IWSESS	DRIVER WSIO
WSIO (#WSX03)	CALL EXIT TRWSIO WSABC	EWSSESS MSWSU PUTS SAVRST*

*This routine is not specifically called, but its code is shared.

Figure 1-6 (Part 5 of 5). ROUTINE CROSS-REFERENCE CHART

EXTERNAL CALLS MADE BY EXECUTION ROUTINES

Figure 1-7 lists the external calls made by the WSU execution routines.

Routine	External Call	Explanation
#WSX11		
#WSX11	\$FIND	Find WSU message member
	\$LOG	Abort: WSU message member not found
	\$INFO	Modify JCB (WSU message member disk address)
	\$INFO	Modify JCB (user message member disk address)
	GETP	Get partition end
	\$INFO	Get NEP attribute
	\$INFO	Get MRTMAX value
	FDIO	Read object program control header
	\$ALOC	Allocate work station DTF
	\$OPEN	Open work station DTF
	\$CLOS	Close work station DTF
	\$FIND	Find #WSX12
	\$LOAD	Load #WSX12
	\$XFER	
	RFINDLIB	Find library
	\$FIND	Find object member
	SYSIN	Read sysin record (utility control statement)
	FDIO	Read object program area sectors
	\$INFO	Get system date
	MSG	Retrieve message
	\$XFER	
	RSLOG	Abort job with appropriate MIC
#WSX12		
#WSX12	\$ALOC	Allocate master/transaction file
	\$OPEN	Open master/transaction file
	\$ALOC	Special allocate work file
	\$OPEN	Open work file
	\$FIND	Find #WSXP
	\$LOAD	Load #WSXP
	AFARW	Get transaction file format 1
	DALOC	Deallocate transaction file (if new and retain code not S or J)
	SALOC	Reallocate transaction file old
	MSG	Retrieve message
	\$XFER	
	RSLOG	Abort job with appropriate MIC
#WSXP		
START	\$FIND	Find transient module object member
CALL	LOGTR	Log trace entry for call to subroutine
EXIT	LOGTR	Log trace entry for exit from subroutine
ABORT	LOGTR	Log trace entry for abort session/job
	\$LOG	Abort: abort already in progress
	\$LOG	Abort session with appropriate MIC
WFIO	FDIO	Perform work file I/O
TRNGET	\$LOAD	Perform transient module load
TRPOOL	LOGTR	Log trace entry for pool storage request
#WSX00		
TRDRVR	\$TOD	Get system time for driver trace entry
	LOGTR	Log trace entry for driver interaction cycle
TRWSIO	LOGTR	Log trace entry for work station I/O
TRDFIO	LOGTR	Log trace entry for data file I/O
	LOGTR	Log trace entry for master file key
	LOGTR	Log trace entry for transaction file trailer

Figure 1-7 (Part 1 of 2). EXTERNAL CALLS MADE BY EXECUTION ROUTINES

Routine	External Call	Explanation
#WSX01 GET	\$GETD	Get data record for read
#WSX02 PUT	\$GETD \$PUTD	Get data file record for update Put data file record
#WSX03 WSIO	\$WSIO	Perform work station I/O
#WSX05 CSPEC	LOGTR	Log trace entry for C-spec operation
#WSX06 IWSESS	\$INFO	Get UPSI switches (U1-U8)
#WSX09 EWSSESS	\$CLOS \$EOJ \$INFO	Close data file DTFs at end-of-job Terminate job Put UPSI switches (U1-U8)
#WSX10 DRIVER	\$WSIO	Accept work station input
#WSX17 TXGET TXPUT	\$GETD \$PUTD	Get transaction file control record Put transaction file control record

Figure 1-7 (Part 1 of 2). EXTERNAL CALLS MADE BY EXECUTION ROUTINES

Directory

This section identifies WSU modules so that quick references can be made to program listings and microfiche. Modules are listed alphabetically by name. Each entry in the list has the following information:

- Module name
- Routine name (if there are multiple routines in the module)
- Descriptive name
- Entry point
- Major functions

Module Name	Routine Name	Descriptive Name	Major Functions
#WSGAF		Data field assignment phase	Updates field name table with relative data field numbers and job data block displacements. Prints field name usage tables for data fields.
#WSGAI		Indicator assignment phase	Uses local copy of system indicator table to update indicator-referencing T, M, S, D, and C records with the corresponding mask/displacements. Prints indicator usage lists.
#WSGAL		Program label assignment phase	Updates field name table with relative command data block numbers and data block displacements for program labels.
#WSGAM		Message assignment phase	Updates C records that have message-defining literals with assigned relative message numbers. Builds MIC table and updates C records that have message-referencing MIC numbers with assigned relative message numbers. Prints list of referenced MICs.
#WSGBF		Field name table building phase	Builds field name table from data field and program label definition/references on T, M, F, S, D, and C records.
#WSGCD		Command data area building phase	Formats the command data area from C records, GCSBDFSB, GCSBDRSD, and GCSBDPSB and puts the area in work file object at the preassigned location.
#WSGCH		Control header building phase	Formats the control header from sections of GCOMMON and puts it in the first work file object sector.
#WSGCS		Command specification area building	Formats the command specification area from C records and puts the area in the work file object at the preassigned location.
#WSGDS		Data specification area building phase	Formats the data specification area from T, M, F, S, D, and C records and the system fields table and puts the area in the work file object at the preassigned location.
#WSGEP		Execution procedure member generating phase	Calculates the minimum execution region size. Formats the execution procedure statements and puts them in the new procedure member. Prints an execution region map and the OCL for the new procedure.
#WSGFO		Object program formatting phase	Builds file and record type specification block displacement tables from T, M, I, and F records. Builds a process segment specification block displacement table and table of display attributes and field counts from S and D records. Determines total object size and assigns relative locations to individual object areas.

Module Name	Routine Name	Descriptive Name	Major Functions
#WSGFS		File specification area building phase	Formats the file specification area from T, M, I, and F records and puts the area in the object work file at the preassigned location.
#WSGMD		Message data area building phase	Formats the message data area from C records and puts the area in the work file object at the preassigned location.
#WSGML		Message length assignment phase	Builds tables from D records that describe display screen format field distribution. Determines the number of bytes allowed for message text on each display screen format. Then updates D records with assigned message lengths. Prints extended diagnostics.
#WSGMS		Message specification area building phase	Formats the literal section of the message specification area from C records and puts this section in the work file object at the preassigned location. Formats the MIC section of the message specification area from C records and puts this section in the work file object after the literal section.
#WSGMT		Diagnostic message text listing and cleanup phase	Prints list of diagnostic message texts cross-referenced to error note numbers and severity codes. If an abort or terminal error has been signaled, #WSGMT abnormally terminates generation.
#WSGOB		Object program member generating phase	Loads the library management PUT/GET sector processor, (\$MAPGS) and copies the preformatted work file object to the new object member.
#WSGOM		WSU generator post-assign initialization phase	Initializes the post-assign GCOMMON extension, overlaying assign-only code and data.
#WSGPS		Process specification area building phase	Formats the process specification area from S records and GCSCRNTB and puts this area in the work file object at the preassigned location.
#WSGRF		Field data block resolution phase	Updates embedded field data blocks in T, M, F, S, D, and C records and system fields table with name definitions and relative number and displacement assignments from corresponding field name table entries.
#WSGSS		Screen specification area building phase	Formats the screen specification area from S and D records and GCSCRNTB and puts this area in the work file object at the preassigned location.
#WSGWF		WSU generator post-syntax initialization phase	Initializes the post-syntax GCOMMON extension, overlaying syntax-only code and data, and moves a copy of the system fields table into GCOMMON.

Module Name	Routine Name	Descriptive Name	Major Functions
#WSG0		WSU generator common initialization phase	Initializes the generator's common data areas (GCOMMON, GCCHGEND, and GCSYNTAX). Prints source records up to first noncomment or end of file.
#WSG1		J specification syntax checking phase	Prints J specification and reads and prints all source records to the first T, M, S, D, or C specification or EOF.
#WSG2		T and M specification, F and I specification syntax checking phase	Prints records in source buffer and reads and prints T, M, F, and I specifications and all source records to the first S, D, or C specification or end of file. Puts T, M, I, and F records in the work file intermediate text.
#WSG3A		S specification syntax checking phase	Prints record in source buffer and reads and prints S specifications and all other source records to first D or C specification or end of file. Puts S record in work file intermediate text.
#WSG3B		D specification syntax checking phase	Prints record in source buffer and reads and prints D specifications and all other source records to next C or S specification or end of file. Puts D records in work file intermediate text.
#WSG4A		C specification control level, conditioning indicators, factor 1 and resulting indicators syntax checking phase	Begins checking the syntax of C specifications. Puts special C and/or S records in the work file intermediate text.
#WSG4B		C specification operation, table, and continuation syntax phase	Completes syntax checking for table operations. Prints C specification and any continuation line and all other source records to next C, S, or D specification or end of file. Puts C record in work file intermediate text.
#WSG4C		C specification operation, MIC/message text, and continuation syntax checking phase	Completes syntax checking for message operations. Prints C specification and reads and prints any continuation line and all other source records to next C, S, or D specification or end of file. Puts C record in work file intermediate text.
#WSG4D		C specification operation, factor 2, result field, and half-adjust syntax checking phase	Completes syntax checking for non message/table operations. Prints C specifications and reads and prints all source records to next C, S, or D specification or end of file. Puts C record in work file intermediate text.
#WSG5		File, record type, and processing level assignment phase	Builds relational tables for file, and record type definitions. Updates C and S records with relative file and record type numbers and processing level codes. Prints extended diagnostics.

Module Name	Routine Name	Descriptive Name	Major Functions
#WSG6		Format assignment phase	Builds relational table for format definitions. Updates C records with relative screen format numbers. Prints extended diagnostics.
#WSG8		Operation code assignment phase	Prints unreferenced, multiply-defined and undefined field names. Prints extended diagnostics. Adjusts field-name-definition-dependent operation codes in C records.
#WSG9		SFGR source member generating phase	Formats SFGR S and D records and puts them in a new source member.
#WSX11		WSU initialization phase (Part 1)	Prepares the environment for WSU execution by transferring the object program (OBJ) specification blocks to high storage permanent elements and opening the work station DTF.
#WSX12		WSU initialization phase (Part 2)	Prepares the environment for WSU execution by opening every data file DTF and allocating, opening, and formatting the work file, transferring object program data blocks and message member texts to pool-element-transient work file data blocks.
#WSXP		WSU processing phase	Contains initial load for low storage resident data/code.
	START	Processing initialization routine	Completes initialization of low storage resident data areas. Cycles through low storage processors and transient routines serially dispatch interactions (for up to MRTMAX work stations). Randomly tends active/new display station requests until the end of work session process or completes for the only active display station (if non-NEP) or until the job ends, the job aborts, or an operator cancels the job. Secures generation/execution variables passed from initialization phase. Collects load information for all WSX transients, and formats pool element storage.
	CALL	Call to subroutine processor	Performs enter linkage for called subroutine, and fetches subroutine-containing transient if necessary.
	EXIT	Exit from subroutine processor	Performs return linkage for called subroutine, and fetches calling-routine-containing transient if necessary.
	TRNGET	Transient-load-module get processor	Fetches specified transient module from WSX OBJ library to storage pool element.
	INTCMP INTRST INTBEG INTEND	Interaction complete/ restart/begin/end processor	Dispatches work session driver to complete, restart, or begin interaction cycle for work station, or work session abort to end cycle.

Module Name	Routine Name	Descriptive Name	Major Functions
#WSXP (continued)	ABORT	Abort session/job processor	Aborts job or session.
	WFIO	Work file I/O processor	Transfers specified number of sectors between work file and storage pool element.
	SHIFT	Shift data block processor	Shifts block of specified length from old starting address to new starting address.
	MOVE	Move data block processor	Moves block of specified length from old ending address to new ending address.
	ALOCAT	Pool element allocation routine	Satisfies storage request involving new pool element.
	REALOC	Pool element reallocation routine	Satisfies storage request involving old pool element.
	FREE	Free pool element creation/expansion routine	Recovers unused space in allocated pool element.
	RLOCAT	Pool element relocation routine	Rearranges relocatable pool elements.
	DEALOC	Pool element deallocation routine	Performs special deallocation (saving WDB or indicating transient routines overlaid) and recovers deallocated pool element space.
	#WSX00		WSU execution transient number 00
	TRDRVR	Trace Driver R	Logs a driver trace entry.
	TRWSIO	Trace work station I/O R	Logs a work station trace entry.
	TRDFIO	Trace data file I/O R	Logs data file trace entries.
#WSX01		WSU execution transient number 01	
	GET	Data file record read routine	Reads record from specified data file, and suspends session if necessary to avoid record contention.
#WSX02		WSU execution transient number 02	
	PUT	Data file record write routine	Writes record to specified data file.

Module Name	Routine Name	Descriptive Name	Major Functions
#WSX03		WSU execution transient number 03	
	TRPOOL	Trace pool	Trace pool element functions if pool trace is active.
	RETURN	Return from subroutine	Restores registers and returns to the subroutine's caller.
	PUTS	Screen display put transient	Displays specified WSU/user format and selects any timely WSU message.
	MSGWSU	WSU message put transient	Erases format if requested and displays specified WSU and/or user message or resets keyboard.
	SAVRST	Screen save/restore transient	Saves and restores screens for menu and review suspension.
	WSIO	Work station I/O processing transient	Relays request involving display screen or keyboard to work station data management (WSDM).
	MDBGET	Message data block get transient	Gets specified MDB text from work file into pool element.
#WSX04		WSU execution transient number 04	
	ARANGE	Alphameric range transient	Processes the RANGE operation for alphameric factors.
#WSX05		WSU execution transient number 05	
	CSPEC	C specification processing transient	Processes all C-specifications for current processing segment.
	CSPO200		GETNH subprocessor.
	CSPO400		GETNR subprocessor.
	CSP0600		GETPR subprocessor.
	CSP0800		GETPH subprocessor.
	CSPOA00		GET subprocessor.
	CSPOCOO		PUT subprocessor
	CSPOEOO		PUTS subprocessor.
	CSP1000		MSG subprocessor.
	CSP1200		IMSG subprocessor.
	CSP1400		EXSR subprocessor.
	CSP1600		SETON subprocessor.
	CSP1800		SETOF subprocessor.
	CSP1A00		GOTO subprocessor.
	CSP4000		COMP (alpha) subprocessor.
	CSP4200		RANGE (alpha) subprocessor.
	CSO4400		MOVE subprocessor.
	CSP4600		MOVE L subprocessor.
	CSP4800		MOVE *BLANK subprocessor.

Module Name	Routine Name	Descriptive Name	Major Functions
#WSX05 (continued)	CSP4A00		COMP (alpha, literal table) subprocessor.
	CSP4C00		COMP (alpha, field table) subprocessor.
	CSP6A00		COMP (numeric, literal table) subprocessor.
	GSP6C00		COMP (numeric, field table) subprocessor.
	CSP6E00		MUTL subprocessor.
	CSP7000		DIVD subprocessor.
	CSP7200		MVR subprocessor.
	CSP7400		ZADD subprocessor.
	CSP7600		ZSUB subprocessor.
	CSP7800		ADD subprocessor.
	CSP7A00		SUB subprocessor.
CSP7C00		RANGE (numeric) subprocessor.	
CSP7E00		COMP (numeric) subprocessor.	
#WSX06		WSU execution transient number 06	
	IWSESS	Work session initiate routine	Performs sign-on initialization of transaction file and session activation, and initiates the IJ, IW, and the first sequence screen processing segments.
#WSX07		WSU execution transient number 07	
	SELECT	Processing segment selection routine	Determines next segment to be processed in enter mode.
#WSX08		WSU execution transient number 08	
	NRMLFN	Normal function handling routine	Handles automatic puts and program-defined processing within current segment.
	CDBGET	Command data block get routine	Gets specified CDB from work file and places it in the pool.
#WSX09		WSU execution transient number 09	
	EWSESS	Work session end routine	Performs session deactivation and sign-off transaction file and work session termination and initiates the EW and EJ processing segments.
#WSX10		WSU execution transient number 10	
	DRIVER	Execution driver routine	Drives interaction cycles, restores get-suspended session, accepts user, sets command/function key indicators, and selects appropriate routine for keyed response or request. Saves or restores normal WDB for cycle turnover.

Module Name	Routine Name	Descriptive Name	Major Functions
#WSX10 (continued)	WDBGGET	Work station data block get routine	Gets normal or alternate WDB from work file into pool element and transfers corresponding sections to work station level JDB fields, common indicators, and common data pointers.
	WDBPUT	Work station data block put routine	Puts work station level JDB fields, common indicators, and common data pointers into corresponding sections of normal or alternate WDB in pool element.
#WSX11		WSU execution transient number 11	
	MULT	Multiply routine	Processes C-spec multiply operation.
#WSX12		WSU execution transient number 12	
	DIVD	Divide routine	Processes C-spec divide operation.
	MVR	MVR routine	Processes C-spec MVR (move remainder) operation.
#WSX13		WSU execution transient number 13	
	ALTCMP	Alphameric literal table compare routine	Processes C-spec table compare operation for table of alphameric literals.
	AFTCMP	Alphameric field table compare routine	Processes C-spec table compare operation for table of alphameric fields.
#WSX14		WSU execution transient number 14	
	NLTCMP	Numeric literal table compare routine	Processes C-spec table compare operation for table of numeric literals.
	NFTCMP	Numeric field table compare routine	Processes C-spec table compare operation for table of numeric fields.
#WSX15		WSU execution transient number 15	
	BINDEC	Binary to decimal conversion routine	Converts specified binary value to decimal character equivalent to JDB.
	DECBIN	Decimal to binary conversion routine	Converts decimal character value in JDB key area to binary equivalent in key area.

Module Name	Routine Name	Descriptive Name	Major Functions
#WSX16		WSU execution transient number 16	
	RESP	WSU message response handling routine	Routes control to appropriate processor for response to WSU message.
	MENU	Menu response handling routine	Routes control to appropriate processor for response to MENU display.
	AWSESS	Work session abort routine	Schedules abort of current work session, and notifies user when possible.
	WSABC	Abnormal work station I/O handling routine	Handles abnormal completion code from WSDM. Schedules appropriate WSU action.
#WSX17		WSU execution transient number 17	
	TXGET	Transaction file control record get routine	Gets trailer information from transaction file record.
	TXPUT	Transaction file control record put routine	Updates trailer information in transaction file record.
#WSX18		WSU execution transient number 18	
	TDRGET	Transaction file record get transient	Performs I/O for C-spec or review of GETPH, GETPR, GETNH, or GETNR operation.
	DRGET	Master file record get transient	Performs I/O for GET C-spec operation.
	DRPUT	Transaction/master file record put transient	Performs I/O for C-spec operation. Formats trailer information transaction file adds and inserts.
	FDBGET	File data block get transient	Allocates FDB pool element.
	KEYBLD	Master file key building transient	Builds master file key in key area.
#WSX19		WSU execution transient number 19	
	REVIEW	Review mode processing transient	Handles review mode processing requests.
	CLRJDB	Job data block clear transient	Zeros or blanks all normal or alternate work station level JDB fields.

Data Areas

Figure 1-8 contains a list of the WSU execution data areas and a description of how the execution routines reference or modify them. Refer to Appendix H of the *Data Areas and Diagnostic Aids Handbook* for explanations of these data areas.

Legend:

- 0 — Area cleared, zeroed, or blanked
- 1 — Area copied from
- 2 — Area copied into
- 3 — Area loaded or read into
- 4 — Area written from
- 5 — Area read into and/or written from
- 6 — Field in area extracted from
- 7 — Field in area extracted into, converted/edited
- C — Area created (allocated/made addressable)
- D — Area destroyed
- R — Field in area referenced, no field modified
- M — Field in area modified
- — Field in area assumed overlaid
- — Area passed at service interface, no field referenced
- * — Individual fields cross-referenced elsewhere

Notes:

1. Some labels locate field ends (for example, RLOCWRK).
2. Some fields overlap (for example, KEYAREA, CSPWRK, EDITJ).
3. In general, . . . WRK areas lose integrity upon exit from any associated routine. Exceptions are the use of CSPWRK by DIVD to pass information about the remainder to CSPEC subroutine MVR and the use of KEYAREA in the interfaces to GET, PUT, and TXPUT transients and the use of WRKIOB in the interface to WFIO.
4. In general, WRK. . labels are used for work areas not involved in any transient interfaces. The same fields, when involved in transient interfaces, are referenced by descriptive names (for example, TX# . . .). However, there are several exceptions. For example:
 - WRK10-11 — User-selected format ID (review)
 - WRK00-01 — PSB address (PUTS)
 - WRK00-01 — MSB address (MDBGET)
 - WRK00-01 — DSB address (BINDEC)
 - WRK00-01 — DSB address (DECBIN)
 - WRK00-01 — FSB address (FDBGET)
 - WRK01 — X'80' to identify alternate JDB request (CLRJDB)
 - WRK01 — X'80' to identify alternate WDB request (WDBGET/WDBPUT)

Figure 1-8 (Part 3 of 3). Data Area Usage

Diagnostic Aids

Figure 1-9 lists the terminal errors that can occur during execution and identifies the module that detects each error.

#WSX11	0617	WSU execution message member not found
	0607	First entry on first sysin record not OCL verb
	0608	WSX verb invalid
	0609	END verb invalid
	0629	Invalid trace parameter
	0614	User object program member name missing
	0615	User message member name missing
	0616	User format member name missing (not specified with library name in OCL)
	0602	I/O error reading object program control header
	0627	FSA missing from object program
	0628	DSA missing from object program
	0635	PSA missing from object program
	0630	FSA format error, invalid block identifier
	0616	User format member name missing (not defaulted in control header)
	0622	Object program library not found
	0623	Message library not found
	0624	Format library not found
	0618	User object program member not found
	0619	User message member not found
	0600	Unexpected SYSIN EOF
	0601	Unrecognized SYSIN I/O return code
	0606	Missing '/' on SYSIN record
	0605	Null OCL statement
	0604	Keyword/parameter/verb on OCL statement too long
	0610	OCL parameter not followed by ' ' or ','
	0633	Invalid character in OCL parameter
	0611	OCL keyword not followed by '-'
	0612	Unrecognized WSX statement keyword
	0613	Duplicate WSX statement keyword
	0602	I/O error reading object program specification block or C-spec data block sector(s)
	0625	DSA format error, not enough system fields
	0634	Insufficient region size
	0603	Unable to retrieve WSU message
	0636	Unable to allocate transaction file
#WSX12	0631	Unable to allocate execution work file
	0602	I/O error reading object program MDA sector
	0603	Unable to retrieve WSU message
#WSXP	0632	Unable to retrieve user message
	0802	Job aborted (no current work station)
	0802	Job aborted (recursive abort for current work station)

Figure 1-9 (Part 1 of 2). Execution Terminal Errors

---- Job/Session Abortion

(for last work station for non-NEP or normally-ending job)

WFIO	00	Abnormal work file I/O completion
ALOCAT	31	Size requested in SRB exceeds pool maximum
	01	Invalid SRB (attribute, priority, data pointer) or previous relocation pending
REALOC	02	Invalid relocation request (deallocation for free, first, or last element) (set nondeallocatable/deallocatable attribute for nonrelocatable element, etc.)
START	03	Unable to find transient module
GET	04	Unable to get record, I/O error
	06	Unable to get record, invalid key
	05	Record not found, and no not-found indicator
PUT	30	Transaction file relative record number overflow
	08	Put requested, but no put pending
	33	Key for put does not match key for pending put
	33	Unable to put record, update (key) error
	07	Unable to get record, I/O error
	29	Unable to get transaction file record, past extent
	09	Unable to put record, I/O error
CSPEC	10	Invalid operation code
IWSESS	16	No inactive session available
	17	Invalid work station control record ID
WSABC	11	No active session and last session aborted
DRIVER	25	No nonsuspended sessions (deadlock)
DIVD	32	Divide by 0
RESP	27	Unrecognized MIC for pending response
WSABC	36	No current session, shutdown requested
	26	No current session and invite input failed
	13	Permanent work station I/O error
	14	Work station released by operator or offline
	15	Unspecified work station I/O error
TXGET	22	Unable to get transaction file record
	28	Out of extent on transaction file GET operation
	19	Record type invalid for job control get
	20	Record type invalid for session control get
	21	Record type invalid for noncontrol get
	21	Data record not marked with work station ID
TXPUT	23	Unable to put transaction file record
DRPUT	12	Header record insertion attempted

---- Normal Session Completion

(for last work station for non-NEP job)

Figure 1-9 (Part 2 of 2). Execution Terminal Errors

- #WSGAF (data field assignment phase)
 - directory 1-73
- #WSGAI (indicator assignment phase)
 - directory 1-73
- #WSGAL (program label assignment phase)
 - directory 1-73
- #WSGAM (message assignment phase)
 - directory 1-73
- #WSGBF (field name table building phase)
 - directory 1-73
- #WSGCD (command data area building phase)
 - directory 1-73
- #WSGCH (control header building phase)
 - directory 1-73
- #WSGCS (command specification area building)
 - directory 1-73
- #WSGDS (data specification area building phase)
 - directory 1-73
- #WSGEP (execution procedure member generating phase)
 - directory 1-73
- #WSGFS (file specification area building phase)
 - directory 1-73
- #WSGF0 (object program formatting phase)
 - directory 1-73
- #WSGI1 (WSU initialization phase, part 1)
 - description 1-36
 - directory 1-76
- #WSGI2 (WSU initialization phase, part 2)
 - description 1-37
 - directory 1-76
- #WSGMD (message data area building phase)
 - directory 1-74
- #WSGML (message length assignment phase)
 - directory 1-74
- #WSGMS (message specification area building phase)
 - directory 1-74
- #WSGMT (diagnostic message text listing and clean up phase)
 - directory 1-74
- #WSGOB (object program member generating phase)
 - directory 1-74
- #WSGOM (WSU generator post assign initialization phase)
 - directory 1-74
- #WSGPS (process specification area building phase)
 - directory 1-74
- #WSGRF (file data block resolution phase)
 - directory 1-74
- #WSGSS (screen specification area building phase)
 - directory 1-74
- #WSGWF (WSU generator post syntax initialization)
 - directory 1-74
- #WSGO (WSU generator common initialization phase)
 - directory 1-75
- #WSG1 (J specification syntax checking phase)
 - directory 1-75
- #WSG2 (T and M, F and I specification syntax checking phase)
 - directory 1-75
- #WSG3A (S specification syntax checking phase)
 - directory 1-75
- #WSG3B (D specification syntax checking phase)
 - directory 1-75
- #WSG4A (C specification control level, conditioning indicators, factor 1 and resulting indicators syntax checking phase)
 - directory 1-75
- #WSG4B (C specification operation, table, and continuation syntax checking phase)
 - directory 1-75
- #WSG4C (C specification operation, MIC/message text, and continuation syntax checking phase)
 - directory 1-75
- #WSG4D (C specification operation, factor 2, result field and half-adjust syntax checking phase)
 - directory 1-75
- #WSG5 (file, record type, and processing level assignment phase)
 - directory 1-75
- #WSG6 (format assignment phase)
 - directory 1-76
- #WSG8 (operation code assignment phase)
 - directory 1-76
- #WSG9 (SPGR source member generation phase)
 - directory 1-76
- #WSXP (WSU processing phase)
 - description 1-41
 - directory 1-76
- #WSX01 (WSU execution transient number 1)
 - description 1-41
 - directory 1-77
- #WSX02 (WSU execution transient number 2)
 - description 1-42
 - directory 1-77
- #WSX03 (WSU execution transient number 3)
 - description 1-43
 - directory 1-78
- #WSX04 (WSU execution transient number 4)
 - description 1-45
 - directory 1-78
- #WSX05 (WSU execution transient number 5)
 - description 1-45
 - directory 1-78
- #WSX06 (WSU execution transient number 6)
 - description 1-49
 - directory 1-79
- #WSX07 (WSU execution transient number 7)
 - description 1-50
 - directory 1-79
- #WSX08 (WSU execution transient number 8)
 - description 1-51
 - directory 1-79
- #WSX09 (WSU execution transient number 9)
 - description 1-52
 - directory 1-79
- #WSX10 (WSU execution transient number 10)
 - description 1-53
 - directory 1-79
- #WSX11 (WSU execution transient number 11)
 - description 1-55
 - directory 1-80

#WSX12 (WSU execution transient number 12)
 description 1-55
 directory 1-80

#WSX13 (WSU execution transient number 13)
 description 1-55
 directory 1-80

#WSX14 (WSU execution transient number 14)
 description 1-55
 directory 1-80

#WSX15 (WSU execution transient number 15)
 description 1-56
 directory 1-80

#WSX16 (WSU execution transient number 16)
 description 1-56
 directory 1-81

#WSX17 (WSU execution transient number 17)
 description 1-58
 directory 1-81

#WSX18 (WSU execution transient number 18)
 description 1-58
 directory 1-81

#WSX19 (WSU execution transient number 19)
 description 1-59
 directory 1-81

abnormal work station I/O handling transient (WSABC)
 directory 1-81

ABORT (abort session/job processor)
 description 1-38
 directory 1-77

abort session/job processor (ABORT)
 description 1-38
 directory 1-77

AFTCOMP (alphameric field table compare transient)
 description 1-55
 directory 1-80

ALOCAT (pool element allocation transient)
 description 1-39
 directory 1-77

alphameric field table compare transient (AFTCOMP)
 description 1-55
 directory 1-80

alphameric range transient (ARRANGE)
 description 1-45
 directory 1-78

alphameric literal table compare transient (ALTCMP)
 description 1-55
 directory 1-80

ALTCMP (alphameric literal table compare transient)
 description 1-55
 directory 1-80

ARRANGE (alphameric range transient)
 description 1-45
 directory 1-78

AWSESS (work session abort transient)
 description 1-57
 directory 1-81

binary to decimal conversion transient (BINDEC)
 description 1-56
 directory 1-80

BINDEC (binary to decimal conversion transient)
 description 1-56
 directory 1-80

C specification control level, conditioning indicators, factor 1 and resulting indicators syntax checking phase (#WSG4A)
 directory 1-75

C specification operation, factor 2, result field and half-adjust syntax checking phase (#WSG4D)
 directory 1-75

C specification operation, MIC/message text, and continuation syntax checking phase (#WSG4C)
 directory 1-75

C specification operation, table, and continuation syntax checking phase (#WSG4B)
 directory 1-75

C specification processing transient (CSPEC)
 description 1-45
 directory 1-78

CALL (call to subroutine processor)
 description 1-38
 directory 1-76

call to subroutine processor (CALL)
 description 1-38
 directory 1-76

CLRJDB (job data block clear transient)
 description 1-62
 directory 1-81

command data area building phase (#WSGCD)
 directory 1-73

command specification area building (#WSGCS)
 directory 1-73

control header building phase (#WSGCH)
 directory 1-73

CSPEC (C specification processing transient)
 description 1-45
 directory 1-78

D specification syntax checking phase (#WSG3B)
 directory 1-75

data areas usage 1-82

data field assignment phase (#WSGAF)
 directory 1-73

data file record read transient (GET)
 description 1-41
 directory 1-77

data file record write transient (PUT)
 description 1-42
 directory 1-77

data specification area building phase (#WSGDS)
 directory 1-73

DEALLOC (pool element deallocation transient)
 description 1-40
 directory 1-77

DECBIN (decimal to binary conversion transient)
 description 1-56
 directory 1-80

decimal to binary conversion transient (DECBIN)
 description 1-56
 directory 1-80

diagnostic aids 1-86

diagnostic message text listing and clean up phase (#WSGMT)
 directory 1-74

DIVD (divide transient)
 description 1-55
 directory 1-80

divide transient (DIVD)
 description 1-55
 directory 1-80

DRGET (master file record get transient)
 description 1-60
 directory 1-81

DRIVER (execution driver transient)
 description 1-53
 directory 1-79

EWSESS (work session end transient)
 description 1-52
 directory 1-79

execution driver transient (DRIVER)
 description 1-53
 directory 1-79

execution procedure member generating phase (#WSGEP)
 directory 1-73

EXIT (exit from subroutine processor)
 description 1-38
 directory 1-76

exit from subroutine processor (EXIT)
 description 1-38
 directory 1-76

external call summary 1-69

F and I specification syntax checking phase (#WSG2)
 directory 1-75

FDBGET (file data block get transient)
 description 1-59
 directory 1-81

file data block get transient (FDBGET)
 description 1-59
 directory 1-81

file specification area building phase (#WSGFS)
 directory 1-74

file, record type, and processing level assignment phase (#WSG5)
 directory 1-75

file data block resolution phase (#WSGRF)
 directory 1-74

format assignment phase (#WSG6)
 directory 1-76

FREE (free pool element creation/expansion transient)
 description 1-40
 directory 1-77

free pool element creation/expansion transient (FREE)
 description 1-40
 directory 1-77

GET (data file record read transient)
 description 1-41
 directory 1-77

hierarchy diagram 1-28

indicator assignment phase (#WSGAI)
 directory 1-73

INTBEG (interaction begin)
 description 1-38
 directory 1-76

INTCMP (interaction complete)
 description 1-38
 directory 1-76

INTEND (interaction end)
 description 1-38
 directory 1-76

interaction begin (INTBEG)
 description 1-38
 directory 1-76

interaction complete (INTCMP)
 description 1-38
 directory 1-76

interaction end (INTEND)
 description 1-38
 directory 1-76

interaction restart (INTRST)
 description 1-38
 directory 1-76

INTRST (interaction restart)
 description 1-38
 directory 1-76

IWSESS (work session initiate transient)
 description 1-49
 directory 1-79

J specification syntax checking phase (#WSG1)
 directory 1-75

job data block clear transient (CLRJDB)
 description 1-62
 directory 1-81

KEYBLD (master file key building transient)
 description 1-59
 directory 1-81

master file key building transient (KEYBLD)
 description 1-59
 directory 1-81
 master file record get transient (DRGET)
 description 1-60
 directory 1-81
 MDBGET (message data block get transient)
 description 1-45
 MENU (menu response handling transient)
 description 1-57
 directory 1-81
 menu response handling transient (MENU)
 description 1-57
 directory 1-81
 message assignment phase (#WSGAM)
 directory 1-73
 message data area building phase (#WSGMD)
 directory 1-74
 message data block get transient (MDBGET)
 description 1-45
 message length assignment phase (#WSGML)
 directory 1-74
 message specification area building phase (#WSGMS)
 directory 1-74
 module flow diagram
 execution 1-18
 generation 1-17
 MOVE (move data block processor)
 description 1-39
 directory 1-77
 move data block processor (MOVE)
 description 1-39
 directory 1-77
 MSGWSU (WSU message put transient)
 description 1-43
 directory 1-78
 MULT (multiply transient)
 description 1-55
 directory 1-80
 multiply transient (MULT)
 description 1-55
 directory 1-80
 MVR (MVR transient)
 directory 1-80
 MVR transient (MVR)
 directory 1-80

 NFTCMP (numeric field table compare transient)
 description 1-55
 directory 1-80
 NLTCMP (numeric literal table compare transient)
 description 1-55
 directory 1-80
 normal function handling transient (NRMLFN)
 description 1-51
 directory 1-79
 NRMLFN (normal function handling transient)
 description 1-51
 directory 1-79
 numeric field table compare transient (NFTCMP)
 description 1-55
 directory 1-80
 numeric literal table compare transient (NLTCMP)
 description 1-55
 directory 1-80

object program formatting phase (#WSGF0)
 directory 1-73
 object program member generating phase (#WSGOB)
 directory 1-74
 operation code assignment phase (#WSG8)
 directory 1-76

pool element allocation transient (ALOCAT)
 description 1-39
 directory 1-77
 pool element deallocation transient (DEALOC)
 description 1-40
 directory 1-77
 pool element reallocation transient (REALOC)
 description 1-40
 directory 1-77
 pool element relocation transient (RLOCAT)
 directory 1-77
 process specification area building phase (#WSGPS)
 directory 1-74
 processing segment selection transient (SELECT)
 description 1-50
 directory 1-79
 program label assignment phase (#WSGAL)
 directory 1-73
 PUT (data file record write transient)
 description 1-42
 directory 1-77
 PUTS (screen display put transient)
 directory 1-78

REALOC (pool element reallocation transient)
 description 1-40
 directory 1-77
 RESP (WSU message response handling transient)
 description 1-56
 directory 1-81
 REVIEW (review mode processing transient)
 description 1-59
 directory 1-81
 review mode processing transient (REVIEW)
 description 1-59
 directory 1-81
 RLOCAT (pool element relocation transient)
 directory 1-77

S specification syntax checking phase (#WSG3A)
 directory 1-75
 SAVRST (screen save/restore transient)
 description 1-44
 directory 1-78
 screen display put transient (PUTS)
 directory 1-78
 screen save/restore transient (SAVRST)
 description 1-44
 directory 1-78
 screen specification area building phase (#WSGSS)
 directory 1-74

SELECT (processing segment selection transient)
 description 1-50
 directory 1-79

SHIFT (shift data block processor)
 description 1-39
 directory 1-77

shift data block processor (SHIFT)
 description 1-39
 directory 1-77

SPGR source member generation phase (#WSG9)
 directory 1-76

T and M specification syntax checking phase (#WSG2)
 directory 1-75

TDRGET (transaction file record get transient)
 description 1-58
 directory 1-81

transaction file control record get transient (TXGET)
 description 1-58
 directory 1-81

transaction file control record put transient (TXPUT)
 description 1-58
 directory 1-81

transaction file record get transient (TDRGET)
 description 1-58
 directory 1-81

transient-load-module get processor (TRNGET)
 description 1-38
 directory 1-76

TRNGET (transient-load-module get processor)
 description 1-38
 directory 1-76

TXGET (transaction file control record get transient)
 description 1-58
 directory 1-81

TXPUT (transaction file control record put transient)
 description 1-58
 directory 1-81

WDBGET (work station data block get transient)
 description 1-54
 directory 1-80

WDBPUT (work station data block put transient)
 description 1-54
 directory 1-80

WFIO (work file I/O processor)
 description 1-39
 directory 1-77

work file I/O processor (WFIO)
 description 1-39
 directory 1-77

work session abort transient (AWSESS)
 description 1-57
 directory 1-81

work session end transient (EWSESS)
 description 1-52
 directory 1-79

work session initiate transient (IWSESS)
 description 1-49
 directory 1-79

work station data block get transient (WDBGET)
 description 1-54
 directory 1-80

work station data block put transient (WDBPUT)
 description 1-54
 directory 1-80

WSABC (abnormal work station I/O handling transient)
 directory 1-81

WSU execution logic flow
 description 1-18

WSU execution transient number 1 (#WSX01)
 description 1-41
 directory 1-77

WSU execution transient number 10 (#WSX10)
 description 1-53
 directory 1-79

WSU execution transient number 11 (#WSX11)
 description 1-55
 directory 1-80

WSU execution transient number 12 (#WSX12)
 description 1-55
 directory 1-80

WSU execution transient number 13 (#WSX13)
 description 1-55
 directory 1-80

WSU execution transient number 14 (#WSX14)
 description 1-55
 directory 1-80

WSU execution transient number 15 (#WSX15)
 description 1-56
 directory 1-80

WSU execution transient number 16 (#WSX16)
 description 1-56
 directory 1-81

WSU execution transient number 17 (#WSX17)
 description 1-58
 directory 1-81

WSU execution transient number 18 (#WSX18)
 description 1-58
 directory 1-81

WSU execution transient number 19 (#WSX19)
 description 1-59
 directory 1-81

WSU execution transient number 2 (#WSX02)
 description 1-42
 directory 1-77

WSU execution transient number 3 (#WSX03)
 description 1-43
 directory 1-78

WSU execution transient number 4 (#WSX04)
 description 1-45
 directory 1-78

WSU execution transient number 5 (#WSX05)
 description 1-45
 directory 1-78

WSU execution transient number 9 (#WSX09)
 description 1-52
 directory 1-79

WSU generator common initialization phase (#WSG0)
 directory 1-75

WSU generator post assign initialization phase (#WSGOM)
 directory 1-74

WSU generator post syntax initialization (#WSGWF)
 directory 1-74

WSU initialization phase, part 1 (#WSGI1)

description 1-36

directory 1-76

WSU initialization phase, part 2 (#WSGI2)

description 1-37

directory 1-76

WSU message put transient (MSGWSU)

description 1-43

directory 1-78

WSU message response handling transient (RESP)

description 1-56

directory 1-81

Introduction

The screen design aid (SDA) portion of the utilities program product provides a convenient method of creating and maintaining:

- SFGR source specifications
- Menu build message member source statements

A skeleton RPG II program can also be created from an SFGR source member using SDA.

The SDA allows the System/34 user to:

- Create, add, update, or delete an entire SFGR source member format.
- Display the formats in an SFGR object member (individually, all, or list).
- Insert, delete, update, and duplicate the actual source statements in an SFGR source member using SEU.
- Create or update menu build source.
- Create an RPG II skeleton program containing related display station specifications.
- Display help text to assist the user.
- Recover user data if the system terminates abnormally.

HOW THE PROGRAM WORKS

The SDA user initiates all functions except help by entering the SDA command and applicable parameters. SDA then calls the option selection module to allow the user to select which function is to be performed and to perform the parameter diagnostics.

When a valid option is selected, a load module is selected to perform that function. Each load module determines the specific action to be performed based on SYSLOG prompting and formatted input.

Note: The library referred to in this discussion is the system library unless a user library is specified on the SDA command.

PHYSICAL CHARACTERISTICS

SDA is composed of 45 library members, 9 executable load modules, 23 load members, and 13 procedures.

The *Directory* section in this chapter explains the 9 executable load modules.

Load Members

#SA#M1	SDA message load member
#SASP	SDA menu syntax checking load module
#SA@CU	formats for #SAMN and #SACU
#SA@DS	formats for #SADS
#SA@DP	formats for #SADP
#SA@HP	formats for #SAHP
#SA@RC	formats for #SARC
#SA@RP	formats for #SARP
#SA@ME	formats for #SAME
#SAP1	reserved for expansion
#SAP2	reserved for expansion
#SAP3	reserved for expansion
#SAP4	reserved for expansion
#SAP5	reserved for expansion
#SAP6	reserved for expansion
#SAP7	reserved for expansion
#SAP8	reserved for expansion
#SAP9	reserved for expansion
#SAPA	reserved for expansion
#SAPB	reserved for expansion
#SAPC	reserved for expansion
#SAPD	reserved for expansion
#SAPE	reserved for expansion

Procedure Members

SDA	parameter substitution.
SDAH	displays help text.
SDALOAD	loads SDA into #LIBRARY.
SDASAVE	saves SDA on diskette.
SDADROP	deletes SDA from #LIBRARY or a specified library.
SDA1	executes code to create, add, or update total formats.
SDA2	executes code to delete entire formats or manipulate SFGR statements directly.
SDA3	displays formats in an SFGR load module.
SDA5	executes code to create or update menu source statements.
SDA6	executes \$SFGR to generate a new SFGR load module.
SDA7	executes code to build a skeleton RPG II display station program.
SDA8	performs user recovery.
SDA9	initiates all SDA functions except help text and recovery. This procedure routes activity to the proper subprocedure.

I/O STORAGE REQUIREMENTS

SDA is designed to operate in 14K (14,336) bytes of main storage.

SYSTEM CONFIGURATION

SDA runs on all models of the System/34. SDA requires SEU to support the update source function.

INPUT/OUTPUT

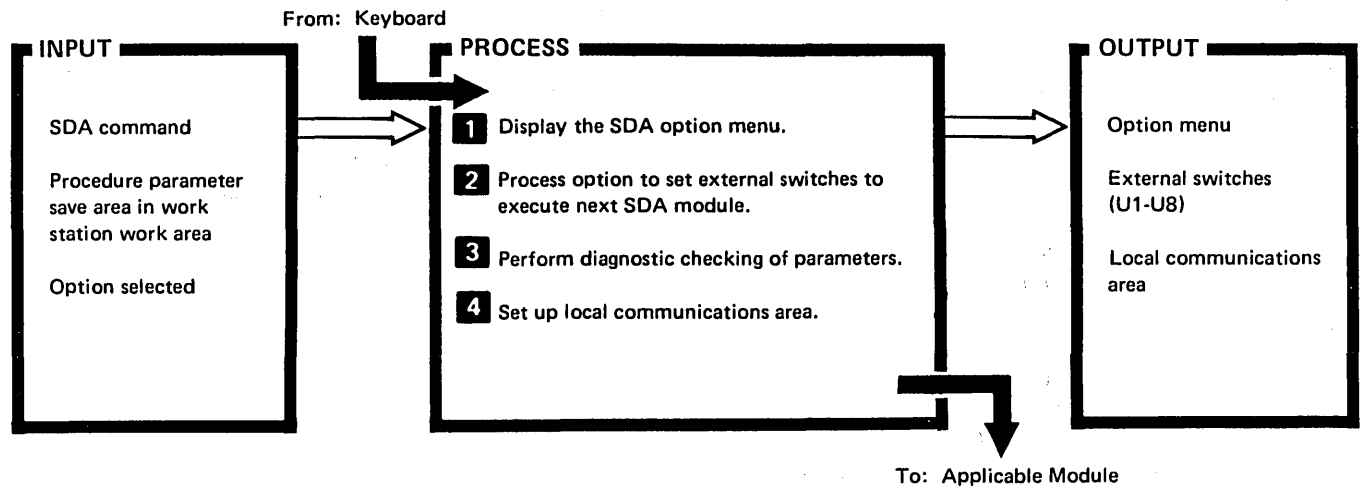
After the user signs on SDA, input to the program consists of:

- A copy of the selected member in the SDA work file if the member exists in a library
- Responses to SDA prompts
- Data entered from the keyboard if the user enters or updates SDA statements

Output from SDA is:

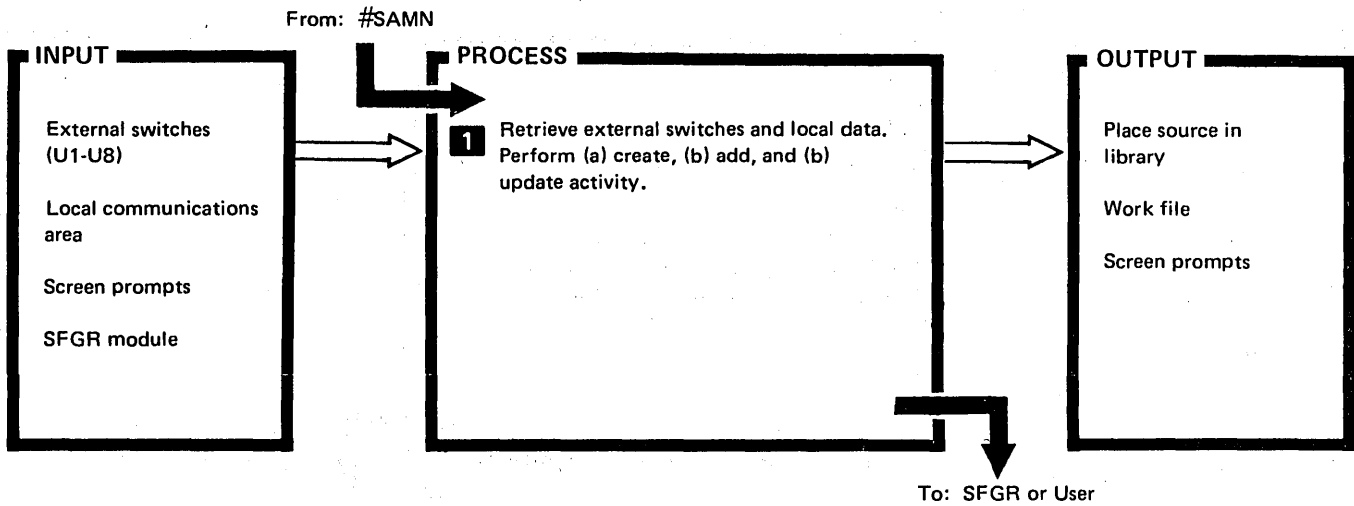
- A new or changed member in the specified library
- Required messages
- Selected printed statements if the print option was used
- An example of the screen image and listing of SFGR source produced when creating, adding, or updating entire formats

Method of Operation



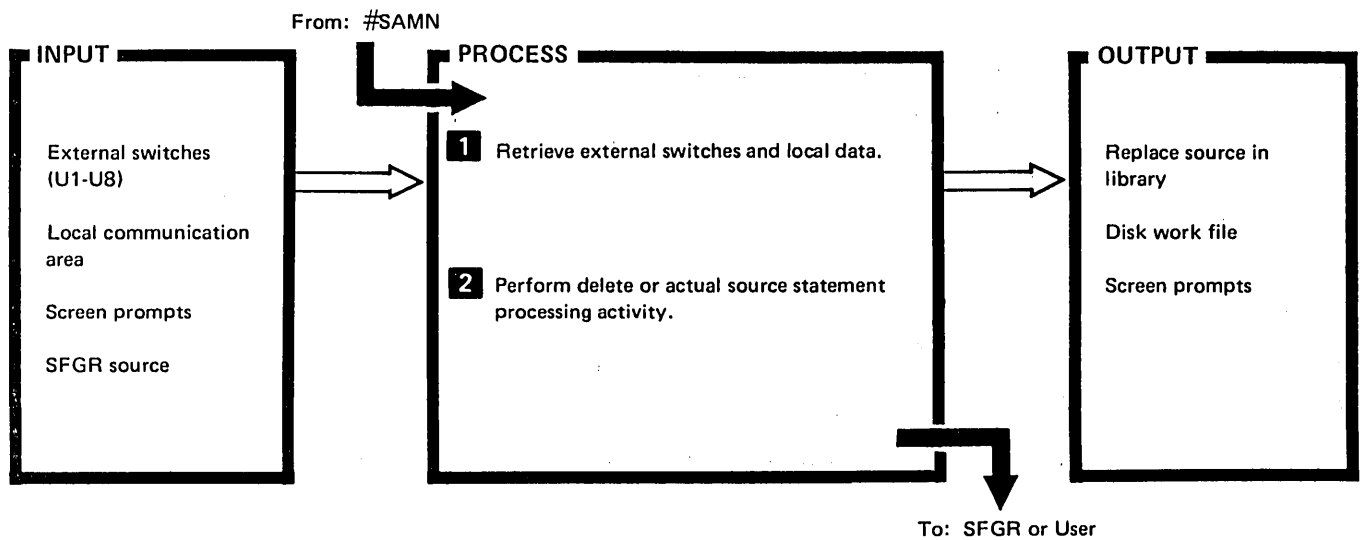
DESCRIPTION	MODULE/ ROUTINE
<ol style="list-style-type: none"> 1 By selecting a menu option the user chooses the function within SDA to perform. If an incorrect option is selected, the menu is redisplayed. 2 External switches (U1-U8) are set and interpreted by the OCL to execute the correct module for the selected function. 3 Checks for syntax and verifies that libraries exist for each parameter. If OCL is used, the module prompts for the necessary parameters. 4 Data is passed to affected modules in the local communications area in the PPSA layout. 	#SAMN

Diagram 2-1. SDA Mainline Router (#SAMN)



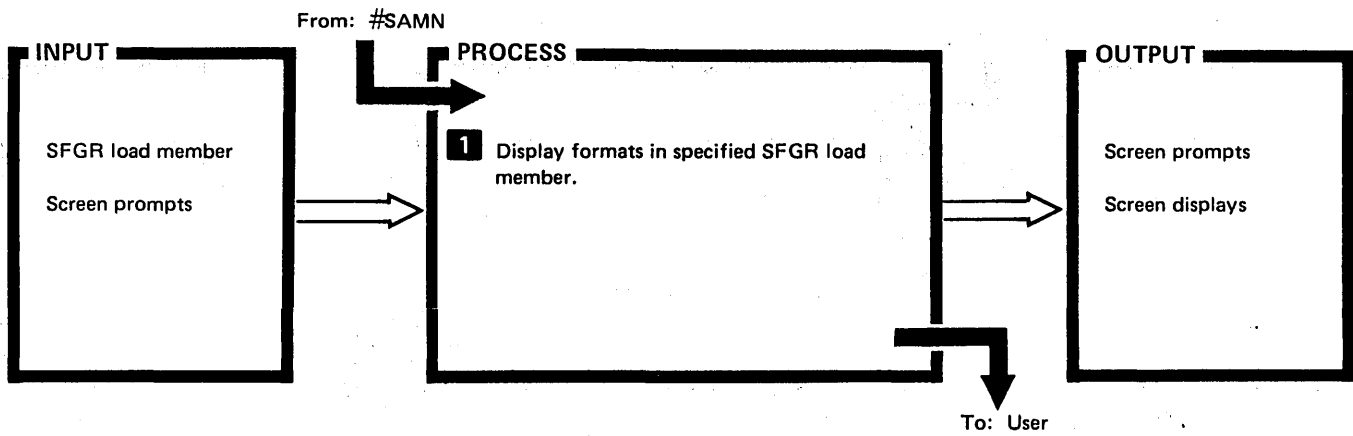
DESCRIPTION	MODULE/ ROUTINE
<p>1 a. The create screen sequence:</p> <ul style="list-style-type: none"> ● Prompts for S specification information. ● Provides blank screen to compose screen image. ● Prompts for field attributes. ● Prompts for overriding attributes if requested in step c. ● Prints the image of the screen composed in step b. <p>1 b. The add function is the same as create, except the existing source is copied into the work file first.</p> <p>1 c. The update function prompts the user to process specific formats or displays the format name encountered so the user can decide to update. Update rebuilds the screen image and returns to the create sequence at step c.</p>	<p>#SACU</p>

Diagram 2-2. SDA Total Format Create/Add/Update (#SACU)



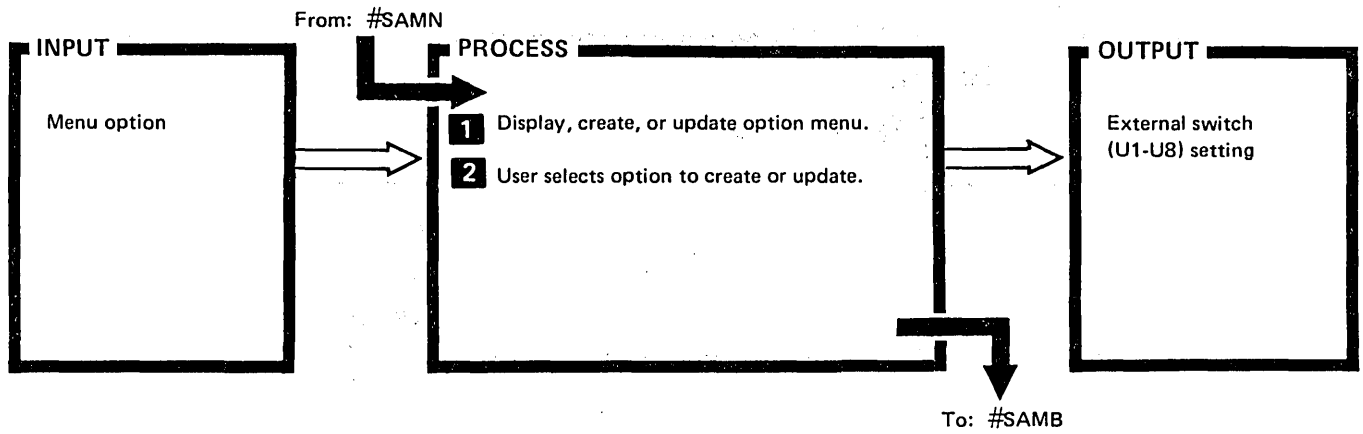
DESCRIPTION	MODULE/ ROUTINE
<p>1 Delete function:</p> <ul style="list-style-type: none"> • Prompts the user to scan for specific formats, and if selected the user enters the format name to scan for. The format name (if found) is displayed and the user can delete it (response to YES/NO prompt). • If no scan is desired each format encountered is displayed by name and deleted or not as described preceding. <p>2 Update source function uses SEU with SDA provided formats.</p>	#SADS

Diagram 2-3. SDA Format Delete (#SADS)



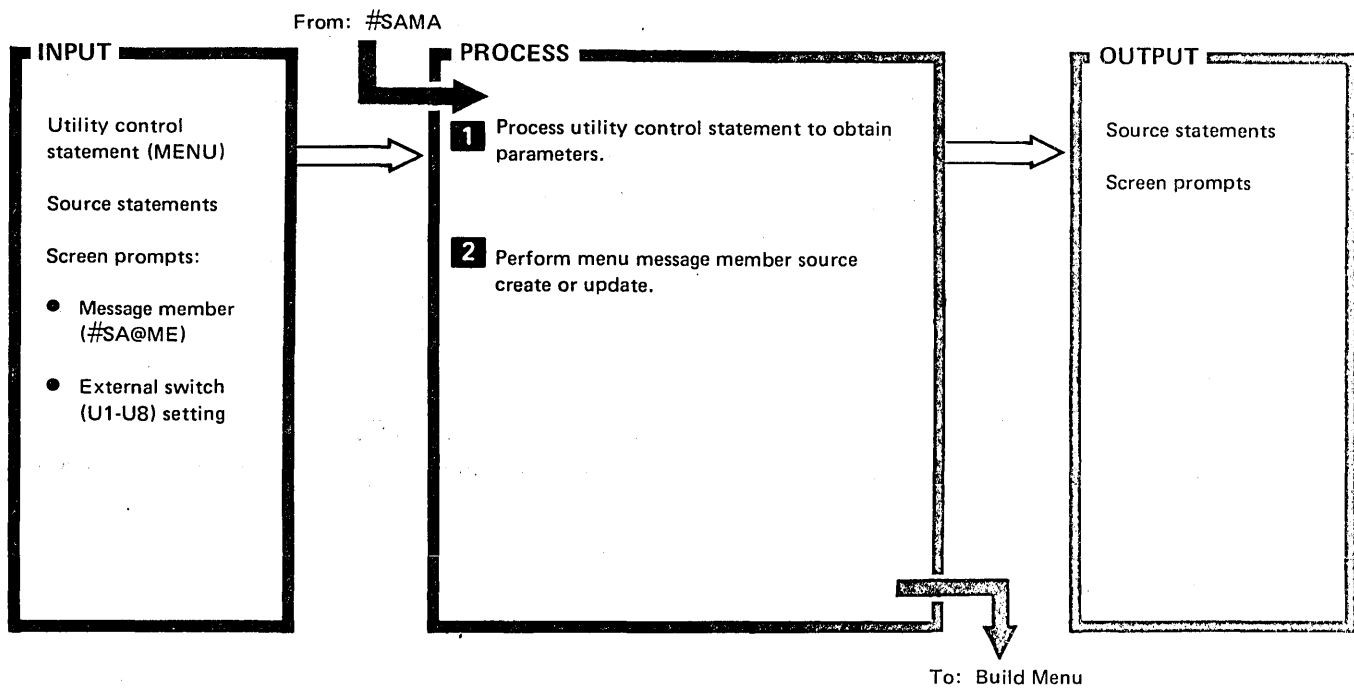
DESCRIPTION	MODULE/ ROUTINE
<p>1 The user can have:</p> <ul style="list-style-type: none"> ● A list of all the formats in an SFGR load member displayed. ● A specific format in an SFGR load member displayed. ● All of the formats in an SFGR load member displayed. <p><i>Note:</i> While a format is being displayed, the user can enter data into it and the format will act as if displayed by the user's program.</p>	#SADP

Diagram 2-4. SDA Display Function (#SADP)



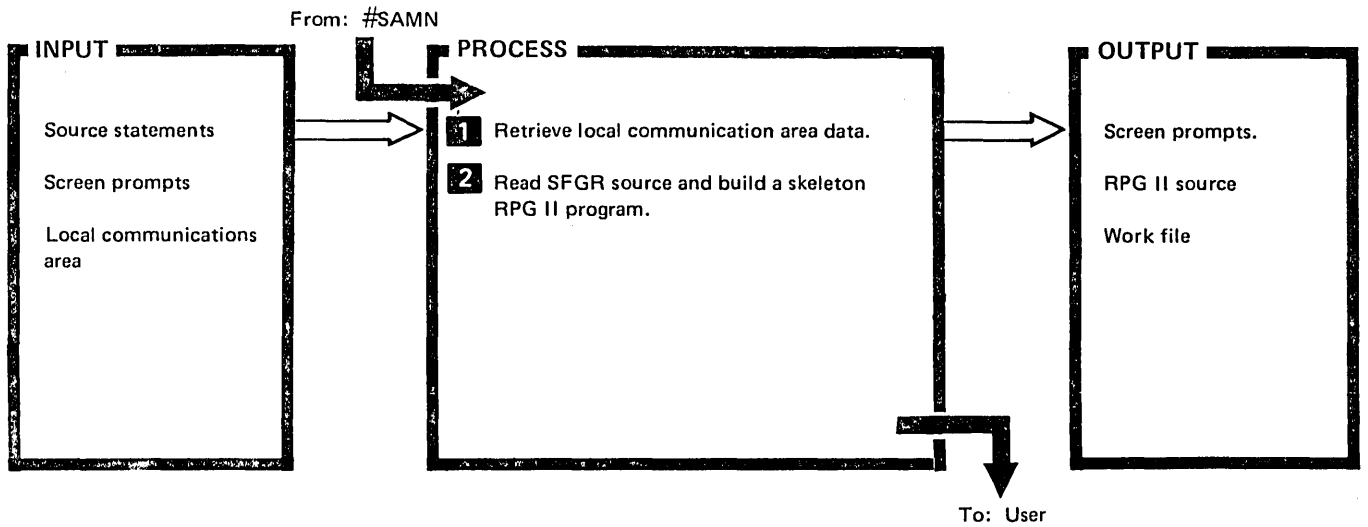
DESCRIPTION	MODULE/ ROUTINE
<ol style="list-style-type: none"> 1 Display, create, or update screen. 2 Based on user selection, turn on external switch 1 or 2. 	#SAMA

Diagram 2-5. SDA Interactive Menu Build (#SAMA)



DESCRIPTION	MODULE/ ROUTINE
<ol style="list-style-type: none"> 1 Read the menu control statement for SDA and perform syntax and diagnostic checking on parameter data. 2 The create function displays a blank skeleton menu to be filled in by the user (if desired). Screens follow to allow the user to enter commands associated with each option number on the skeleton menu. Update rebuilds the same screens from message member source and allows the user to modify it. 	#SAMB

Diagram 2-6. SDA Interactive Menu Build (#SAMB)



DESCRIPTION	MODULE/ ROUTINE
<p>1 Obtain library and SFGR source member to process information from local communications area parameter data.</p> <p>2 When building RPG II source:</p> <ul style="list-style-type: none"> ● Prompt for H and F specification information. ● Build I specifications prompting for decimal positions on numeric fields. ● Build O specifications 	<p>#SARP</p>

Diagram 2-7. SDA RPG Skeleton Program Build (#SARP)

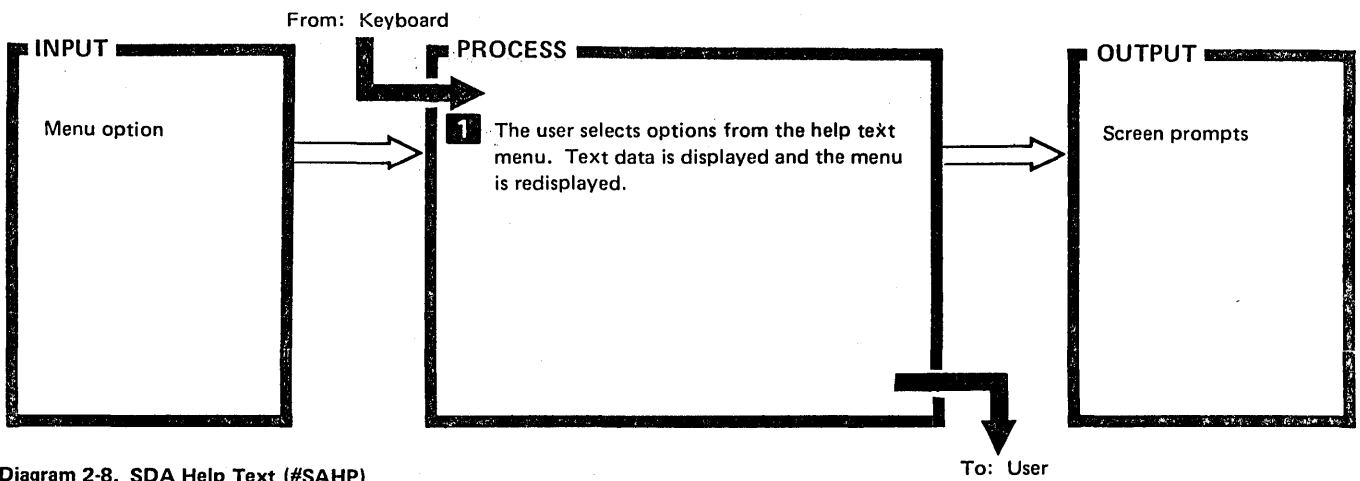


Diagram 2-8. SDA Help Text (#SAHP)

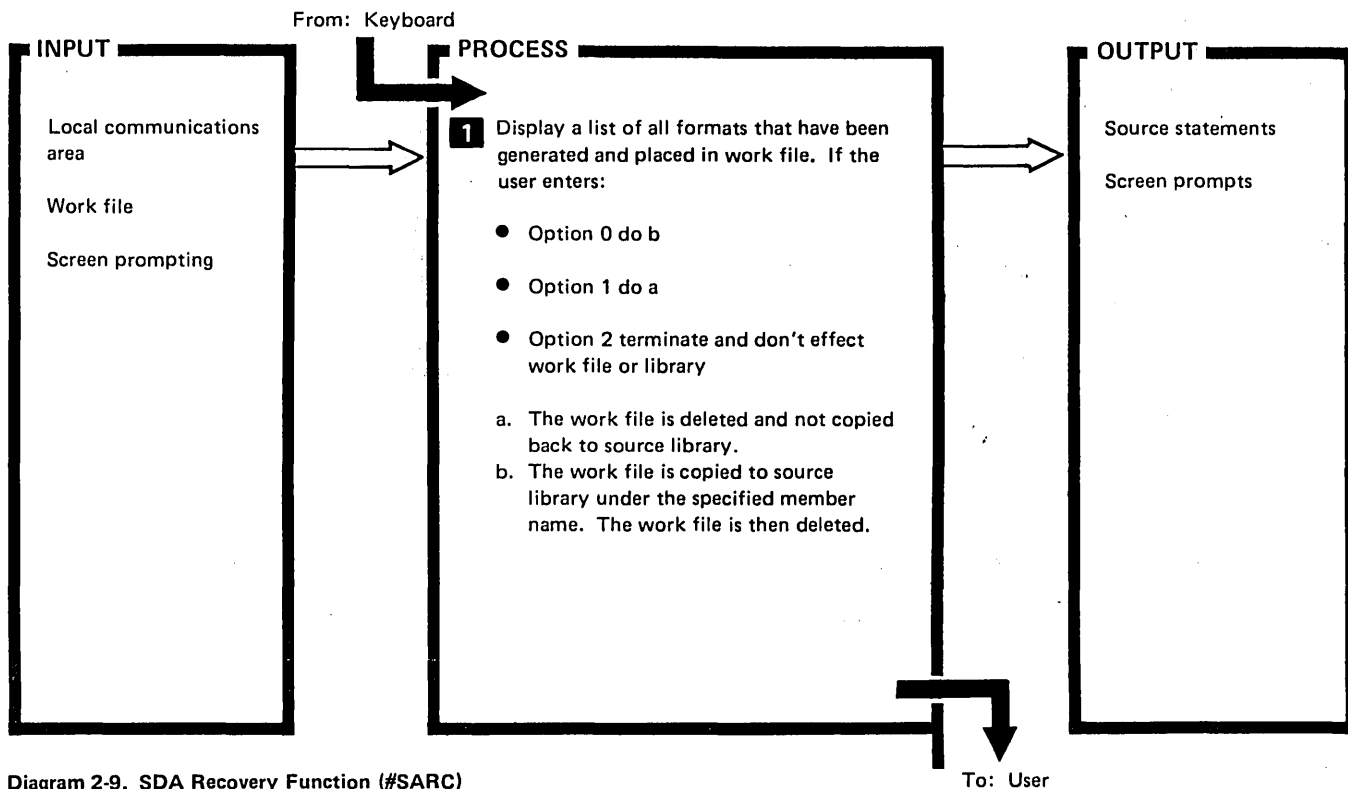


Diagram 2-9. SDA Recovery Function (#SARC)

Program Organization

The storage map (Figure 2-1) in this section is representative of the layout for all SDA modules.

X'0000'

Copyright information
Executable code
DTF Data buffers Work areas Accumulator values (See the Data Areas section of each module listing to determine individual usage.)

Figure 2-1. SDA Storage Map

Directory

This directory identifies the SDA load modules for quick reference to program listings and microfiche. Modules are listed alphabetically by name, and each entry in the directory provides the following information:

- *Module Name* is the symbolic label identifying the module in the program listings and on microfiche.
- *Major Functions* details a brief list of specific functions the module performs within the program.

Module Name	Major Functions
#SACU	Retrieves external switches and local data. Performs create/add/update activity.
#SADP	Displays formats in specified SFGR load member.
#SADS	Retrieves external switches and local data. Performs delete processing activity.
#SAHP	Allows user to select options from the help text menu. Displays text data and redisplay menu.
#SAMA	Displays create or update option menu.
#SAMB	Processes utility control statements for parameters. Performs menu message member source create or updating.
#SAMN	Displays SDA option menu. Processes option to set external switches (U1-U8) to execute next SDA option. Performs diagnostic checking of parameters. Sets up local communications area.
#SARC	Displays list of SFGR formats placed in work file.
#SARP	Retrieves local communications area data. Reads SFGR source and builds a skeleton RPG II program.

Data Areas

LOCAL COMMUNICATIONS AREA

The local communications area passes data between the modules. It is defined using the same displacements as for the procedure parameter save area (macro PPSEQ).

Parameter 1 is in PPSVAR01.

Parameter 2 is in PPSVAR02.

Parameter 3 is in PPSVAR03.

Parameter 4 is in PPSVAR04.

Parameter 5 is in PPSVAR05.

Parameter 6 is in PPSVAR06.

Parameter 7 contains the time of day in displayable format.

Parameter 8 contains the date in displayable format.

Diagnostic Aids

Figure 2-2 lists the message identification codes (MICs) and the modules that diagnose and issue the corresponding error messages.

MESSAGES

Message	Module	Message	Module
0001	#SAMN	0025	#SAMN
0002	#SAMN	0026	#SAMN
0003	#SADS	0027	#SAMB
0004	#SACU	0028	#SAMB
0005	#SAMN	0029	#SAMB
0008	#SAMN	0030	#SAMB
0009	#SAMN	0031	#SAMB
0010	#SACU	0032	#SAMB
0011	#SACU	0033	#SAMB
0012	#SACU	0034	#SAMB
0013	#SACU	0035	#SAMB
	#SADS	0036	#SAMB
0014	#SACU	0037	#SARP
	#SADS	0038	#SAMB
0015	#SAMN	0039	#SAMB
0016	#SAMN	0040	#SACU
0018	#SAMN	0041*	#SACU
0019	#SAMN	0047	#SAMN
0020	#SAMN	0048	#SAMN
0021	#SACU	0049*	#SACU
	#SADS	0050	#SAMB
	#SARP		
0022	#SACU		
	#SADP		
	#SAHP		
	#SAMA		
	#SAMB		
	#SAMN		
	#SARC		
	#SARP		

Note: All messages are SYSLOG messages except those annotated with an asterisk. The asterisked messages are displayed messages.

Figure 2-2. SDA Message-to-Module Cross Reference

PROCEDURES

SDA Mainline

This procedure executes SDA mainline where:

- Parameter 1: source member name to process
- Parameter 2: SDA inlib
- Parameter 3: SFGR load member name
- Parameter 4: yes/no to SFGR printing
- Parameter 5: SDA outlib
- Parameter 6: SFGR outlib

```
// MEMBER PROGRAM1-#SA#M1,USER1-#SA#M1
// IF ?1R'0001'?/ SWITCH 00000000
// IFF ?2'#LIBRARY'?/#LIBRARY SWITCH 00000000
// IFF ?4'YES'?/ SWITCH 00000000
// IF ?5?/ IF ?6?/ INCLUDE SDA9 ?1?,?2?,?3?,?4?,?2?,?2?
// IFF ?5?/ IF ?6?/ INCLUDE SDA9 ?1?,?2?,?3?,?4?,?5?,?2?
// IF ?5?/ IFF ?6?/ INCLUDE SDA9 ?1?,?2?,?3?,?4?,?2?,?6?
// IFF ?5?/ IFF ?6?/ INCLUDE SDA9 ?1?,?2?,?3?,?4?,?5?,?6?
```

SDADROP

This procedure removes SDA from #library or a specified library.

```
// REMOVE #SA,ALL,LIBRARY,?1'#LIBRARY'?
// REMOVE SDA,ALL,LIBRARY,?1'#LIBRARY'?
```

SDAH

This program executes program #SAHP which is the help function.

```
// LOAD #SAHP
// RUN
```

SDALOAD

This procedure loads SDA into the system.

```
// MEMBER PROGRAM1-#SA#M1,USER1-#SA#M1
// * 0049
```

SDASAVE

This procedure saves SDA from #library or a specified library.

```
// IF DATA1-SDA DELETE SDA,I1
// FROMLIBR SDA,ALL,LIBRARY,,,999,PPUTIL (SEE SDADROP)
// FROMLIBR #SA,ALL,LIBRARY,SDA,,ADD,PPUTIL (SEE SDADROP)
```

SDA1

This procedure executes program #SACU which creates, adds, and updates total formats.

```
// LOAD $FBLD
// RUN
// FILE LABEL-#SD.?WS?,ATTRIB-D,BLOCKS-157,RECL-80,RETAIN-T
// END
// LOAD #SACU
// FILE NAME-IOWORK,RETAIN-T,LABEL-#SD.?WS?
// RUN
// IF SWITCH1-1 CANCEL
// IF DATAF1-#SD.?WS? DELETE #SD.?WS?,F1
// IFF ?3?/ INCLUDE SDA6 ?1?,?2?,?3?,?4?,?5?,?6? SFGR
// SWITCH 00000000
// IF SWITCH8-0 RESET SDA ?1?,?2?,?3?,?4?,?5?,?6?
```

SDA2

This procedure executes program #SADS which is the delete format and SEU generate option.

```
// IF SWITCH3-1 SEU ?1?,S,#SA@DS,80,?5?
// IF SWITCH4-1 LOAD #SADS
// IF SWITCH4-1 FILE NAME-IOWORK,RETAIN-T,LABEL-#SD.?WS?,BLOCKS-157
// IF SWITCH4-1 RUN
// IF SWITCH1-1 CANCEL
// IF DATAF1-#SD.?WS? DELETE #SD.?WS?,F1
// IFF ?3?/ INCLUDE SDA6 ?1?,?2?,?3?,?4?,?5?,?6? SFGR
// SWITCH 00000000
// IF SWITCH8-0 RESET SDA ?1?,?2?,?3?,?4?,?5?,?6?
```

SDA#

This procedure executes program #SADP which is the format display function.

```
// IFF ?2?/#LIBRARY LIBRARY NAME-?2?  
// LOAD #SADP  
// RUN  
// SWITCH 00000000  
// IF SWITCH8-0 RESET SDA ?1?,?2?,?3?,?4?,?5?,?6?
```

SDA5

This procedure executes program #SAMA (interactive menu option select) and SAMB (interactive menu create or update).

```
// MEMBER PROGRAM1-#SA#M1,USER1-#SA#M1  
// IF ?1'SCRNSP'?/ SWITCH 00000000 SET SWITCHES TO 00000000  
// ELSE SWITCH 00000000 BEFORE SAMA  
// LOAD #SAMA  
// RUN  
// IF SWITCH1-0 IF SWITCH2-0 RESET SDA ?1?,?2?,?3?,?4?,?5?,?6?  
// IFF ?2?/?5? IF SWITCH2-1 LOAD $MAINT  
// IFF ?2?/?5? IF SWITCH2-1 RUN  
// IFF ?2?/?5? IF SWITCH2-1 COPY FROM-?2?,TO-?5?,LIBRARY-S,NAME-?1?##,RETAIN-R  
// IFF ?2?/?5? IF SOURCE-'?1?DT,?2?' IF SWITCH2-1 COPY FROM-?2?,TO-?5?,RETAIN-R,  
// IFF ?2?/?5? IF SOURCE-'?1?DT,?2?' IF SWITCH2-1 LIBRARY-S,NAME-?1?DT  
// IFF ?2?/?5? IF SWITCH2-1 END  
// LOAD #SAMB  
// RUN  
// MENU INPMSG-?1?##,MENMSG-?1?DT,INLIB-?5?,REPLACE-NO  
// END  
// IFF SOURCE-'?1?DT,?5?' BLDMENU ?1?, ,?5?,?5?,REPLACE  
// IF SOURCE-'?1?DT,?5?' BLDMENU ?1?,?1?DT,?5?,?5?,REPLACE  
// SWITCH 00000000  
// IF SWITCH8-0 RESET SDA ?1?,?2?,?3?,?4?,?5?,?6?
```

SDA6

This procedure executes SFGR.

```
// IF ?1'SCRNSPEC'?/ SWITCH 00000000  
// * 0041  
// LOAD $SFGR  
// RUN  
// LOADMBR REPLACE-YES,NAME-?3?  
// INOUT OUTLIB-?6?,INLIB-?5?,PRINT-?4?  
// CREATE SOURCE-?1?,NUMBER-32  
// END  
// SWITCH 00000000  
// IF SWITCH8-0 RESET SDA ?1?,?2?,?3?,?4?,?5?,?6?
```

SDA7

This procedure executes program #SARP which is the RPG II skeleton program create option.

```
// LOAD #SARP
// FILE NAME-IOWORK,RETAIN-T,LABEL-#SD.?WS?,BLOCKS-157
// RUN
// IF SWITCH1-1 CANCEL
// IF DATAF1-#SD.?WS? DELETE #SD.?WS?,F1
// SWITCH 00000000
// IF SWITCH8-0 RESET SDA ?1?,?2?,?3?,?4?,?5?,?6?
```

SDA8

This procedure executes program #SARC which is the recovery procedure data set.

```
// SWITCH 10000000 CAUSES LOCAL AREA TO BE SET UP
// LOAD #SAMN
// RUN
// SWITCH 00000000
// LOAD #SARC
// FILE NAME-IOWORK,RETAIN-T,LABEL-#SD.?WS?
// RUN
// IF SWITCH1-1 CANCEL
// IF DATAF1-#SD.?WS? DELETE #SD.?WS?,F1
```

SDA9

This procedure executes SDA mainline and routes to the subprocedure shown below where:

- Parameter 1: source member name to process (defaults to SCRNSPEC)
- Parameter 2: SDA inlib (defaults to #library)
- Parameter 3: SFGR load member name
- Parameter 4: yes/no to SFGR printing (defaults to yes)
- Parameter 5: SDA outlib (defaults to parameter 2)
- Parameter 6: SFGR outlib (defaults to parameter 2)

```
// IF DATAF1-#SD.?WS? INCLUDE SDA8 ?1?,?2?,?3?,?4?,?5?,?6?
// SWITCH 00000000
// LOAD #SAMN
// RUN
// IF SWITCH1-1 INCLUDE SDA1 ?1?,?2?,?3?,?4?,?5?,?6? CREATE,ADD,UP
// IF SWITCH2-1 INCLUDE SDA2 ?1?,?2?,?3?,?4?,?5?,?6? DELETE,SEU
// IF SWITCH3-1 INCLUDE SDA3 ?1?,?2?,?3?,?4?,?5?,?6? DISPLAY
// IF SWITCH4-1 INCLUDE SDA5 ?1?,?2?,?3?,?4?,?5?,?6? MENU
// IF SWITCH5-1 INCLUDE SDA7 ?1?,?2?,?3?,?4?,?5?,?6? RPG
// IF SWITCH8-1 CANCEL
```


SDA Index

#SACU (performs create/add/update activity)
diagram 2-4
directory 2-13

#SADP (displays formats in specified SFGR load member)
diagram 2-6
directory 2-13

#SADS (performs delete)
diagram 2-5
directory 2-13

#SAHP (allows user to select options from the help text menu)
diagram 2-10
directory 2-13

#SAMA (displays create or update option menu)
diagram 2-7
directory 2-13

#SAMB (displays the blank or updated menu formats)
diagram 2-8
directory 2-13

#SAMN (displays SDA option menu)
diagram 2-3
directory 2-13

#SARC (displays list of SFGR formats placed in work file)
diagram 2-11
directory 2-13

#SARP (reads SFGR source and builds a skeleton RPG program)
diagram 2-9
directory 2-13

Allows user to select options from the help text menu
diagram 2-10
directory 2-13

Displays create or update option menus
diagram 2-7
directory 2-13

Displays formats in specified SFGR load member
diagram 2-6
directory 2-13

Displays SDA option menu
diagram 2-3
directory 2-13

Displays a list of SFGR formats placed in work file
diagram 2-11
directory 2-13

Performs create/add/update activity
diagram 2-4
directory 2-13

Performs delete
diagram 2-5
directory 2-13

Reads SFGR source and builds a skeleton RPG program
diagram 2-9
directory 2-13

Introduction

The data file utility (DFU) portion of the utilities program product provides a convenient method of creating, maintaining, displaying, and listing data files.

DFU operates interactively, allowing the System/34 user to:

- Create indexed data files.
- Update indexed data files.
- Inquire into indexed data files.
- List indexed, sequential, or direct data files. Data files can be sorted prior to being listed. The following capabilities are available within the list function:
 - Extract and print data from a related master file.
 - Select records for printing based on field value.
 - Calculate and print a result field by combining, arithmetically, fields and/or constant values.

HOW THE PROGRAM WORKS

The DFU user signs on by entering one of the DFU commands: ENTER, UPDATE, INQUIRY, or LIST. DFU then calls setup or execution depending on the existence or nonexistence of the DFU format named by the second parameter of the command. The DFU format describes the processing to be performed. If the DFU format name does not exist as a subroutine member in the library, DFU setup is called to create the format. If the DFU format named on the initial command exists as a subroutine member in the library, the corresponding execution phase is called.

Note: The library referred to in this discussion is the system library, unless a user library is specified on the DFU command.

PHYSICAL CHARACTERISTICS

DFU is composed of 36 library members. The *Directory* in this chapter explains the 14 executable load modules.

Nonexecutable Load Modules

#DF#MG	DFU message load member.
#DF@24	display screen load member.
#DFS0-#DFS10	reserved for expansion to keep DFU a fixed size in the library.

Procedures

ENTER	allows creation of a data file.
UPDATE	allows alteration of an existing data file.
INQUIRY	allows display of a data file.
LIST	allows printing of a data file.
DFULOAD	loads DFU into a specified library.
DFUDROP	deletes DFU from a specified library.
DFUSAVE	saves DFU on diskette.
#DFST	calls modules required for a list with sort.
#DFMP	calls job setup modules.

INPUT/OUTPUT

After the user signs on DFU, input to set up and execution consists of:

Setup

- Source member of RPG II file description and input specifications describing the files to be processed.
- Source member of saved DFU specifications.
- Operator responses to DFU prompts.

Execution

- DFU format description (subroutine members) describing the processing to be performed.
- Display format load member describing execution time output displays.
- Data file(s) to be updated, displayed, listed.
- Operator responses to DFU prompts.

Output from DFU setup and execution consists of:

Setup

- Source member of saved DFU specifications.
- DFU format description (subroutine member) describing the processing to be performed.
- Display format load member describing execution time output displays.
- SYSLIST output of DFU attributes and specifications.
- Source member of display format specifications describing execution time displays.

Execution

- Created or updated data file.
- Printed output of processing performed.
- Displayed data records at the display station.

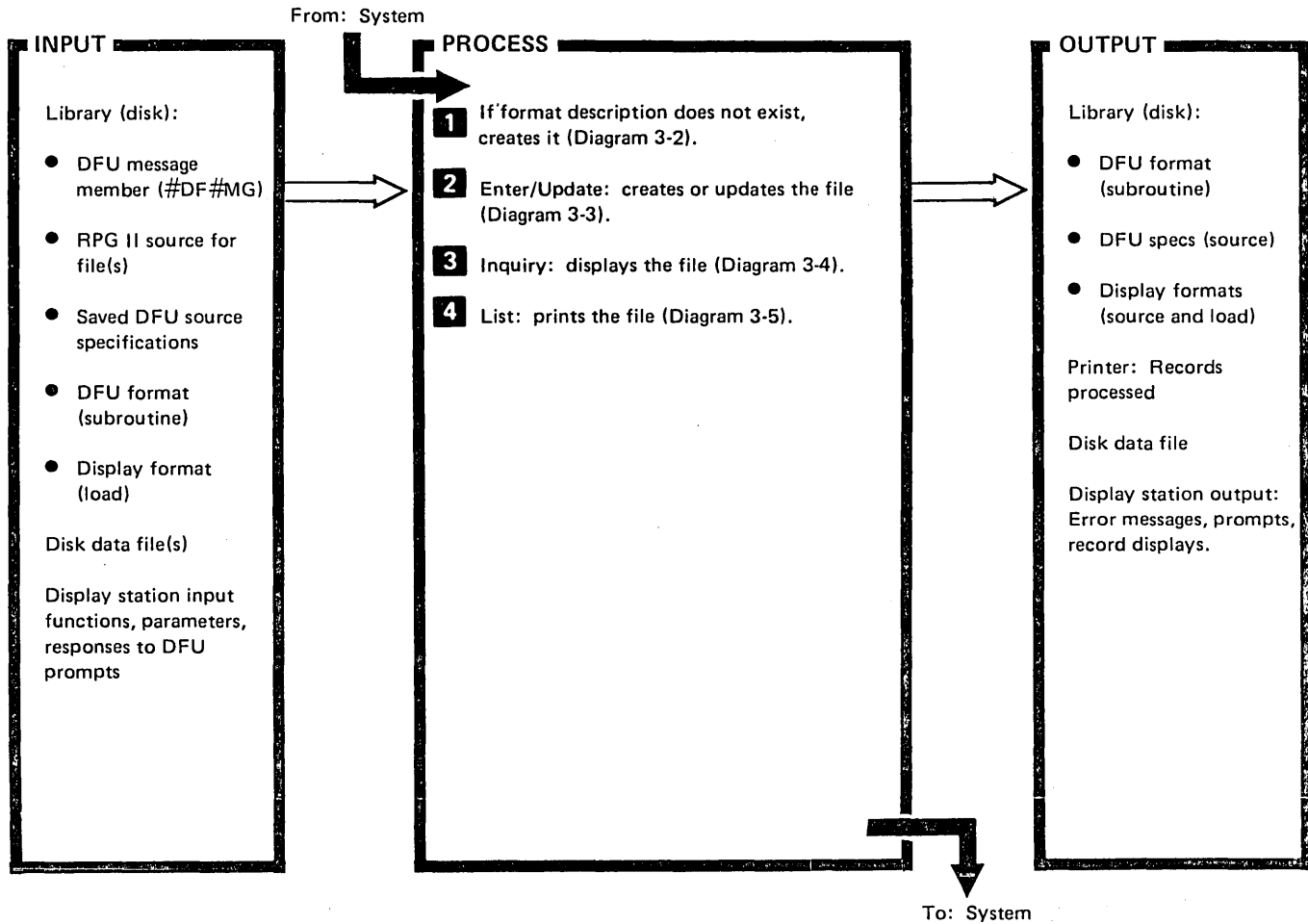
I/O STORAGE REQUIREMENTS

DFU requires 14K (14,336) bytes of main storage.

SYSTEM CONFIGURATION

DFU runs on all models of the System/34. DFU requires the Sort program when listing with sort.

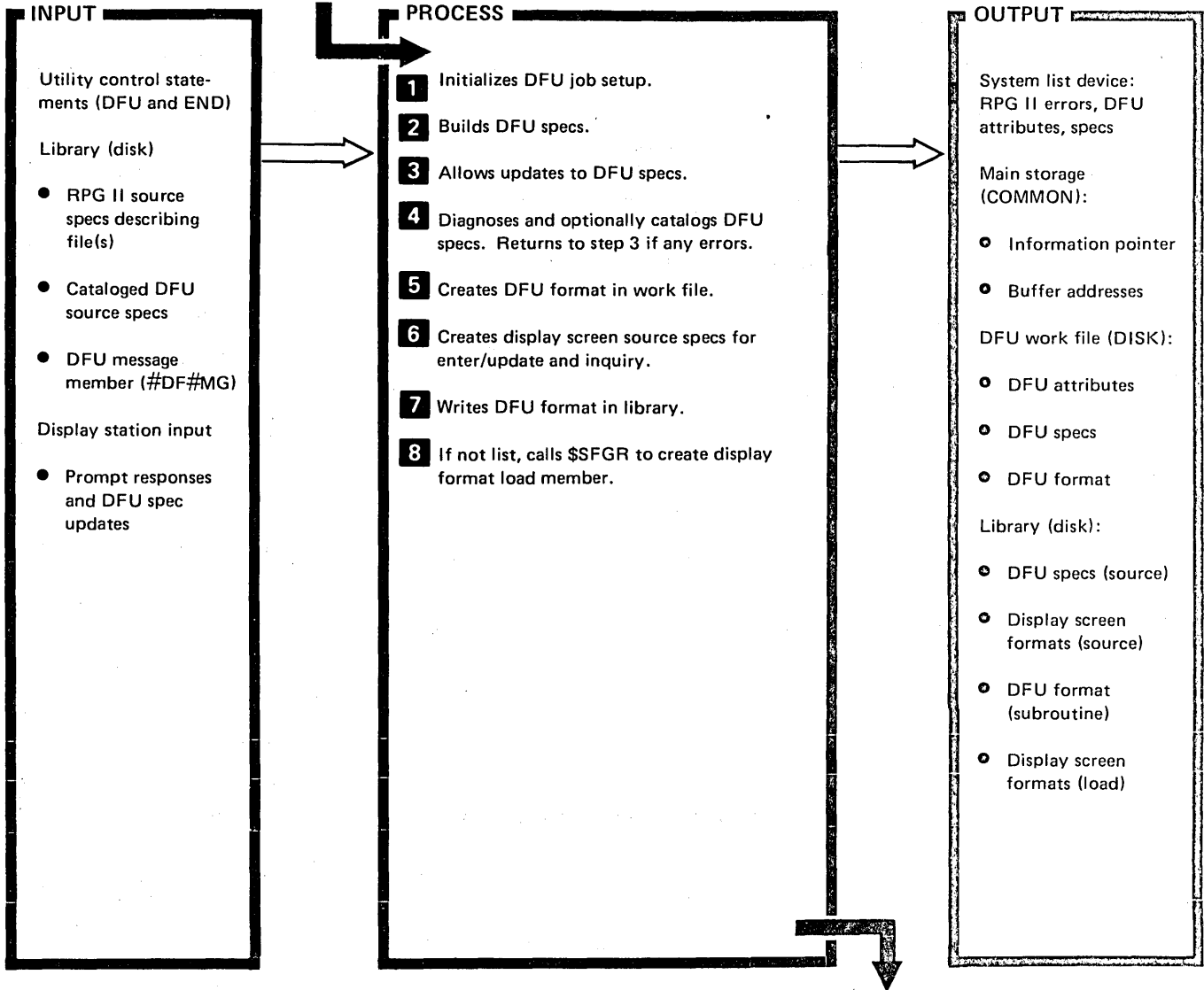
Method of Operation



DESCRIPTION	MODULE/ROUTINE
<p>1 The DFU format describes the file processing required. If it does not exist as a subroutine member in the library at job initiation, the DFU job setup modules are called to create it. For enter/update and inquiry, corresponding display format source specifications are created to describe display station screens. These are stored in the library under the default name generated by DFU, unless a name is specified on the initial command. The default name is a combination of: #DF + display station ID + 1 (if not in inquiry mode), or 2 (if in inquiry mode).</p> <p>DFU then calls the SSP utility program, \$SFGR, to convert these source specifications to the load member form required for execution; the load member has the same name as the DFU format. The source input specifications are then removed from the library, unless a name was specified on the initial command, indicating they were to be saved.</p>	#DFMP
<p>2 The operator can create or update records in a file. Fields can be totaled, and new, updated, or deleted records can be printed.</p>	#DFEI, #DFUD
<p>3 The operator can select any prime data area record in the file for display at the display station. Records can be selected by record key or by scrolling through the file. Any displayed record can be printed.</p>	#DFIN
<p>4 The file is listed as specified in the DFU format. The file can be sorted before listing data for the list can be extracted from a related master file.</p>	#DFLS

Diagram 3-1. DFU Overview

From: DFU Overview
(Diagram 3-1)



To: Execution Step for
Enter/Update (Diagram 3-3),
Inquiry (Diagram 3-4), or
List (Diagram 3-5)

Diagram 3-2 (Part 1 of 2). DFU Job Setup

DESCRIPTION	MODULE/ ROUTINE
<p>1 Allocates a scratch work file on disk (DFUWORKA), initializes COMMON in main storage, and converts RPG II specifications to DFU attributes in the work file. If a master file is present, prompts for the RPG II member and creates the corresponding master file attributes. If DFU specifications are saved, reads them and puts them in the work file.</p>	#DFMP
<p>2 Prompts the operator for a job description and builds DFU specifications for enter, update, inquiry, or list.</p>	#DFQL— list #DFQI— inquiry #DFQE— enter/update
<p>3 Displays DFU attributes or specifications and allows updating of DFU specifications.</p>	#DFUP
<p>4 Diagnoses the DFU specifications in the work file. Saves the DFU specifications, if requested, when they are error free.</p>	#DFDI
<p>5 Merges DFU attributes/specifications and creates DFU format in the work file.</p>	#DFFB
<p>6 If enter/update or inquiry, creates source specifications describing display screens for the job.</p>	#DFWS
<p>7 Writes the DFU format as a subroutine member in the library and calls end of job to terminate the job setup step.</p>	#DFWS
<p>8 If enter/update or inquiry, called (as a job step) to convert display screen source to display screen load format.</p>	\$\$FGR

Diagram 3-2 (Part 2 of 2). DFU Job Setup

From: DFU Overview (Diagram 3-1) if
format existed, or Job Setup
(Diagram 3-2) if format did not exist.

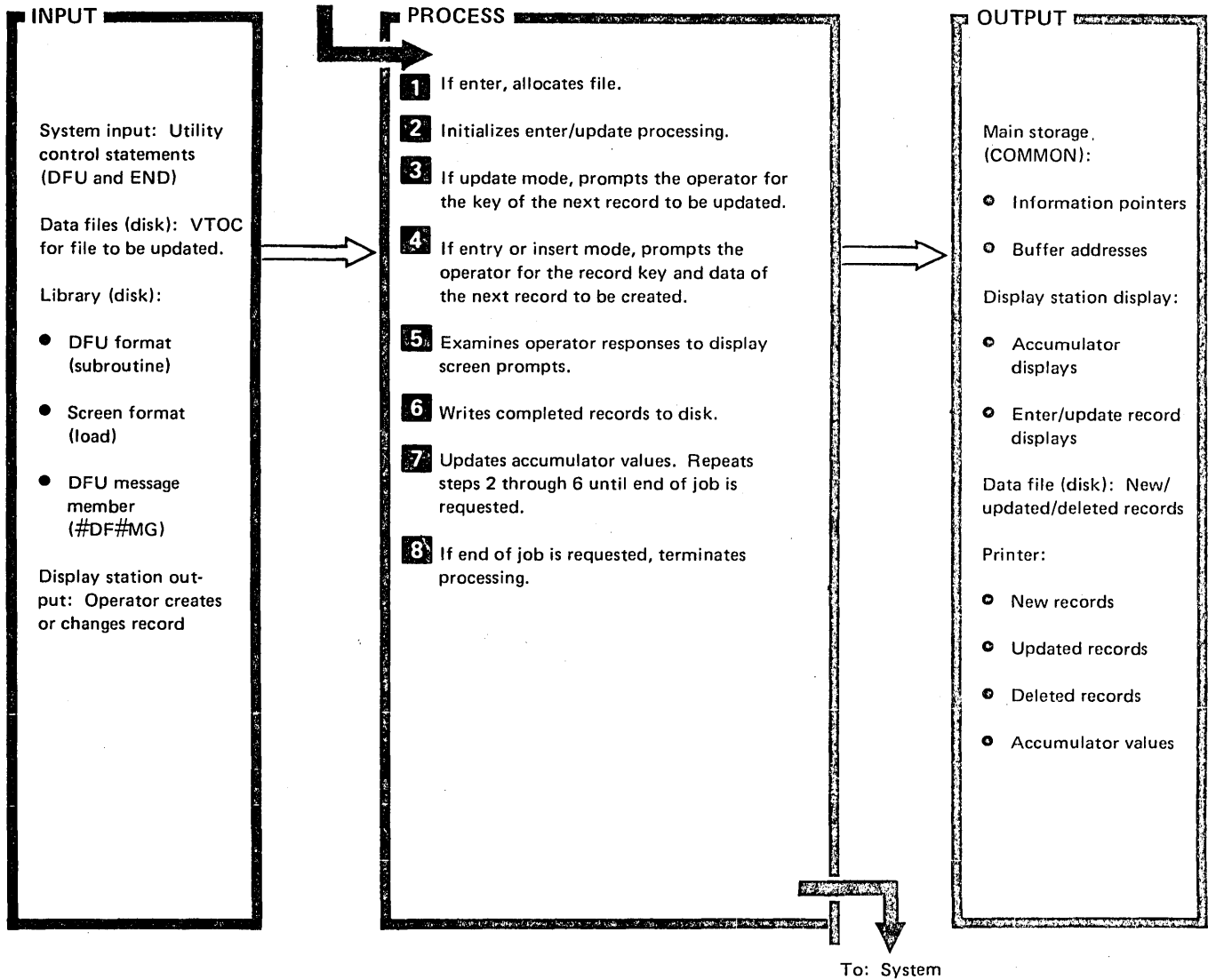
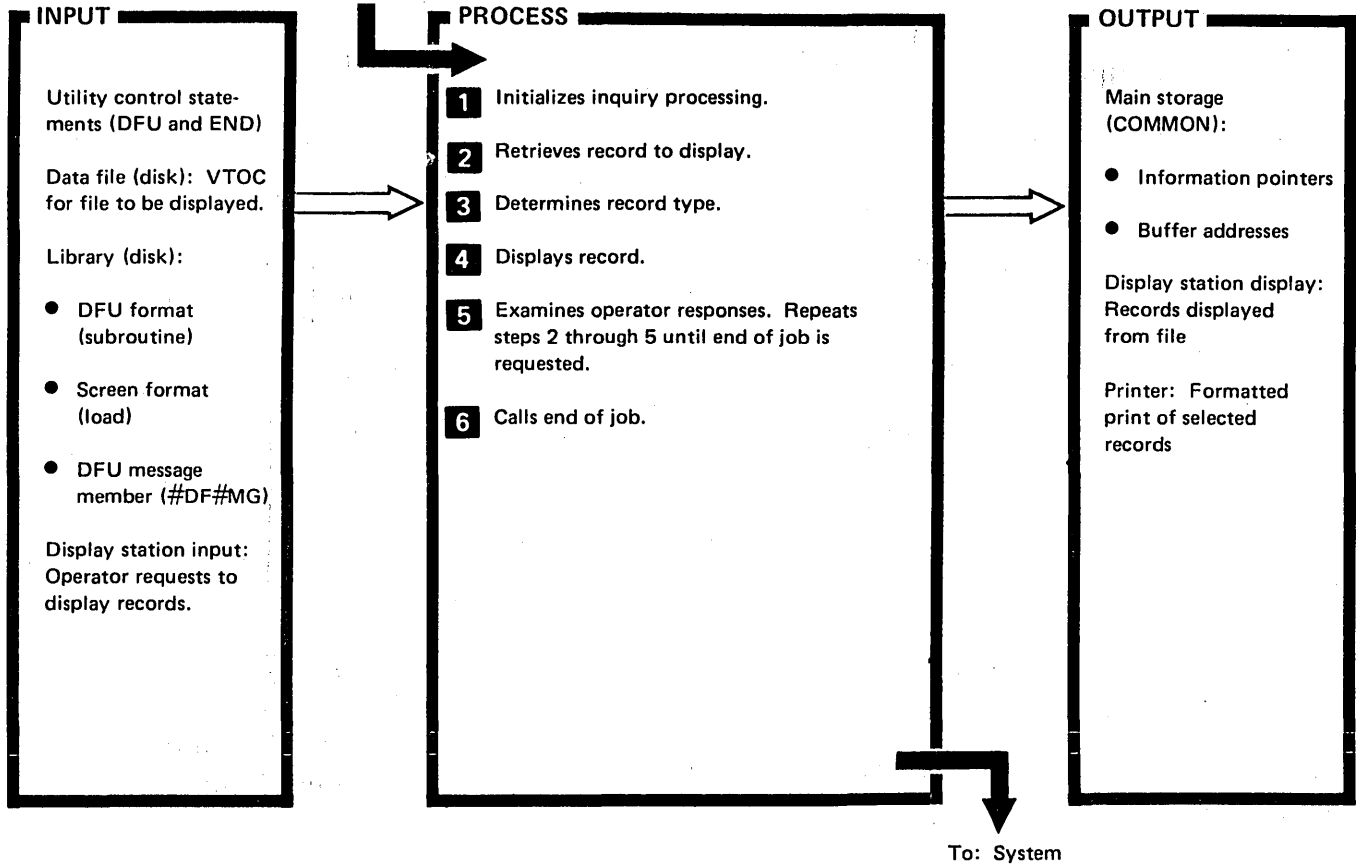


Diagram 3-3 (Part 1 of 2). DFU Enter/Update Execution

DESCRIPTION	MODULE/ ROUTINE
1 Allocates and creates VTOC entry for a new file.	#DFEX
2 Reads and processes utility control statements, allocates buffers for all files, and locates and reads the DFU format. Allocates and opens the data file and display station and initializes the COMMON area.	#DFEX
3 In update, prompts the operator for the key of the record to be updated. This key must exist in the file.	#DFUD
4 If creating records, the record key and associated data is prompted for at the same time. The record key must not exist in the file.	#DFEI
5 Interprets all operator function control and command function key responses.	#DFEI (entry/insert) #DFUD (update)
6 When a record is completed, writes new/updated record or deletes record. If requested in the DFU format, the record is printed using printer data management. The printer is opened the first time it is requested.	#DFEI (entry/insert) #DFUD (update)
7 Changes or adds to the accumulator hold areas. If requested by the operator, displays batch accumulators; they are also printed if the printer is opened.	#DFEI (entry/insert) #DFUD (update)
8 Displays new/updated/deleted record counts, and displays batch and final accumulators. Prints this information if the printer is opened. Calls end of job.	#DFEI (entry/insert) #DFUD (update)

Diagram 3-3 (Part 2 of 2). DFU Enter/Update Execution

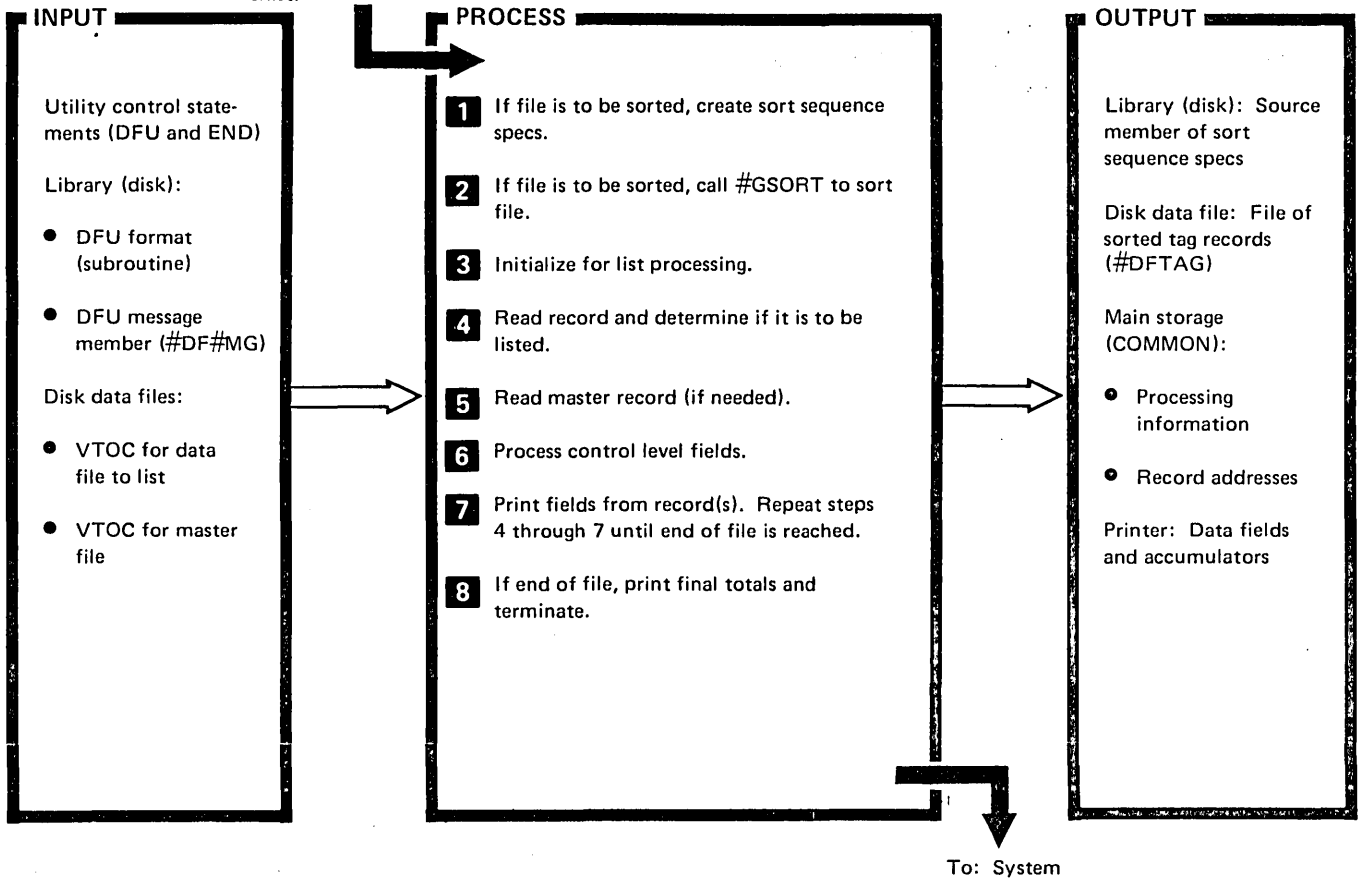
From: DFU Overview (Diagram 3-1) if format existed, or Job Setup (Diagram 3-2) if format did not exist.



DESCRIPTION	MODULE/ROUTINE
1 Reads and processes utility control statements, allocates buffers for all files, and locates and reads the DFU format. Allocates and opens the data file and initializes the execution COMMON area.	#DFEX
2 Initially retrieves the first record in the file, later retrieves the record specified by the operator. The operator request records by a specific record key, or scrolling the file.	#DFIN
3 Examines the record information sectors of the format description to check if the retrieved record matches any defined record types. If a match is found, reads the field information sectors that indicate the fields to display for this record type.	#DFIN
4 Displays the record as described in the DFU format.	#DFIN
5 Examines operator responses, prints the record if requested (via printer data management), and allocates and opens the printer on first print request. Determines operator's next record key request.	#DFIN
6 When operator requests end of job, calls end of job to terminate the job and close all files.	#DFIN

Diagram 3-4. DFU Inquiry Execution

From: DFU Overview (Diagram 3-1) if format existed, or Job Setup (Diagram 3-2) if format did not exist.



DESCRIPTION	MODULE/ROUTINE
1 If file is to be sorted before listing, this module is called via OCL to create sort sequence specifications from information in the DFU format.	#DFSB
2 If file is to be sorted before listing, the sort module is called via OCL to create a file of sorted tag records. This file has RETAIN-J; so it exists only for this job.	#GSORT
3 Reads and processes utility control statements, allocates buffers for all files. Allocates and opens the printer and all data files. Locates and reads the DFU format. Initializes the COMMON area. <i>Note:</i> All printing is done via printer data management.	#DFEX
4 If sorted, reads a tag record and then the associated data record. If not sorted, reads next record from data file. Determines record type. If found, determines if the record satisfies record selection criteria (if specified).	#DFLS
5 Reads the master record if necessary.	#DFLS
6 Determines if a control field changed value. If so, prints and updates accumulators.	#DFLS
7 If detail records are printed, formats and prints data including generated result fields.	#DFLS
8 Prints final accumulator and calls end of job.	#DFLS

Diagram 3-5. DFU List Execution

Program Organization

This section contains module flow charts (Figures 3-1 through 3-3) to define interaction within the program. Additionally, storage maps (Figures 3-4 and 3-5) for the DFU job setup and job execution are included in this section.

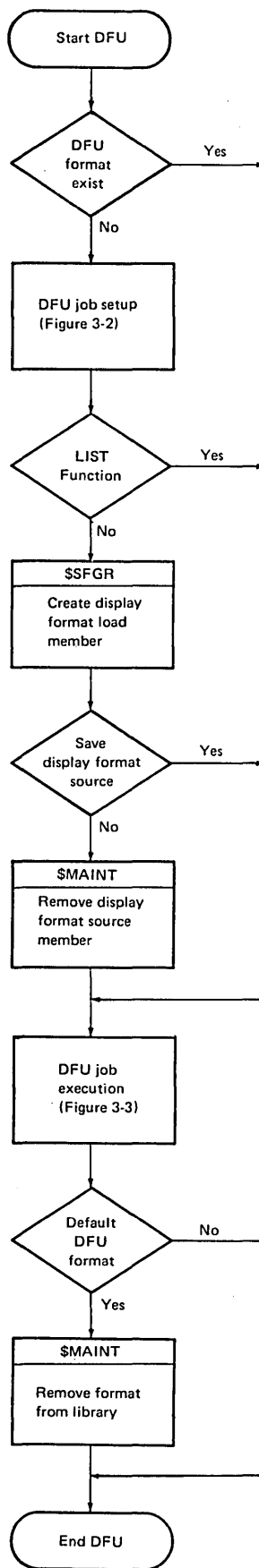


Figure 3-1. DFU Module Flow Overview

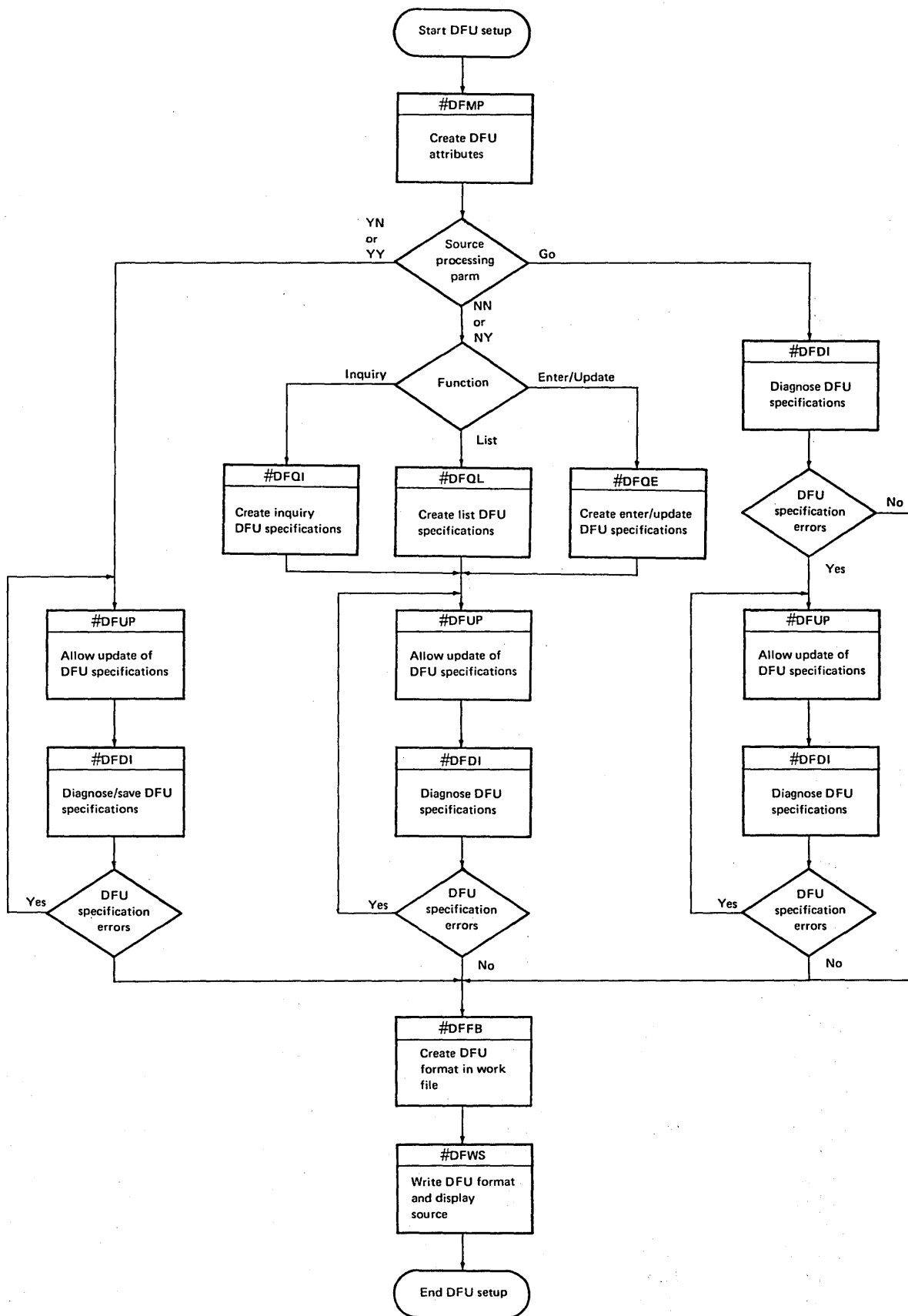


Figure 3-2. DFU Job Setup Module Flow

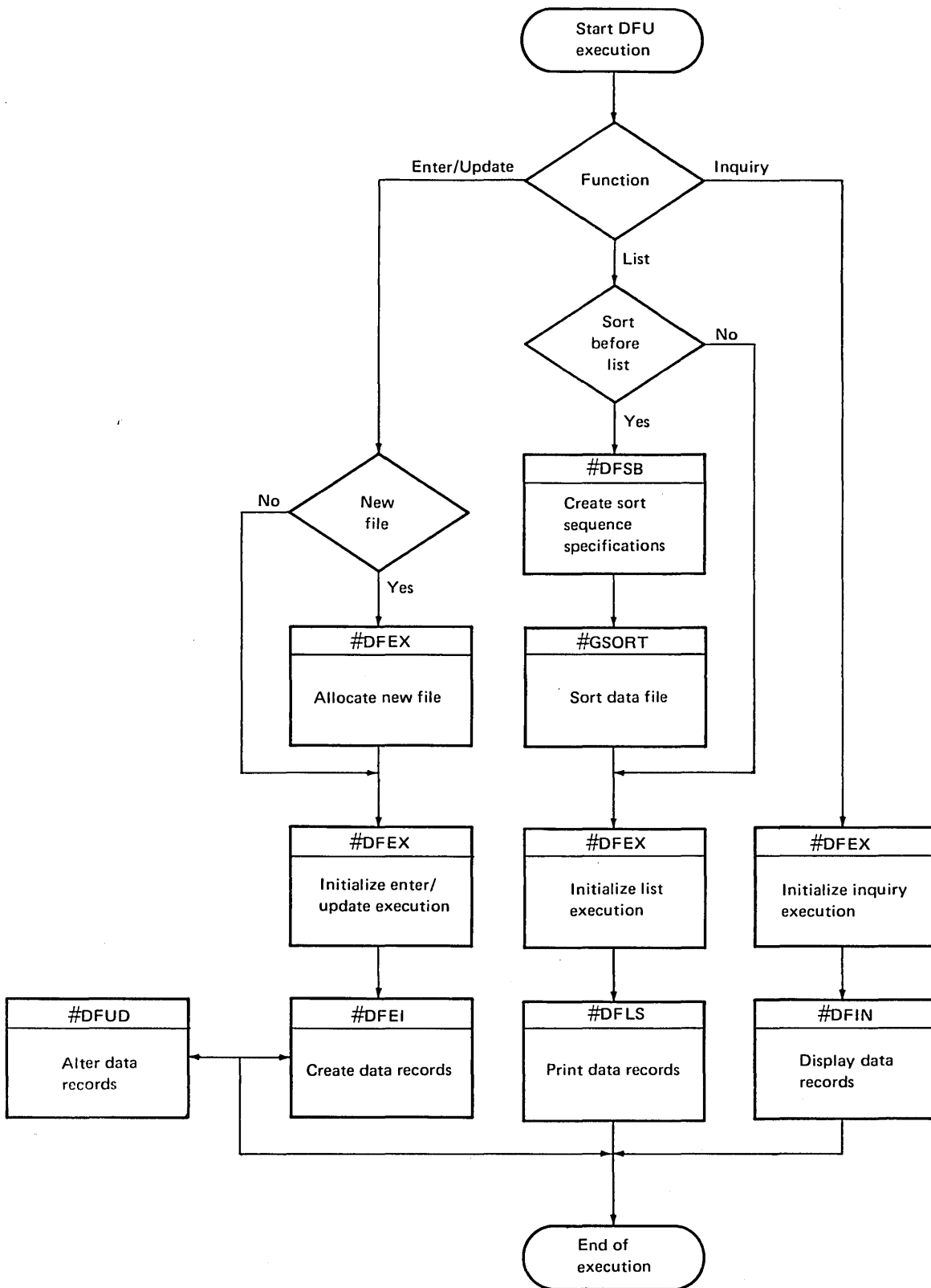


Figure 3-3. DFU Job Execution Module Flow

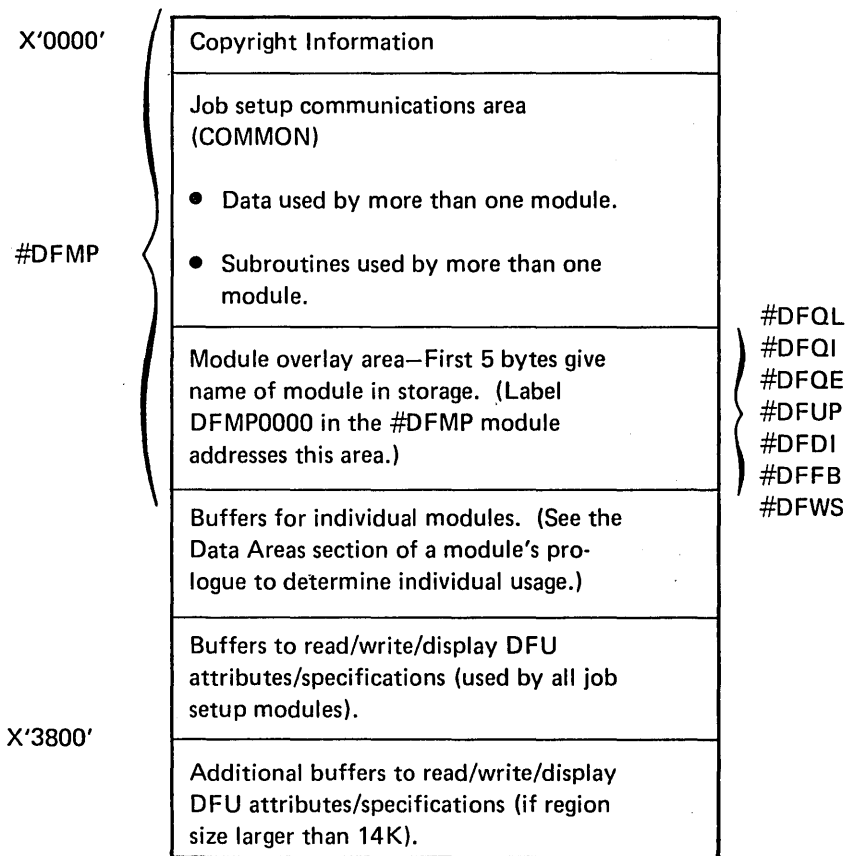
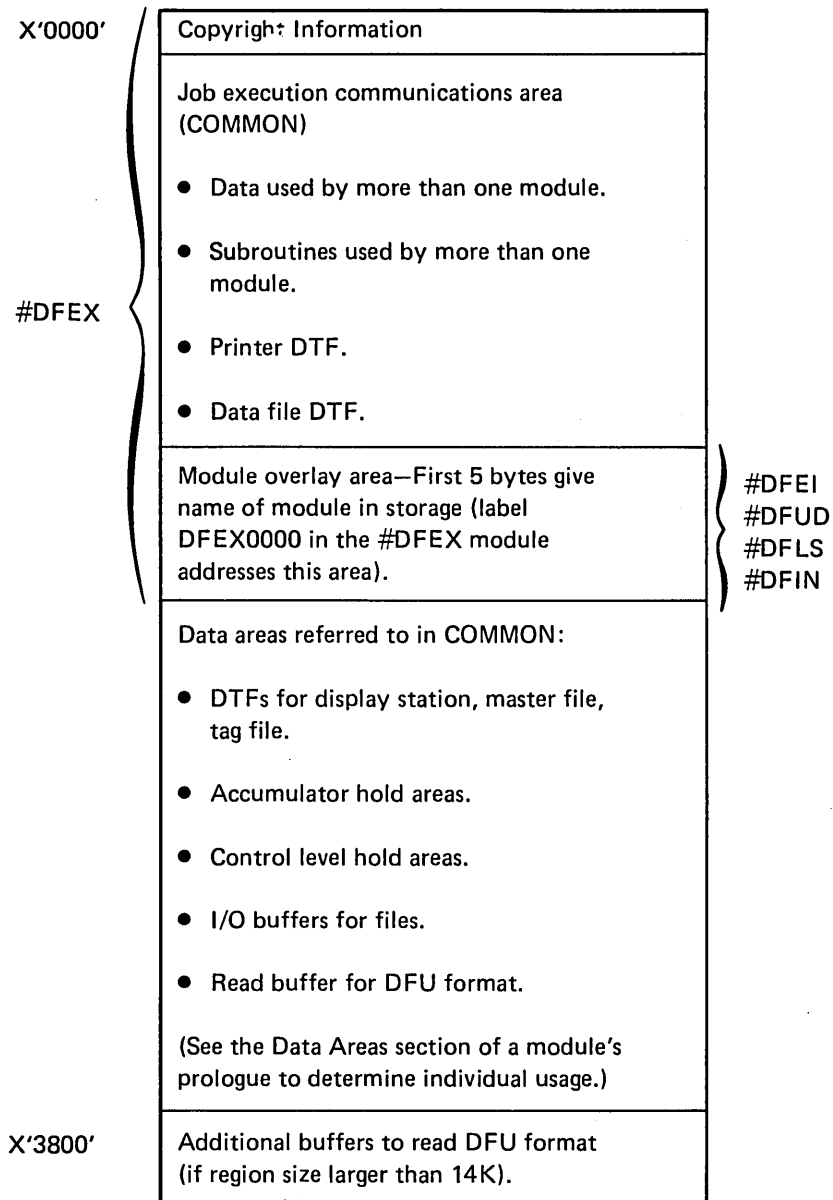


Figure 3-4. DFU Job Setup Storage Map



Note: The module to create sort sequence specifications, #DFSB, is loaded via OCL as a separate job step. Since the module contains no overlays, it is not described in a storage map.

Figure 3-5. DFU Job Execution Storage Map

Directory

This directory identifies the DFU executable load modules for quick reference to program listings and microfiche. Modules are listed alphabetically by name, and each entry in the directory provides the following information:

- *Module Name* is the symbolic label identifying the module in the program listings and on microfiche.
- *Major Functions* is a brief list of specific functions the module performs within the program.

Module Name

Major Functions

- | | |
|-------|--|
| #DFDI | <ul style="list-style-type: none"> ● Diagnoses DFU specifications. ● Saves the DFU source specifications if requested when they are error free. |
| #DFEI | <ul style="list-style-type: none"> ● Prompts the operator for key and data fields for a new record (entry/insert modes). ● Creates the record from operator responses and writes it to disk. Totals accumulator fields. ● Allocates a printer, if necessary, and prints new/updated/deleted records. ● Processes end-of-job request. |
| #DFEX | <ul style="list-style-type: none"> ● Reads and processes DFU utility control statements. ● Allocates a new file for the enter function. ● Initializes job execution communications area (COMMON) for update, inquiry, and list functions. ● Passes control to the execution module to process the file. |
| #DFFB | <ul style="list-style-type: none"> ● Reads DFU specifications from the work file. ● Reads DFU attributes from the work file. ● Merges DFU attribute/specification information to create DFU format in the work file. |

Module Name

Major Functions

- | | |
|-------|---|
| #DFIN | <ul style="list-style-type: none"> ● Retrieves the key and data fields of the requested record from the file and places the fields into the display buffer. ● Puts the requested record out to the display station and gets the next operator response. ● Calls indexed random/sequential input data management to get one of the following: <ul style="list-style-type: none"> – Requested record using key entered by operator. – Next record (Roll Up function control key). – Preceding record (Roll Down function control key). |
| #DFLS | <ul style="list-style-type: none"> ● Prints record if requested. ● Processes end-of-job request. ● Reads records from the file. ● Determines if record is to be listed. ● Reads the master file record if necessary. ● Prints accumulator/control break fields when control fields change. ● Prints detail information from record. |
| #DFMP | <ul style="list-style-type: none"> ● Allocates and opens the display station and work file. ● Initializes the job setup communications area. ● Reads from SYSIN and processes the parameters from the DFU utility control statement. ● Reads and processes the records from the RPG II source member. ● Builds DFU attributes and writes them in the work file. |

Module Name	Major Functions	Module Name	Major Functions
#DFMP (continued)	<ul style="list-style-type: none"> ● Prompts for description of related master file and builds attributes for the master file. ● Reads DFU specifications and writes them in the work file. 	#DFUD	<ul style="list-style-type: none"> ● Allows the operator to review/change previous record processed. ● Prompts the operator for the key of the record to update. ● Displays and allows the operator to change/delete an existing record. ● Totals accumulator field changes. ● Allocates a printer, if necessary, and prints new/updated/deleted records. ● Processes end-of-job request.
#DFQE	<ul style="list-style-type: none"> ● Prompts the operator, checks the syntax of the responses, and builds DFU specifications from the following enter/update displays: <ul style="list-style-type: none"> – General information – Key field specification – Record key description – Record type selection – Data field specification 	#DFUP	<ul style="list-style-type: none"> ● Displays DFU specifications/attributes, including any errors diagnosed. ● Prints DFU attributes/specifications if requested by operator. ● Allows the updating, deletion, and addition of DFU specifications.
#DFQI	<ul style="list-style-type: none"> ● Prompts the operator, checks the syntax of the responses, and builds DFU specifications from the following inquiry displays: <ul style="list-style-type: none"> – General information – Key field specification – Record key description – Record type selection – Data field specification 	#DFWS	<ul style="list-style-type: none"> ● Reads the DFU format from the work file. ● Creates display station source specifications for enter/update/inquiry functions. ● Rewrites DFU format in work file with enter/update/inquiry information inserted. ● Writes DFU format as a subroutine member in the library.
#DFQL	<ul style="list-style-type: none"> ● Prompts the operator, checks the syntax of the responses, and builds DFU specifications from the following list displays: <ul style="list-style-type: none"> – General information – Record key description – Record type selection – Data field specification – Result field specification – Sort field specification – Control field specification – Select field specification 		
#DFSB	<ul style="list-style-type: none"> ● Reads DFU and END DFU utility control statements. ● Finds DFU format and determines if information matches file to be sorted. ● Creates sort sequence specifications as a source member in the library to describe how file is to be sorted. 		

Data Areas

This section discusses the following data areas used by DFU:

- Job setup communications area (COMMON)
- Job setup disk work file
- DFU format description
- Job execution communications area (COMMON)

An explanation is provided for the type of data in each data area. The exact data layout is not provided; rather, references are made to the module listings (microfiche) containing the detailed data layout.

JOB SETUP COMMUNICATIONS AREA (COMMON)

This area passes information between the job setup modules and is initialized by the first setup module (#DFMP). Succeeding setup modules are loaded after the area, and do not overlay any data in the area. Index register 1 addresses this area whenever a new job setup module is loaded. This area contains constants, keyword translations, work file pointers, display station DTF, and several subroutines used by all setup modules. The label COMMON in the following job setup modules can be used to locate the start of this area:

```
#DFMP
#DFQL
#DFQI
#DFQE
#DFUP
#DFDI
#DFFB
#DFWS
```

JOB SETUP DISK WORK FILE (DFUWORKA)

The job setup disk work file stores the following information about the current job:

- DFU attributes—describes the file to be processed (see ATTRFIL in job setup communications area).
- DFU specifications—describes the processing to be performed on the file (see DFUFIL in job setup communications area).

Additionally, the area is used as an intermediate area for creating and storing the DFU format, before it is written as a library subroutine member. This data is written beginning in the first new sector after the DFU specifications.

This disk work file is allocated by the first job setup module (#DFMP) with a minimum size of 50 sectors, and a maximum size of 500 sectors. The first sector address is found at the label ATTRFIL, and the first sector past the end of the work file is found at the label ENDWORKA in the job setup communications area.

DFU FORMAT DESCRIPTION

The DFU format description (DFU format) is a library subroutine member that describes the current job processing. A detailed description of the data in the DFU format can be found starting at the label DFUFORMT in the following DFU modules:

```
#DFFB
#DFWS
#DFSB
#DFEX
#DFEI
#DFUD
#DFIN
#DFLS
```

Generally, the DFU format looks as follows:

- The first sector is the file information sector. It describes the general characteristics of the job: file description and file processing. See label DFUFILE for a description of the file information.
- The second sector is the first (or only) sector of the record information. Each record type in the file to be processed is described in a record identification entry. See label DFURTYPE for a description of a record type entry. Entry RCDFPTR is a pointer to the first sector of field information for a record.
- The third sector is the first field information sector. More than one sector can be required to describe the fields for a given record type. See label DFUFIELD for a description of the field information entries.

JOB EXECUTION COMMUNICATIONS AREA (COMMON)

This area passes information between the various job execution modules and is initialized by the first job execution module (#DFEX). Succeeding execution modules are loaded after the area, and do not overlay any data in the area. Index register 1 addresses this area whenever a new job execution module is loaded.

This area contains constants, keywords, control block pointers, and several subroutines used by all job execution modules. The label COMMON in the following job execution modules can be used to locate the start of this area.

#DFEX
#DFIN
#DFLS
#DFE1
#DFUD

Diagnostic Aids

The following list (Figure 3-6) details the message identification codes (MICs) and the modules that diagnose and issue the displayed error messages (described in the Displayed Messages Guide). Refer to the DFU Reference Manual (Appendix B) for a list of printed messages issued while module #DFMP is diagnosing the RPG II specifications.

MESSAGES

Message	Module	Message	Module	Message	Module	Message	Module
0001*	#DFEI #DFUD	0025	#DFUP #DFMP	0056*	#DFDI #DFQL	0076*	#DFQL #DFDI
0002*	#DFEI		#DFEX		#DFQI	0077*	#DFQL
0003*	#DFEI		#DFEI		#DFQE		#DFDI
0004	#DFEI #DFUD #DFIN #DFLS	0026* 0029*	#DFUD #DFIN #DFUD	0057* 0058*	#DFDI #DFDI #DFQE	0078* 0079*	#DFQL #DFDI #DFQL
0005*	#DFEI #DFUD	0031*	#DFIN	0059*	#DFIN #DFEI	0080*	#DFQL #DFDI
0006	#DFEX #DFEI #DFUD #DFIN #DFLS	0032* 0033*	#DFIN #DFEI #DFUD		#DFQE #DFUD #DFQI	0081* 0082*	#DFQL #DFDI #DFQL
0007*	#DFUD #DFIN	0035 0037	#DFLS #DFEX #DFEI #DFIN	0060*	#DFQL #DFQI #DFDI #DFQL	0083 0084 0085 0089	#DFWS #DFSB #DFDI #DFEI
0008	#DFEX		#DFUD		#DFQE		#DFUD
0009*	#DFEI #DFUD #DFIN	0039 0040*	#DFSB #DFEI	0061* 0062*	#DFQE #DFQL		#DFUP #DFMP
0010	#DFEX	0041	#DFMP	0063*	#DFQE		#DFQL
0011	#DFEX #DFSB	0044 0045	#DFMP #DFMP	0064*	#DFQL #DFDI		#DFQI #DFQE #DFIN
0012	#DFEX #DFSB	0046	#DFMP		#DFDI		#DFLS
0013	#DFEX #DFSB		#DFQI #DFQL	0065* 0066*	#DFQL #DFQL	0090*	#DFDI
0014	#DFMP #DFSB #DFEX	0047* 0048*	#DFQE #DFUP #DFDI	0067* 0068*	#DFQL #DFQL #DFDI	0111* 0112* 0113*	#DFDI #DFDI #DFDI
0015	#DFIN	0049*	#DFQE #DFDI	0069*	#DFQL #DFDI	0114* 0115*	#DFDI #DFDI
0016	#DFSB #DFEX	0050*	#DFQE #DFDI	0070*	#DFQL #DFDI	0116* 0117*	#DFDI #DFDI
0017*	#DFIN #DFEI #DFUD	0051* 0053*	#DFDI #DFDI #DFQE	0071* 0072*	#DFQL #DFQL	0118* 0119* 0120*	#DFDI #DFDI #DFDI
0018*	#DFEI		#DFQE	0073*	#DFQL	0121*	#DFDI
0019*	#DFEI #DFUD	0054* 0055*	#DFQI #DFDI #DFDI	0074* 0075*	#DFQL #DFDI #DFQL	0122* 0123* 0124*	#DFDI #DFDI #DFDI
0022	#DFUP				#DFDI	0125*	#DFDI

Note: All messages are SYSLOG messages unless annotated with an asterisk. The asterisked messages are displayed messages.

Figure 3-6 (Part 1 of 2). DFU Message-to-Module Cross Reference

Message	Module	Message	Module	Message	Module
0126*	#DFDI	0205	Procedures	0290	Procedures
0127*	#DFDI	0206	Procedures	0291*	#DFDI
0128*	#DFDI		#DFMP		
	#DFQL	0209	Procedures		
	#DFQI	0212	Procedures		
	#DFQE	0214	Procedures		
0129*	#DFDI	0216	Procedures		
0130*	#DFDI		#DFMP		
0131*	#DFDI	0217	#DFMP		
0132*	#DFDI		#DFQL		
0133*	#DFDI		#DFQI		
0134*	#DFDI		#DFQE		
0135*	#DFDI		#DFUP		
0136*	#DFDI		#DFEI		
0137*	#DFDI		#DFUD		
0139*	#DFDI		#DFIN		
0141*	#DFDI	0218	#DFMP		
0142*	#DFDI		#DFQL		
0143	#DFUP		#DFQE		
0145*	#DFDI		#DFQI		
0146*	#DFDI		#DFUP		
0152*	#DFDI		#DFEI		
0161*	#DFDI		#DFUD		
0162	#DFMP		#DFIN		
0163*	#DFDI	0224	#DFSB		
0164	#DFMP		#DFMP		
0167	#DFMP		#DFEX		
	#DFQI		Procedures		
	#DFQE	0254*	#DFMP		
	#DFQL	0255*	#DFMP		
	#DFFB	0256*	#DFMP		
0169	#DFMP	0258*	#DFMP		
	#DFQI	0259*	#DFDI		
	#DFQE	0264*	#DFDI		
	#DFQL	0265*	#DFDI		
	#DFUP	0266*	#DFDI		
	#DFDI	0269*	#DFDI		
	#DFFB	0270*	#DFDI		
	#DFWS	0275	Procedures		
0172*	#DFDI	0276	Procedures		
0174*	#DFDI	0277	Procedures		
0181*	#DFDI	0278	Procedures		
0182*	#DFDI	0280*	#DFDI		
0183*	#DFDI	0281	#DFEX		
0188*	#DFDI	0282	#DFEX		
0189*	#DFDI	0283	#DFEX		
0201	Procedures	0284	#DFEX		
0203	Procedures	0286*	#DFDI		
0204	Procedures	0289	#DFLS		

Note: All messages are SYSLOG messages unless annotated with an asterisk. The asterisked messages are displayed messages.

Figure 3-6 (Part 2 of 2). DFU Message-to-Module Cross Reference

PROCEDURES

The following procedures are invoked by DFU when the applicable commands are entered.

Enter Procedure

When the ENTER command is entered, DFU invokes the following procedure.

```
* ENTER FILENAME,FORMAT,RPGNAME,# REC,DATA TYPE,CATL,LIBNAME,,USERLIB,DISPLAY FORMAT SOURCE NAME
// MEMBER PROGRAM1-#DF#MG
// MEMBER USER1-#DF#MG
// IFF ?5'D'?/D #ERR 0201,C,DFU
// IF DATAF1-?1R'0202'? #ERR 0203,C,DFU
// IFF ?9'#LIBRARY'?/#LIBRARY IFF DATAF1-?9? #ERR 0224,C,DFU
// IF INQUIRY-NO IF ?11'1'?/ *
// ELSE IF ?11'2'?/ *
// IF ?2?/ * 0261
// IF ?2R'0262'?/ #DFMP ?1?,?2?,?3?,?10?,?6?,?7?,?9?,E,#DF?WS??11?
// ELSE IFF SUBR-'?2?,?9?' #DFMP ?1?,?2?,?3?,?10?,?6?,?7?,?9?,E,#DF?WS??11?
// IFF ?2?/ IFF LOAD-'?2?,?9?' #ERR 0278,C,DFU
// LOAD #DFEX
// FILE NAME-#DFDATA,LABEL-?1?,DISP-NEW,RETAIN-P,RECORDS-?4R'0207'?
// RUN
// DFU FL-?1?,FT-?2?,LB-?9?,DF-#DF?WS??11?,UT-E
// END
// LOAD #DFEX
// FILE NAME-#DFDATA,LABEL-?1?,DISP-SHR,RETAIN-P
// RUN
// DFU FL-?1?,FT-?2?,LB-?9?,DF-#DF?WS??11?,UT-U
// END
// IFF ?2?/ RETURN
// LOAD $MAINT
// RUN
// DELETE NAME-#DF?WS??11?,LIBRARY-ALL,LIBRNAME-?9?
// END
```


Update Procedure

When the UPDATE command is entered, DFU invokes the following procedure.

```
* UPDATE FILENAME,FORMAT,RPGNAME,,DATA TYPE,CATL,LIBNAME,,USERLIB,DISPLAY FORMAT SOURCE NAME
// MEMBER PROGRAM1-#DF#MG
// MEMBER USER1-#DF#MG
// IFF ?5'D'?/D #ERR 0201,C,DFU
// IFF DATAF1-?1R'0208'? #ERR 0209,C,DFU
// IFF ?9'#LIBRARY'?/#LIBRARY IFF DATAF1-?9? #ERR 0224,C,DFU
// IF INQUIRY-NO IF ?11'1'?/ *
// ELSE IF ?11'2'?/ *
// IF ?2?/ * 0261
// IF ?2R'0262'?/ #DFMP ?1?,???,?3?,?10?,?6?,?7?,,,?9?,U,#DF?WS??11?
// ELSE IFF SUBR-'?2?,?9?' #DFMP ?1?,???,?3?,?10?,?6?,?7?,,,?9?,U,#DF?WS??11?
// IFF ?2?/ IFF LOAD-'?2?,?9?' #ERR 0278,C,DFU
// LOAD #DFEX
// FILE NAME-#DFDATA,LABEL-?1?,RETAIN-P,DISP-SHR
// RUN
// DFU FT-???,UT-U,DF-#DF?WS??11?,LB-?9?,FL-?1?
// END
// IFF ?2?/ RETURN
// LOAD $MAINT
// RUN
// DELETE NAME-#DF?WS??11?,LIBRARY-ALL,LIBRNAME-?9?
// END
```

Inquiry Procedure

When the INQUIRY command is entered, DFU invokes the following procedure.

```
* INQUIRY FILENAME,FORMAT,RPGNAME,,DATA TYPE,CATL,LIBNAME,,USERLIB,DISPLAY FORMAT SOURCE NAME
// MEMBER PROGRAM1-#DF#MG
// MEMBER USER1-#DF#MG
// IFF ?5'D'?/D #ERR 0201,C,DFU
// IFF DATAF1-?1R'0210'? #ERR 0209,C,DFU
// IFF ?9'#LIBRARY'?/#LIBRARY IFF DATAF1-?9? #ERR 0224,C,DFU
// IF INQUIRY-NO IF ?11'1'?/ *
// ELSE IF ?11'2'?/ *
// IF ?2?/ * 0261
// IF ?2R'0262'?/ #DFMP ?1?,???,?3?,?10?,?6?,?7?,,,?9?,I,#DF?WS??11?
// ELSE IFF SUBR-'?2?,?9?' #DFMP ?1?,???,?3?,?10?,?6?,?7?,,,?9?,I,#DF?WS??11?
// IFF ?2?/ IFF LOAD-'?2?,?9?' #ERR 0278,C,DFU
// LOAD #DFEX
// FILE NAME-#DFDATA,LABEL-?1?,RETAIN-P,DISP-SHR
// RUN
// DFU FT-???,UT-I,LB-?9?,DF-#DF?WS??11?,FL-?1?
// END
// IFF ?2?/ RETURN
// LOAD $MAINT
// RUN
// DELETE NAME-#DF?WS??11?,LIBRARY-ALL,LIBRNAME-?9?
// END
```

List Procedure

When the LIST command is entered, DFU invokes the following procedure.

```
* LIST FILENAME,FORMAT,RPGNAME, SORT/NOSORT,DATA TYPE,CATL,LIBNAME,MASTERNAME,USERLIB
// MEMBER PROGRAM1-#DF#MG
// MEMBER USER1-#DF#MG
// IFF ?5'D'?/D #ERR 0201,C,DFU
// IFF DATAF1-?1R'0211'? #ERR 0209,C,DFU
// IFF ?8?/ IFF DATAF1-?8? #ERR 0290,C,DFU
// IFF ?9'#LIBRARY'?/#LIBRARY IFF DATAF1-?9? #ERR 0224,C,DFU
// IFF ?4R'0274'?/ ,IFF ?4?/NOSORT IFF ?4?/SORT #ERR 0212,C,DFU
// IF INQUIRY-NO IF ?11'1'?/ *
// ELSE IF ?11'2'?/ *
// IF ?2?/ * 0261
// IF ?2R'0262'?/ #DFMP ?1?,???,?3?,?4?,?6?,?7?,?8?,?9?,L,#DF?WS??11?
// ELSE IFF SUBR-'???,?9?' #DFMP ?1?,???,?3?,?4?,?6?,?7?,?8?,?9?,L,#DF?WS??11?
// IF ?4'NOSORT'?/SORT IF JOBQ-NO #DFST ?1?,???,?8?,?9?,#DF?WS??11?
// ELSE IF ?4?/SORT #DFST ?1?,???,?8?,?9?,#DF?WS?3
// IF JOBQ-NO * 0223
// LOAD #DFEX
// FILE NAME-#DFDATA,LABEL-?1?,RETAIN-P,DISP-SHR
// IF ?4?/SORT FILE NAME-#DFTAG,LABEL-#DFTAG,RETAIN-S
// IFF ?8?/ FILE NAME-#DFMAST,LABEL-?8?,RETAIN-P,DISP-SHR
// RUN
// DFU FT-???,DS-?4?,MF-?8?,LB-?9?,UT-L,DF-#DF?WS??11?
// END
// IFF ?2?/ RETURN
// LOAD $MAINT
// RUN
// DELETE NAME-#DF?WS??11?,LIBRARY-R,LIBRNAME-?9?
// END
```

#DFMP Procedure

#DFMP is invoked by DFU when DFU must enter the setup step.

```
* #DFMP FILENAME,FORMAT,RPGNAME,SORT/NOSORT/DISPLAY SOURCE,CATL,LIBNAME,MASTERNAME,USERLIB,UTILTYPE,DEFAULT NAME
// IFF ?9?/L IFF ?2?/ IF LOAD-'?2?,?8?' #ERR 0276,C,DFU
// IFF ?9?/L IFF ?4?/ IF SOURCE-'?4?,?8?' #ERR 0277,C,DFU
// IFF ?5'NN'?/NN IFF ?5?/NY IFF ?5?/YN IFF ?5?/YY IFF ?5?/GO #ERR 0205,C,DFU
// IFF ?5?/NN IF ?6?/ #ERR 0214,C,DFU
// IFF ?5?/NN IFF ?5?/NY IFF SOURCE-'?6?,?8?' #ERR 0206,C,DFU
// IF ?5?/NY IF SOURCE-'?6?,?8?' #ERR 0204,C,DFU
// IFF ?9?/L IFF ?4?/ IF ?4?/?6? #ERR 0275,C,DFU
// IFF SOURCE-'?3R'0215'?,?8?' #ERR 0216,C,DFU
// LOAD #DFMP
// WORKSTN RESTORE=YES,UNIT=?WS?
// RUN
// DFU FL-?1?,FT-?2?,RG-?3?,DS-?4?,SP-?5?,SN-?6?,MF-?7?,LB-?8?,UT-?9?,DF-?10?
// END
// IF ?9?/L RETURN
// LOAD $SFGR
// RUN
// IF ?2?/ LOADMBR NAME-?10?,REPLACE=YES
// ELSE LOADMBR NAME-?2?
// INOUT PRINT=NO,INLIB-?8?,OUTLIB-?8?
// IF ?4?/ CREATE SOURCE-?10?,NUMBER=32
// ELSE CREATE SOURCE-?4?,NUMBER=32
// END
// IFF ?4?/ RETURN
// LOAD $MAINT
// RUN
// DELETE NAME-?10?,LIBRARY=S,LIBRNAME-?8?
// END
```

#DFST Procedure

DFU invokes this procedure when modules are required for a list with sort.

```
* #DFST FILENAME,FORMAT,MASTER NAME,USER LIBRARY,DEFAULT NAME
// IF JOBQ=NO * 0219
// LOAD #DFSB
// RUN
// DFU FL-?1?,FT-?2?,MF-?3?,LB-?4?,DF-?5?
// END
// LOAD #G SORT
// FILE NAME=INPUT,LABEL-?1?,DISP=SHR
// FILE NAME=OUTPUT,LABEL=#DFTAG,RETAIN=J,BLOCKS=10
// RUN
// SOURCE ?5?,?4?
// LOAD $MAINT
// RUN
// DELETE NAME-?5?,LIBRARY=S,LIBRNAME-?4?
// END
```

DFUDROP Procedure

DFU invokes this procedure when requested to delete DFU from a specified library.

```
* 5726-UT1 COPYRIGHT IBM CORP 1977 LICENSED MATERIAL - PROGRAM PROPERTY OF IBM
* DFUDROP USERLIB
// MEMBER PROGRAM1-#DF#MG
// MEMBER USER1-#DF#MG
// IF JOBQ-NO * 0221
// LOAD $MAINT
// RUN
// DELETE NAME-ENTER,LIBRARY-P,LIBRNAME-?1'#LIBRARY'?
// DELETE NAME-UPDATE,LIBRARY-P,LIBRNAME-?1'#LIBRARY'?
// DELETE NAME-INQUIRY,LIBRARY-P,LIBRNAME-?1'#LIBRARY'?
// DELETE NAME-LIST,LIBRARY-P,LIBRNAME-?1'#LIBRARY'?
// DELETE NAME-DFUSAVE,LIBRARY-P,LIBRNAME-?1'#LIBRARY'?
// DELETE NAME-DFULOAD,LIBRARY-P,LIBRNAME-?1'#LIBRARY'?
// DELETE NAME-#DF.ALL,LIBRARY-P,LIBRNAME-?1'#LIBRARY'?
// DELETE NAME-#DF.ALL,LIBRARY-O,LIBRNAME-?1'#LIBRARY'?
// DELETE NAME-DFUDROP,LIBRARY-P,LIBRNAME-?1'#LIBRARY'?
// END
```

DFULOAD Procedure

DFU invokes this procedure when requested to load DFU into a specified library.

```
// MEMBER PROGRAM1-#DF#MG
// MEMBER USER1-#DF#MG
// IF JOBQ-NO * 0222
```

DFUSAVE Procedure

DFU invokes this procedure when requested to save DFU on diskette.

```
* DFUSAVE USERLIB-NAME
// MEMBER PROGRAM1-#DF#MG
// MEMBER USER1-#DF#MG
// IF JOBQ-NO * 0220
// LOAD $MAINT
// FILE PACK-PPUTIL,UNIT-11,RETAIN-999,NAME-DFU
// RUN
// COPY FROM-?1'F1'?,TO-DISK,FILE-DFU,NAME-DFUDROP,LIBRARY-P
// COPY FROM-?1'F1'?,TO-DISK,FILE-DFU,NAME-ENTER,LIBRARY-P,ADD-YES
// COPY FROM-?1'F1'?,TO-DISK,FILE-DFU,NAME-UPDATE,LIBRARY-P,ADD-YES
// COPY FROM-?1'F1'?,TO-DISK,FILE-DFU,NAME-INQUIRY,LIBRARY-P,ADD-YES
// COPY FROM-?1'F1'?,TO-DISK,FILE-DFU,NAME-LIST,LIBRARY-P,ADD-YES
// COPY FROM-?1'F1'?,TO-DISK,FILE-DFU,NAME-DFUSAVE,LIBRARY-P,ADD-YES
// COPY FROM-?1'F1'?,TO-DISK,FILE-DFU,NAME-DFULOAD,LIBRARY-P,ADD-YES
// COPY FROM-?1'F1'?,TO-DISK,FILE-DFU,NAME-#DF.ALL,LIBRARY-P,ADD-YES
// COPY FROM-?1'F1'?,TO-DISK,FILE-DFU,NAME-#DF.ALL,LIBRARY-O,ADD-YES
// END
```


Introduction

The source entry utility (SEU) portion of the Utilities Program Product is used to create and maintain source and procedure members in System/34 libraries. SEU operates in five modes:

- Enter/update, to enter and change statements under control of display screen formats.
- Delete, to delete statements from members.
- Move, to move statements within a member.
- Include, to include statements in a member from another or from the same member.
- Scan, to scan a member for a particular statement.

HOW THE PROGRAM WORKS

The operator signs on SEU by entering the SEU command. Parameters in the command specify, either explicitly or by default: member name, member type, format member name, statement length, and library name.

After sign-on, SEU is in the enter/update mode. The operator can select a different mode by pressing one of the SEU command function keys. The operator requests specific functions in each mode by entering responses to the SEU prompts and by pressing command function keys.

Data entered and changes made by the operator are stored in the SEU work file until SEU end of job. The SEU work file is a disk work area labeled #SEU#.nn, where nn is the ID of the SEU operator's display station. At normal SEU end of job, the contents of the work file are copied to the library identified by the SEU command, and the work file is deleted.

PHYSICAL CHARACTERISTICS

SEU is composed of 32 load members, four procedure members, two source members, and 16 expansion buffers.

Load Members

- #SEU—common. #SEU contains the following routines, and must not be overlaid:
 - SEUPNT—printer interface
 - SEUKEY—display station interface
 - SEUDM—work file data management
 - SEUMSG—warning message
- #SEIN—initialization
- #SELL—Roll ↑ (Roll Up), Roll ↓ (Roll Down), and search end of source interface.
- #SEDX—work file data management for deleting statements, counting statements, and chaining new index sectors to the index area in the work file.
- Mode execution:
 - #SEET—enter/update
 - #SEDL—delete
 - #SEMV—move
 - #SEIC and #SEID—include
 - #SEAN—scan
- Syntax checkers for RPG II and auto report specifications:
 - #SERA, #SERB, #SERC, and #SERD—calculation
 - #SERE—control, option, copy, extension, and line counter
 - #SERF, #SERG, and #SERH—file description
 - #SERI and #SERJ—input
 - #SERL—extension
 - #SERO and #SERP—output
 - #SERT and #SERU—telecommunications
- #SEEO and #SEEJ—SEU end of job.
- #SE@FMT—display screen formats for SEU prompts and messages.

- #SE@FORM and #SE@XTRA—supplied display screen formats for entering and updating statements.
- #SE#M1 and #SE#M2—level 1 and level 2 messages, respectively.

The control flow between load members is illustrated in *Program Organization*, in this chapter. An alphabetical listing of member names with a short description of each member is contained in *Directory*, in this chapter.

Procedure Members

SEU	signs on SEU.
SEUDROP	deletes SEU from a library.
SEULOAD	issues a load message.
SEUSAVE	saves SEU on a diskette.

The contents of the procedure members are listed in the *Program Organization* section of this chapter.

Source Members

#SE@FORM and #SE@XTRA contain the display screen format specifications that define the display screen formats supplied with SEU.

Expansion Buffers

So that SEU remains a fixed size, SEU contains 16 expansion buffers to absorb any increase in code caused by changes to SEU. The buffers are labeled #SES0 through #SES15.

INPUT/OUTPUT

After the operator signs on SEU, input to the program consists of:

- A copy of the selected member in the SEU work file if the member already exists in a library.
- Responses to SEU prompts.
- Data entered from the keyboard if the operator enters or updates statements.

Output from SEU is:

- A new or changed member in the specified library.
- Required messages.
- Selected printed statements if the print option is used.
- A listing of the new or changed member if a listing is requested.

I/O STORAGE REQUIREMENTS

SEU requires 14K (14,336) bytes of main storage.

SYSTEM CONFIGURATION

SEU runs on all models of System/34.

Method of Operation

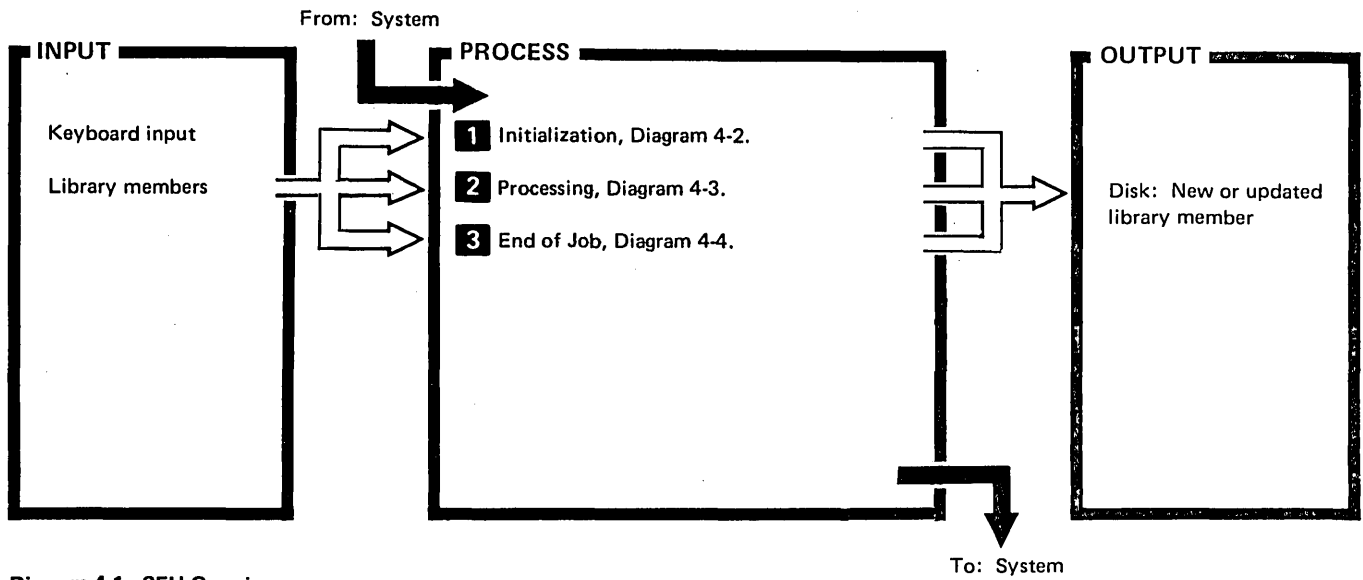


Diagram 4-1. SEU Overview

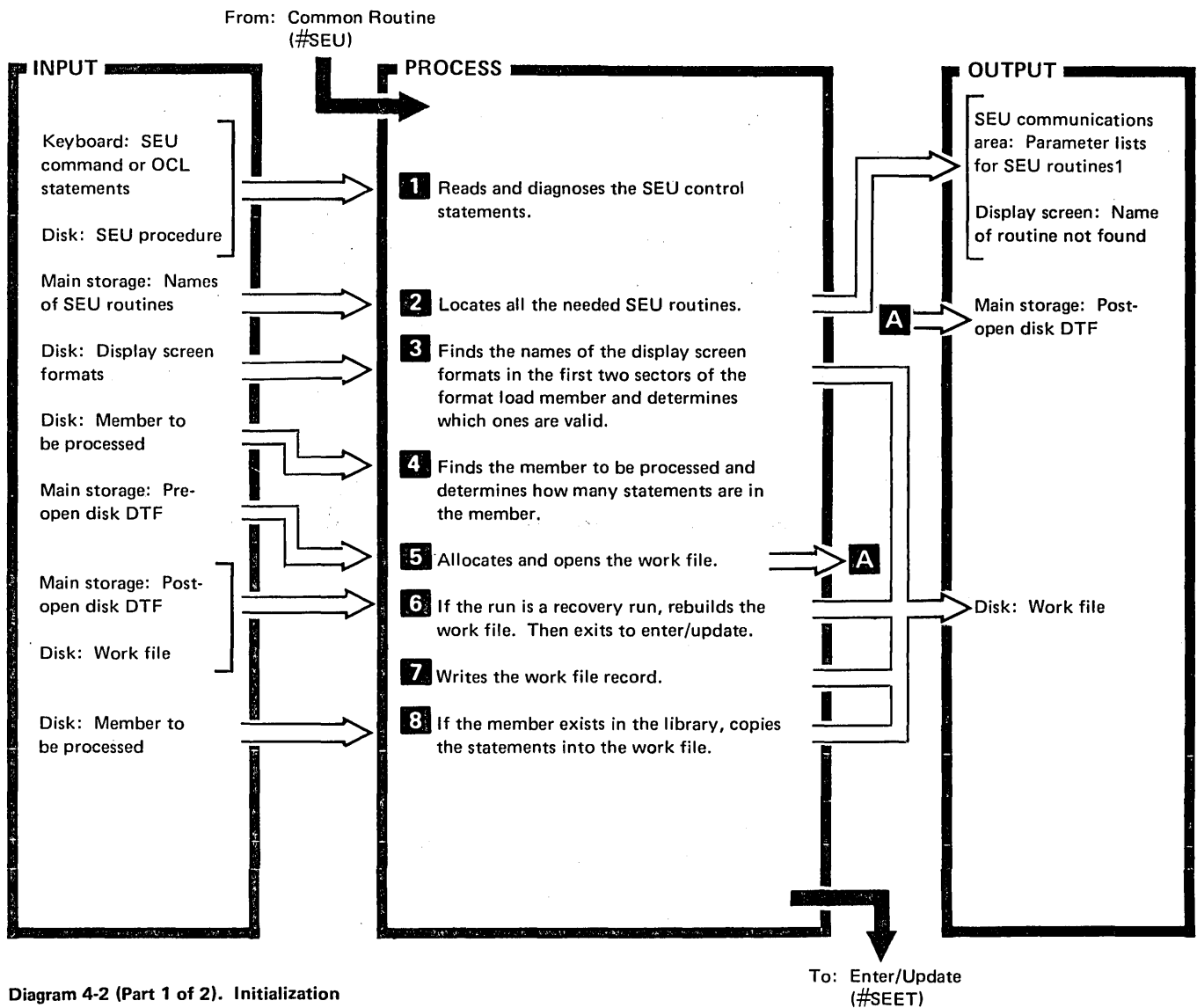
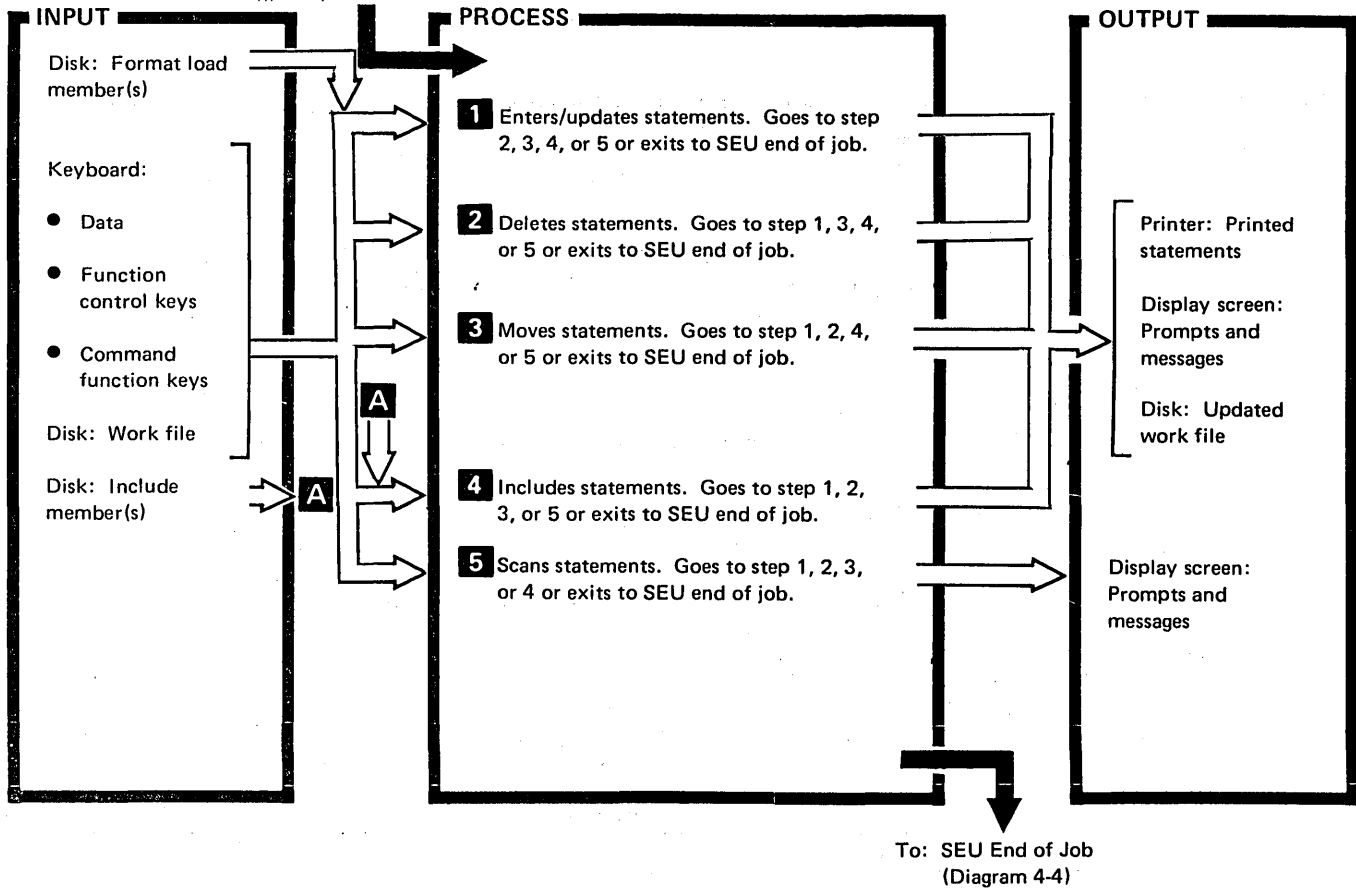


Diagram 4-2 (Part 1 of 2). Initialization

DESCRIPTION	MODULE/ ROUTINE
<ol style="list-style-type: none"> 1 Reads, by way of SYSIN, the SEU control statements SEU NAME- . . . and END. Diagnoses the two statements. If either is in error, calls SYSLOG to issue a message and cancels the job. SYSIN and SYSLOG are described in the publication <i>SSP Logic System</i>. 2 Finds the SEU routines. Builds parameter lists for loading the routines and saves the parameter lists in the SEU communications area. If a routine is not found, calls SYSLOG to issue an error message and cancels the job. 3 Finds directory entries for the format load member(s). Finds in the first two sectors of the format member(s) indexes of the display screen formats and determines which formats are valid. The format length of valid formats must be equal to or greater than 1, and equal to or less than 120. The indexes of valid formats are saved by #SEIN. 4 Finds the member to be processed and determines the number of statements in the member. If the member is not found, initialization assumes the member will be created. 5 Allocates and opens the work file. Calculates the start and end extents of the index for the work file, and calculates the start and end extents of the statement area. 6 If the job is a recovery run, rebuilds the index from the statement area. Also determines whether or not the parameters in the SEU command for the recovery run are the same as the parameters saved in the work file record (the work file record is described in <i>Data Areas</i>, in this chapter). If the parameters are not the same, calls SYSLOG to issue an error message and cancels the job. When the work file is rebuilt, passes control to enter/update (#SEET). 7 Creates the work file record and writes it to the work file. 8 If the job is not a recovery run and the member to be processed exists, calls the source library get routine (#MASGT) to copy statements from the member to the work file. The source library get routine is described in the publication <i>SSP Logic: System</i>. 	#SEIN

Diagram 4-2 (Part 2 of 2). Initialization

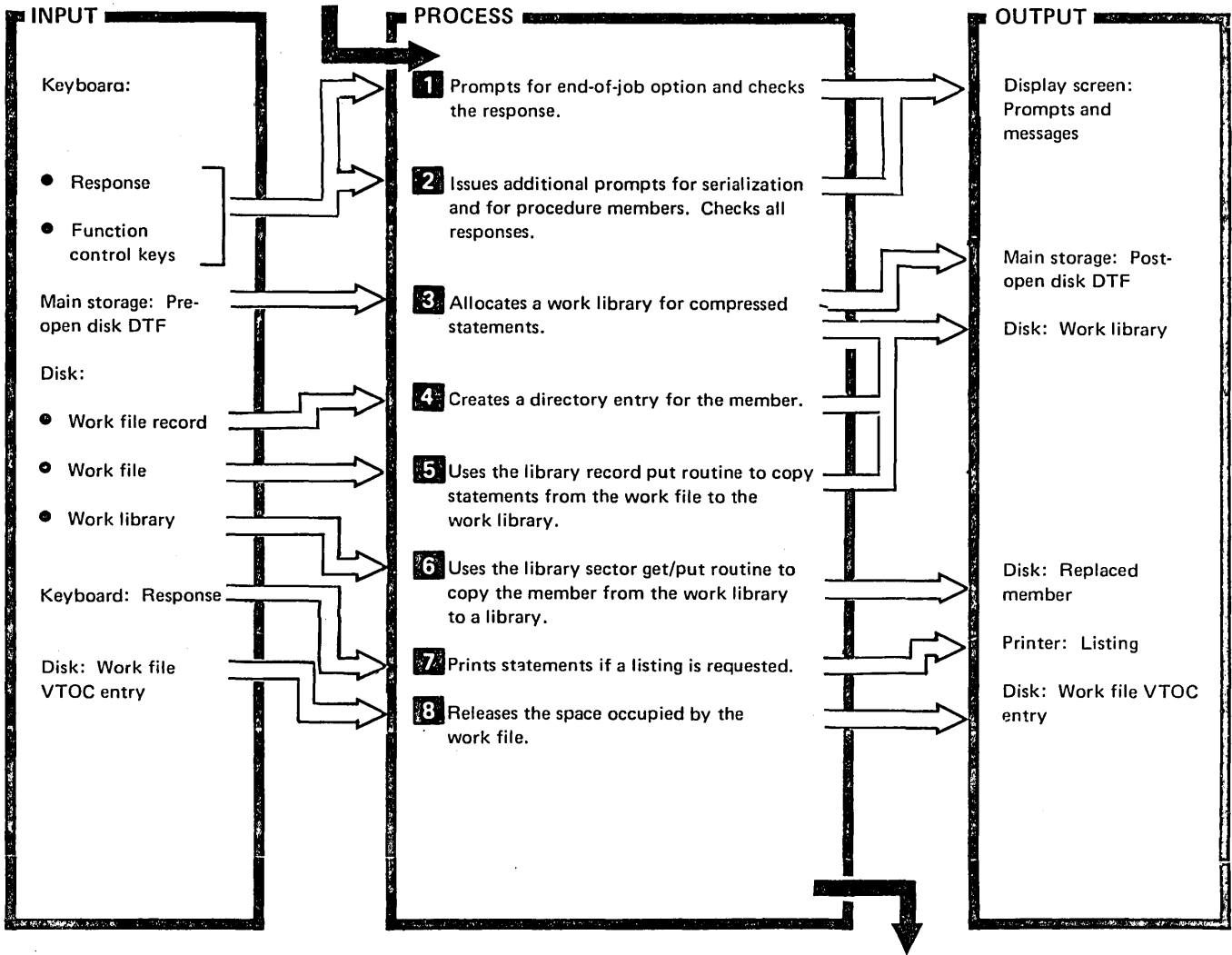
From: Initialization (Diagram 4-2) or
Display Station Interface
(SEUKEY) in Common Routine
(#SEU)



DESCRIPTION	MODULE/ ROUTINE
1 Enter/update allows the operator to enter new statements into a member and to update existing statements in a member. Statements are entered or updated under control of display screen formats. If the member type specified in the SEU command is A or R, SEU can check the syntax of statements entered or updated under control of the supplied RPG II or auto report formats. Statements that are entered or updated can also be printed.	#SEET
2 Delete allows the operator to delete statements from a member. If no statements exist in the member, the delete mode is not allowed. Statements that are deleted can also be printed.	#SEDL
3 Move allows the operator to move statements from one location in a member to a new location in the same member. The statements are deleted from the original location. If no statements exist in the member, the move mode is not allowed. Statements that are moved are not printed.	#SEMV
4 Include allows the operator to include existing statements in the member being processed. Statements are included from a source or procedure member that exists in a library. Statements that are included can also be printed.	#SEIC
5 Scan searches a member for a statement that contains a string of characters specified by the operator. If no statements exist in the member, the scan mode is not allowed.	#SEAN

Diagram 4-3. Processing

- From:
- Display Station Interface (SEUKEY) in Common Routine (#SEU)
 - Enter/Update (#SEET)
 - Delete (#SEDL)
 - Move (#SEMV)
 - Include (#SEIC)
 - Scan (#SEAN)



SSP Termination Interface (#CTEIF)

Diagram 4-4 (Part 1 of 2). End of Job

DESCRIPTION	MODULE/ ROUTINE
<p>1 Displays end-of-job options 0 through 5. Checks the validity of the response.</p> <p>2 If serialization is requested (end-of-job option 3 or 4) and the member in the work file contains statements, end of job issues additional prompts and checks the validity of each response:</p> <ul style="list-style-type: none"> • If the member in the work file is not a procedure member (member type not P) and the statement length of the member is 80 or more, end of job prompts for an indication of program name duplication. • End of job prompts for a serial start position. • If the member in the work file is a procedure member (member type P), end of job prompts for options related only to procedures. <p>3 Calculates size of work library and requests work library space to contain the statements in the work file after the statements are compressed.</p> <p>4 Creates a directory entry for the member that was created or changed during the job.</p> <p>5 Uses the library record put routine (\$MAPUR) to move statements one at a time from the work file to the work library. Statements are compressed as they are moved. If serialization is requested, statements are serialized as they are placed in the work library. The library record put routine is described in the publication <i>SSP Logic: System</i>.</p> <p>6 Uses the library sector get/put routine (\$MAPGS) to copy the member from the work library to the library specified by the SEU command. The library sector get/put routine is described in the publication <i>SSP Logic: System</i>.</p> <p>7 If a listing is requested, uses the printer data management routine (#DPDM) to list statements from the work file (if serialization is also requested, serializes the statements before printing them). Statements are printed after they are copied to the library specified by the SEU command.</p> <p>8 Changes the retain status of the work file from T to S.</p>	<p>#SEEO</p> <p>#SEEJ</p>

Diagram 4-4 (Part 2 of 2). End of Job

Program Organization

This section illustrates control flow between SEU members for SEU sign-on and SEU operation. Figure 4-1 illustrates sign-on, and Figure 4-2 illustrates operation.

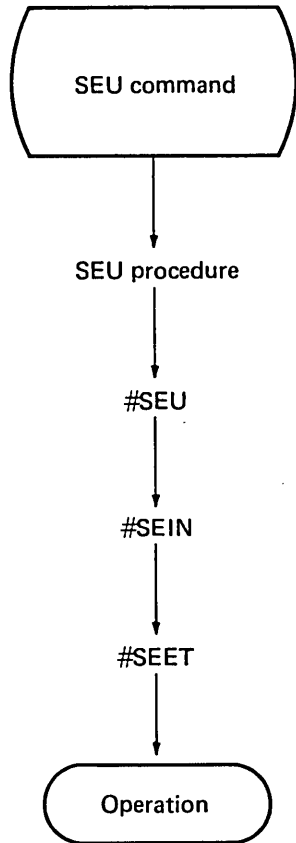


Figure 4-1. Control Flow for SEU Sign-On

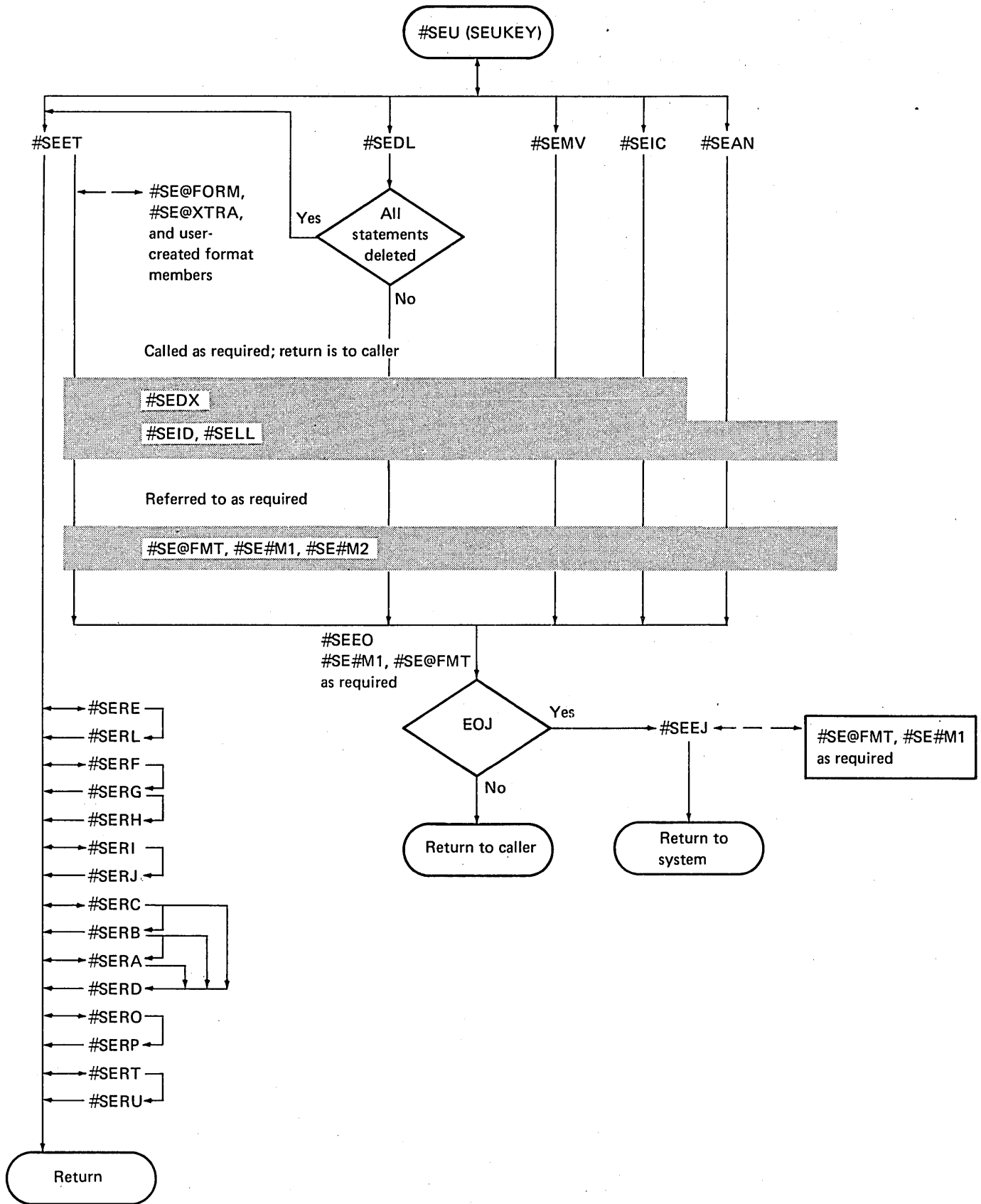


Figure 4-2. Control Flow for SEU Operation

Directory

Module Name	Major Functions	Module Name	Major Functions
#SE#M1	Contains level 1 messages issued by SEU.	#SEET	Enters statements into the member or updates statements that already exist in the member. Retrieves selected display screen formats. If print option is on, uses printer data management (#DPDM) to print statements.
#SE#M2	Contains level 2 messages issued by SEU.		
#SE@FMT	Contains display screen formats for SEU prompts and messages.	#SEIC	Includes statements from a member in a library into the member being worked on. Performs roll up and search end of source functions for include members. If print option is on, uses printer data management (#DPDM) to print statements.
#SE@FORM	Contains the SEU-supplied free-form formats and SEU-supplied display screen formats for display screen format, RPG II and auto report, and work station utility specifications.		
#SE@XTRA	Contains SEU-supplied display screen formats for assembler, magnetic character reader, sort, and FORTRAN IV specifications.	#SEID	Checks responses to INCLUDE LIBRARY NAME and INCLUDE MEMBER NAME. Changes the roll factor.
#SEAN	Scans a member for a statement that contains a specified character string.	#SEIN	Initializes variables and diagnoses the SEU control statements. If it already exists, copies the member being worked on into the work file from the library.
#SEU	Loads the SEU communications area and the four main storage resident routines (SEUDM, SEUKEY, SEUMSG, and SEUPNT).	#SELL	Performs roll up, roll down, and search end of source functions on all members except include members.
#SEDL	Deletes statements from the member being worked on.	#SEMV	Moves statements to a new location and deletes them from their old location in the member being worked on.
#SEDX	Performs the following data management operations on the SEU work file: deletes statements, counts statements, and chains new index sectors to index. Checks and reformats statement number responses.	#SERA	Continues syntax-checking RPG II calculation specification (phase 3 of 4). Syntax checks all specifications that have a slash (/) in position 7.
#SEEJ	Copies the statements from the SEU work file into the library member. Performs statement serialization, program name duplication, and member listing as requested. Deletes the SEU work file.	#SERB	Continues syntax-checking RPG II calculation specification (phase 2 of 4).
		#SERC	Syntax checks RPG II calculation specification (phase 1 of 4).
#SEEO	Allocates a printer to SEU if printing is requested, changes the number of display screen lines per statement, displays the command function key display, and displays and handles the response to the end-of-job options.	#SERD	Continues syntax-checking RPG II calculation specification (phase 4 of 4).
		#SERE	Syntax checks RPG II control, extension, and line counter specifications; and auto report option and /COPY specifications (phase 1 of 2).

Module Name	Major Functions
#SERF	Syntax checks RPG II file description specification (phase 1 of 3).
#SERG	Continues syntax-checking RPG II file description specification (phase 2 of 3).
#SERH	Continues syntax-checking RPG II file description specification (phase 3 of 3).
#SERI	Syntax checks RPG II input specification (phase 1 of 2).
#SERJ	Continues syntax-checking RPG II input specification (phase 2 of 2).
#SERL	Continues syntax-checking RPG II extension specification (phase 2 of 2).
#SERO	Syntax checks RPG II output specification (phase 1 of 2).
#SERP	Continues syntax-checking RPG II output specification (phase 2 of 2).
#SERT	Syntax checks RPG II telecommunications specification (phase 1 of 2).
#SERU	Continues syntax-checking RPG II telecommunications specification (phase 2 of 2).
#SES0 through #SES15	Reserve space for expansion.
SEUDM	Performs data management functions for the SEU disk work file.
SEUKEY	Provides SEU with an interface to the display station.
SEUMSG	Retrieves messages needed by SEU from #SE#M1 and #SE#M2.
SEUPNT	Provides SEU with an interface to the printer.

Data Areas




This section describes the SEU communications area and the SEU work file, both of which are used by more than one SEU module. For a detailed description of field contents of all SEU data areas, see the SEU microfiche. For a description of system data areas used by SEU, see the *Data Areas and Diagnostic Aids Handbook*.

COMMUNICATIONS AREA

The SEU communications area contains constants, variables, pointers, keywords, and DTFs that are used by more than one SEU module. The communications area is initialized by #SEU and is included in each executable SEU load member. The communications area is 3328 (hex 0D00) bytes long and begins in main storage at label SEUINCOM.

WORK FILE

The SEU work file is organized as follows:

Work File Record	Index Area	Data Area
		
Always 6 sectors	Variable length	Variable length

Work File Record

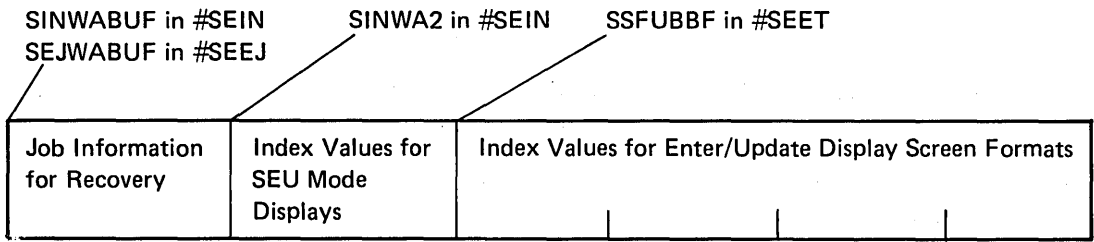
SEUWBEXT in the SEU communications area contains the 3-byte disk address of the start of the work file record. The work file record, which is initialized by #SEIN, occupies the first six sectors of the SEU work file.

The first sector contains information about the SEU job. This information is required for recovery of the work file in the event of a system failure. The sector definition begins in the load member #SEIN at SINWABUF, and in the load member #SEEJ at SEJWABUF.

The second sector of the work file record contains index values for all displays shown on the display screen by each of the five SEU modes. The sector definition begins in the load member #SEET at SSFUBBF.

The last four sectors of the work file record contain index values for the display screen formats available for entering and updating statements. The values refer to display screen formats in #SE@FORM and, if the member is specified in the SEU command, #SE@XTRA or a format member created by the user. The four sectors can contain a maximum of 64 index values (64 is the maximum number of statement display screen formats available for one SEU job). The definition of the four sectors begins in the load member #SEET at SSFUBBF.

Labels and contents of the work file record are summarized in the following illustration.



Index Area

The index area in the SEU work file contains an index of the statements in the work file data area. The following fields in the SEU communications area identify and define the index area:

- SEUXBEXT—Contains the 3-byte disk address of the start of the index area.
- SEUXLEXT—A one-byte field that identifies the number of sectors in the index area.
- SDMXSECT—A 256-byte definition of each sector in the index area.

Data Area

The data area in the SEU work file contains one record for each statement in the member being created or changed. Each record contains a packed statement number and one statement. The following fields in the SEU communications area identify the data area:

- SEUDBEXT—Contains the 3-byte disk address of the start of the data area.
- SEUDLEXT—A one-byte field that identifies the number of sectors in the data area.

Diagnostic Aids

This section presents diagnostic information unique to SEU. Diagnostic aids common to all of System/34 are described in the *Data Areas and Diagnostic Aids Handbook*.

MESSAGES

Figures 4-3 and 4-4 relate all SEU message identification codes (MICs) to the SEU modules that diagnose error conditions. Messages from the syntax checkers are described in the *SEU Reference Manual*. All other messages are described in the *Displayed Messages Guide*.

MIC	Diagnosing Module	MIC	Diagnosing Module
0101	SEU procedure	0517	#SEET
0102	SEU procedure	0518	#SEET
0103	SEULOAD procedure	0519	#SEID
0104	SEU procedure	0522	#SEEJ
0105	SEUDROP procedure	0523	#SEEO
0106	SEUSAVE procedure	0524	#SEEJ
0201	SEU procedure	0525	#SEEJ
0204	SEU procedure	0526	#SEEJ
0301	#SEIN (issued by SYSLOG)	0528	#SEEO
0302	#SEIN (issued by SYSLOG)	0530	#SEMV
0305	#SEIN (issued by SYSLOG)	0531	#SEET
0307	#SEIN (issued by SYSLOG)	0533	#SEID
0309	#SEIN (issued by SYSLOG)	0538	#SEIN (issued by SYSLOG)
0313	#SEIN (issued by SYSLOG)	0539	#SEIN
0315	#SEIN (issued by SYSLOG)	0540	#SEID
0327	#SEIN	0543	#SEIN
0435	#SEDL	0544	#SEIN
0436	#SEMV	0545	#SEIN (issued by SYSLOG)
0437	#SEIC	0546	#SEIN (issued by SYSLOG)
0501	#SEET	0549	#SEEJ (issued by SYSLOG)
0502	#SEDL, #SEET, #SEIC, #SEMV	0550	#SEEJ (issued by SYSLOG)
0503	#SEDL, #SEMV	0552	#SEIN (issued by SYSLOG)
0504	#SEDL	0553	#SEIN (issued by SYSLOG)
0505	#SEMV	0555	#SEEJ
0506	#SEMV	0558	#SEET
0507	#SEMV	0561	#SEEO (issued by SYSLOG)
0508	#SEET	0562	#SEEO (issued by SYSLOG)
0509	#SEET	0564	#SEIN
0510	#SEAN, #SEDL, #SEEJ, #SEET, #SEIC, #SEID, #SEMV	0565	#SEAN
0511	#SEIC, #SEID	0566	#SEAN
0512	#SEIC	0567	#SEAN
0513	#SEIC	0568	#SEAN
0514	#SEIC	0569	#SEAN, #SEET
0515	#SEIC	0570	#SEEJ
0516	#SEIC	0571	#SEEO

Figure 4-3. Level 1 Messages for SEU

MIC	Diagnosing Module
0307	#SEIN
0309	#SEIN
0313	#SEIN
0315	#SEIN
0538	#SEIN
0545	#SEIN
0546	#SEIN
0550	#SEEJ
0552	#SEIN
0553	#SEIN
0561	#SEEO
1001	#SEET, #SERA
1002	#SERE, #SERF, #SERI, #SERO, #SERT
1009	#SERA
1081	#SERE
1101	#SERF, #SERH
1102	#SERG
1103	#SERG
1104	#SERG
1105	#SERG
1106	#SERH
1107	#SERH
1108	#SERG
1109	#SERH
1110	#SERH
1111	#SERH
1112	#SERH
1113	#SERG
1114	#SERF
1115	#SERF
1116	#SERH
1117	#SERG
1118	#SERG
1120	#SERG
1121	#SERG
1122	#SERH
1123	#SERH
1124	#SERH
1125	#SERH
1126	#SERH
1127	#SERH
1128	#SERG
1129	#SERH
1130	#SERH
1132	#SERF
1133	#SERF
1134	#SERF
1201	#SERE
1202	#SERE
1203	#SERE
1204	#SERL

MIC	Diagnosing Module
1205	#SERL
1206	#SERL
1207	#SERL
1208	#SERE
1209	#SERL
1210	#SERL
1251	#SERE
1252	#SERE
1301	#SERT
1302	#SERT
1303	#SERT
1305	#SERT
1306	#SERT
1307	#SERU
1308	#SERU
1309	#SERU
1310	#SERU
1311	#SERT
1312	#SERT
1313	#SERU
1314	#SERU
1315	#SERT
1316	#SERT
1317	#SERT
1401	#SERI
1402	#SERI, #SERJ
1403	#SERI
1404	#SERJ
1405	#SERJ
1406	#SERI
1407	#SERJ
1408	#SERJ
1410	#SERI
1411	#SERJ
1412	#SERJ
1413	#SERJ
1414	#SERJ
1415	#SERD
1416	#SERJ
1417	#SERJ, #SERP
1418	#SERJ
1419	#SERI
1420	#SERI
1421	#SERI
1501	#SERD
1502	#SERD
1503	#SERB, #SERC
1504	#SERB, #SERC
1505	#SERB, #SERC
1506	#SERC, #SERD
1507	#SERA, #SERB, #SERD

Figure 4-4 (Part 1 of 2). Level 2 Messages for SEU

MIC	Diagnosing Module
1508	#SERB, #SERD
1509	#SERB
1510	#SERA
1511	#SERA
1512	#SERA
1513	#SERA
1514	#SERA
1515	#SERA
1601	#SERO
1602	#SERO
1603	#SERO
1604	#SERO
1605	#SERP
1606	#SERO, #SERP
1607	#SERO
1608	#SERP
1609	#SERO
1610	#SERO
1611	#SERO
1612	#SERP

Figure 4-4 (Part 2 of 2). Level 2 Messages for SEU

MAIN STORAGE STATUS FOR SEU MEMBERS

Three fields in main storage identify SEU members present in main storage.

- *SEUINSUB* + (X'07' through X'0B'). Contains the hexadecimal representation of the name of the SEU load member last called by the executing SEU mode member.
- *SEUMLOAD* + (X'07' through X'0B'). Contains the hexadecimal representation of the name of the SEU mode member currently in main storage. Mode members are #SEAN (scan), #SEDL (delete), #SEET (enter/update), #SEIC (include), and #SEMV (move). #SEEJ (end of job) is named at the same location.
- *SEUMODE*. A 1-byte field that indicates the current mode of SEU:

<i>Value</i>	<i>Meaning</i>
X'80'	Enter/update
X'40'	Delete
X'20'	Move
X'10'	Include
X'08'	Scan
X'00'	Initialization or end of job

PROCEDURES

SEU

The procedure member SEU contains:

```
* 5726-UT1 COPYRIGHT IBM CORP 1977 REFER TO COPYRIGHT
* INSTRUCTIONS FORM NO. G120-2083.
// MEMBER USER1-#SE#M1
// MEMBER USER2-#SE#M2
// MEMBER PROGRAM1-#SE#M1
// MEMBER PROGRAM2-#SE#M2
// * 0104
// IF ?1R'0101'?/ #ERR 0201,C,SEU
// IF ?2R'0102'?/ #ERR 0204,C,SEU
// LOAD #SEU
// WORKSTN RESTORE=YES,UNIT=?WS?
// RUN
// SEU NAME-?1?,TYPE-?2?,FORMAT-?3?,LENGTH-?4?,LIBRARY-?5?
// END
```

The SEU procedure is called by the SEU command:

```
SEU p1,p2,p3,p4,p5
```

where the positional parameters have the following meanings:

<i>Parameter</i>	<i>Meaning</i>
p1	Member name
p2	Member type
p3	Format member name
p4	Statement length
p5	Library name

The OCL statements generated by the SEU procedure are:

```
// MEMBER USER1-#SE#M1
// MEMBER USER2-#SE#M2
// MEMBER PROGRAM1-#SE#M1
// MEMBER PROGRAM2-#SE#M2
// LOAD #SEU
// WORKSTN RESTORE=YES,UNIT=?WS?
// RUN
// SEU NAME-p1,TYPE-p1,FORMAT-p3,LENGTH-p4,LIBRARY-p5
// END
```

SEUDROP

SEUDROP contains:

```
// MEMBER USER1-#SE#M1
// * 0105
// LOAD $MAINT
// RUN
// DELETE NAME-#SE.ALL,LIBRARY-ALL,LIBRNAME-?1'#LIBRARY'?
// DELETE NAME-SEU,LIBRARY-P,LIBRNAME-?1'#LIBRARY'?
// DELETE NAME-SEULOAD,LIBRARY-P,LIBRNAME-?1'#LIBRARY'?
// DELETE NAME-SEUSAVE,LIBRARY-P,LIBRNAME-?1'#LIBRARY'?
// DELETE NAME-SEUDROP,LIBRARY-P,LIBRNAME-?1'#LIBRARY'?
// END
```


SEULOAD

SEULOAD contains:

```
// MEMBER USER1-#SE#M1  
// * 0103  
* THIS PROCEDURE IS HERE ONLY FOR S-32 COMPATIBILITY
```

SEUSAVE

SEUSAVE contains:

```
// MEMBER USER1-#SE#M1  
// * 0106  
// LOAD $MAINT  
// FILE PACK-PPUTIL,UNIT-11,RETAIN-999,NAME-SEU  
// RUN  
// COPY FROM-?1'F1'?,TO-DISK,FILE-SEU,NAME-#SE.ALL,LIBRARY-ALL  
// COPY FROM-?1'F1'?,TO-DISK,FILE-SEU,NAME-SEU.ALL,LIBRARY-P,ADD-YES  
// END
```

Introduction

The sort program is designed to sort records from a disk file. Sorting is based on control field values from within each input record. Records are selected for or omitted from a sort according to their record types. The record types are determined by comparing fields to constants, to other fields within the same record, or to all or part of the program date. The include sets and omit sets that differentiate the record types are created by using OR conditions, AND conditions, or a combination of both. The sort program performs three types of ascending and descending sorts: an addrout sort, a tagalong sort, or a summary tagalong sort.

The sort program manipulates data contained in a user data file on disk. The user data file remains intact unless it is overlaid by the output file. The output of the sort program is part or all of the input data or 3-byte binary relative record numbers for some or all of the records in the input file.

HOW THE PROGRAM WORKS

The sort program has two parts: generation and execution. During generation, the sort sequence specifications are read and diagnosed; the specifications and diagnostic messages are printed if requested; the select/build routine is generated; and the sort is designed. During execution, the input records are read; the select/build routine determines which input records to include in the sort; a work record is built for each input record to be included in the sort; the work records are sorted, generating strings on a work file; the strings are merged in ascending or descending sequence until the number of strings remaining can be handled by a final merge; then the sorted work records are written to the output file as the final merge is performed.

PHYSICAL CHARACTERISTICS

The sort program consists of the following:

- Thirty-four load modules
- One message member (#GS#MM)
- Four procedures
- Fifteen expansion buffers (#GSS0-#GSS14) used to maintain the sort program at a fixed size on disk

INPUT/OUTPUT

The output of an addrout sort is an output file (addrout file) containing 3-byte binary relative record numbers of some or all of the input records. Zero indicates the first record in the output file. The output of a tagalong sort or summary tagalong sort consists of sorted output records. These output records may contain all the information in the input records or only selected fields from the input records.

In addition to creating the output file, the sort program, if requested by the sort sequence specifications, produces a report. This report may consist of images of all the sort sequence specifications, as well as diagnostic and status messages. Input to sort consists of information from OCL statements, the sort sequence specifications, and the input file.

I/O STORAGE REQUIREMENTS

The minimum region size for the sort program is 14K (14,336) bytes of main storage.

SYSTEM CONFIGURATION

The sort program runs on all models of System/34.

<i>Phase</i>	<i>Description</i>
--------------	--------------------

- | | |
|----------|---|
| Phase 0A | <ul style="list-style-type: none"> ● Partially initializes COMMON. ● Determines the location of the sort sequence specification statements. ● Determines if input, output, and work file (FILE) statements are present. |
| Phase 0B | <ul style="list-style-type: none"> ● Completes initialization of COMMON. ● Allocates and dummy opens the input file. ● Reads alternate collating sequence (ALTSEQ) statements. ● Allocates the work file, if a work file (FILE) statement is present. |
| Phase 0C | <ul style="list-style-type: none"> ● Reads the sequence specification statements and checks them for errors. ● Builds the select/build routine. ● Creates the summary table. |

<i>Phase</i>	<i>Description</i>
--------------	--------------------

- | | |
|----------|--|
| Phase 0D | <ul style="list-style-type: none"> ● Designs the sort by determining specific fields in COMMON. |
| Phase 0E | <ul style="list-style-type: none"> ● Allocates the output file, if the output file does not overlay the input file. ● Allocates the work file, if a work file (FILE) statement is not present. ● Determines if the work file is large enough to hold at least one work block. |
| Phase 0G | <ul style="list-style-type: none"> ● Handles error conditions for generation phases. |
| Phase 1 | <ul style="list-style-type: none"> ● Prepares variable-length strings and places them in the work file. ● Reads the input records and builds a work record for each input record selected for inclusion in the sort. |
| Phase 2 | <ul style="list-style-type: none"> ● Merges strings until only one pass remains. |
| Phase 3 | <ul style="list-style-type: none"> ● Performs the final pass and places the sorted records into the output file. |
| Phase 4 | <ul style="list-style-type: none"> ● Ends the job (cancels the job if any errors are present or indicates a successful completion). |

Method of Operation

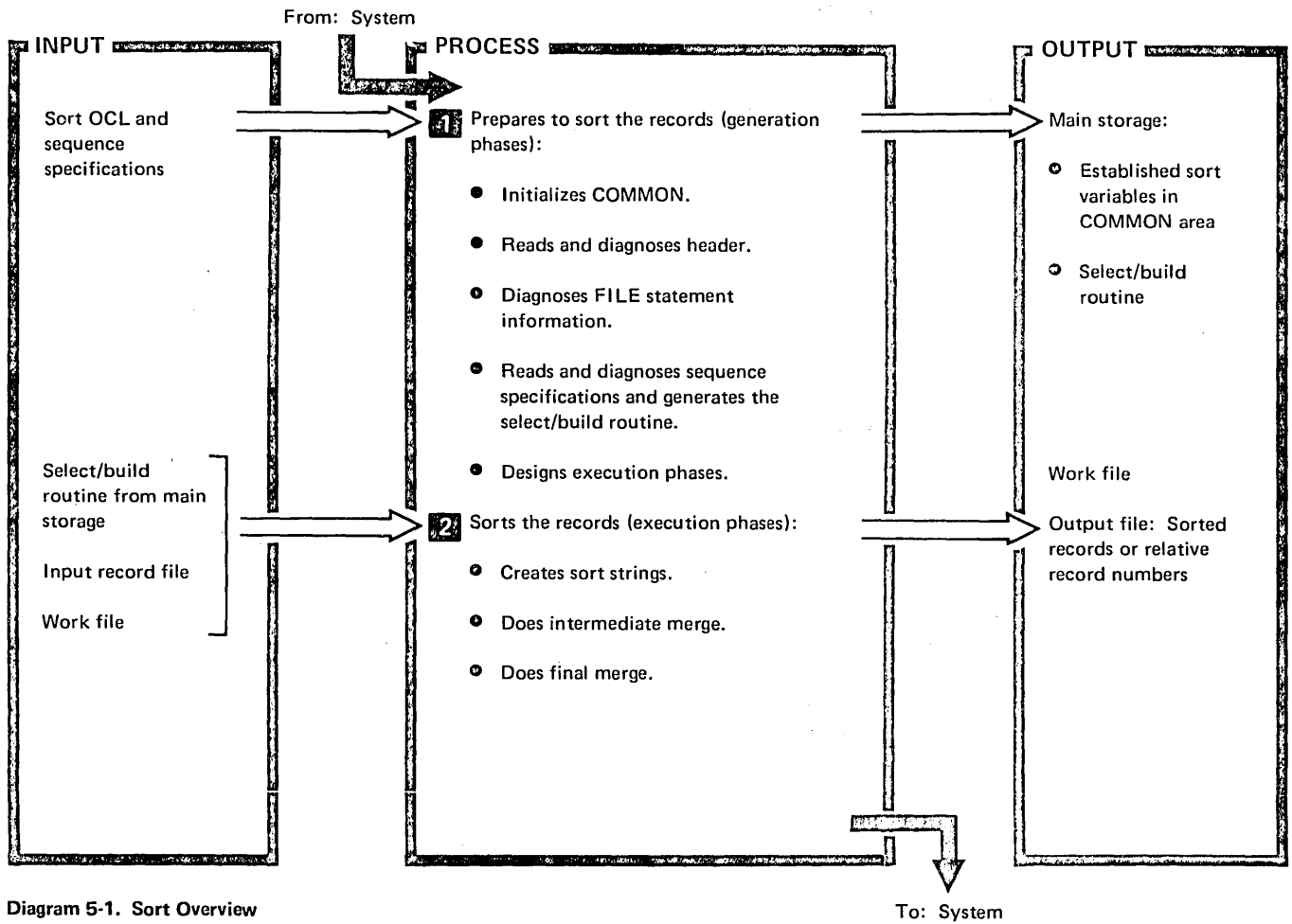


Diagram 5-1. Sort Overview

Program Organization

Diagram 5-2 shows the program organization of the sort program generation phases.

Diagram 5-3 shows the program organization of the sort program execution phases.

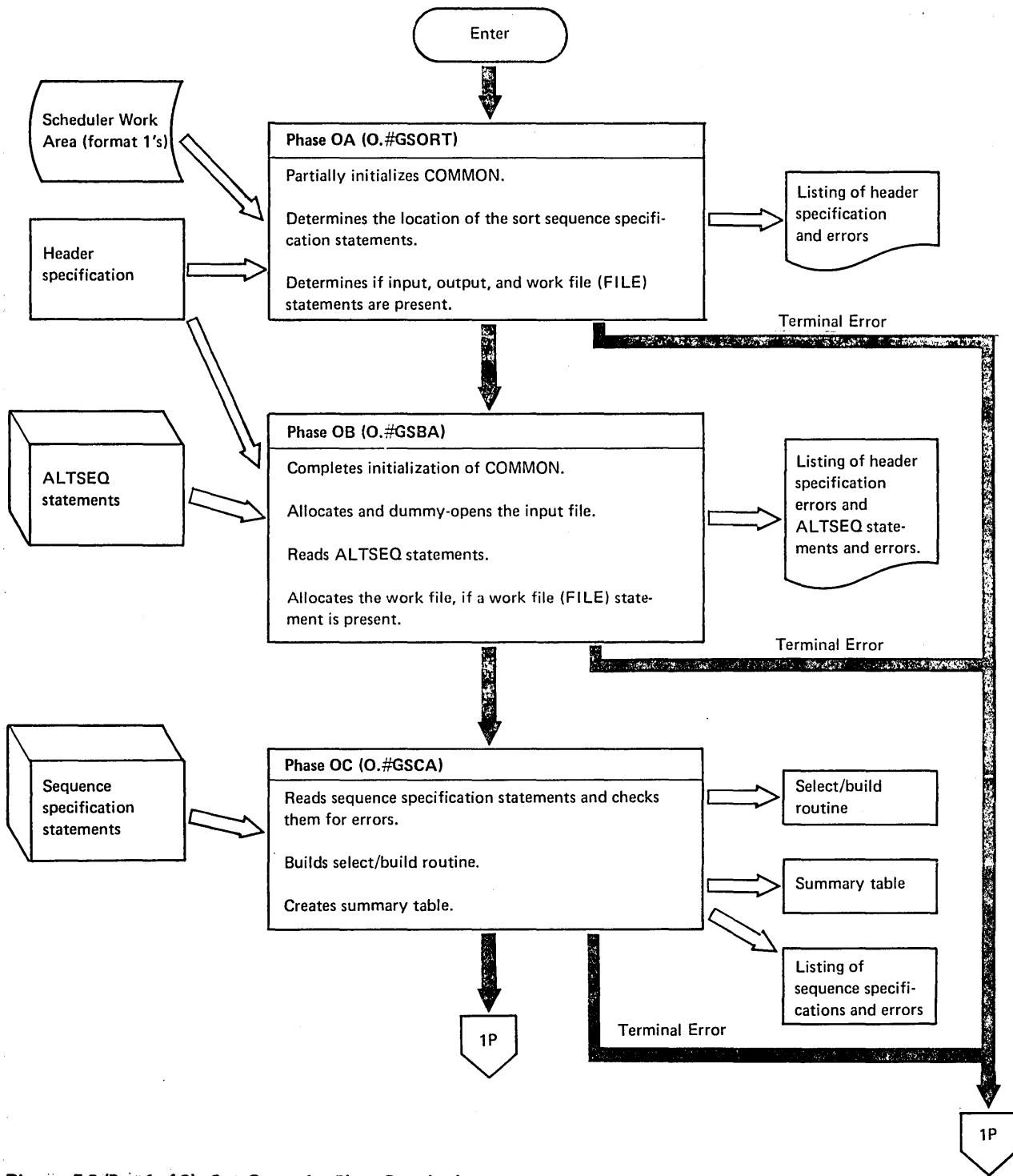


Diagram 5-2 (Part 1 of 2). Sort Generation Phase Organization

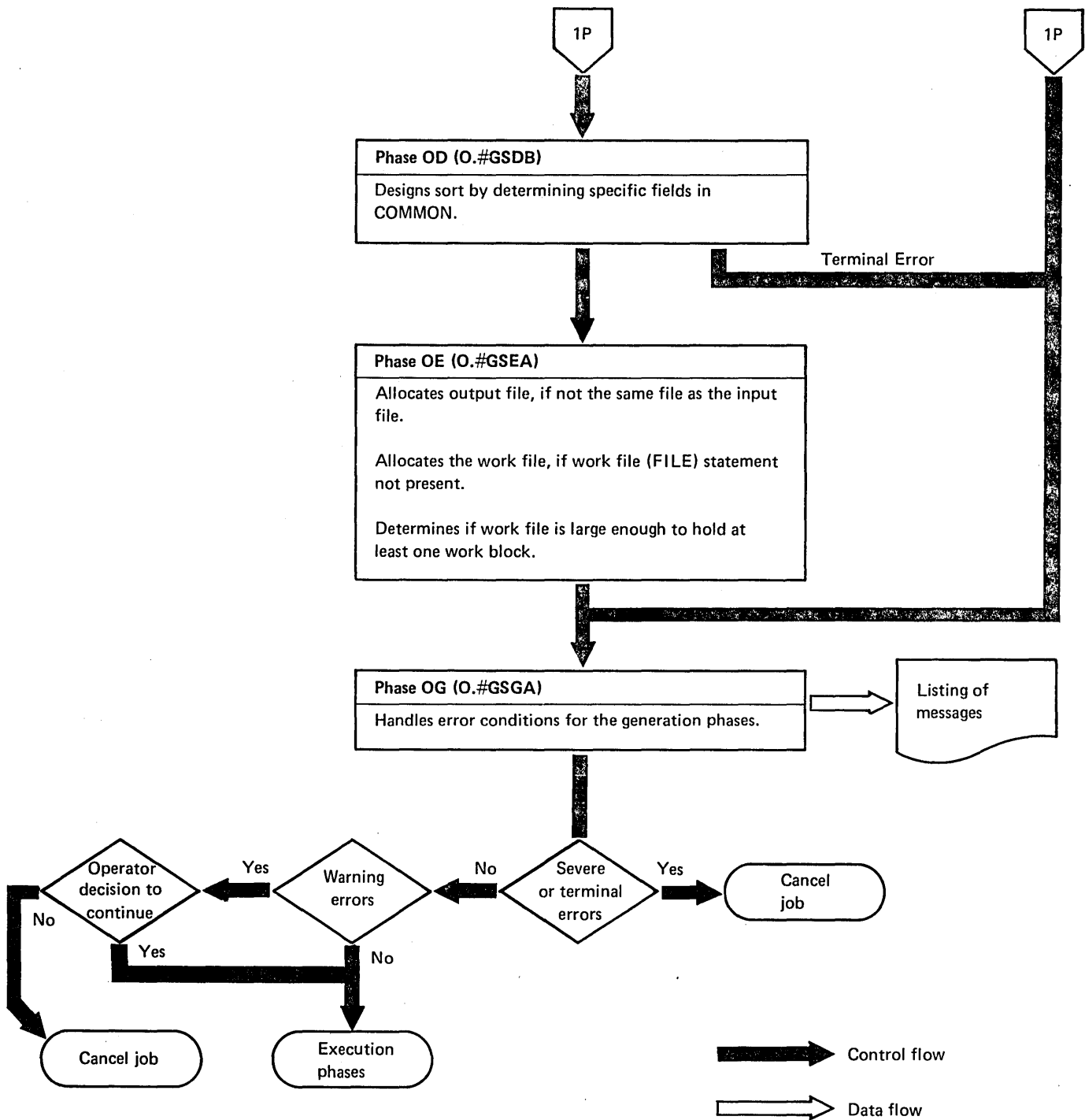


Diagram 5-2 (Part 2 of 2). Sort Generation Phase Organization

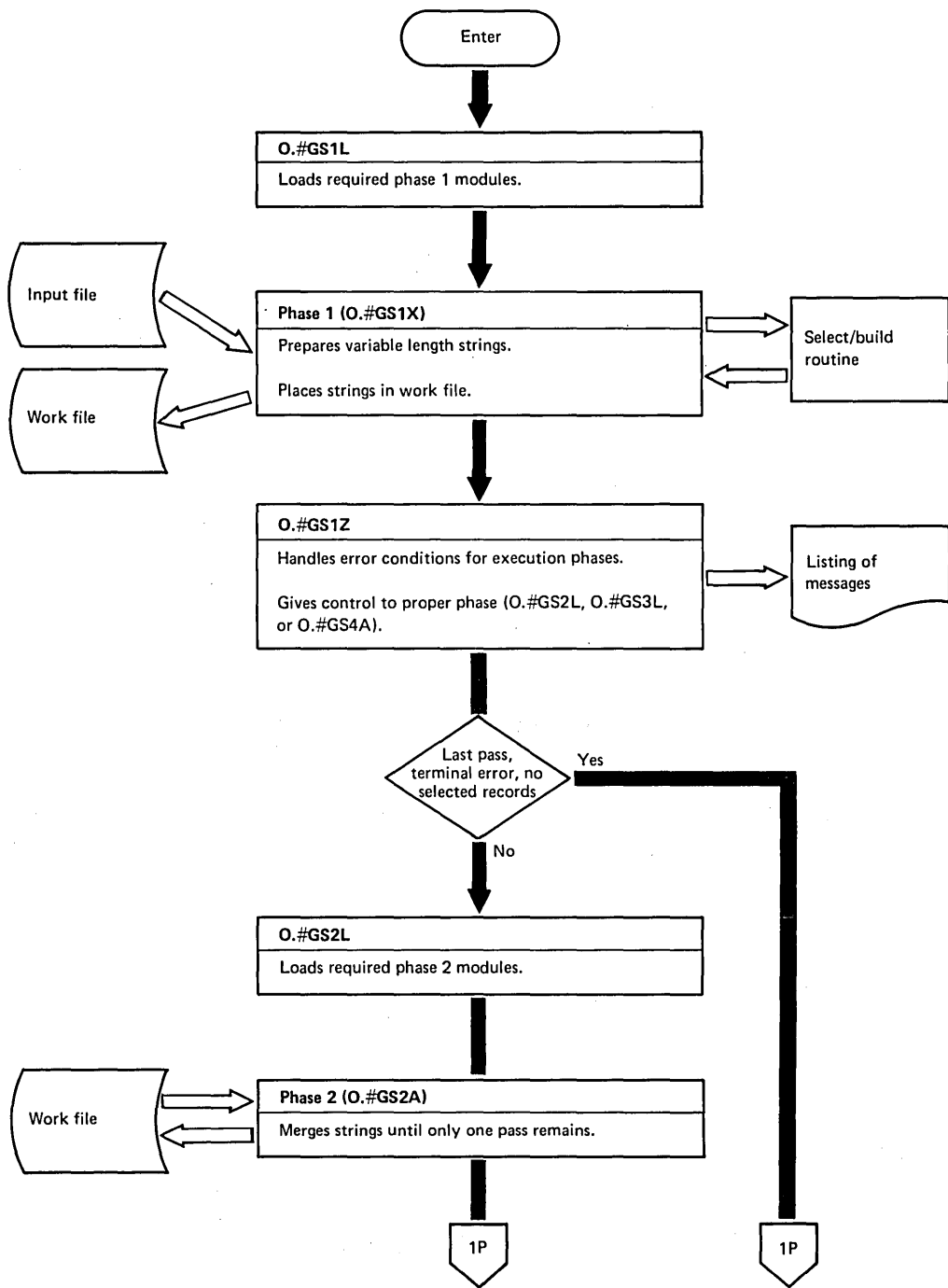


Diagram 5-3 (Part 1 of 3). Sort Execution Phase Organization

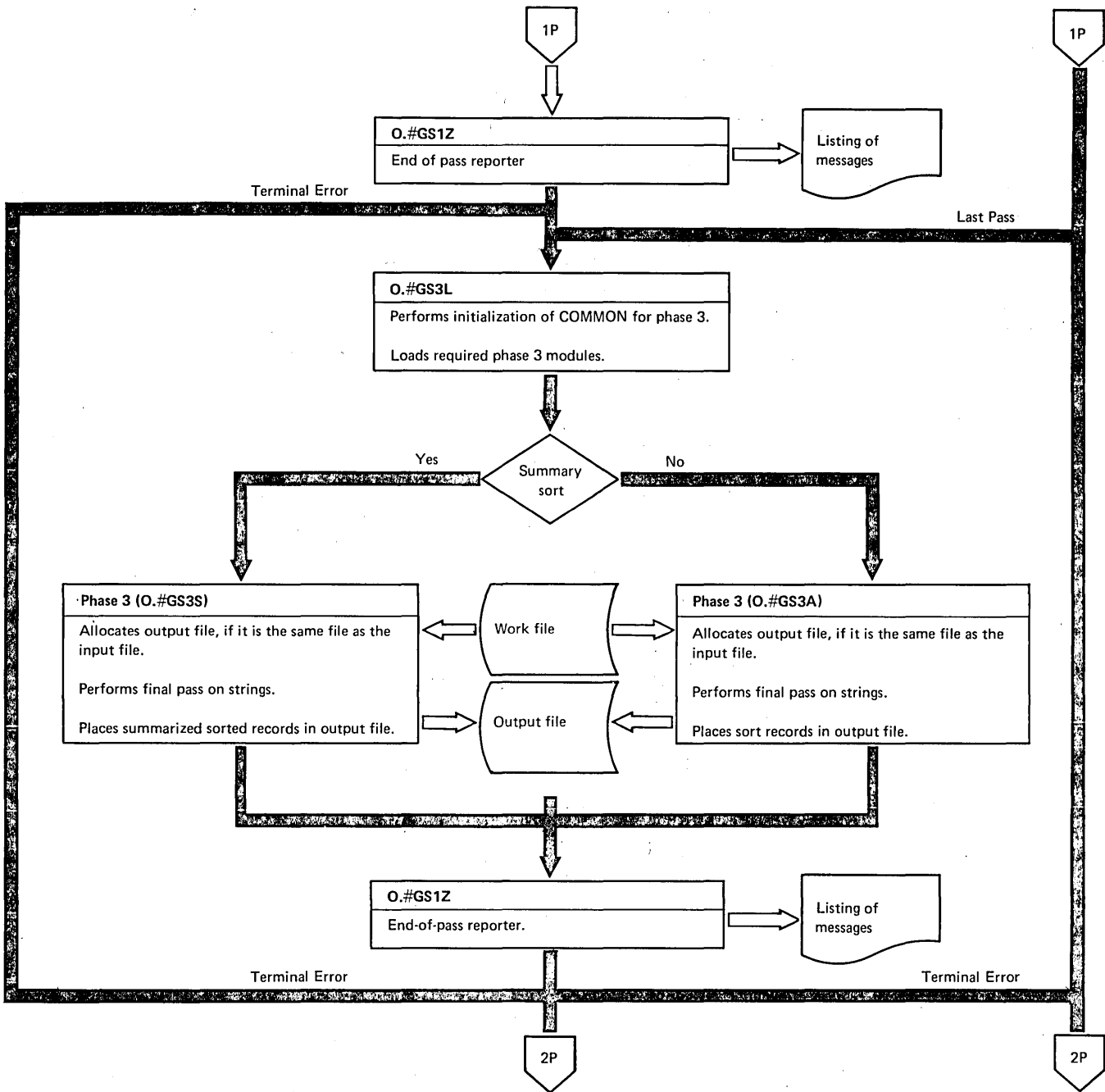


Diagram 5-3 (Part 2 of 3). Sort Execution Phase Organization

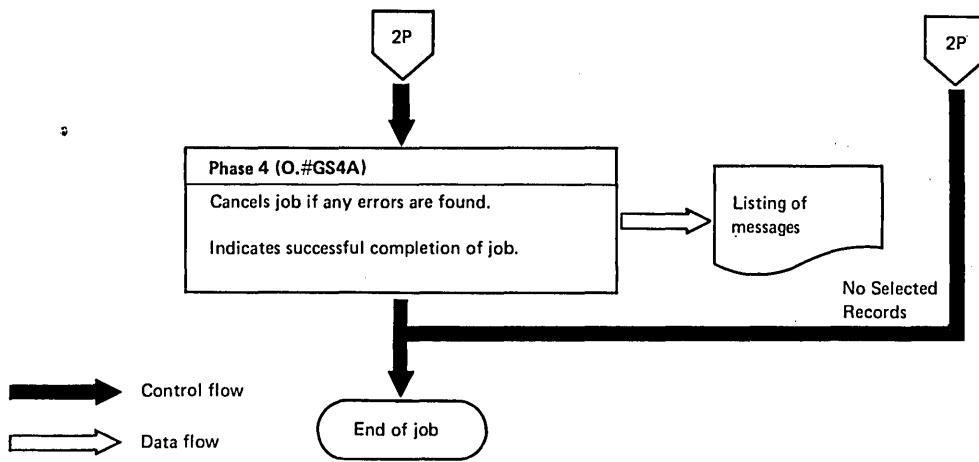


Diagram 5-3 (Part 3 of 3). Sort Execution Phase Organization

Directory

This section contains a quick-reference table that can be used to find sort phases on microfiche. The module names are indexed on the microfiche records.

The table consists of the following:

- *Module Name* is the symbolic label identifying the module in the program listings and on microfiche.
- *Major Functions* is a brief list of specific functions the module performs within the program.

Module Name

Major Functions

- | | |
|----------|---|
| 0.#GS\$A | <ul style="list-style-type: none"> • Prints sort's COMMON area with labels after phases 0A, 0B, 0D, 0E, and all passes. |
| @GSZE | <ul style="list-style-type: none"> • Converts 4-bit groups into their equivalent 8-bit EBCDIC values. |
| @GSZF | <ul style="list-style-type: none"> • Converts 1 byte of storage to printable EBCDIC values, 1 bit at a time. |
| 0.#GS\$D | <ul style="list-style-type: none"> • Phase 0D COMMON constants • Serves as input to 0.#GS\$A when dumping COMMON. |
| 0.#GS\$E | <ul style="list-style-type: none"> • Determines if there is enough available storage to load 0.#GS\$A (sort COMMON dump routine). • If there is enough available storage, loads 0.#GS\$A. |
| 0.#GS\$X | <ul style="list-style-type: none"> • Phase 1, 2, and 3 COMMON constants. • Serves as input to 0.#GS\$A when dumping COMMON. |
| #GSAA | <ul style="list-style-type: none"> • Determines the region size that the sort program has available to execute in. • Determines if the statement read by #GSAB is a header specification statement. • Partially diagnoses the header statement and issues messages if any errors are detected. |

Module Name

Major Functions

- | | |
|--------------------|--|
| #GSAB | <ul style="list-style-type: none"> • Determines the location of the sort sequence specification statements. • Reads first sort sequence specifications statement. |
| #GSAF | <ul style="list-style-type: none"> • Determines if there are input, output, and work file (FILE) statements. • Determines if the input and output files are the same file. |
| 0.#GSBA (phase 0B) | <ul style="list-style-type: none"> • Finishes diagnosing the header statement and issues messages if any errors are detected. • Allocates and dummy opens the input file. • Allocates the work file, if work file (FILE) statement present. • Reads the statement following the header statement. • If ALTSEQ specified, loads and calls 0.#GSBC. |
| 0.#GSCA (phase 0C) | <ul style="list-style-type: none"> • Reads and diagnoses sort sequence specifications for one sort run. |
| #GSCA1 | <ul style="list-style-type: none"> • Initializes the COMMON fields used by Phase 0C. |
| #GSCA2 | <ul style="list-style-type: none"> • If the print option entry is 0, prints the <i>current</i> statement. • Partially diagnoses the current statement. • Determines which phase 0C module(s) (0.#GSCE, 0.#GSCF, 0.#GSCK, 0.#GSCL, and 0.#GSCZ) to call to finish diagnosing the current statement and to generate code for it. |

<i>Module Name</i>	<i>Major Functions</i>	<i>Module Name</i>	<i>Major Functions</i>
#GSCA5	<ul style="list-style-type: none"> ● Moves code generated in 0.#GSCE and 0.#GSCF to the generated code area. ● Determines if there is enough storage available for more code to be generated. 	0.#GSCC	<ul style="list-style-type: none"> ● Moves the select/build routine to its final position in storage. ● Moves the summary table behind select/build routine if summary sort is used.
#GSCA6	<ul style="list-style-type: none"> ● Determines the zone of a given byte by testing the zone portion of that byte. 	0.#GSCE	<ul style="list-style-type: none"> ● Diagnoses the sequence specification statements to determine if they are valid include or omit statements.
#GSCA8	<ul style="list-style-type: none"> ● Builds the error table starting with the highest available address. ● Checks error table; if full, passes control to 0.#GSCZ. 	0.#GSCF	<ul style="list-style-type: none"> ● Generates code segments from the information on the include/omit statements. ● Diagnoses the sequence specification statements to determine if they are valid field statements.
0.#GSBC	<ul style="list-style-type: none"> ● Reads and diagnoses ALTSEQ statements and issues messages if any errors are detected. ● Modifies the 256-byte alternate collating sequence table as specified on the ALTSEQ statements that are read in. 	0.#GSCK	<ul style="list-style-type: none"> ● Generates code segments from the information on the field statements. ● Diagnoses include/omit record type specifications with a factor 2 keyword.
@GSDC	<ul style="list-style-type: none"> ● Computes the initial area in phase 1 to be assigned to the internal sort area. 	0.#GSCL	<ul style="list-style-type: none"> ● Generates a valid include/omit sequence specification statement. ● Calculates module length.
@GSEG	<ul style="list-style-type: none"> ● Determines the space needed for the work file and uses special allocate to automatically allocate it. 	@GSCA9	<ul style="list-style-type: none"> ● Diagnoses errors in the factor 1 and factor 2 field lengths for include/omit record type specifications.
@GSZB	<ul style="list-style-type: none"> ● Converts an unsigned zoned decimal number (up to 7 bytes long) to a 3-byte hex number. 		<ul style="list-style-type: none"> ● Diagnoses errors in the from and to location entries for field specifications.
@GSZC	<ul style="list-style-type: none"> ● Converts a 3-byte hex number to a 7-byte zoned decimal number and concatenates a sign byte to the result. 		<ul style="list-style-type: none"> ● Calculates displacement values for include, omit, and field statements.
0.#GSCB	<ul style="list-style-type: none"> ● Issues error messages based on the error table generated in phase 0C. ● Determines if any terminal errors were diagnosed by phase 0C; if they were, passes control to 0.#GSGA. 	@GSCA7	<ul style="list-style-type: none"> ● Converts the to and from fields of factors 1 and 2, and the to and from location entries for field specifications, to binary.

<i>Module Name</i>	<i>Major Functions</i>	<i>Module Name</i>	<i>Major Functions</i>
0.#GSCZ	<ul style="list-style-type: none"> • Completes select/build code. • Moves the summary table to phase 1 location behind COMMON. 	0.#GS1X (phase 1)	<ul style="list-style-type: none"> • Performs a tournament sort. • Produces variable length strings of sequenced work records.
0.#GSDA	<ul style="list-style-type: none"> • Determines the active program lengths of modules 0.#GS1X, 0.#GS2A, 0.#GS3A, and 0.#GS3S. 		<ul style="list-style-type: none"> • Places a string of a variable number of work record blocks onto the work file. • At end of pass for a summary sort, moves the summary table from the next byte after the select/build code to the next byte after COMMON.
0.#GSDB (phase 0D)	<ul style="list-style-type: none"> • Designs the execution phases of the sort program (phase 1, phase 2, and phase 3). • Checks for terminal error in design of sort; if found, passes control to 0.#GSGA. 	0.#GS1Z	<ul style="list-style-type: none"> • Issues messages indicating errors found in and information determined in the preceding execution phase.
0.#GSEA (phase 0E)	<ul style="list-style-type: none"> • Allocates output file, if not the same file as the input file. • If work file (FILE) statements omitted, calls automatic work file allocation (@GSEG). 	0.#GS2A (phase 2)	<ul style="list-style-type: none"> • Merges strings created in phase 1 until the total number of strings is less than or equal to the sort's order of merge.
0.#GSGA (phase 0G)	<ul style="list-style-type: none"> • Issues messages indicating errors found in and information determined in the preceding generation phases. 	0.#GS2L	<ul style="list-style-type: none"> • Loads into storage the required phase 2 modules for 0.#GS2A.
0.#GSORT (phase 0A)	<ul style="list-style-type: none"> • Initial sort load module. • Contains COMMON and initializes portions of it. 	0.#GS3A (phase 3)	<ul style="list-style-type: none"> • Allocates the output file, if it is the same file as the input file. • Performs last pass of the merge. • Places the sorted records consecutively in the output file.
0.#GSZA	<ul style="list-style-type: none"> • Moves a given number of bytes from one area in storage to another. 	0.#GS3L	<ul style="list-style-type: none"> • Performs initialization for the phase 3 modules (0.#GS3A/0.#GS3S).
0.#GS1D	<ul style="list-style-type: none"> • Calculates and places in COMMON the sector address (SSS) of the current work block. • Calculates and places in COMMON the SSS of the next work block. 		<ul style="list-style-type: none"> • Loads into storage the modules required by 0.#GS3A/0.#GS3S to perform phase 3. • If the number of remaining strings is less than the order of merge, redesigns the sort for phase 3.
0.#GS1L	<ul style="list-style-type: none"> • Performs initialization for 0.#GS1X. • Loads into storage the required phase 1 modules for 0.#GS1X. 		

*Module
Name*

Major Functions

0.#GS3S
(phase 3)

- Allocates the output file, if it is the same file as the input file.
- Performs last pass of the merge.
- If no summary table exists, deletes duplicate records.
- If a summary table exists, summarizes duplicate records as specified.
- Places the sorted records consecutively in the output file.

@GSZD

- Divides a 3-byte hex number into another 3-byte hex number.

@GSZM

- Multiplies two 3-byte hex numbers.

0.#GS4A
(phase 4)

- Issues displayed message and ends the job after unsuccessful completion via the SYSLOG transient routine.
- Terminates the sort program after successful completion.

0.#GS9G

- Provides the logical get I/O function for the work file.

0.#GS9I

- Reads sort sequence specification statements from a procedure member or entered through the display station keyboard.

0.#GS9P

- Provides the logical put I/O function for the work file.

0.#GS9S

- Reads sort sequence specification statements from a source member.

Data Areas

This section describes data areas used by the sort program:

<i>Data Area</i>	<i>Description</i>
COMMON	<p>COMMON is a 256-byte-plus-two-DTF-lengths interphase table used by phases in both generation and execution portions of sort. COMMON is loaded into storage as the first 256-bytes-plus-two-DTF-lengths of the phase 0A load member.</p> <p>#GSC in COMMON, the first 4 bytes of COMMON, contains C'#GSC'. This field serves as an identification for sort COMMON.</p> <p>The macro DSEQU generates the labels required by each individual module to access fields it needs in COMMON. To locate these labels and the offsets for them, look up #GSC in the cross-reference table of any module.</p>
PHASE@	<p>PHASE@ in COMMON contains either the address of the current phase or the address of the next phase to be loaded. Immediately before control is to be passed to the next phase, PHASE@ is loaded with the address of where to load that phase. During execution of the current phase (before the point where control is to be passed to the next phase), PHASE@ contains the address of where the current executing phase was loaded.</p>
ERROR TABLE	<p>The error table is built backwards by 0.#GSCA; that is, the first entry is at a higher location in storage than the second entry. A 3-byte entry is placed in the table for each error found on the sequence specification statements. The first 2 bytes contain the statement number where the error was found; the third byte contains the error number.</p> <p>BEGER1 in COMMON contains the address of the error table.</p>

Data Area

Description

ALTERNATE COLLATING SEQUENCE TABLE	<p>The alternate collating sequence table, if requested, is located in the last 256 bytes in main storage until phase 2 and contains all the hex values. The table is modified by 0.#GSBC according to the ALTSEQ statements.</p>																
OLDS STRUCTURED ELEMENT ARRAY	<p>The OLDS structured element array, built by 0.#GS9G, contains information on a maximum of 18 old sort strings. OLDS@ in COMMON contains the address of this array. Each 16-byte entry contains:</p> <table border="1"> <thead> <tr> <th><i>Byte</i></th> <th><i>Contents</i></th> </tr> </thead> <tbody> <tr> <td>0-2</td> <td>Sector address (SSS) of current block of strings</td> </tr> <tr> <td>3-4</td> <td>Total number of records in block</td> </tr> <tr> <td>5-6</td> <td>Number of records taken from block</td> </tr> <tr> <td>7-8</td> <td>Buffer address</td> </tr> <tr> <td>9-10</td> <td>Block address</td> </tr> <tr> <td>11-12</td> <td>Address of current record in block</td> </tr> <tr> <td>13-15</td> <td>Sector address (SSS) of next block of string</td> </tr> </tbody> </table>	<i>Byte</i>	<i>Contents</i>	0-2	Sector address (SSS) of current block of strings	3-4	Total number of records in block	5-6	Number of records taken from block	7-8	Buffer address	9-10	Block address	11-12	Address of current record in block	13-15	Sector address (SSS) of next block of string
<i>Byte</i>	<i>Contents</i>																
0-2	Sector address (SSS) of current block of strings																
3-4	Total number of records in block																
5-6	Number of records taken from block																
7-8	Buffer address																
9-10	Block address																
11-12	Address of current record in block																
13-15	Sector address (SSS) of next block of string																
AVAIL TABLE	<p>0.#GS9G builds the AVAIL table so that the first entry is at a higher location in storage than the second entry. The address of the table is found at @AVAIL in COMMON. Each 5-byte entry contains this information about the available blocks on disk:</p> <table border="1"> <thead> <tr> <th><i>Byte</i></th> <th><i>Contents</i></th> </tr> </thead> <tbody> <tr> <td>0-2</td> <td>Sector address (SSS) of block</td> </tr> <tr> <td>3-4</td> <td>Total number of records in block</td> </tr> </tbody> </table>	<i>Byte</i>	<i>Contents</i>	0-2	Sector address (SSS) of block	3-4	Total number of records in block										
<i>Byte</i>	<i>Contents</i>																
0-2	Sector address (SSS) of block																
3-4	Total number of records in block																

Data Area

Description

**SUMMARY
TABLE**

This table, built by 0.#GSCF, contains information about the type, length, and location of the summary fields within the work record. The table consists of 1-25 entries. If bit 3 of ATTRQ1 in COMMON is 1, then the first entry is that of the overflow indicator field. Otherwise, the first entry is the first summary data field. The entries contain:

Overflow Indicator

Byte Contents

- 0 XL1'FF'=Overflow indicator
- 1 Overflow indicator character (from record character column)
- 2-3 Displacement of field from start of work record

First Summary Field

Byte Contents

- 0 XL1'00'=Digit or unpacked decimal
XL1'80'=Character (integer)
XL1'40'=Packed decimal
- 1 Field length minus 1
- 2-3 Displacement of first field from start of work record (rightmost byte)

Following Summary Fields

Byte Contents

- 0 Type
- 1 Field length minus 1
- 2-3 Displacement of this field from previous field (rightmost byte)

Data Area

Description

**TABLE of WORK
FILE EXTENTS**

0.#GSBA or 0.#GSEA builds this 21-byte table in COMMON. The high-order byte is MVFTBL in COMMON. The first 3 bytes in the table contain three 1-byte device addresses (MVFQ1-MVFQ3), one address for each extent. The remainder of the table gives 6 bytes of information for each extent in this format:

Byte Contents

- 0-2 Disk SSS
- 3-5 Accumulated number of sectors with this extent

The last 3 bytes of each 6-byte entry are converted by 0.#GSDA. Each entry then contains:

Byte Contents

- 0-2 Disk SSS
- 3-5 Accumulated number of work file blocks with this extent (MVF#1-MVF#3)

Diagnostic Aids

This section lists the displayed and printed messages diagnosed by sort and the modules that diagnose them.

MESSAGES

Message Module

SORT-7002 SORT procedure
 SORT-7004 SORT procedure
 SORT-7005 SORT procedure
 SORT-7007 SORT procedure
 SORT-7101 #GSORT
 SORT-7102 #GSORT
 SORT-7103 #GSORT
 SORT-7104 #GSORT
 SORT-7105 #GSORT
 SORT-7150 #GSCB
 SORT-7151 #GSBA
 SORT-7152 #GSBA
 SORT-7154 #GSBA
 SORT-7155 #GSBA
 SORT-7156 #GSBA
 SORT-7157 #GSBA
 SORT-7158 #GSBA
 SORT-7159 #GSBA
 SORT-7160 #GSBA
 SORT-7161 #GSBA
 SORT-7162 #GSBA
 SORT-7176 #GSBC
 SORT-7177 #GSBC
 SORT-7178 #GSBC
 SORT-7179 #GSBC
 SORT-7181 #GSBC
 SORT-7202 #GSCA
 SORT-7204 #GSCA
 SORT-7206 #GSCA
 SORT-7208 #GSCA
 SORT-7210 #GSCA CK
 SORT-7212 #GSCA
 SORT-7214 #GSCA
 SORT-7216 #GSCZ
 SORT-7218 #GSCA
 SORT-7220 #GSCA
 SORT-7222 #GSCF
 SORT-7224 #GSCK CA9
 SORT-7225 #GSCK
 SORT-7226 @CSCA9
 SORT-7228 @GSCA9

Message

Module

SORT-7230 #GSCA
 SORT-7232 #GSCE
 SORT-7234 #GSCE
 SORT-7236 #GSCE
 SORT-7238 @GSCA9
 SORT-7240 @GSCA9
 SORT-7242 @GSCA9
 SORT-7244 #GSCF CA9
 SORT-7246 #GSCA
 SORT-7248 #GSCF
 SORT-7250 #GSCF
 SORT-7252 @GSCA9
 SORT-7254 #GSCF
 SORT-7256 @GSCA9
 SORT-7258 @GSCA7
 SORT-7262 @GSCA9
 SORT-7264 #GSCB
 SORT-7266 #GSCE
 SORT-7268 @GSCA9
 SORT-7276 #GSCF
 SORT-7278 #GSCF
 SORT-7280 #GSCF
 SORT-7282 #GSCZ
 SORT-7284 #GSCZ
 SORT-7286 #GSCF
 SORT-7288 @GSCA9
 SORT-7391 #GSDB
 SORT-7395 @GSEG
 SORT-7401
 SORT-7402
 SORT-7403
 SORT-7404
 SORT-7422 #GSCB
 SORT-7423 #GS9I
 SORT-7425 #GSCB DB EG
 SORT-7450
 SORT-7451
 SORT-7452
 SORT-7453
 SORT-7461
 SORT-7462
 SORT-7600
 SORT-7601
 SORT-7602
 SORT-7603
 SORT-7620 #GS1X
 SORT-7681 #GS1Z
 SORT-7690
 SORT-7691

Figure 5-1 (Part 1 of 2). Sort Message - to - Module Cross Reference

Message	Module
SORT-7692	
SORT-7693	#GS1X
SORT-7694	
SORT-7695	
SORT-7721	#GS9S
SORT-7722	#GSCB
SORT-7723	#GS9I
SORT-7724	#GS1X
SORT-7725	#GSCB DB EG
SORT-7727	#GS\$A
SORT-7728	#GS1X
SORT-7729	#GS1X 9G 9P
SORT-7730	#GS3A 3S
SORT-7731	#GS1D
SORT-7732	#GS3A 3S
SORT-7733	@GSEG
SORT-7781	#GSCB, 1Z, 2A, 3L, 3A, 3S, 4A, 9G
SORT-7800	#GS\$E
SORT-7801	
SORT-7901	
SORT-7902	
SORT-7928	#GS1X
SORT-7929	#GS1X 9G 9P
SORT-7930	#GS3A 3S
SORT-7931	#GS1D
SORT-7932	#GS3A 3S

Figure 5-1 (Part 2 of 2). Sort Message - to - Module Cross Reference

PROCEDURES

When the SORT command is entered, the following procedure is executed:

```
// MEMBER PROGRAM1-#GS#MM
// MEMBER USER1-#GS#MM
// LOAD #GSORT
// FILE NAME-INPUT,LABEL-input file label,DISP-SHR
// FILE NAME-OUTPUT,LABEL-output file label,RECORDS-number of
  records
// RUN
// SOURCE source member [,user library name]
```

Appendix. Acronyms and Abbreviations

The following acronyms and abbreviations are used in this manual:

APAR	Authorized Program Analysis Report	SAV	Screen save/restore area
CDB	C-spec data block	SCB	Storage control block
CSA	C-spec specification area	SDA	Screen design aid
CSB	C-spec specification block	SDB	Screen data buffer
CSP	C-specification	SEU	Source entry utility
DFU	Data file utility	SFGR	Screen format generation routine
DSA	Data specification area	SRB	Storage request block
DSB	Data specification block	SSA	Screen specification area
DTF	Define the file	SSB	Screen specification block
EJ	End of WSU job	SSP	System Support Program Product
EOJ	End of job	SSS	The format for representing the physical address of a data field (sector) on disk.
ES	End of sequence	TRN	Transient
EW	End of work session	TSA	Transient specification area
FDB	File data buffer	TSB	Transient specification block
FIA	Format index area	VTOC	Volume table of contents
FSA	File specification area	WCA	Work station control area
FSB	File specification block	WCB	Work station control block
hex	hexadecimal	WCR	Work station control record
HIPO	Hierarchy, plus input, process, output identification	WDB	Work session data block
ID	identification	WSDTF	Work station DTF
IPB	Initialization parameter block	WSU	Work station utility
IJ	Job initiation		
IW	Work session initiation		
JCR	Job control record		
JDB	Job data block		
K	1024 (bytes)		
MDB	Message data block		
MIA	Master index area		
MIC	Message identification code		
MSA	Message specification area		
MSB	Message specification block		
NEP	Never ending program		
OCL	Operation control language		
PPSA	Procedure parameter save area		
PSA	Process specification area		
PSB	Process specification block		
QSA	Queued save area		
QSB	Queued save area blocks		
RID	Record identifier		
RSA	Routine specification area		
RSB	Routine specification block		
RTN	WSU execution routine		

READER'S COMMENT FORM

Please use this form only to identify publication errors or request changes to publications. Technical questions about IBM systems, changes in IBM programming support, requests for additional publications, etc, should be directed to your IBM representative or to the IBM branch office nearest your location.

Error in publication (typographical, illustration, and so on). **No reply.**

Page Number *Error*

Inaccurate or misleading information in this publication. Please tell us about it by using this postage-paid form. We will correct or clarify the publication, or tell you why a change is not being made, provided you include your name and address.

Page Number *Comment*

Note: All comments and suggestions become the property of IBM.

Name _____

Address _____

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

Cut Along Line

IBM System/34 Utilities Program Product Program Logic Manual (File No. S34-32) Printed in U.S.A. LY21-0563-0

Fold

Fold

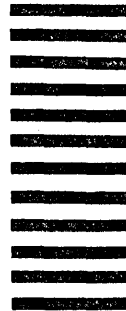
FIRST CLASS
PERMIT NO. 40
ARMONK, N. Y.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
General Systems Division
Development Laboratory
Publications, Dept. 245
Rochester, Minnesota 55901



Fold

Fold



International Business Machines Corporation

**General Systems Division
4111 Northside Parkway N.W.
P.O. Box 2150
Atlanta, Georgia 30301
(U.S.A. only)**

**General Business Group/International
44 South Broadway
White Plains, New York 10601
U.S.A.
(International)**