SY28-0650-0

Systems

OS/VS2 TSO Terminal Monitor Program and Service Routines Logic

VS2 Release 1

IBM

First Edition

This edition applies to release 1 of OS/VS2 and to all subsequent releases until otherwise indicated in new editions or Technical Newletters. Changes are continually made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest IBM System/360 and System/370 Bibliography, Order No. GN20-0360, for the editions that are applicable and current.

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

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Publications Development, Department D58, Building 706-2, PO Box 390, Poughkeepsie, N. Y. 12602. Comments become the property of IBM.

Preface

This publication describes the internal logic and organization of the terminal monitor program (TMP) and the TSO service routines. It is written for persons who maintain or modify TSO and is not necessary for persons who use TSO to process programs or who write programs that are processed by TSO.

External information is available in the following publications:

- OS/VS2 TSO Guide, GC28-0644, which describes what TSO is and what it can do.
- OS/VS2 TSO Terminal User's Guide, GC28-0645, which describes typical operations that a terminal user may perform.
- OS/VS2 TSO Command Language Reference, GC28-0646, which describes the commands, subcommands, and operands of the TSO command language.
- OS/VS2 TSO Guide to Writing a Terminal Monitor Program or a Command Processor, GC28-6764, which tells how to write foreground programs that will interact with or replace those supplied by TSO.

This book consists of six parts, which are preceded by an introduction and followed by a glossary and an index. Each part is really a separate program logic manual, with its own introduction, method of operation, program organization, directory, data areas, and diagnostic aids sections.

The "Introduction" describes the purpose and use of the terminal monitor program (TMP) and the TSO service routines and describes their relationship to the system. You should understand the "Introduction" before reading any of the other parts.

"Part 1: Terminal Monitor Program" describes the internal logic and organization of the terminal monitor program and its relationship to other programs including the TSO Control Program, the LOGON/LOGOFF Scheduler, the TSO command processors, and the TSO service routines.

"Part 2: Terminal I/O Service Routines" describes the internal logic and organization of STACK, PUTLINE, GETLINE, and PUTGET and their relationship to other programs including the TSO Control Program, the Terminal Monitor Program and the TSO command processors.

"Part 3: Command Scan and Parse Service Routines" describes the internal logic and organization of command scan and parse and their relationship to other programs including the terminal monitor program and the TSO command processors.

"Part 4: Dynamic Allocation Routines" describes the internal logic and organization of the dynamic allocation interface routine (DAIR) and the SVC 99 dynamic allocation routines and their relationship to each other and to other programs, including the LOGON/LOGOFF scheduler, the terminal monitor program, the TSO command processors.

"Part 5: Default Service Routine" describes the internal logic and organization of default and its relationship to other programs including PUTLINE, PUTGET, and the catalog information routine.

"Part 6: Catalog Information Routine" describes the internal logic and organization of the catalog information routine and its relationship to other programs including the routines invoked by the LOCATE macro instruction.

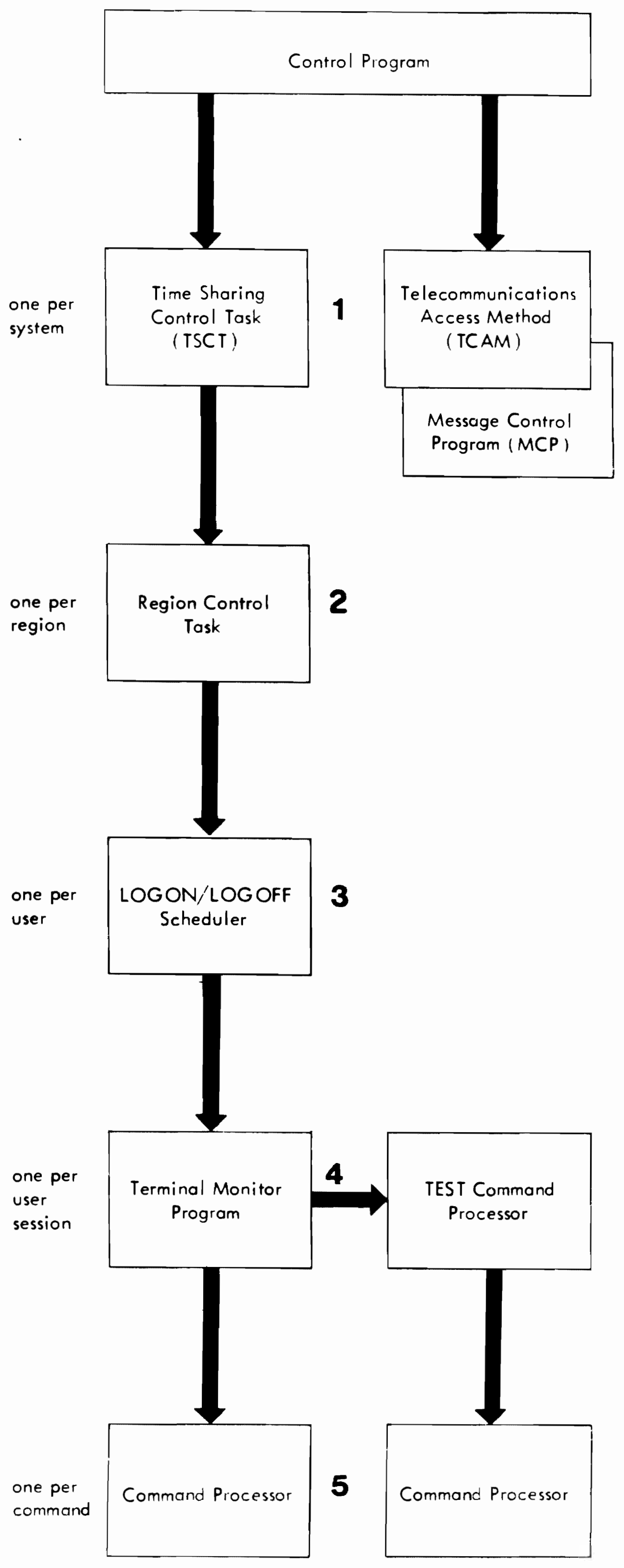
Contents

Illustrations

Figures

Charts

Method of Operation Diagrams



1 At the highest level are the Time Sharing Control Task (TSCT) and the Message Control Program (MCP). The TSCT handles system-wide functions such as the initialization procedures required when the operator starts time sharing, and the swapping of foreground jobs in and out of real storage. The MCP is a part of the Telecommunications Access Method (TCAM) and handles I/O for all terminals.

2 Below the TSCT is a Region Control Task (RCT) for each foreground region. The RCT supervises the foreground jobs assigned to its region, including the quiescing and restoring of job activities before and after swapping.

3 The Logon/Logoff Scheduler (LOGON) is invoked by the RCT whenever a user wants to log on or off the system and defines his foreground job using parameters in the logon procedure, user profile, and operands of the LOGON command.

4 LOGON invokes a problem program specified by the user logon procedure. This program, normally the Terminal Monitor Program (TMP), handles TSO and user-supplied commands. One of these commands invokes the TEST command processor which has the same task level as the TMP.

5 TSO command processors, and their subtasks, are at the lowest level. They perform the work required by the terminal user.

Figure 1. TSO System Overview

Introduction

The Time Sharing Option (TSO) extends the capabilities of the operating system to include general purpose time sharing from terminals supported by the Telecommunications Access Method (TCAM).

You should remember three things about TSO:

- The operating system supervises the execution of all TSO programs.
- TSO provides time sharing.
- TCAM provides terminal support.

TSO System

Figure 1 shows the relationship between major programs in a TSO system. Both TSO and TCAM execute as problem programs under the operating system control program, but as far as the terminal user is concerned, TSO is a system -- the only system he needs to know about.

The terminal user describes the work he wants done by entering TSO commands. These commands are received by the terminal monitor program (TMP) which gives control to the appropriate TSO command processor. One TSO command invokes the TEST command processor which executes at the same task level as the TMP. All other command processors execute as subtasks of the TMP.

As the TMP and the command processors execute, they may invoke the TSO service routines to perform the following operations:

- Handling input/output operations to or from terminals supported by TCAM.
- Searching input buffers for TSO commands and TSO command parameters.
- Allocating and freeing data sets and performing other data management functions.

All service routines execute at the same task level as the program that invokes them.

Terminal Monitor Program

The terminal monitor program obtains TSO commands, gives control to TSO command processors, and monitors their execution as shown in Figure 2.

The TMP is a problem program executed by the IBM-supplied user logon procedure. An installation may write a similar program and substitute it for the TMP as described in the publication OS/VS2 TSO Guide to Writing a Terminal Monitor Program or a Command Processor, GC28-0648.

The TMP does the following things:

- Obtains a new command and gives control to the appropriate command processor.
- Handles attention requests.
- Attempts to recover from errors in a command processor or one of its subtasks.
- Attempts to recover from errors in its own routines.
- Returns control to the LOGON/LOGOFF scheduler when the operator issues a STOP command or when the terminal user enters a LOGON or LOGOFF command.

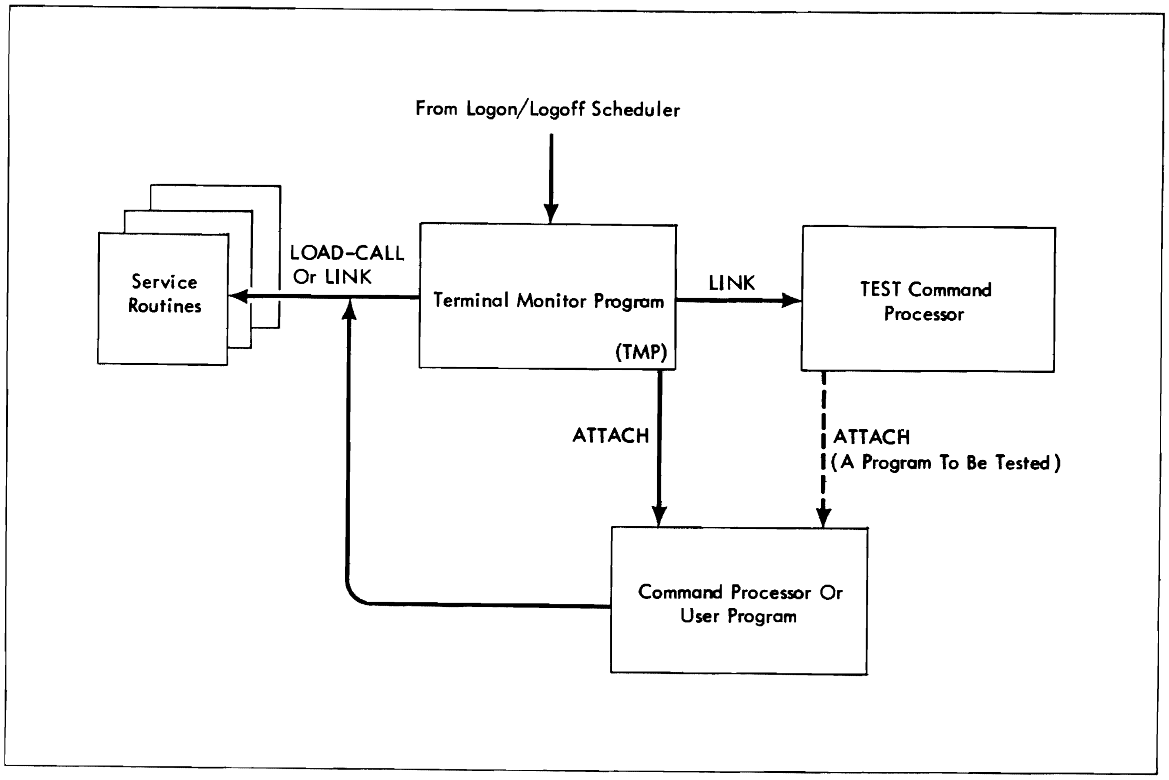

From Logon/Logoff Scheduler
Service Routines
LOAD-CALL Or LINK
Terminal Monitor Program (TMP)
LINK
TEST Command Processor
ATTACH
ATTACH (A Program To Be Tested)
Command Processor Or User Program

Figure 2. Terminal Monitor Program and Service Routines

TSO Command Processors and User Programs

TSO command processors are problem programs that perform the operations requested by a TSO command such as EDIT, CALL, RUN, ALLOCATE, etc. Figure 3 shows the functions performed by IBM-supplied TSO command processors. An installation may write similar programs for use as command processors as described in the publication OS/VS2 TSO Guide to Writing a Terminal Monitor or a Command Processor, GC28-0648.

Some TSO command processors call on standard system processors to perform the requested function. For example, the COBOL command processor sets up a standard calling sequence according to the options selected by the user and transfers control to the ANS COBOL compiler to compile the user's program. Except for the special formatting of output and messages, the compiler operates exactly as it would in a non-TSO environment.

TEST Command Processor

The TEST command processor allows the terminal user to test a command processor or other user program. Since it must be able to control the execution of command processors, the TEST command processor executes at the same task level as the TMP -- receiving control by a LINK, rather than an ATTACH, macro instruction. For further information about the TEST command processor, refer to the publication OS/VS2 Command Processs sor Logic Volume IV (TEST), SY35-0004.

TSO Service Routines

The TSO service routines are used by the TMP, TEST, and other TSO command processors. In general, they perform services needed by all TSO problem programs and their use as subroutines saves repetitive coding in the command processors. Figure 4 shows the functions performed by the TSO service routines.

TERMINAL I/O SERVICE ROUTINES

The terminal I/O service routines handle terminal input/output operations required by the LOGON/LOGOFF Scheduler, the terminal monitor program, the TSO command processors, and other TSO problem programs.

There are four terminal I/O service routines:

- STACK -- which maintains a list of input sources and defines the current source of input.
- GETLINE -- which obtains a line of input from the terminal or from the current source of input.
- PUTLINE -- which sends output or messages to the terminal.
- PUTGET -- which sends a message to the terminal and obtains a line of input from the current source of input.

COMMAND SCAN AND PARSE SERVICE ROUTINES

Command scan and parse search the command buffer for TSO commands and their parameters. In general, command scan is invoked by the terminal monitor program while parse is invoked by TSO command processors. Command scan is also invoked by the TEST command processor and by TSO command processors that accept subcommands.

DYNAMIC ALLOCATION INTERFACE ROUTINE AND SVC 99

The dynamic allocation interface routine (DAIR) handles the allocation and freeing of data sets needed by the terminal monitor program, the TSO command processors, and other TSO problem programs. In general, DAIR obtains information about a data set and, if necessary, invokes the SVC 99 dynamic allocation routines to perform the requested function.

DEFAULT SERVICE ROUTINE AND CATALOG INFORMATION ROUTINE

The default service routine constructs a data set name that follows TSO data set naming conventions. The catalog information routine obtains information from the system catalog.

Figure 3 summarizes the functions performed by each TSO command processor and shows how each command processor receives control from the TMP.

Command	Function	Given control by
ALLOCATE	Allocate data sets.	ATTACH Macro Instruction
ACCOUNT	Update the user attribute data set.	ATTACH Macro Instruction
ASM[1]	Invoke Assembler Language Compiler.	ATTACH Macro Instruction
ATTRIB	Build a list of data set attributes.	ATTACH Macro Instruction
CALC[1]	Invoke the ITF:PL/1 or CODE&GO FORTRAN	ATTACH Macro Instruction
CALL	Load and execute a load module.	ATTACH Macro Instruction
CANCEL	Cancel a foreground-initiated background job.	ATTACH Macro Instruction
COBOL[1]	Invoke the ANS COBOL compiler.	ATTACH Macro Instruction
CONVERT[1]	Convert ITF:PL/1 or CODE&GO FORTRAN source programs to standard PL/1 or FORTRAN.	ATTACH Macro Instruction
COPY[1]	Copy a data set (sequential or partitioned) to another data set.	ATTACH Macro Instruction
DELETE	Delete and uncatalog a data set or member of a partitioned data set.	ATTACH Macro Instruction
EDIT	Create and/or edit a data set.	ATTACH Macro Instruction
EXEC	Invoke a command processor or list of command processors.	ATTACH Macro Instruction
FORMAT[1]	Format a data set.	ATTACH Macro Instruction
FORT[1]	Invoke the FORTRAN IV Compiler.	ATTACH Macro Instruction
FREE	Free an allocated data set or an attribute list.	ATTACH Macro Instruction
HELP	Display information about command or subcommand.	ATTACH Macro Instruction
IPLI[1]	Invoke the ITF:IPLI compiler.	ATTACH Macro Instruction
LINK	Invoke the linkage editor.	ATTACH Macro Instruction
LIST[1]	List one or more data sets.	ATTACH Macro Instruction
LISTALC	List allocated data sets.	ATTACH Macro Instruction
LISTBC	List messages from operator or other users as entered on the broadcast data set.	ATTACH Macro Instruction

Figure 3. Functions Performed by TSO Command Processors (Part 1 of 2)

Command	Function	Given Control by
LISTCAT	List catalog entries.	ATTACH Macro Instruction
LISTDS	List the attributes of one or more data sets.	ATTACH Macro Instruction
LOADGO	Load and execute an object module.	ATTACH Macro Instruction
LOGOFF	End a terminal session	ATTACH Macro Instruction
LOGON	Begin a terminal session.	ATTACH Macro Instruction
MERGE[1]	Combine data sets.	LINK Macro Instruction
OPERATOR	Make the terminal an operator's console.	ATTACH Macro Instruction
OUTPUT	Direct the output for a foreground-initiated background job.	ATTACH Macro Instruction
PLI[1]	Invoke the optimizing PL/1 compiler.	ATTACH Macro Instruction
PLIC[1]	Invoke the checkout PL/1 compiler.	ATTACH Macro Instruction
PROFILE	Update the user profile table (UPT).	ATTACH Macro Instruction
PROTECT	Create or modify a password.	ATTACH Macro Instruction
RENAME	Rename a data set or a member of a partitioned data set.	ATTACH Macro Instruction
RUN	Compile, load, and execute a program.	ATTACH Macro Instruction
SEND	Send a message to the operator or to a terminal user.	ATTACH Macro Instruction
STATUS	List the status of a foreground-initiated background job.	ATTACH Macro Instruction
SUBMIT	Submit a job for interpretation and execution in the background.	ATTACH Macro Instruction
TERMINAL	Update the protected step control block (PSCB).	ATTACH Macro Instruction
TEST	Test a command processor or user program.	LINK Macro Instruction
TESTFORT[1]	Test a FORTRAN program using symbolic references.	ATTACH Macro Instruction
TIME	List CPU time.	Branch-and-link-register instruction
WHEN	Establish condition for initiation or termination of a command processor.	ATTACH Macro Instruction

[1]Optional program products; available for a license fee.

Figure 3. Functions Performed by TSO Command Processors (Part 2 of 2)

As the TMP, TEST, and other command processors execute, they may request services from any of the TSO service routines shown in Figure 4.

Functional Group	Common Name	Module Name	Function
I/O Service Routines	STACK	IKJPTGT	Maintains stack of input sources. The top element describes the current source of input. The bottom element describes the terminal as a source of input.
	GETLINE	IKJPTGT	Gets a line of input from the current source or from the terminal.
	PUTLINE	IKJPTGT	Sends a line of output to the terminal.
	PUTGET	IKJPTGT	Sends a line of output to the terminal; gets a line of input from the terminal.
Dynamic Allocation Interface Routine	DAIR	IKJDAIR	Allocates data sets.
Command Scan/Parse Service Routines	Command Scan	IKJSCAN	Checks command names for valid syntax.
	Parse	IKJPARS	Checks command parameters for valid syntax.
Default Service Routine	Default	IKJEHDEF IKJDFLT	Constructs a data set name according to TSO naming conventions.
Catalog Information Routine	CIR	IKJEHCIR	Searches the system catalog for information about data sets.

Figure 4. Functions Performed by TSO Service Routines

These service routines are invoked using system macro instructions. The internal logic of these routines is described in separate parts of this book.

Part 1: Terminal Monitor Program

1

Section 1: Introduction

The terminal monitor program obtains TSO commands, gives control to TSO command processors, and monitors their execution.

The TMP is a problem program executed by the IBM-supplied user logon procedure. An installation may write a similar program and substitute it for the TMP as described in the publication OS/VS2 TSO Guide to Writing a Terminal Monitor Program or a Command Processor, GC28-0648.

The TMP obtains its first command from the user logon procedure and gives control to the appropriate command processor. After the first command has been processed, the TMP does one of the following:

- Obtains a new command and gives control to the appropriate command processor.
- Handles attention requests.
- Attempts to recover from errors in a command processor or one of its subtasks.
- Attempts to recover from errors in its own routines.
- Returns control to the LOGON/LOGOFF scheduler when the operator issues a STOP command or when the user enters a LOGON or LOGOFF command.

As supplied with TSO, the TMP will reside in SYS1.LINKLIB and will execute in the user's foreground region with the protection key assigned to that region. The installation may choose to make the TMP resident in the TSO Link Pack Area (TSLPA) in the region assigned to the Time Sharing Control Task (TSCT).

Section 2: Method of Operation

This section describes the method of operation of the terminal monitor program. It includes six method of operation diagrams:

- Method of Operation Diagram 1: Terminal Monitor Program -- which shows the basic functions performed by the TMP.
- Method of Operation Diagram 2: TMP Initialization -- which shows how the TMP completes the logon process by setting up tables and control blocks that define the user's environment in the foreground region.
- Method of Operation Diagram 3: Handling Commands -- which shows how the TMP obtains commands from the terminal and gives control to the appropriate command processor.
- Method of Operation Diagram 4: Handling Attention Requests -- which shows how the TMP handles terminal attention requests.
- Method of Operation Diagram 5: Handling STAI Requests -- which shows how the TMP attempts to recover from errors in a command processor or one of its programs.
- Method of Operation Diagram 6: Handling STAE Requests -- which shows how the TMP attempts to recover from errors in its own programs.

Each method of operation diagram includes a cross-reference table to help you find the appropriate assembly listing.

Overview

Method of Operation Diagram 1 shows how the TMP obtains TSO commands, gives control to TSO command processors, and monitors their execution.

Briefly, here is what happens:

- The TMP receives control from the LOGON/LOGOFF scheduler as the program named by the first EXEC statement in the user logon procedure.
- The TMP completes the logon process by setting up tables and control blocks that define the user's environment in the foreground region.
- The TMP obtains TSO commands and gives control to the appropriate TSO command processors, one at a time.
- When a TSO command processor completes normally, the TMP obtains a new TSO command and gives control to a new TSO command processor.
- When a TSO command processor is interrupted by an attention at the TMP level, the TMP attention exit routine handles the attention request.

- When a TSO command processor begins to terminate abnormally, the TMP STAI Exit Routine attempts to recover from the error.
- When the TMP begins to terminate abnormally, the TMP STAE exit routine attempts to recover from the error.
- The TMP returns control to the LOGON/LOGOFF scheduler when the user enters a LOGON or LOGOFF command or when the operator issues a STOP command.

Figure 5 summarizes the functions performed by the TMP.

Event	Action Taken
Command processor completes normally.	TMP obtains a command and gives control to the next command processor. See Diagram 3 (foldout).
Command processor interrupted by attention.	TMP attention exit routine displays second-level messages in response to a question mark or obtains a new command to replace the command that was interrupted. See Diagram 4 (foldout).
Command processor begins to terminate abnormally.	TMP STAI exit routine attempts to recover from the error. See Diagram 5 (foldout).
TMP begins to terminate terminate abnormally.	TMP STAE exit routine attempts to recover from the error. See Diagram 6 (foldout).
Operator or terminal user has requested logoff or re-logon.	TMP returns control to the LOGON/LOGOFF Scheduler.

Figure 5. Functions Performed by the Terminal Monitor Program

Initialization

Method of Operation Diagram 2 shows how the TMP completes the logon process by setting up tables and control blocks that define the user's environment in the foreground region.

Briefly, here is what happens:

- The TMP receives control from the LOGON/LOGOFF scheduler as the program named by the first EXEC statement in the user logon procedure.
- At entry, register 1 contains the address of a buffer that contains the PARM value from the first EXEC statement in the user logon procedure.
- The TMP uses an EXTRACT macro instruction to obtain the address of the protected step control block, which contains the address of the user profile table (UPT).
- The TMP sets up the environment control table (ECT) and two internal word areas: the TMP Work Area (TMPWORKA) and the TMP retry work area (TMPWA2).

- The TMP sets up the TMP attention exit by issuing a STAX macro instruction and loading the TMP Attention Exit Routine.

- The TMP sets up the TMP STAE exit by issuing a STAE macro instruction and loading the TMP STAE Exit Routine.

- The TMP partially sets up the STAI exit by loading the STAI exit routine. Later, when the TMP attaches a command processor, the STAI operand on the ATTACH macro instruction provides supervisor linkage to the STAI exit routine.

- The TMP loads the TIME command processor.

- The TMP sets up the command buffer and obtains its first command from the PARM field of the first EXEC statement in the user logon procedure.

When initialization is complete, the TMP is ready to process the first command.

Handling TSO Commands

Method of Operation Diagram 3 shows how the TMP obtains commands from the terminal and gives control to the appropriate command processor.

Briefly, here is what happens:

- The TMP uses the PUTGET service routine to obtain a line of input from the terminal or from an in-storage list. The command buffer (CBUF) receives the line of input.

- The TMP uses the command scan service routine to search the command buffer for a command name, a question mark, or a null line. The TMP tests for four special cases and takes the appropriate action:

Buffer Contains	Action Taken by TMP
TIME	Branch to TIME command processor.
TEST	Link to TEST command processor.
Question mark	Display all chained second level messages. (Done by PUTGET.)
Null	Ignore line. Obtain another line.

- The TMP searches the command library to obtain the TSO command processor that corresponds to the TSO command. If it cannot find the command processor, the TMP assumes that the intended command is EXEC and that the command buffer contains a valid member name.

- The TMP attaches the command processor as a subtask and waits for it to complete as shown in Method of Operation Diagram 1 (foldout).

Handling Attention Requests

Method of Operation Diagram 4 shows how the TMP handles terminal attention requests.

During initialization, the TMP loads the attention exit routine and sets up supervisor linkage to it by issuing a STAX macro instruction. The STAX service routine builds a terminal attention exit element (TAXE) and a terminal attention interrupt element (TAIE) and chains them to the TMP's TCB.

Later, when the terminal user signals attention, program execution is interrupted and the attention scheduler (a part of the region control task) gets control. The attention scheduler checks the TCB of the interrupted program for a terminal attention exit element and schedules the most recently specified attention exit routine.

If the interruption occurs while the TMP is processing, the TMP's attention exit routine receives control. If the interruption occurs while a command processor is processing, the command processor's attention exit routine receives control unless the attention is signaled twice. Signaling attention two times in quick succession will cause the next higher-level attention exit routine to receive control.

The TMP attention exit routine uses the command scan service routine to search the attention buffer for question mark, null line, or command name. The contents of the attention buffer determine the action taken by the TMP attention exit routine, as shown in Figure 6.

Buffer Contains	Action Taken by TMP Attention Exit Routine
Question mark	Writes second-level message to the terminal using PUTLINE service routine. Prompts terminal for another line of input using PUTLINE and GETLINE service routines.
Null line	Returns to attention scheduler which restarts interrupted program from the point of interruption.
Invalid Command	Writes error message using the PUTLINE service routine. Prompts terminal for another line of input using the PUTLINE and GETLINE service routines.
TIME	Branches to TIME command processor to obtain elapsed time, CPU time, and time for the terminal session.
Other Valid Commands	Places command name in command waiting field of the TMP work area (TMPWORKA) and posts the TMP attention ECB. The TMP will detach a previously attached command processor, if any, and obtain the new command. See Diagram 3 (foldout).

Figure 6. Functions Performed by TMP Attention Exit Routine

Handling STAI Requests

Method of Operation Diagram 5 shows how the TMP attempts to recover from errors in a TSO command processor or one of its programs.

During initialization, the TMP loads the STAI exit routine. Later, when the TMP attaches a command processor as a subtask, the TMP includes a STAI operand on the ATTACH macro instruction specifying the entry point of the STAI Exit Routine.

The STAE service routine builds a STAI control block and chains it to the TCBNSTAE field of the command processor's TCB. Note that the STAE service routine handles both STAE and STAI exit routines.

When an error occurs in the command processor or one of its subtasks, interrupted program for a terminal attention exit element and see if any STAI or STAE control blocks have been chained and, finding one, passes control to the ABEND/STAE interface routine. Note that the ABEND/STAE interface routine handles both STAE and STAI requests.

The ABEND/STAE interface routine quiesces all active I/O, purges all ready I/O, and schedules the most recently specified STAE exit routine by issuing a SYNCH macro instruction. Normally, the command processor's STAE exit routine will get control.

The TMP attention exit routine uses the command scan service routine attempts to diagnose the cause of the error, uses a return code to mark it recoverable or unrecoverable, and returns control to the ABEND/STAE interface routine. This routine passes control to the STAE retry routine (if one is specified) which will attempt to restart the interrupted program.

If the retry is not successful, or if no retry routine was specified, the ABEND/STAE interface routine checks the STAE/STAI chain for the most recently specified STAI exit. Note that only one STAE exit routine is ever executed, while any number of STAI exit routines may be executed.

The TMP STAI exit routine uses the PUTLINE service routine to write a message to the terminal indicating that the task has failed. It then checks to see whether the error occurred in a command processor running under the TMP or running under the TEST command processor.

If the error occurred in a command processor running under the TEST command processor, the STAI exit routine posts the TMP STAI ECB and transfers control to the TEST command processor.

The STAI exit routine then waits on the command processor's ECB. The TEST command processor posts this ECB immediately, specifying a retry address. The STAI exit routine marks the ABEND recoverable and returns control to the ABEND/STAE interface routine. The ABEND/STAE interface routine passes control to the retry address, which is an entry point in the TEST command processor.

If the error occurred in a command processor running under the TMP, the TMP STAI exit routine posts the TMP STAI ECB and immediately transfers control to the TMP mainline routine.

The STAI exit routine then waits on the command processor's ECB. The TMP posts this ECB whenever it is impossible to recover from the error. The TMP mainline routine then prompts the terminal to enter another command using the PUTGET service routine.

The TMP mainline routine uses the command scan service routine to search the buffer for the TEST command name and, upon finding it, transfers control to the TEST command processor.

Except for the TIME or TEST commands, any valid command in the input buffer causes the command processor to be detached, thereby canceling the outstanding STAI request.

If the buffer contains a null line, the ECB for the abnormally terminating command processor is posted. The STAI exit routine then checks the post code, marks the ABEND unrecoverable, and returns control to the ABEND/STAE interface routine.

All other input is processed as though the STAI condition had not occurred.

The ABEND/STAE interface routine checks the TCB of the failing task for another STAI exit and, finding none, returns control to the ABEND routine, which terminates the task.

Handling STAE Requests

Method of Operation Diagram 6 shows how the TMP attempts to recover from errors in its own code.

During initialization, the TMP loads a STAE exit routine and sets up supervisor linkage to it by issuing a STAE macro instruction. The STAE service routine builds a STAE Control Block and chains it to the TMP's TCB.

When an error occurs in the TMP, the ABEND routine checks the TMP's TCB for a STAE control block and, finding one, passes control to the ABEND/STAE interface routine, which marks the task non-dispatchable and passes control to the TMP's STAE exit routine.

The TMP mainline routine uses the command scan service routine to ABEND/STAE interface routine, which marks the task non-dispatchable and passes control to the TMP's STAE exit routine.

The TMP's STAE exit routine attempts to diagnose the cause of the error and returns to the ABEND/STAE interface routine with a return code that indicates whether the error is recoverable or not recoverable. If the error is not recoverable, the ABEND/STAE interface routine returns control to the ABEND routine, which terminates the task. If the error is recoverable, the ABEND/STAE interface routine passes control to the TMP's STAE retry routine, which attempts to restart the TMP.

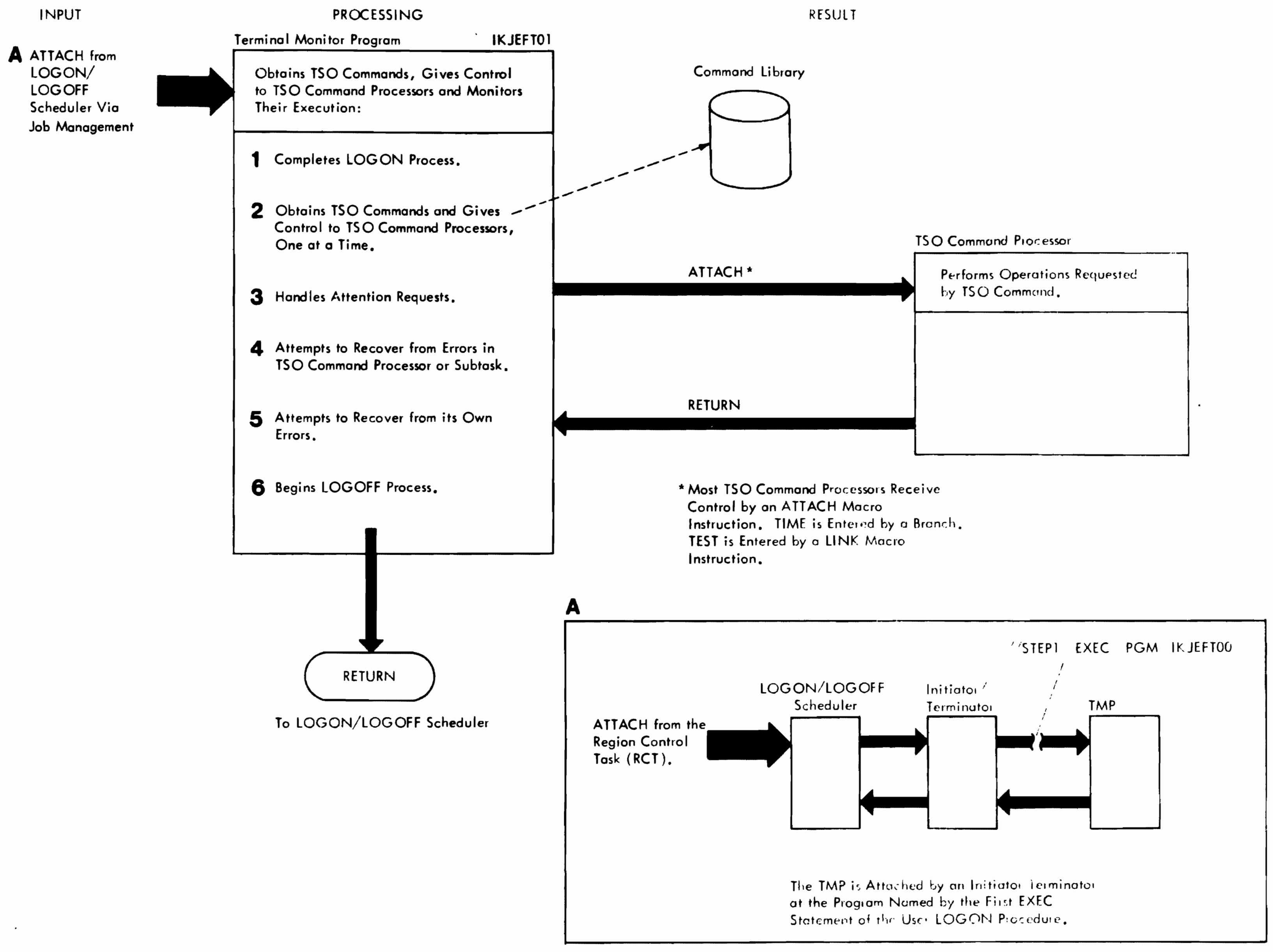
The TMP's STAE retry routine determines whether a recovery has been attempted for this command processor. If so, the STAE retry routine deletes all TMP modules and transfers control to the TMP initialization routine, IKJEFT01. Otherwise it transfers control to the TMP mainline routine, IKJEFT02.

If the retry is successful, TMP processing continues as if nothing had happened. If the retry is not successful, the ABEND routine passes control to the ABEND/STAE interface routine which passes control to the TMP's STAE routine for cleanup operations before returning to the ABEND/STAE interface routine.

The ABEND/STAE interface routine then checks the TMP's TCB for the most recently specified STAI exit routine and, finding none, returns to the ABEND routine which terminates the TMP.

Method of Operation Diagrams

1


INPUT
PROCESSING
RESULT
A ATTACH from LOGON/ LOGOFF Scheduler Via Job Management
Terminal Monitor Program IKJEFT01
Obtains TSO Commands, Gives Control to TSO Command Processors and Monitors Their Execution:
1 Completes LOGON Process.
2 Obtains TSO Commands and Gives Control to TSO Command Processors, One at a Time.
3 Handles Attention Requests.
4 Attempts to Recover from Errors in TSO Command Processor or Subtask.
5 Attempts to Recover from its Own Errors.
6 Begins LOGOFF Process.
Command Library
TSO Command Processor
Performs Operations Requested by TSO Command.
ATTACH *
RETURN
* Most TSO Command Processors Receive Control by an ATTACH Macro Instruction. TIME is Entered by a Branch. TEST is Entered by a LINK Macro Instruction.
RETURN
To LOGON/LOGOFF Scheduler
A
//STEP1 EXEC PGM IKJEFT00
LOGON/LOGOFF Scheduler
Initiator/ Terminator
TMP
ATTACH from the Region Control Task (RCT).
The TMP is Attached by an Initiator Terminator at the Program Named by the First EXEC Statement of the User LOGON Procedure.

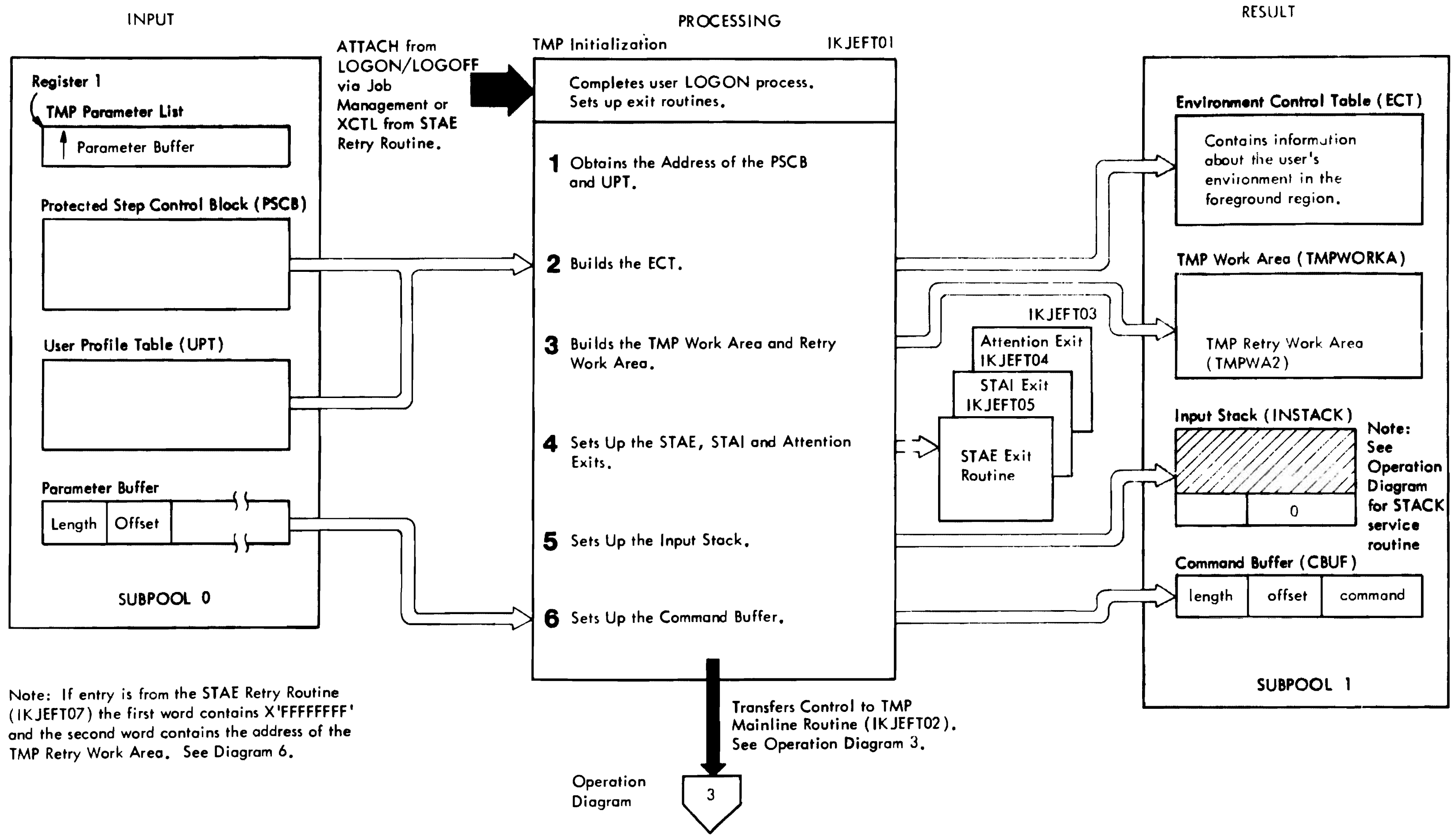
Method of Operation Diagram 1. Terminal Monitor Program

Method of Operation Diagram 1. Terminal Monitor Program (Part 1 of 2)

CROSS REFERENCE TABLE

Key Description	Routine	Label	Diagram
1 The TMP is attached by an initiator/terminator as the program named by the first EXEC statement in the user logon procedure. The TMP completes the logon process by setting up tables and control blocks that define the user's environment in the foreground region.	Initialization	IKJEFT01	2
2 The TMP obtains commands from the terminal and gives control to the appropriate command processor, one at a time. Most command processors are attached as subtasks of the TMP. Exceptions are TIME (which is branched to) and TEST (which is linked to).	Mainline	IKJEFT02	3
3 The TMP handles attention interrupts by displaying second-level messages in response to a question mark or by obtaining a new command to replace the one that was interrupted.	Attention Exit Routine	IKJEFT03	4
4 The TMP attempts to recover from errors in a command processor or one of its subtasks by allowing the user to enter a TEST command.	STAI Exit Routine	IKJEFT04	5
5 The TMP attempts to recover from errors in its own code by diagnosing the cause of the error and, if possible, by restarting the TMP.	STAE Exit Routine STAE Retry Routine	IKJEFT05 IKJEFT07	6 6
6 The TMP performs cleanup operations and returns to the LOGON/LOGOFF scheduler when the operator issues a STOP or MODIFY command or when the user issues a LOGON or LOGOFF command.	Mainline	IKJEFT02	2

Method of Operation Diagram 1. Terminal Monitor Program (Part 2 of 2)


INPUT
PROCESSING
RESULT
Register 1
TMP Parameter List
Parameter Buffer
Protected Step Control Block (PSCB)
User Profile Table (UPT)
Parameter Buffer
Length
Offset
SUBPOOL 0
ATTACH from LOGON/LOGOFF via Job Management or XCTL from STAE Retry Routine.
TMP Initialization
IKJEFT01
Completes user LOGON process. Sets up exit routines.
1 Obtains the Address of the PSCB and UPT.
2 Builds the ECT.
3 Builds the TMP Work Area and Retry Work Area.
4 Sets Up the STAE, STAI and Attention Exits.
5 Sets Up the Input Stack.
6 Sets Up the Command Buffer.
IKJEFT03
Attention Exit IKJEFT04
STAI Exit IKJEFT05
STAE Exit Routine
Environment Control Table (ECT)
Contains information about the user's environment in the foreground region.
TMP Work Area (TMPWORKA)
TMP Retry Work Area (TMPWA2)
Input Stack (INSTACK)
0
Note: See Operation Diagram for STACK service routine
Command Buffer (CBUF)
length
offset
command
SUBPOOL 1
Transfers Control to TMP Mainline Routine (IKJEFT02). See Operation Diagram 3.
Operation Diagram
3

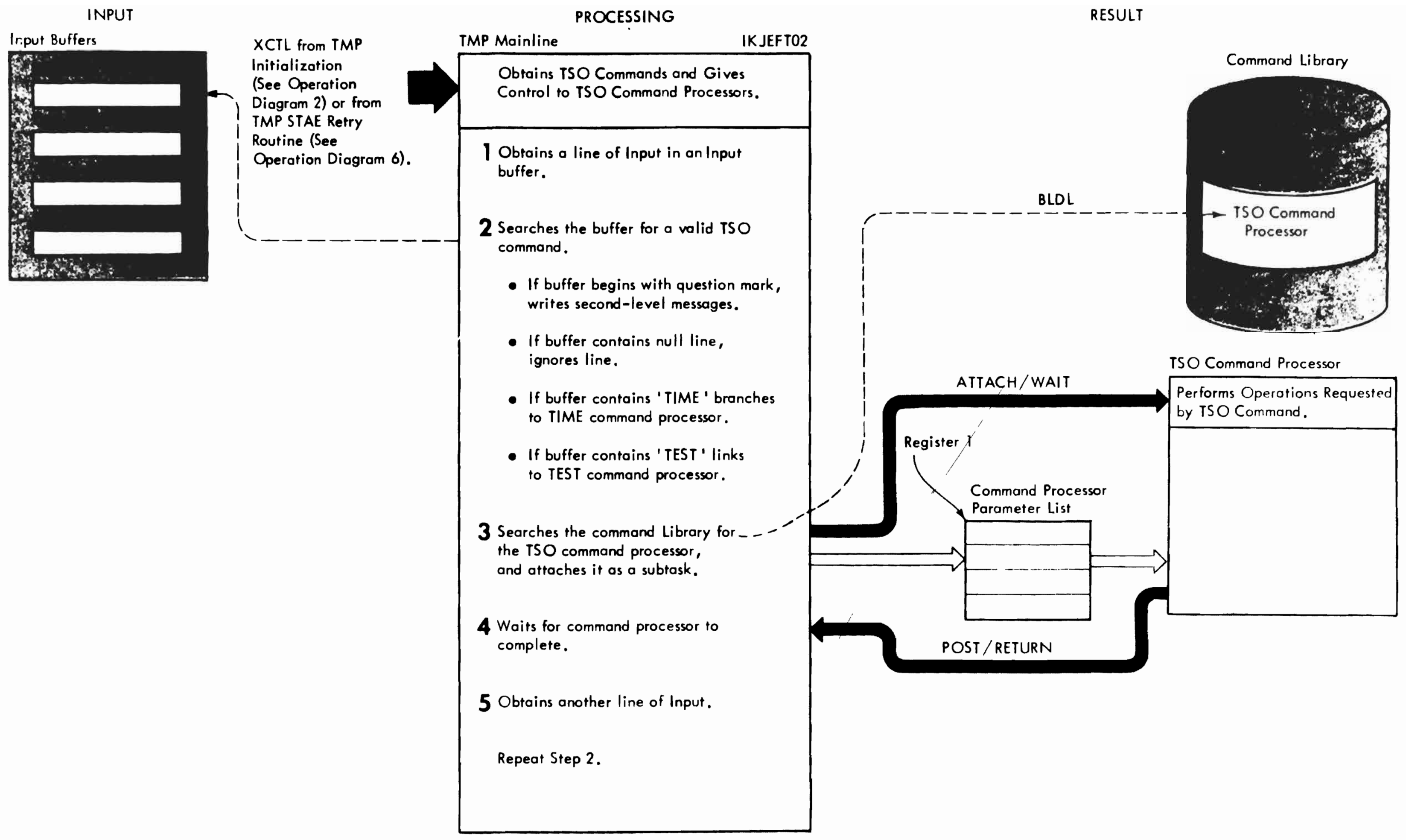
Note: If entry is from the STAE Retry Routine (IKJEFT07) the first word contains X'FFFFFFFF' and the second word contains the address of the TMP Retry Work Area. See Diagram 6.

Method of Operation Diagram 2. TMP Initialization (Part 1 of 2)

CROSS REFERENCE TABLE

Key Description	Routine	Label
1 The TMP uses an EXTRACT macro instruction to obtain the address of the Protected Step Control Block (PSCB) which contains the address of the User Profile Table (UPT).	Initialization	IKJEFT01
2 The TMP constructs the Environment Control Table (ECT) from information contained in the PSCB and UPT.		
3 The TMP builds two major internal work areas: the TMP Work Area (TMPWORKA) and the TMP Retry Work Area (TMPWA2). TMPWORKA contains parameter lists and control information for normal operation. TMPWA2 contains information needed by the TMP STAE Retry Routine.		
4 The TMP sets up the STAE exit by issuing a STAE macro instruction and loading the STAE exit routine.		LDSTAE
The TMP sets up the Attention exit by issuing a STAX macro instruction and loading the Attention exit routine.		
The TMP sets up the STAI exit by loading the STAI exit routine and (later on, in Diagram 5) by including the STAI operand on the ATTACH macro instruction when giving control to a command processor.		
The TMP loads the TIME command processor.		
5 The TMP initializes the first element on the Input Stack to describe the terminal as the current source of input.		STACK
6 The TMP sets up the Command Buffer (CBUF) and initializes it with the value from the PARM field of the first EXEC statement in the user logon procedure.		FCM001

Method of Operation Diagram 2. TMP Initialization (Part 2 of 2)


INPUT
Input Buffers
XCTL from TMP Initialization (See Operation Diagram 2) or from TMP STAE Retry Routine (See Operation Diagram 6).
PROCESSING
TMP Mainline
IKJEFT02
RESULT
Command Library
BLDL
TSO Command Processor
ATTACH/WAIT
Register 1
Command Processor Parameter List
TSO Command Processor
Performs Operations Requested by TSO Command.
POST/RETURN

Obtains TSO Commands and Gives Control to TSO Command Processors.

1 Obtains a line of Input in an Input buffer.

2 Searches the buffer for a valid TSO command.

- If buffer begins with question mark, writes second-level messages.
- If buffer contains null line, ignores line.
- If buffer contains 'TIME' branches to TIME command processor.
- If buffer contains 'TEST' links to TEST command processor.

3 Searches the command Library for the TSO command processor, and attaches it as a subtask.

4 Waits for command processor to complete.

5 Obtains another line of Input.

Repeat Step 2.

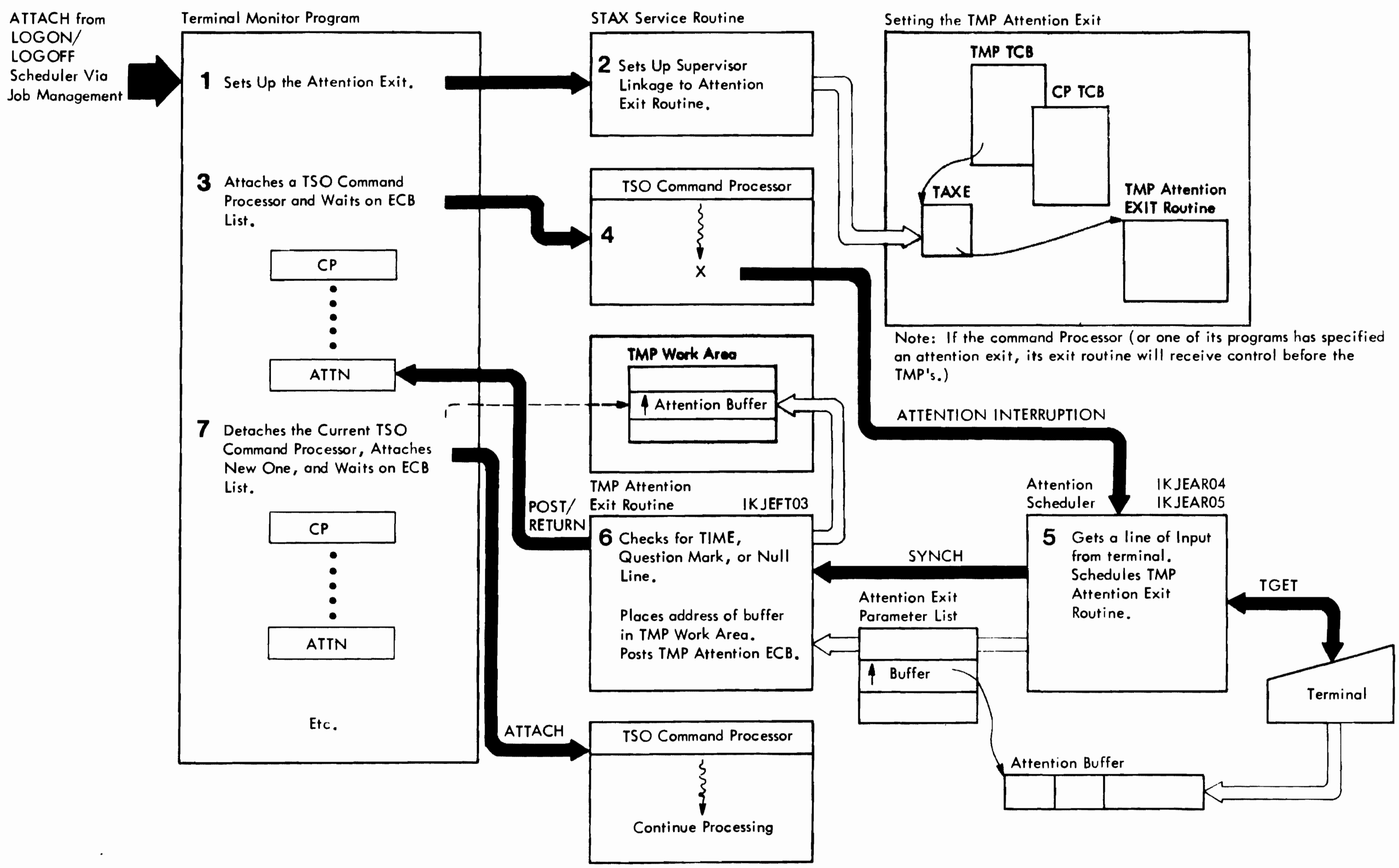
Method of Operation Diagram 3. Handling TSO Commands

Method of Operation Diagram 3. Handling TSO Commands (Part 1 of 2)

CROSS REFERENCE TABLE

Key Description	Routine	Label	Diagram
1 The TMP obtains a TSO command from one of four input buffers, depending upon the situation:			3
• At entry from the LOGON/LOGOFF scheduler, the TMP obtains a command from the PARM Buffer.	TMP Initialization	IKJEFT01	2
• When handling an attention request, the TMP may obtain a command from the Attention Buffer pointed to by the CMDWAIT field of the TMP Work Area.	TMP Initialization	ATTNPOST	4
• When handling a STAI request, the TMP may obtain a command from the STAI Buffer pointed to by the CMDWAIT field of the TMP Work Area.		STAIPOST	5
• When a command processor has completed, the TMP obtains a command from the command buffer.		GETCMD	3
2 The TMP searches the buffer for a valid command name. In most cases, the TMP then searches the user command library to find the appropriate command processor and attaches it as a subtask (Step 3). In five cases, the TMP processes the contents of the buffer without having to search the user command library or attach a subtask. These cases are:		SCAN	3
<u>Buffer Contains</u> — <u>Action Taken</u>			
question mark — Writes chained second-level messages to the terminal.			
null line — Ignores the line.			
invalid command — Writes an error message to the terminal.			
TIME — Branches to the TIME command processor.			
TEST — Links to the TEST command processor.			
The TMP then prompts the terminal to enter another command (Step 1).			
3 The TMP searches the user command library to obtain the command processor corresponding to the command name and attaches it as a subtask. If the command processor is not found, the TMP assumes that the intended command processor is EXEC and that the buffer contains a valid member name.		BLDL IMPLEXEC	3 3
4 The TMP waits on the command processor ECB. The dispatcher gives control to the command processor and allows it to execute.		WAIT	3
5 When the TMP regains control from the dispatcher following a post of an ECB, it determines why it got control and takes the appropriate action:		LIST	3
<u>ECB Posted</u> — <u>Action Taken</u>			
CP — Obtains another command. See Step 1.			
ATTN — Obtains another command. See Step 1.			
STAI — Obtains another command. See Step 1.			
LOGOFF — Performs cleanup operations before returning to the LOGON/LOGOFF scheduler.			

Method of Operation Diagram 3. Handling TSO Commands (Part 2 of 2)

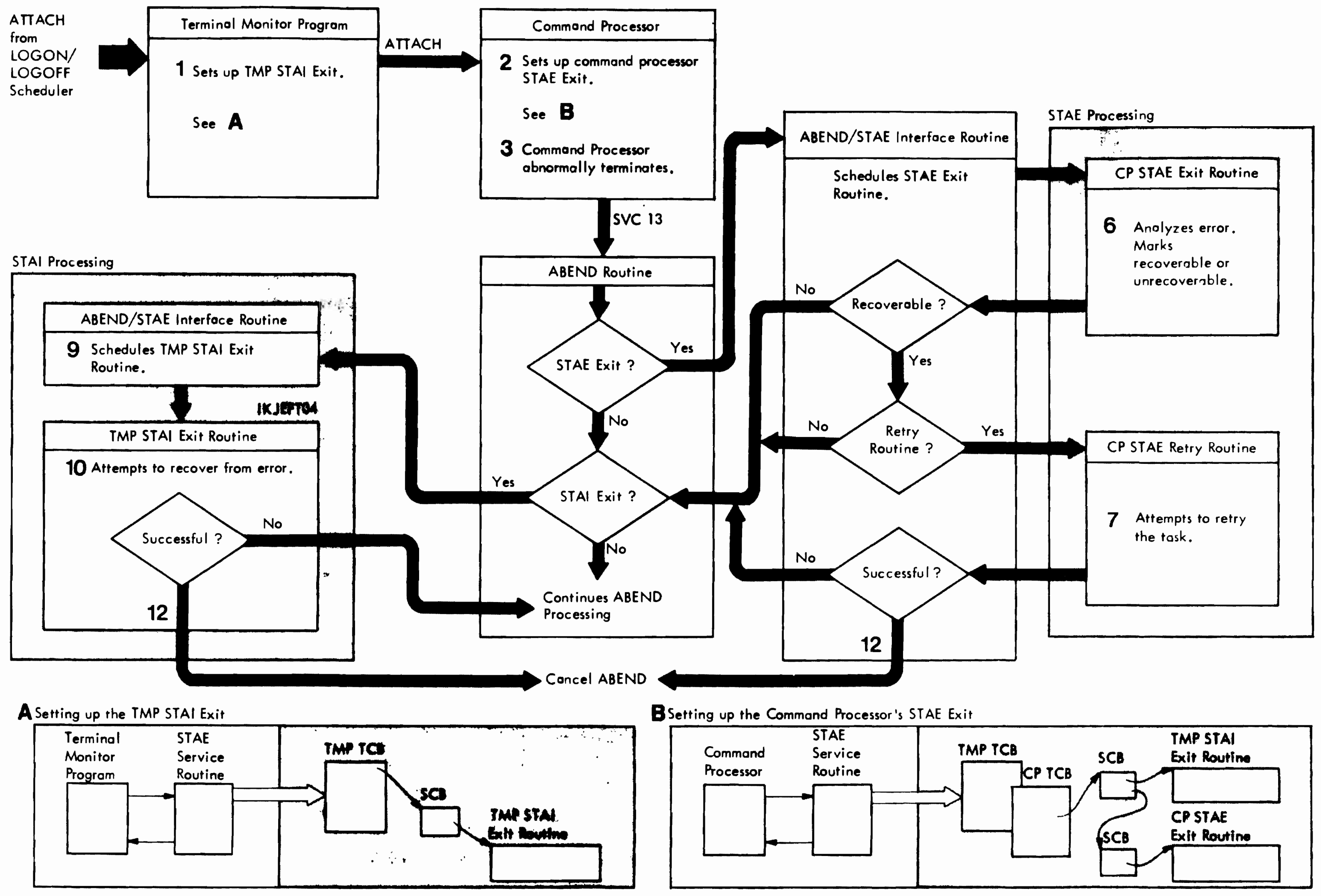

ATTACH from LOGON/ LOGOFF Scheduler Via Job Management
Terminal Monitor Program
1 Sets Up the Attention Exit.
3 Attaches a TSO Command Processor and Waits on ECB List.
CP
ATTN
7 Detaches the Current TSO Command Processor, Attaches New One, and Waits on ECB List.
CP
ATTN
Etc.
STAX Service Routine
2 Sets Up Supervisor Linkage to Attention Exit Routine.
TSO Command Processor
4
X
TMP Work Area
Attention Buffer
POST/ RETURN
TMP Attention Exit Routine IKJEFT03
6 Checks for TIME, Question Mark, or Null Line.
Places address of buffer in TMP Work Area. Posts TMP Attention ECB.
ATTACH
TSO Command Processor
Continue Processing
Setting the TMP Attention Exit
TMP TCB
CP TCB
TAXE
TMP Attention EXIT Routine
Note: If the command Processor (or one of its programs has specified an attention exit, its exit routine will receive control before the TMP's.)
ATTENTION INTERRUPTION
Attention Scheduler IKJEAR04 IKJEAR05
5 Gets a line of Input from terminal. Schedules TMP Attention Exit Routine.
SYNCH
Attention Exit Parameter List
Buffer
TGET
Terminal
Attention Buffer

Method of Operation Diagram 4. Handling Attention Requests (Part 1 of 2)

CROSS REFERENCE TABLE

Key Description	Routine	Label	Diagram
1 The TMP issues a STAX macro instruction specifying an Attention Exit Routine.	TMP Initialization	IKJEFT01	4
2 The STAX Service Routine sets up the necessary control blocks and returns control to the TMP.	STAX Service Routine		4
3 The TMP continues processing, obtaining commands, attaching command processors, waiting on an ECB list, etc.	TMP Mainline	IKJEFT02	3
4 At some point during the processing of a command, the user presses the Attention key.			
5 The Attention scheduling routine in the Region Control Task (RCT) gets control and issues a TPUT SVC to write a "READY" message to the terminal and a TGET SVC to get a line of input from the terminal. It then schedules the TMP Attention Exit Routine.	ATTN Scheduling Routine	IKJEAR04 IKJEAR05	4
6 The Attention Buffer is scanned for a valid command using Command Scan. TIME and question mark requests are handled directly. Otherwise, the interrupted task is marked non-dispatchable using the STATUS macro instruction with a STOP operand. The new command is moved into the Command Waiting field of the TMP Work Area and the ATTN ECB is posted.	TMP Attention Exit	IKJEFT03	4
7 When the TMP regains control, it finds that its ATTN ECB has been posted. The TMP detaches the Interrupted command processor and attaches the new command processor.	TMP Mainline	IKJEFT02	3

Method of Operation Diagram 4. Handling Attention Requests (Part 2 of 2)

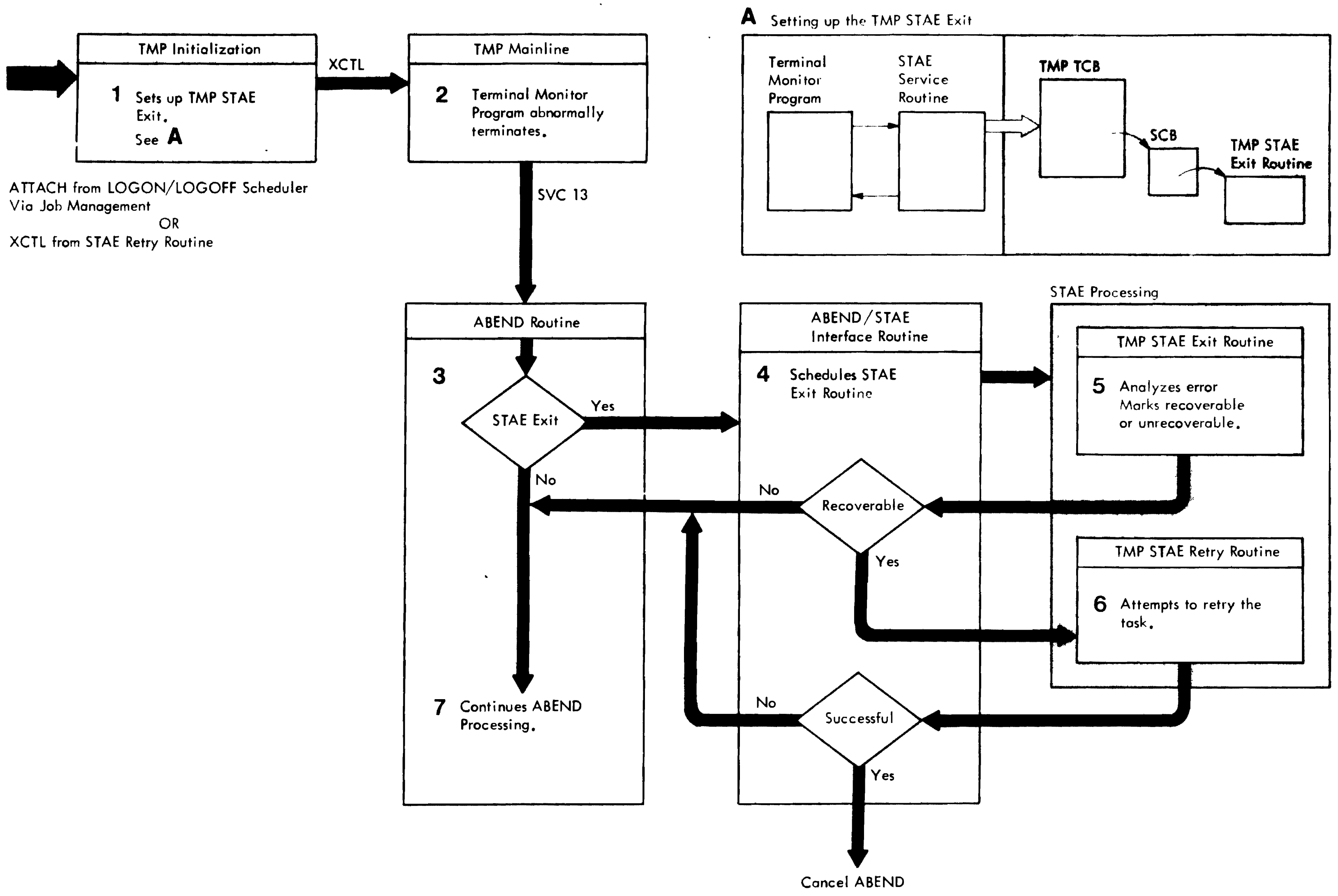


Method of Operation Diagram 5. Handling STAI Requests (Part 1 of 2)

CROSS REFERENCE TABLE

Key	Description	Routine	Label	Diagram
1	During initialization, the TMP loads the TMP STAI Exit Routine. Later, when the TMP gives control to a command processor, it issues an ATTACH macro instruction with a STAI operand. The STAE service routine builds a STAI Control Block specifying the address of the STAI Exit Routine and chains it to the TCBNSTAE field of the command processor's TCB.	TMP Initialization TMP Mainline STAE Service Routine	IKJEFT01 IKJEFT02	2 3
2	The command processor, during its initialization process, issues a STAE macro instruction specifying the address of a STAE exit routine and, possibly, a STAE retry routine. The STAE service routine builds a STAE Control Block and chains it to the TCBNSTAE field of the command processor's TCB.	Command Processor STAE Service Routine		
3	When an error in the command processor results in an ABEND, control passes to the ABEND Routine.	Command Processor		
4	The ABEND routine recognizes that there is a STAE exit routine and passes control to the ABEND/STAE interface routine.	ABEND Routine		
5	The ABEND/STAE interface routine quiesces all active I/O, purges all ready I/O, attempts to establish a work area, and schedules the command processor's STAE exit routine by issuing a SYNCH macro instruction.	ABEND/STAE Interface		
6	The command processor's STAE Exit Routine analyzes the error and marks it recoverable or unrecoverable.	Command Processor		
7	If the error is recoverable, the ABEND/STAE interface routine passes control to the STAE retry routine, if any, which attempts to retry the failing task.	Command Processor STAE Retry		
8	If the attempt is successful, the ABEND is cancelled. Otherwise, control is returned to the ABEND routine which recognizes a STAI Exit Routine.			
9	The TMP STAI Exit Routine is scheduled using a SYNCH macro instruction.	ABEND/STAE Interface		
10	If the Command Processor's STAE Exit Routine has marked the error unrecoverable, control is returned to the ABEND Routine. Otherwise the TMP STAI ECB is POSTed. If the TEST command processor had control, TEST is re-entered. Otherwise, the user is prompted for a command and given a chance to attempt to recover from the error.	TMP STAI Exit TMP Mainline	IKJEFT04 IKJEFT02 STAIPOST	5 3
11	If the attempt is not successful, control is returned to the ABEND routine.	ABEND Routine		
12	If the attempt is successful, the ABEND is cancelled.			

Method of Operation Diagram 5. Handling STAI Requests (Part 2 of 2)


TMP Initialization
1 Sets up TMP STAE Exit. See A
XCTL
TMP Mainline
2 Terminal Monitor Program abnormally terminates.
ATTACH from LOGON/LOGOFF Scheduler Via Job Management
OR
XCTL from STAE Retry Routine
SVC 13
A Setting up the TMP STAE Exit
Terminal Monitor Program
STAE Service Routine
TMP TCB
SCB
TMP STAE Exit Routine
ABEND Routine
3
STAE Exit
Yes
No
ABEND/STAE Interface Routine
4 Schedules STAE Exit Routine
STAE Processing
TMP STAE Exit Routine
5 Analyzes error Marks recoverable or unrecoverable.
No
Recoverable
Yes
TMP STAE Retry Routine
6 Attempts to retry the task.
7 Continues ABEND Processing.
No
Successful
Yes
Cancel ABEND

Method of Operation Diagram 6. Handling STAE Requests

Method of Operation Diagram 6. Handling STAE Requests (Part 1 of 2)

CROSS REFERENCE TABLE

Key Description	Routine	Label	Diagram
1 The TMP issues a STAE macro instruction that sets up parameters for the STAE service routine. The STAE service routine builds a STAE Control Block (SCB) that contains the address of the TMP STAE exit routine.	TMP Initialization	IKJEFT01	2
2 When an error occurs in any TMP routine, an ABEND (SVC 13) is issued.	ABEND SVC		
3 The ABEND routine checks the TCBNSTAE field of the TCB for the abnormally terminating task and, finding that a STAE exit is specified, passes control to the ABEND/STAE interface routine.	ABEND SVC		
4 The ABEND/STAE interface routine quiesces active I/O, purges ready I/O, and attempts to establish a work area. It then schedules the TMP STAE exit routine by issuing a SYNCH macro instruction.	ABEND/STAE interface routine		
5 The TMP STAE exit routine diagnoses the cause of the error and marks the error recoverable or not recoverable.	TMP STAE exit routine	IKJEFT05	6
6 If the error is recoverable, the TMP is re-initialized, if necessary, and processing resumes.	TMP STAE Retry Routine	IKJEFT07	6
7 If the error is not recoverable, control is returned to ABEND for abnormal termination processing.	ABEND SVC		

Method of Operation Diagram 6. Handling STAE Requests (Part 2 of 2)

Section 3: Program Organization

This section describes the organization of the terminal monitor program. It contains information about the hierarchy of the load modules, the assembly modules, and the control sections that constitute the program. Figure 7 is a graphic representation of this hierarchy.

The module operation information briefly describes the processing operations that occur within each TMP module.

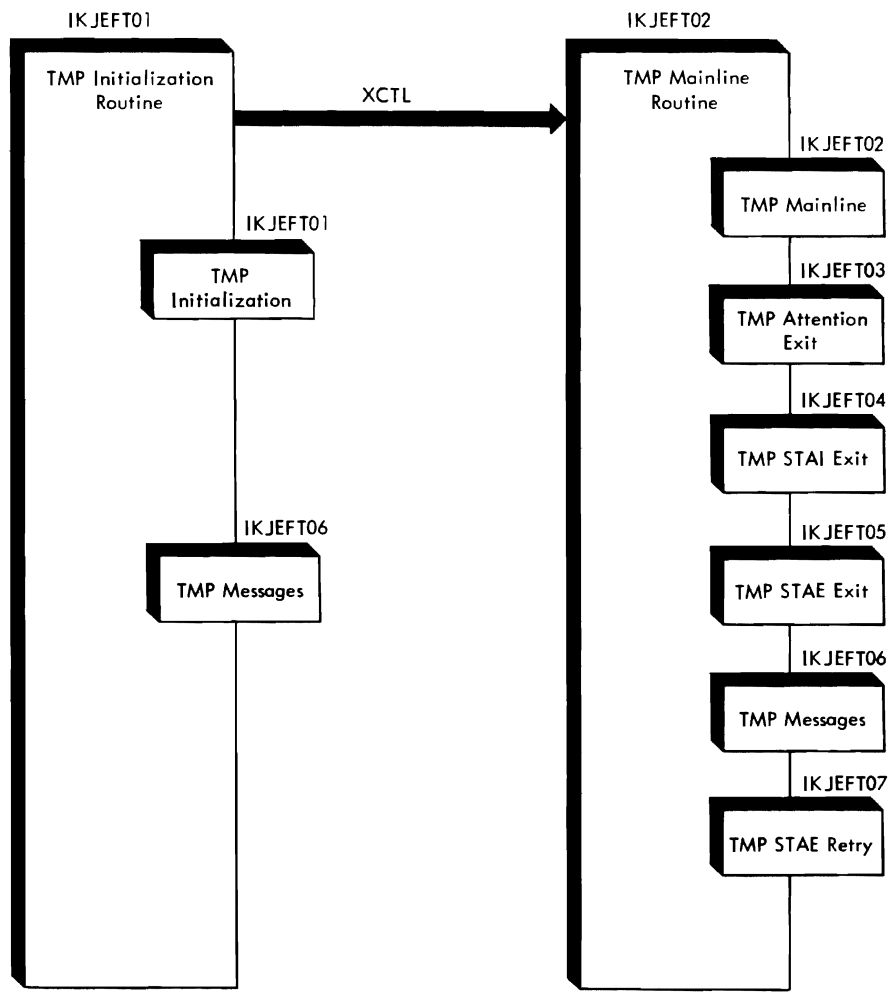
For a summary of functions performed by subroutines, refer to the Directory in Section 4.

Program Hierarchy

The TMP has two load modules as shown in Figure 7. The TMP initialization routine (IKJEFT01) completes the logon process by setting up tables and control blocks and setting up the STAE, STAI and attention exit routines. The TMP mainline routine (IKJEFT02) obtains TSO commands, gives control to TSO command processors, and monitors their execution.

The TMP consists of seven separate routines. They are:

IKJEFT01	- TMP initialization routine.
IKJEFT02	- TMP mainline routine.
IKJEFT03	- TMP attention exit routine.
IKJEFT04	- TMP STAI exit routine.
IKJEFT05	- TMP STAE exit routine.
IKJEFT06	- TMP message module.
IKJEFT07	- TMP STAE retry routine.



Load Module Names	Normal Residence	Approximate Sizes
IKJEFT01	SYS1.LINKLIB	2K bytes
IKJEFT02	SYS1.LINKLIB	7K bytes

Figure 7. Program Hierarchy: Terminal Monitor Program

Module Operation

The following descriptions briefly describe the processing operation in each executable module of the TMP.

IKJEFT01 -- TMP INITIALIZATION ROUTINE

Builds tables and work areas, sets up exit routines, builds the first element on the input stack, and initializes the command buffer before transferring control to TMP mainline IKJEFT02.

- Gets main storage for the TMP work area.
- Sets up the environment control table (ECT).
- Sets up the TMP STAE exit.
- Sets up the TMP ATTN exit.
- Loads the TMP STAI exit routine.
- Loads the TIME command processor.
- Sets up the input stack.
- Gets main storage for the command buffer.
- Gets first command and places it in command buffer.

IKJEFT02 -- TMP MAINLINE ROUTINE

Obtains a commandname, gives control to the appropriate command processor, waits for it to complete before obtaining another command.

- Obtains first command from IKJEFT01, subsequent commands using PUTGET service routine.
- Checks commandname for validity using command scan.
  - if invalid, issues diagnostic message.
  - if null, obtains another command.
  - if question mark, sends any second-level messages queued by the last command processor using PUTLINE service routine.
  - if TIME, obtains running time for the terminal session by branching to the TIME command processor.
  - if TEST, allows the user to test a program by linking to the TEST command processor.
- Searches the user command library using a BLDL macro instruction to obtain the appropriate command processor.
  - if not found, the TMP assumes that the command processor is EXEC and that the invalid command name is a valid member name for use by the EXEC command processor.
- Abnormally detaches a previous command processor, if any.
- Attaches the appropriate command processor.
- Waits on an ECB list. The operating system dispatcher will give control to the appropriate command processor.
- On return from the dispatcher following a POST, does one of the following:
  - Detaches a normally completed command processor and gets another command. (CP ECB posted.)
  - Abnormally detaches a command processor and gets another command. (Attention ECB posted.)
  - Attempts to recover from error. (STAI ECB posted.)
  - Returns to LOGON/LOGOFF scheduler. (STOP/MODIFY ECB posted.)

IKJEFT03 -- TMP ATTENTION EXIT ROUTINE

Obtains command from attention buffer. Checks command syntax using command scan service routine. Prompts for valid command using PUTLINE and GETLINE. Special cases:

- Time-Obtains running time for terminal session.
- Question mark-Sends messages to the terminal. Prompt for additional input using PUTLINE/GETLINE.
- Null-returns to caller.

Other commands:
- Marks interrupted task non-dispatchable using STATUS STOP macro.
- Moves new command into the TMPCMDWT field of the TMP work area.
- Posts the TMP attention ECB. When the TMP gets control it will obtain the new command to replace the interrupted command.

IKJEFT04 -- TMP STAI EXIT ROUTINE

Inform terminal that a task is termianting abnormally.

If the command processor's STAE exit routine has marked the task unrecoverable, returns to the ABEND/STAE interface routine.

If the command processor was executing under TEST when the ABEND occurred, posts the TMP STAI ECB.

Otherwise:
- Prompts the terminal for a command using STACK and PUTGET.
- Posts the TMP STAI ECB.
- Waits on STAI Exit ECB.

If recovery successful, marks recoverable, returns to ABEND.

If recovery unsuccessful, returns to ABEND.

IKJEFT05 -- TMP STAE EXIT ROUTINE

Determines whether recovery is possible using branch table. If not possible, returns to ABEND/STAE interface routine with "no-retry" code. If possible, takes a SNAP dump of the user's region if SYSABEND or SYSUDUMP was specified. Loads the STAE retry routine IKJEFT07, and returns to the ABEND/STAE interface routine with a "retry" code.

If the retry attempt by the TMP STAE retry routine fails, the TMP STAE exit routine is reentered from the ABEND/STAE interface routine. It again dumps the user's region if SYSABEND or SYSUDUMP was specified, detaches all subtasks and frees subpools 1-127 before returning to the ABEND/STAE interface routine.

IKJEFT07 -- TMP STAE RETRY ROUTINE

Depends upon whether a retry has been attempted for this command processor.

If no retry has been attempted, control is passed to TMP mainline IKJEFT02 for a retry.

If retry has been attempted, control is passed to TMP initialization, IKJEFT01, for re-initialization of IKJEFT02, 03, 04, 05, and 25.

Section 4: Directory

This table contains information that will help you find the appropriate program description or assembly listing. It correlates information from three sources:

- The source code.
- The executable load modules.
- This manual.

Label	Common Name	Load Module Name	Assembly Module Name	Control Section Name	Description	Diagram
IKJEFT01	TMP Initial-ization	IKJEFT01	IKJEFT01	IKJEFT01	Builds tables and work areas, sets up exit routines.	2
IKJEFT02	TMP Mainline	IKJEFT02	IKJEFT02	IKJEFT02	Gets commands, supervises execution of TSO command processors.	3
IKJEFT03	TMP ATTN Exit	IKJEFT03	IKJEFT03	IKJEFT03	Handles ATTN request directed to the TMP.	4
IKJEFT04	TMP STAI Exit	IKJEFT02	IKJEFT04	IKJEFT04	Intercepts abnormally terminating command processors or program tasks.	5
IKJEFT05	TMP STAE Exit	IKJEFT02	IKJEFT05	IKJEFT05	Intercepts abnormally terminating TMP or TEST command processor.	6
IKJEFT06	TMP Messages	IKJEFT01 IKJEFT02	IKJEFT06	IKJEFT06	Contains TMP messages.	
IKJEFT07	TMP STAE Retry Routine	IKJEFT02	IKJEFT07	IKJEFT07	Deletes IKJEFT02, 03 04, 05 and 25. Transfers control to IKJEFT01 for re-initialization.	6

Section 5: Data Areas

This section describes the major data areas used by TMP routines, including:

Command buffer (CBUF)
Command processor parameter list (CPPL)
Environment control table (ECT)
Protected step control block (PSCB)
Terminal attention exit element (TAXE)
Terminal attention interrupt element (TAIE)
Test parameter list (TPL)
TMP parameter list
TMP retry work area (TMPWA2)
TMP work area (TMPWORKA)
User profile table (UPT)

For each data area, the following inforamation appears:

- Size in bytes.
- Name(s) of the routine(s) that creates it.
- Name(s) of the routine(s) that use and/or update it.
- Field names, displacements, size, and contents.
- Cross-references to method of operation diagrams.

COMMAND BUFFER (CBUF)

Size:	Variable.
Constructed by:	IKJEFT01.
Located in:	Subpool 1.
Updated by:	IKJEFT02 using PUTGET service routine.
Used by:	IKJEFT02.
Contents:	Commands, subcommands, and/or operands.

Operation Diagrams
3

Displacement Dec.	Hex.	Field Name	Size in Byte	Contents
0	0	CBUFLNG	2	Length of command buffer
2	2	CBUFOFF	2	Offset to data field
4	4	CBUFDATA	VAR	Commands, subcommands, and/or operands.

COMMAND PROCESSOR PARAMETER LIST (CPPL)

Size:	16 bytes.
Constructed by:	IKJEFT01.
Located in:	Subpool 1.
Updated by:	Command processors.
Used by:	All command processors except the TEST command processor.
Contents:	Parameter List.

Operation Diagrams
3

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	CPPLCBUF	4	↑Command buffer (CBUF)
4	4	CPPLUPT	4	↑User profile table (UPT)
8	8	CPPLPSCB	4	↑Protected step control block (PSCB)
12	C	CPPLECT	4	↑Environment control table (ECT)

ENVIRONMENT CONTROL TABLE (ECT)

Size:	40 bytes.
Constructed by:	IKJEFT01.
Located in:	Subpool 1.
Updated by:	TSO command processors and service routines.
Used by:	TMP and TSO command processors and service Routines.
Contents:	Information about the user's environment in the foreground region.

Operation Diagrams
2

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	2	ECTRTCF	1	ABEND Flags. Bit settings, as follows: Bit Meaning when on 0 Command processor abnormally terminated. 1-7 Reserved (0).
1	1	ECTRTCD	3	Return code from ABEND
4	4	ECTIOWA	4	↑I/O service routine list (IOSRL).
8	8	ECTMSGF	1	Message Flags. Bit settings, as follows: Bit Meaning when on 0 Delete second-level messages. 1-7 Reserved (0).
9	9	ECTSMSG	3	↑Second level message chain, or zero if no messages are chained.
12	C	ECTPCMD	8	Command Name
20	14	ECTSCMD	8	Subcommand Name
28	1C	ECTSWS	4	ECT Switches. Bit settings, as follows: Bit Meaning when on
		ECTNOPD		0 No parameters exist in command buffer.
				1 Reserved (0).

(Continued)

Displacement Dec.	Hex.	Field	Size in Bytes	Contents
		ECTATRM		2 The TMP is terminating the command processor by using a DEACH macro instruction with a STAE operand.
		ECTLOGF		3 LOGON or LOGOFF command processor has requested re-logon or logoff.
		ECTNMAL		4 No user messages at logon.
		ECTNNOT		5 No system messages at logon.
				6-7 Reserved (0).
29	1D	ECTDDNUM	3	Counter used by dynamic allocation SVC routines when assigning temporary ddnames.
32	20	ECTUSER	4	Reserved for installation.
36	24		4	Reserved (0).

PROTECTED STEP CONTROL BLOCK (PSCB)

Size:	72 Bytes.
Located in:	Subpool 0.
Created by:	LOGON/LOGOFF Scheduler.
Referenced by:	TSO Command Processors.
Contents:	Information about a terminal user's job.

Operation Diagrams
2

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	PSCBUSER	7	User identification (padded right with blanks).
7	7	PSCBUSRL	1	Length of user identification.
8	8	PSCBGPNM	8	Group name initialized by LOGON.
16	10	PSCBATRI	1	User authorization flags.
				Bit Meaning when on
		PSCBCTRL		0 Terminal user authorized to use OPERATOR commands.
		PSCBACCT		1 Terminal user authorized to use ACCOUNT commands.
		PSCBJCL		2 Terminal user authorized to use SUBMIT, CANCEL, STATUS, and OUTPUT commands.
				3-15 Reserved (0).
17	11		1	Reserved for IBM use (0).
18	12	PSCBATR2	1	Installation attribute flags.
19	13		1	Reserved for installation use (0).
20	14	PSCBCPU	4	Cumulative CPU time used during session.
24	18	PSCBSWP	4	Cumulative time resident in the region.
28	1C	PSCBLTIM	4	Actual logon time of day.
32	20	PSCBTCPU	4	Total CPU time used in this accounting period, excluding this session.
36	24	PSCBTSWP	8	Total time user job has been resident in region during this accounting period, excluding this session.

(Continued)

Displacement Dec.	Hex.	Field	Size in Bytes	Contents
40	28	PSCBTCON	8	The total "connect" time for the user during this accounting period, excluding the current session. Note: All times are in 26.04166 microsecond timer units.
44	2C	PSCBTCO1	4	Second word of PSCBTCON.
48	30	PSCBRLGB	4	↑ Relogon buffer.
52	34	PSCBUPT	4	↑ User profile table.
56	38	PSCBUPTL	2	Length of UPT.
58	3A		2	Reserved for IBM, (0).
60	3C	PSCBRSZ	4	Region size requested in 2K units.
64	40	PSCBU	8	Reserved for installation, (0).

TERMINAL ATTENTION EXIT ELEMENT (TAXE)

Size:	144 bytes.
Constructed by:	IKJEFT01 using the STAX macro instructions.
Located in:	Subpool 1.
Updated by:	STAX service routine
Used by:	Region control task (RCT).
Contents:	An interrupt request block (IRB), an interrupt queue element (IQE), and a work area used to schedule the attention exit when an attention interrupt occurs.

Operation Diagrams
4

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	TAXEIRB	96	IRB
96	60	TAXENIQE	4	↑Next available IQE
100	64	TIQELNK	4	↑Next IQE
104	68	TIQEPARM	4	↑Parameter to asynchronous exit routine
108	6C	TIQEIRB	4	↑IRB to schedule
112	70	TAXETCB	4	↑TCB
116	74	TAXELNK	4	↑Next TAXE
120	78	TAXEXPSW	4	Left half of PSW for attention exit routine
124	7C	TAXEEXIT	4	↑Attention exit routine
128	80	TAXESTAT	1	TAXE status flags. Bit settings, as follows: __Bit__ __Meaning when on__ 0 Problem key. 1 Problem mode. 2 Requested TAXE. 3-7 Reserved (0).
129	81	TAXEPARM	3	↑STAX parameter list
132	84	TAXETAIE	4	↑TAIE
136	88	TAXEIBUF	4	↑Attention buffer
140	8C	TAXEUSER	4	↑User parameter area

TERMINAL ATTENTION INTERRUPT ELEMENT (TAIE)

Size:	72 bytes.
Constructed by:	IKJEFT01 using the STAX macro instruction.
Located in:	Subpool 1.
Updated by:	Region control task (RCT).
Used by:	IKJEFT03.
Contents:	Interrupt address and contents of general registers 0-15 when interrupt occurred.

Operation Diagrams
4

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	TAIEMSGL	2	Length in bytes of a message placed in an input buffer specified by the STAX macro instruction. If no input buffer is specified, the field is 0.
2	2	TAIETGET	1	Return code from TGET macro instruction issued by attention prologue routine in the region control task (RCT) and checked by attention exit routine IKJEFT03.
3	3		1	Reserved(0).
4	4	TAIEIAD	4	Interrupt address. Right half of the interrupted PSW. Address at which TMP mainline IKJEFT02 (or a previous attention exit routine) was interrupted.
8	8	TAIERSAV	64	Contents of general registers 0-15 of interrupted programs.

TEST PARAMETER LIST (TPL)

Size	60 Bytes.
Constructed by:	IKJEFT01.
Located in:	Subpool 1.
Updated by:	TEST command processor.
Used by:	TEST command processor.
Contents:	Addresses of data areas used by TEST command processor.

Operation Diagrams
1,5

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	TPLCBUF	4	↑Command buffer
4	4	TPLUPT	4	↑UPT
8	8	TPLPSCB	4	↑PSCB
12	C	TPLECT	4	↑ECT
16	10	TPLTBUF	4	↑TEST buffer
20	14	TPLCTCB	4	↑Attached command processor's TCB
24	18	TPLSTAI	4	↑TMP STAI exit routine
28	1C	TPLSPLS	4	↑STAI parameter list
32	20	TPLNECB	4	↑ECB for an abnormally terminating command processor
36	24	TPLNTCB	4	↑TCB for an abnormally terminating command processor
40	28	TPLMECB	4	↑STOP/MODIFY ECB
44	2C	TPLCECB	4	↑Attached command processor's ECB
48	30	TPLIECB	4	↑TMP STAI ECB
52	34	TPLAECB	4	↑TMP attention ECB
56	38	RESV	4	Reserved (0).

TMP PARAMETER LIST

Size:	8 bytes.
Constructed by	LOGON/LOGOFF Scheduler or TMP STAE retry routine IKJEFT07.
Located in:	Subpool 1.
Updated by:	IKJEFT07.
Used by:	IKJEFT01.
Contents:	Parameter list for TMP initialization routine IKJEFT01.

Operation Diagrams
2

Displacement Dec.	Hex.	Field	Size in Bytes	Contents
0	0	FSTCMD	4	↑First command or X'FFFFFFFF' if a STAE retry is in process.
4	4	RETRYWAP	4	↑TMP retry work area (if a STAE retry is in process).

TMP RETRY WORK AREA (TMPWA2)

Size:	16 bytes.
Constructed by:	IKJEFT01.
Located in:	Subpool 1.
Updated by:	IKJEFT05, IKJEFT07.
Used by:	IKJEFT01.
Contents	Information to be used when re-initializing the TMP during STAE retry processing.

Operation Diagrams
2

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	TMPWAPTR	4	↑TMP work area.
4	4	SAVAPTR	4	↑Original register save area.
8	8	RETRYFP	4	↑Retry flags.
12	C	RETRYFLG	1	TMP Retry Flags. Bit settings, as follows:
				<u>Bit</u> <u>Meaning when on</u>
		TMPRINIT		0 Reinitialization is in progress.
		TMPRTRY		1 Retry is in progress.
				2-7 Reserved (0).
13	D		3	Reserved (0).

TMP WORK AREA (TMPWORKA)

Size	462 Bytes.
Constructed by:	IKJEFT01.
Located in:	Subpool 1.
Updated by:	IKJEFT02, IKJEFT03, IKJEFT04, IKJEFT05, IKJEFT07.
Used by:	IKJEFT02, IKJEFT03, IKJEFT04, IKJEFT05, IKJEFT07.
Contents:	Control information and addresses of data areas used by the terminal monitor program.

Operation Diagrams
2

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	TPL	60	Test parameter list.
60	3C	TMPNECB	4	†TMP STAI ECB.
64	40	TMPCECB	4	†Attached command processor's ECB.
68	44	TMPIECB	4	ECB for STAI Post.
72	48	TMPAECB	4	ECB for attention post.
76	4C	TMPCMDWT	4	†Command passed from TMP attention exit routine.
80	50	TMPTIME	4	†Time routine
84	54	TMPSWS	4	TMP internal switches. Bit settings as follows:
				<u>Bit</u> <u>Meaning when on</u>
		TMPTEST		0 TEST program in control.
		TMPCMDW		1 Command waiting.
		TMPNFCMD		2 First command is processed.
		TMPACTRL		3 TMP attention exit is in control.
		TMPSCTRL		4 TMP STAI exit is in control.

(Continued)

Displacement Dec.	Hex.	Field	Size in Bytes	Contents
88	58	MULTLST	40	Multilevel messages list.
128	80	AMSGLIST	12	ATTN message list.
140	8C	ASRPARM	20	ATTN service routine parameter area.
160	A0	TMPZEROS	4	Dummy command buffer.
164	A4	RCODE	4	Return code save area.
168	A8	ARCODE	4	↑ATTN return code save area.
172	AC	SCANFLG	4	Scan flags.
176	B0	ASCANFLG	4	ATTN scanflags.
180	B4	ATTCHPTR	4	↑ATTACH parameter list.
184	B8	CPPLPTR	4	↑Command processor parameter list (CPPL).
188	BC	DYNAPPTR	4	↑Dynamic allocation.
192	C0	GTPBPTR	4	↑GETLINE parameter list.
196	C4	PGPBPTR	4	↑PUTGET parameter list.
200	C8	PTPBPTR	4	↑PUTLINE parameter list.
204	CC	READYPTR	4	↑TMP MODE message.
208	D0	SCANAP	4	↑SCAN answer area.
212	D4	ASCANAP	4	↑ATTN SCAN answer area.
216	D8	SRPLPTR	4	↑Service routine parameter list.
220	DC	ASRPLPTR	4	↑ATTN service routine parameter list.
224	E0	STAXPTR	4	↑STAX parameter list.
228	E4	STPBPTR	4	↑STACK parameter list.
232	E8	WARPTR	4	↑TMP retry work area.
236	EC	BLDLLIST	62	BLDL list.
298	12A	------	2	Padding to get to word boundary.
300	12C	------	20	Reserved.

USER PROFILE TABLE (UPT)

Size:	16 Bytes.
Located in:	Subpool 0.
Created by:	LOGON/LOGOFF scheduler.
Updated by:	PROFILE command processor.
Referenced by:	IKJEFT45, IKJEFT56.
Contents:	Information about terminal user.

Operation Diagrams
2

Displacement Dec.	Displacement Hex.	Field Name	Size in Bytes	Contents
0	0	--------	2	Reserved (0).
2	2	UPTUSER	10	Reserved for installation use.
12	C	UPTSWS	1	User environment switches. Bit settings, as follows:
				Bit 0 — Meaning when on: Reserved (0).
		UPTNPRM		1 — Prompt; if zero, no prompt.
		UPTMID		2 — Message identifiers; if zero, message identifiers will be removed.
		UPTNCOM		3 — No user communication using the SEND command; if zero, user communication is allowed.
		UPTPAUS		4 — PAUSE was specified; if zero, NOPAUSE was specified.
		UPTALD		5 — Attention is a line delete character; if zero, attention is not a line delete character.
				6-7 — Reserved (0).
13	D	UPTCDEL	1	Character delete character.
14	E	UPTLDEL	1	Line delete character.
15	F	--------	1	Reserved (0).

Section 6: Diagnostic Aids

This section contains the following information:

- Messages (Figure 8) -- a list of messages issued by TMP routines.
- Register Usage (Figure 9) -- a summary of the use of general registers 0-15.
- Return Codes (Figure 10) -- a summary of return codes and their meanings. Unless otherwise specified, return codes are contained in register 15.

Other useful diagnostic information is contained in the TMP work area (TMPWORKA) and TMP retry work area (TMPWA2). These data areas are described in Section 5.

Message ID	Message	Issued by
IKJ56621I	INVALID COMMAND SYNTAX	IKJEFT02,IKJEFT03,
IKJ56622I	COMMAND NOT FOUND	IKJEFT02
	READY	IKJEFT02,IKJEFT03,
IKJ56641I	command ENDED DUE TO ERROR+	IKJEFT02/TEST
IKJ5664I	System User ABEND CODE xxxx	IKJEFT02/TEST
IKJ56600I	COMMAND SYSTEM ERROR	IKJEFT05
IKJ56601I	COMMAND SYSTEM RESTARTING DUE TO CRITICAL ERROR	IKJEFT05
IKJ56602I	COMMAND SYSTEM RESTARTING DUE TO ERROR	IKJEFT05

Figure 8. Messages: Terminal Monitor Program

	IKJEFT01		IKJEFT02		IKJEFT03		IKJEFT04		IKJEFT05		IKJEFT07	
Register	Name	Use	Name	Use	Name	Use	Name	Use	Name	Use	Name	Use
0	R0	Work Register	R0	Work Register	R0	Work Register	R0	Entry Code ①	R0	Entry Code ①	R0	Work Register
1	R1	↑ TMP parameter list	R1	↑ Command buffer	R1	↑ Attention Exit	R1	↑ Work area or ABEND code ②	R1	↑ Work area or ABEND code ②	R1	↑ STAE parameter list
2	R2	Work Register	R2	Work Register	R2	Work Register	R2	↑ STAI ④ parameter list	R2	↑ STAE ④ parameter list	R2	Work Register
3	R3	Work Register	R3	Work Register	R3	Work Register	R3	Work Register	R3	Work Register	R3	Work Register
4	R4	Work Register	R4	Work Register	--	Work Register	--	Work Register	--	Work Register	--	Work Register
5	R5	Work Register	R5	Work Register	--	Work Register	--	Work Register	--	Work Register	--	Work Register
6	R6	Work Register	R6	Work Register	--	Work Register	--	Work Register	--	Work Register	--	Work Register
7	--	Work Register	--	Work Register	--	Work Register	R7	Work Register	R7	Work Register	--	Work Register
8	--	Work Register	SRPARMP	↑ Service routine parameters	--	Work Register	--	Work Register	--	Work Register	--	Work Register
9	WORK-APTR	↑TMP work area	WORK-APTR	Work Register	--	Work Register	WORK-APTR	↑ TMP work area	WORK-APTR	Work Register	--	Work Register
10	--	Work Register	--	Work Register	--	Work Register	--	Work Register	--	Work Register	--	Work Register
11	--	Base Register	--	Base Register	--	Base Register	--	Base Register	--	Base Register	--	Base Register
12	--	Base Register	--	Base Register	--	Base Register	--	Base Register	--	Base Register	--	Work Register
13	R13	↑ Register save area ①	R13	↑ Register save area	--	↑ Register save area	R13	↑ Register save area	R13	↑ Register save area	R13	↑ Register save area
14	--	↑Return point①	R14	↑ Return point	--	↑ Return point	R14	↑ Return point	R14	↑ Return point	--	↑ Return point
15	R15	↑ Entry point/ return code ②	R15	↑ Entry point	R15	↑ Entry point/ return point①	R15	↑ Entry point/ return point③	R15	↑ Entry point	--	↑ Entry point
NOTES:	① In LOGON/LOGOFF scheduler ② 0 - Normal 4 - Unable to open command library		① If entry is from IKJEFT07, the buffer length field is 0		① 0 - No change to program flow 4 - Address of next executable instruction		① 0 - I/O quiesed 4 - I/O halted 8 - No I/O 12 - No work area ④ If no work area was obtained (R0=12)		② Depends on entry code ③ 0 - Retry 4 - No retry			

Figure 9. Register Usage: Terminal Monitor Program

Routine	Return Code Hexadecimal	Meaning
IKJEFT01	101	BLDL Error.
	102	DAIR Error.
	103	PUTLINE Error.
	104	STACK Error.
	105	STAX Error.
	Note:	These are error exit (user ABEND) codes used by IKJEFT01 when going to IKJEFT05.
IKJEFT02	201	BLDL Error.
	202	DAIR Error.
	203	PUTLINE Error.
	204	STACK Error.
	205	STAX Error.
	Note:	These are error exit (user ABEND) codes used by IKJEFT02 when going to IKJEFT05.
IKJEFT03	00	No change is made in program flow.
	↑ next instruction to be executed.	For change in program flow.
IKJEFT04	0	Unrecoverable error.
	4	Recoverable error.
IKJEFT05	00	Retry is not to be attempted.
	04	Retry is to be attempted.
IKJEFT06	None	
IKJEFT07	None	

Figure 10. Return Codes: Terminal Monitor Program

Part 2: Terminal I/O Service Routines

2

Section 1: Introduction

The terminal I/O service routines handle terminal input/output operations required by the LOGON/LOGOFF Scheduler, the terminal monitor program, the TSO command processors, and other TSO problem programs.

There are four terminal I/O service routines:

- STACK -- which determines the current source of input: (1) from the terminal, or (2) from an in-storage list.
- GETLINE -- which obtains a line of input from the terminal or from the current source of input.
- PUTLINE -- which sends line(s) of data to the terminal, formats messages and sends them to the terminal.
- PUTGET -- which sends a message to the terminal and obtains a line of input from the current source of input.

The terminal I/O service routines can be invoked directly (by using the LINK or LOAD/CALL macro instructions) or they can be invoked using system macro instructions: STACK, GETLINE, PUTLINE, and PUTGET.

There are two forms of these macro instructions: the list form and the execute form. The list form generates most of the control blocks and parameter lists required, while the execute form generates executable code that includes a LINK SVC instruction. The LINK SVC results in a branch-and-link-register instruction to the appropriate entry point.

For further information about the terminal I/O macro instructions, refer to OS/VS2 TSO Guide to Writing a Terminal Monitor Program or a Command Processor, GC28-648.

As supplied with TSO, the terminal I/O service routines reside in SYS1.LINKLIB and will execute in the user's foreground region with the protection key assigned to that region. The installation may choose to make the terminal I/O service routines resident in the TSO link pack area (TSLPA) in the region assigned to the Time Sharing Control Task (TSCT).

Section 2: Method of Operation

This section describes the method of operation of the terminal I/O service routines. It includes the following method operations diagrams:

- Method of Operation Diagram 7: Terminal I/O Service Routines, Overview -- which shows how the terminal I/O service routines are used by the terminal monitor program, TSO command processors, and other TSO problem programs.

- Method of Operation Diagram 8: STACK Service Routine -- which shows how the STACK service routine determines the current source of input (1) from the terminal, or (2) from an in-storage list.

- Method of Operation Diagram 9: GETLINE Service Routine -- which shows how the GETLINE service routine obtains a line of input from the terminal or from the current source of input.

- Method of Operation Diagram 10: PUTLINE service routine -- which shows how the PUTLINE service routine sends line(s) of data to the terminal, formats messages, and sends messages to the terminal.

- Method of Operation Diagram 11: PUTGET service routine (Command Mode) -- which shows how the PUTGET service routine obtains commands from the current source of input.

- Method of Operation Diagram 12: PUTGET service routine (Prompting Mode) -- which shows how the PUTGET service routine obtains operands and data from the terminal in response to a prompting message.

Each method of operation diagram includes a cross-reference table to help you find the appropriate assembly listing.

Overview

Method of Operation Diagram 7 shows how the terminal I/O service routines are used by the terminal monitor program, the TSO command processors, and other TSO problem programs.

Briefly, here is what happens:

- The terminal monitor program uses STACK to set up the first (bottom) element on the input stack to define the terminal as the current source of input. Later, the TMP uses PUTGET to obtain commands from the terminal and uses PUTLINE to write informational messages to the terminal.

- TSO command processors may use STACK to set up other elements on the input stack to define either the terminal or an in-storage list as the current source of input. TSO command processors use PUTGET to obtain subcommands, GETLINE to obtain data, and PUTLINE to send data or informational messages to the terminal.

- When a TSO command processor or problem program begins to terminate abnormally, the command processor uses STACK to delete all elements from the input stack (except the bottom element).

- Other TSO problem programs, including parse and other TSO service routines, may use any or all of the terminal I/O service routines.

INPUT STACK

The input stack is a variable-sized control block that contains one or more elements, each of which defines a source of input:

- From the terminal, or
- From an in-storage list.

There are two types of in-storage lists: a source list and a procedure list. A source list contains source language statements or data. A procedure list is a list of TSO commands.

I/O SERVICE ROUTINE LIST

The I/O service routine list (IOSRL) contains the address of the bottom element (a terminal element) and the top element (a terminal element or a storage element). The top element defines the current source of input, while the bottom element always describes the terminal as a source of input.

The GETLINE and PUTGET service routines refer to the IOSRL to determine the current source of input, but they cannot update it. Only STACK can update the IOSRL and the input stack.

STACK Service Routine

Method of Operation Diagram 8 shows how the STACK service routine creates the I/O service routine list (IOSRL) and input stack (INSTACK) and adds or deletes elements to or from the input stack.

ENTRY TO STACK

STACK is the entry to by a branch and link to entry point IKJSTCK in load module IKJPTGT. The calling program may invoke STACK directly, by issuing a LINK macro instruction, or indirectly, by issuing a STACK macro instruction which results in a LINK SVC.

On entry to STACK , register 1 points to the I/O parameter list (IOPL) which contains the address of the STACK parameter block (STPB).

MANAGING THE INPUT STACK

STACK performs one of the following functions:

- Adds an element to the top of the stack.
- Deletes an element from the top of the stack.
- Deletes the current procedure element from the stack.
- Deletes all elements from the stack except the bottom element.

Before adding an element to the stack, STACK checks to see if storage is available and, if necessary, obtains storage for a new input stack that is 32 bytes larger than the current one.

If STACK is to delete a procedure element but the top element is not a procedure element, STACK deletes all elements from the top of the stack down to and including the first procedure element, until it reaches the bottom element.

STACK updates the IOSRL to point to the top and bottom elements or the Input Stack before returning control to the calling program.

RETURN TO CALLING PROGRAM

STACK issues a RETURN macro instruction to return control to the calling program. At exit from STACK, register 15 contains one of the following return codes:

Code	Meaning
X'00'	Normal. Element(s) added to or deleted from the input stack.
X'04'	Error. Invalid input to STACK. Either an invalid operation code or an invalid record in a in-storage list.

GETLINE Service Routine

Method of Operation Diagram 9 shows how the GETLINE service routine obtains lines of input from the terminal or from the current source of input, which may either be the terminal or an in-storage list.

ENTRY TO GETLINE

GETLINE is the entry to by a branch and link to entry point IKJEGTL in load module IKJPTGT. The calling program may invoke GETLINE directly, by issuing a LINK macro instruction, or indirectly, by issuing a GETLINE macro instruction which results in a LINK SVC.

On entry to GETLINE, register 1 points to the I/O parameter list (IOPL) which contains the address of the GETLINE parameter block (GTPB).

OBTAINING LINES FROM THE TERMINAL

If terminal is specified, or if the terminal is the current source of input, GETLINE obtains a physical line or a logical line of input from the terminal using the TGET macro instruction. A physical line is a line of input entered from the terminal. A logical line may consist of one or more physical lines, where '-' is the continuation character.

OBTAINING LINES FROM A LIST

If the current source of input is an in-storage list, GETLINE obtains the next record and calls STACK to update the I/O service routine list (IOSRL). If end-of-data is reached, GETLINE calls STACK to delete the current element from the input stack.

RETURN TO CALLING PROGRAM

GETLINE returns control to the calling program by a branch on register 14. At exit from GETLINE, register 15 contains one of the following return codes:

Code	Meaning
X'00'	Normal. Line of input obtained from the terminal.
X'04'	Normal. Line of input obtained from an in-storage list.
X'08'	Error. Communications ECB's post bit was on or user was disconnected.
X'0C'	Error. NOWAIT was specified as a TGET option and no buffer was available to TGET.
X'10'	Normal. EOD returned from an in-storage list.
X'14'	Error. Invalid parameters to GETLINE or TGET.
X'18'	Error. A conditional GETMAIN was executed and no space was available.

2

PUTLINE Service Routine

Method of Operation Diagram 10 shows how the PUTLINE service routine sends lines of data to the terminal, formats messages, and sends messages to the terminal.

ENTRY TO PUTLINE

PUTLINE is entered by a branch and link to entry point IKJPUTL in load module IKJPTGT. The calling program may invoke PUTLINE directly, by issuing a LINK macro instruction, or indirectly, by issuing a PUTLINE macro instruction which results in a LINK SVC.

On entry to PUTLINE, register 1 points to the I/O parameter list (IOPL) which contains the address of the PUTLINE parameter block (PTPB).

SENDING MESSAGES TO THE TERMINAL

PUTLINE formats messages and sends them to the terminal using the TPUT macro instruction. Messages are formatted by joining message segments, if necessary, and stripping off message identifiers, if specified. (If "format only" was specified, PUTLINE formats the message but does not send it to the terminal.) Second-level messages, if supplied, are chained to the ECTSMSG field of the environment control table (ECT) where they are available to the PUTGET service routine when the terminal user enters a question mark to request additional information.

SENDING DATA TO THE TERMINAL

PUTLINE sends data to the terminal (as received from the calling program) using the TPUT macro instruction. Each line of chained, multi-lined data is sent to the terminal until end-of-chain is reached.

RETURN TO CALLING PROGRAM

PUTLINE returns control to the calling program by a branch on register 14. At exit from PUTLINE, register 15 contains one of the following return codes:

Code	Meaning
X'00'	Normal. One of the following has occurred: • Line(s) of output were sent to the terminal. • Message(s) were sent to the terminal and second-level messages were chained. • Text was inserted (format only was specified).
X'04'	Error. Communication ECB's post bit was on.
X'08'	Error. NOWAIT was specified as a TPUT option and no buffer was returned by TPUT.
X'0C'	Error. Invalid parameters were sent to PUTLINE.
X'10'	Error. A conditional GETMAIN was executed and no space was available.

PUTGET Service Routine

The PUTGET service routine has two very different uses:

- Obtaining commands (Command Mode).
- Obtaining operands and data (Prompting Mode).

ENTRY TO PUTGET

PUTGET is entered by a branch and link to entry point IKJPTGT in load module IKJPTGT. The calling program may invoke PUTGET directly, by issuing a LINK macro instruction, or indirectly, by issuing a PUTGET macro instruction which results in a LINK SVC.

On entry to PUTGET, register 1 points to the I/O parameter list (IOPL) which contains the address of the PUTGET parameter block (PGPB).

OBTAINING COMMANDS

Method of Operation Diagram 11 shows how the PUTGET service routine obtains commands from the current source of input.

If the current source of input is the terminal, PUTGET sends a mode message to the terminal and obtains a line of input from the terminal. Examples of mode messages are:

Mode Message	Routine
READY	Terminal monitor program
EDIT	EDIT command processor
TEST	TEST command processor

If the current source of input is an in-storage list, PUTGET obtains a line of input from the in-storage list. When end-of-data is reached, PUTGET invokes STACK to delete the current procedure element. If the line begins with a question mark, PUTGET sends a second-level message to the terminal and obtains another line of input. (This process can continue until all second-level messages have been sent to the terminal.)

OBTAINING OPERANDS AND DATA

Method of Operation Diagram 12 (foldout) shows how the PUTGET service routine obtains operands and data from the terminal in response to a prompting message.

If the current input source is an in-storage list, or if the terminal user has specified "NO PROMPT" as one of his PUTGET options, an error return code is returned to the calling routine.

If the current source of input is the terminal, PUTGET sends a prompting message to the terminal and obtains a line of input from terminal. Examples of prompting messages are:

Prompting Message	Routine
ENTER USERID -	LOGON/LOGOFF Scheduler
REENTER -	Parse service routine

If the line begins with a question mark, PUTGET sends a second-level prompting message to the terminal (if one is available) and obtains another line of input. (This process can continue until all second-level messages have been sent to the terminal.)

2

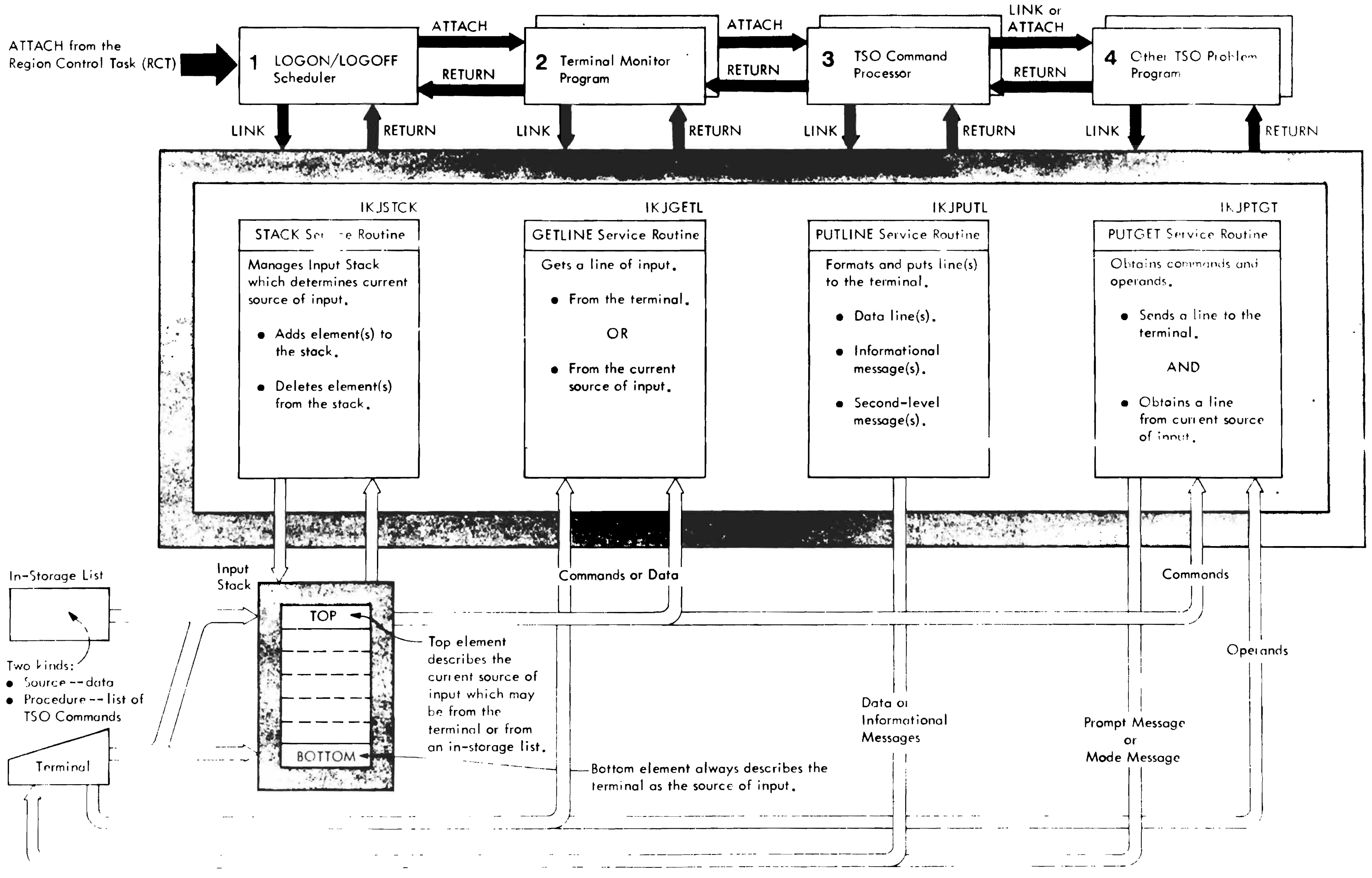
RETURN TO CALLING PROGRAM

PUTGET returns control to the calling probram by a branch on register 14. At exit from PUTGET, register 15 contains one of the following return codes:

Code	Meaning
X'00'	Normal. PUTGET sent a line to the terminal and received a line from the terminal.
X'04'	Normal. PUTGET sent a line to the terminal and received a line from an in-storage list.
X'08'	Error. The communications ECB's post bit was on.
X'0C'	Error. If in command mode -- NOPAUSE was specified and input was from an in-storage list. If in prompting mode -- NOPROMPT was specified or input was from an in-storage procedure.
X'10'	Error. NOWAIT was specified as a TPUT option and no TIOC buffer was available to TPUT.
X'14'	Error. NOWAIT was specified as a TGET option and TGET returned no buffer.
X'18'	Error. PUTGET received invalid parameters.
X'1C'	Error. For conditional GETMAIN. No space was available. available.

Method of Operation Diagrams

2


ATTACH from the Region Control Task (RCT)
1 LOGON/LOGOFF Scheduler
ATTACH
RETURN
2 Terminal Monitor Program
ATTACH
RETURN
3 TSO Command Processor
LINK or ATTACH
RETURN
4 Other TSO Problem Program
LINK
RETURN
LINK
RETURN
LINK
RETURN
LINK
RETURN
IKJSTCK
STACK Service Routine
Manages Input Stack which determines current source of input.
• Adds element(s) to the stack.
• Deletes element(s) from the stack.
IKJGETL
GETLINE Service Routine
Gets a line of input.
• From the terminal.
OR
• From the current source of input.
IKJPUTL
PUTLINE Service Routine
Formats and puts line(s) to the terminal.
• Data line(s).
• Informational message(s).
• Second-level message(s).
IKJPTGT
PUTGET Service Routine
Obtains commands and operands.
• Sends a line to the terminal.
AND
• Obtains a line from current source of input.
In-Storage List
Two kinds:
• Source -- data
• Procedure -- list of TSO Commands
Terminal
Input Stack
TOP
BOTTOM
Top element describes the current source of input which may be from the terminal or from an in-storage list.
Bottom element always describes the terminal as the source of input.
Commands or Data
Data or Informational Messages
Prompt Message or Mode Message
Commands
Operands

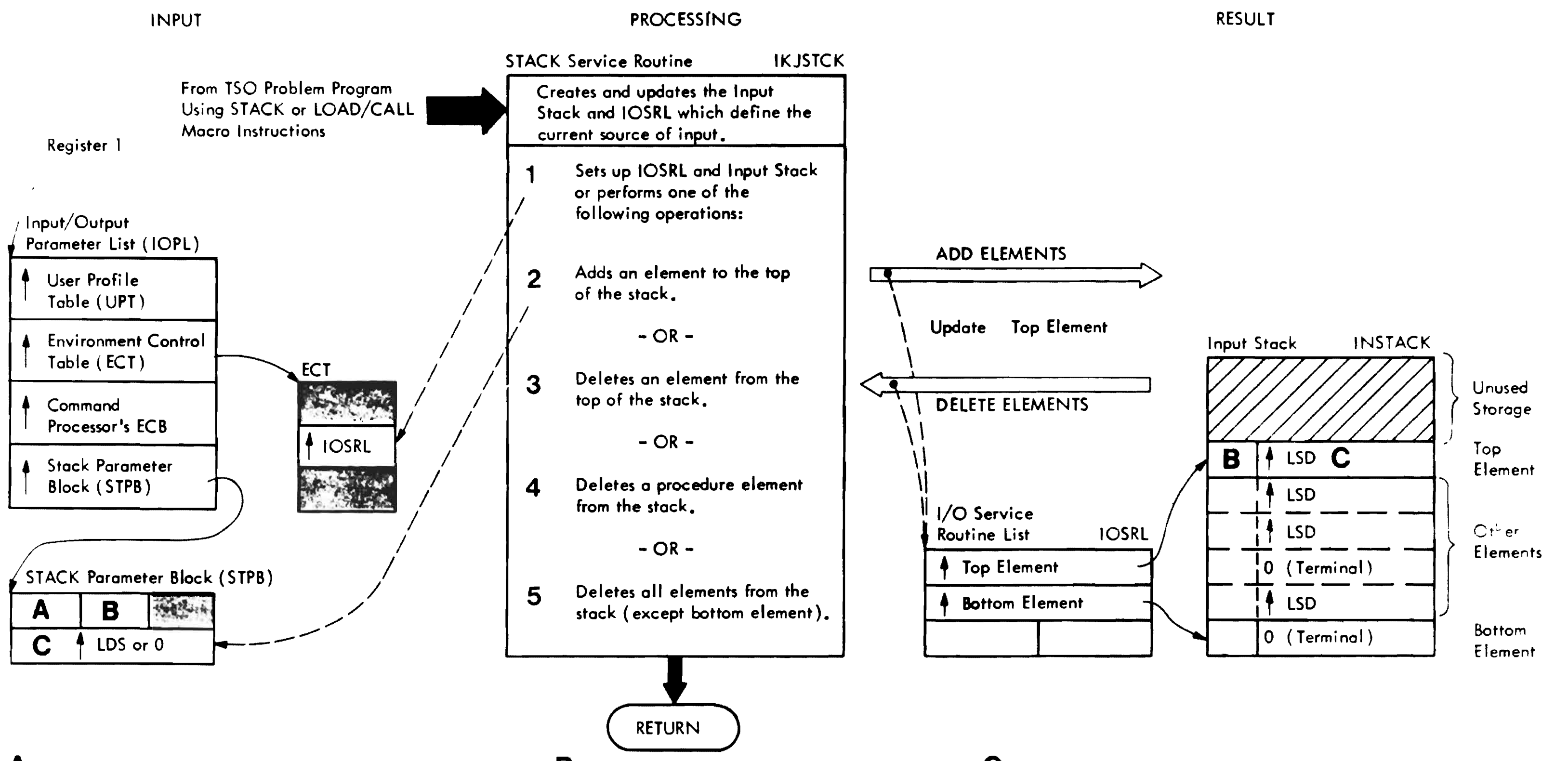
Method of Operation Diagram 7. Terminal I/O Service Routines (Overview) (Part 1 of 2)

CROSS REFERENCE TABLE

Key Description	Routine	Label	Diagram
1 The LOGON/LOGOFF Scheduler uses STACK to set up the first (bottom) element to describe the terminal as the current source of input and uses other I/O Service Routines for terminal I/O.	STACK GETLINE PUTLINE PUTGET	IKJEFT30 IKJEFT55 IKJEFT40 IKJEFT45	8 9 10 11-12
2 The TMP uses STACK to set up the first (bottom) element to describe the terminal as the current source of input. This first (bottom) element is never removed.	STACK	IKJEFT30	8
The TMP uses PUTGET to obtain commands. (an exception: The TMP Attention Exit Routine uses PUTLINE and GETLINE to obtain commands from the terminal. This is done because GETLINE optionally gets a line directly from the terminal, while PUTGET always gets a line from the current source of input.)	PUTGET	IKJEFT45	11
The TMP uses PUTLINE to write error messages.	PUTLINE	IKJEFT40	10
3 The TSO command processors may use STACK to set up in-storage lists. There are two kinds of in-storage lists: source (data), and procedure (TSO commands).	STACK	IKJEFT30	8
TSO command processors use PUTGET to obtain operands and subcommands. (Subcommand processors use PUTGET to obtain their operands.)	PUTGET	IKJEFT45	11-12
TSO Command processors use GETLINE to obtain data and use PUTLINE to write first and second level messages to the terminal or to write data to the terminal.	GETLINE PUTLINE	IKJEFT55 IKJEFT40	9 10
4 Other TSO problem programs may use any or all of the Terminal I/O Service Routines if they share subpool 78 and use the proper parameter lists.	STACK GETLINE PUTLINE PUTGET	IKJEFT30 IKJEFT55 IKJEFT40 IKJEFT45	8 9 10 11-12

Method of Operation Diagram 7. **Terminal I/O Service Routines (Overview) (Part 2 of 2)**

2


INPUT
PROCESSING
RESULT
Register 1
From TSO Problem Program Using STACK or LOAD/CALL Macro Instructions
Input/Output Parameter List (IOPL)
User Profile Table (UPT)
Environment Control Table (ECT)
Command Processor's ECB
Stack Parameter Block (STPB)
ECT
IOSRL
STACK Parameter Block (STPB)
A
B
C
LDS or 0
STACK Service Routine IKJSTCK
Creates and updates the Input Stack and IOSRL which define the current source of input.
1 Sets up IOSRL and Input Stack or performs one of the following operations:
2 Adds an element to the top of the stack.
- OR -
3 Deletes an element from the top of the stack.
- OR -
4 Deletes a procedure element from the stack.
- OR -
5 Deletes all elements from the stack (except bottom element).
RETURN
ADD ELEMENTS
Update Top Element
DELETE ELEMENTS
I/O Service Routine List IOSRL
Top Element
Bottom Element
Input Stack INSTACK
Unused Storage
B
LSD
C
Top Element
LSD
LSD
0 (Terminal)
LSD
0 (Terminal)
Other Elements
Bottom Element

A

Operation Code:

Bit	Meaning when set
0	Add one element to top of stack.
1	Delete one element from top of stack.
2	Delete current procedure element. If top element is not a procedure element, delete all elements down to and including first procedure element.
3	Delete all elements except bottom element.
4-7	Reserved (0).

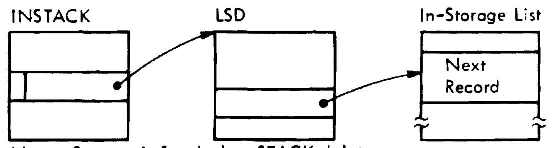
B

Element Code:

Bit	Meaning when set
0	Terminal element.
1	Storage element.
2-5	Reserved (0).
6	Procedure element; if 0, source element.
7	List TSO commands to the terminal.

C

↑ List Source Descriptor (LSD) for In-Storage List;
0 for a Terminal Element.



Note: Storage is freed when STACK deletes one or more elements from the stack.

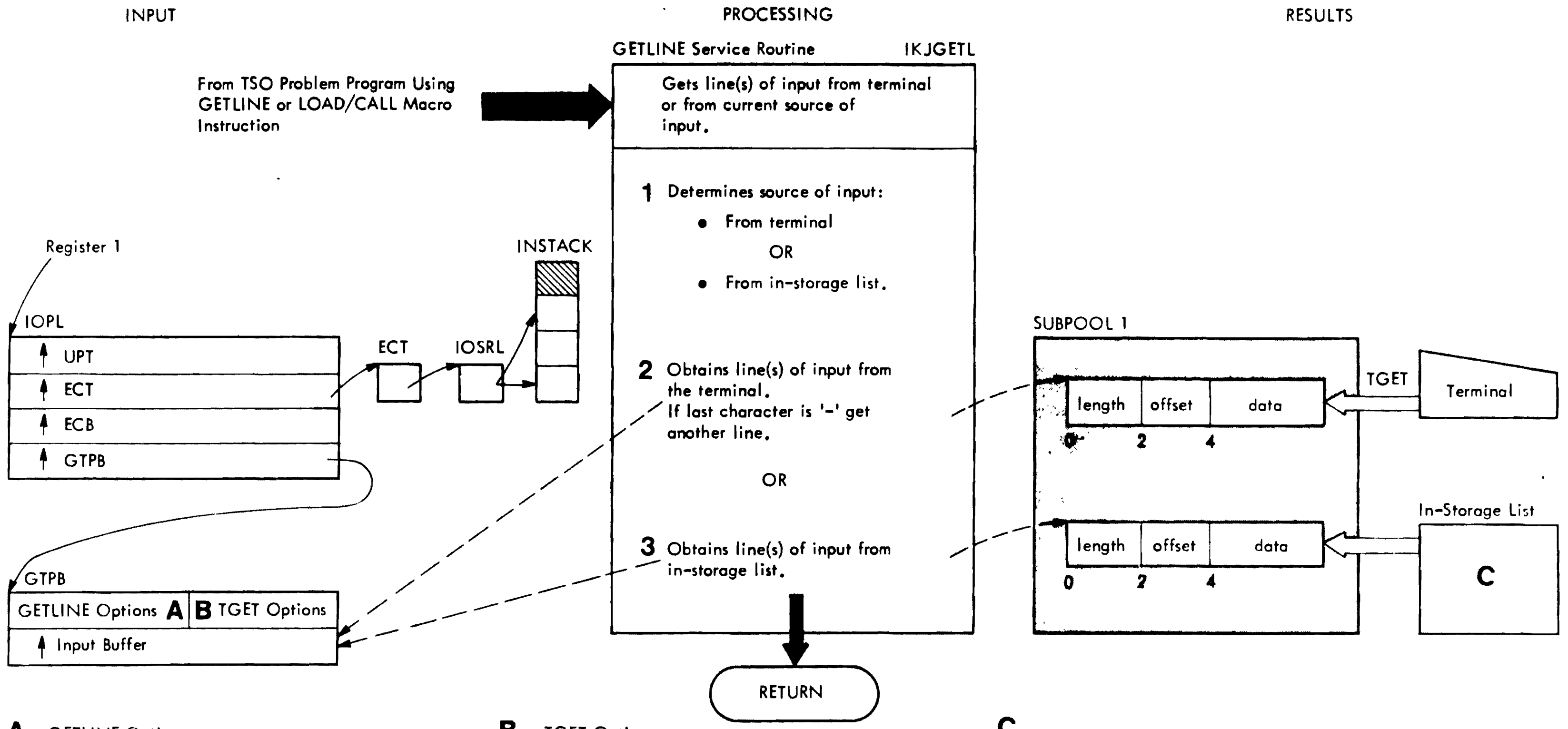
Method of Operation Diagram 8. STACK Service Routine

Method of Operation Diagram 8. STACK Service Routine (Part 1 of 2)

CROSS REFERENCE TABLE

Key Description	Routine	Label
1 If the ECTIOWA field of the ECT is zero, STACK gets main storage for the Input Stack and the IOSRL. STACK then checks the STACK options field of the STPB to determine the function requested.	STACK	IKJEFT30
2 Before adding an element to the stack, STACK checks to see if space is available and, if necessary, gets main storage for a new stack that is 32 bytes larger than the current stack.		
3 STACK deletes an element from the stack by freeing the LSD and in-storage list of any non-terminal element and changing the Top Element Pointer in the IOSRL to point to the next lower element.		DELETE
4 STACK finds the next procedure element on the stack and deletes it by freeing the LSD and in-storage list of any non-terminal element and changing the Top Element Pointer in the IOSRL to point to the next lower element. Any elements <u>above</u> the procedure element are also deleted.		
5 STACK deletes all elements from the stack by setting the Top Element Pointer in the IOSRL to point to the first element in the stack and freeing the LSD and in-storage list of any non-terminal element.		UNSTACK

Method of Operation Diagram 8. STACK Service Routine (Part 2 of 2)



Gets line(s) of input from terminal or from current source of input.

1 Determines source of input:
- From terminal

OR
- From in-storage list.

2 Obtains line(s) of input from the terminal. If last character is '-' get another line.

OR

3 Obtains line(s) of input from in-storage list.

A GETLINE Options

Bit settings that indicate the operation to be performed, as follows:

Bit	Meaning when set
0-2	Reserved (0).
3	The line of input is a physical line; if zero, it is a logical line.
4	The source of input is the terminal; if zero, it is as described by the top element the input stack.
5-15	Reserved (0).

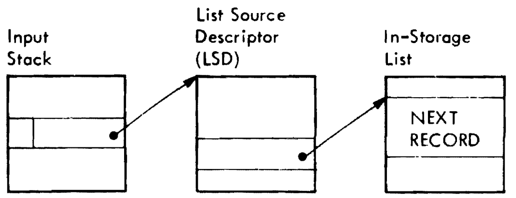
B TGET Options

Bit settings, as follows:

Bit	Meaning when set
0	Always set for TGET.
1-2	Reserved (0).
3	NOWAIT was specified; if zero, WAIT was specified.
4-6	Reserved (0).
7	ASIS was specified; if zero, EDIT was specified.
8-15	Reserved (0).

C

The In-Storage List must have been placed on the Input Stack using the STACK service routine.

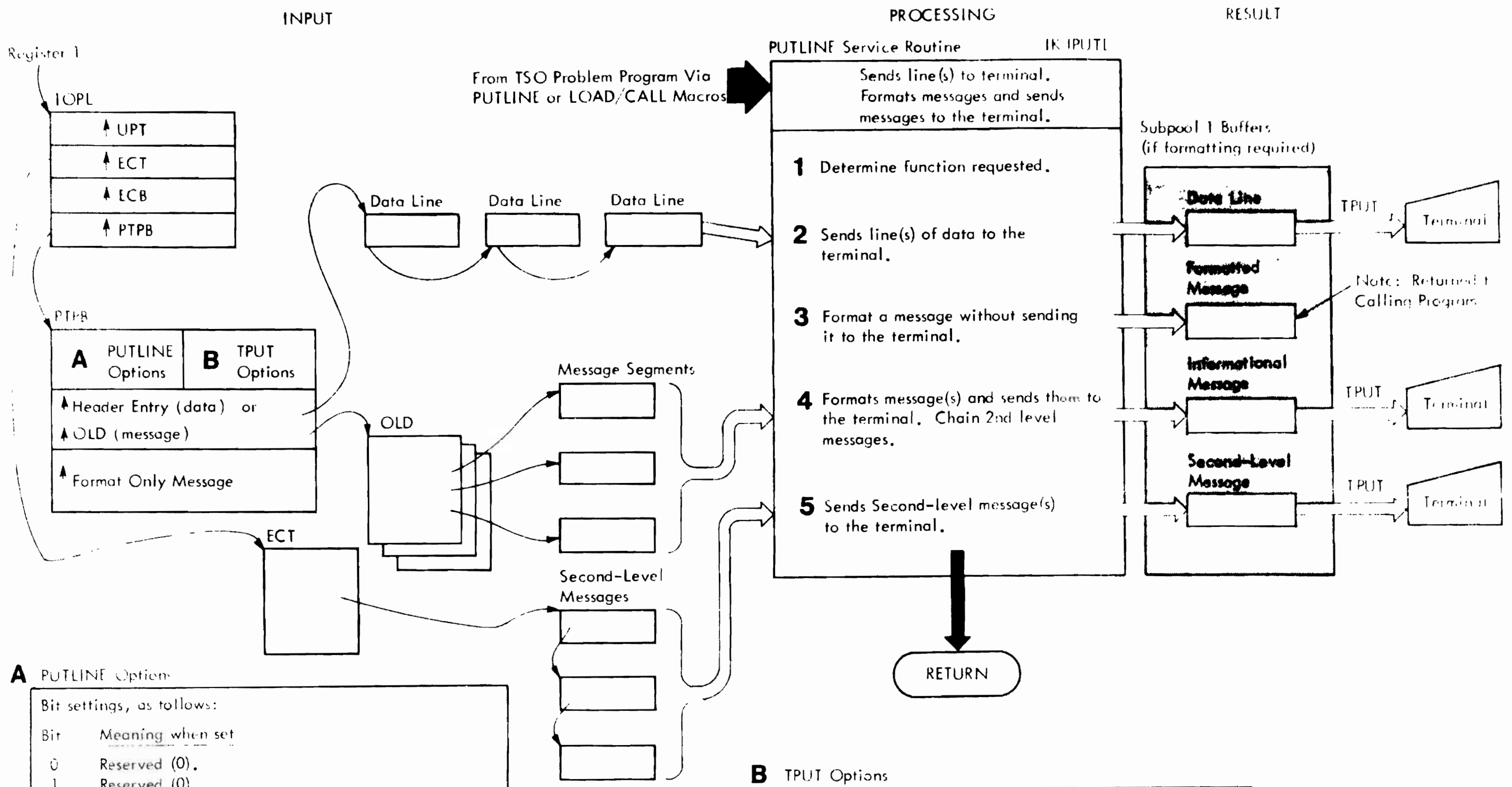


Method of Operation Diagram 9. GETLINE Service Routine (Part 1 of 2)

CROSS REFERENCE TABLE

Key Description	Routine	Label
1 Checks the GTPB to see if the source of input must be the terminal regardless of the current source. If not, checks the Top Element Pointer of the IOSRL to determine the current source of input.	GETLINE	IKJEFT55
2 If the terminal is the current source of input GETLINE gets a line of input from the terminal. If logical line processing is requested, GETLINE recognizes "-" as a continuation character (if it comes at the end of the line) and gets all physical lines contained in the logical line.		
3 If the current source of input is an in-storage list (Top Element Pointer points to List Source Descriptor), GETLINE gets the next record pointed to by a field within the List Source Descriptor (LSD) and updates the pointer to the next record.		
If an end-of-data is reached on a in-storage list, GETLINE links to the STACK service routine to delete an element from the stack. No line is returned to the calling program.	STACK	IKJEFT30

Method of Operation Diagram 9. GETLINE Service Routine (Part 2 of 2)


INPUT
PROCESSING
RESULT
Register 1
IOPL
UPT
ECT
ECB
PTPB
PTPB
A PUTLINE Options
B TPUT Options
Header Entry (data) or
OLD (message)
Format Only Message
ECT
Data Line
Data Line
Data Line
OLD
Message Segments
Second-Level Messages
From TSO Problem Program Via PUTLINE or LOAD/CALL Macros
PUTLINE Service Routine
IKJPUTL
Sends line(s) to terminal. Formats messages and sends messages to the terminal.
1 Determine function requested.
2 Sends line(s) of data to the terminal.
3 Format a message without sending it to the terminal.
4 Formats message(s) and sends them to the terminal. Chain 2nd level messages.
5 Sends Second-level message(s) to the terminal.
RETURN
Subpool 1 Buffers (if formatting required)
Data Line
Formatted Message
Informational Message
Second-Level Message
TPUT
Terminal
Note: Returned to Calling Program
TPUT
Terminal
TPUT
Terminal

A PUTLINE Options

Bit settings, as follows:

Bit	Meaning when set
0	Reserved (0).
1	Reserved (0).
2	Line contains data. If 0, line contains message.
3	Line is single-level or single-line.
4	Multi-line format (data).
5	Multi-level format (messages).
6	Informational message.
7	Reserved (0).
8	Reserved (0).
9	Reserved (0).
10	Format Only. (Do not send message to the terminal).
11-15	Reserved (0).

B TPUT Options

Bit settings that indicate the TPUT options requested, as follows:

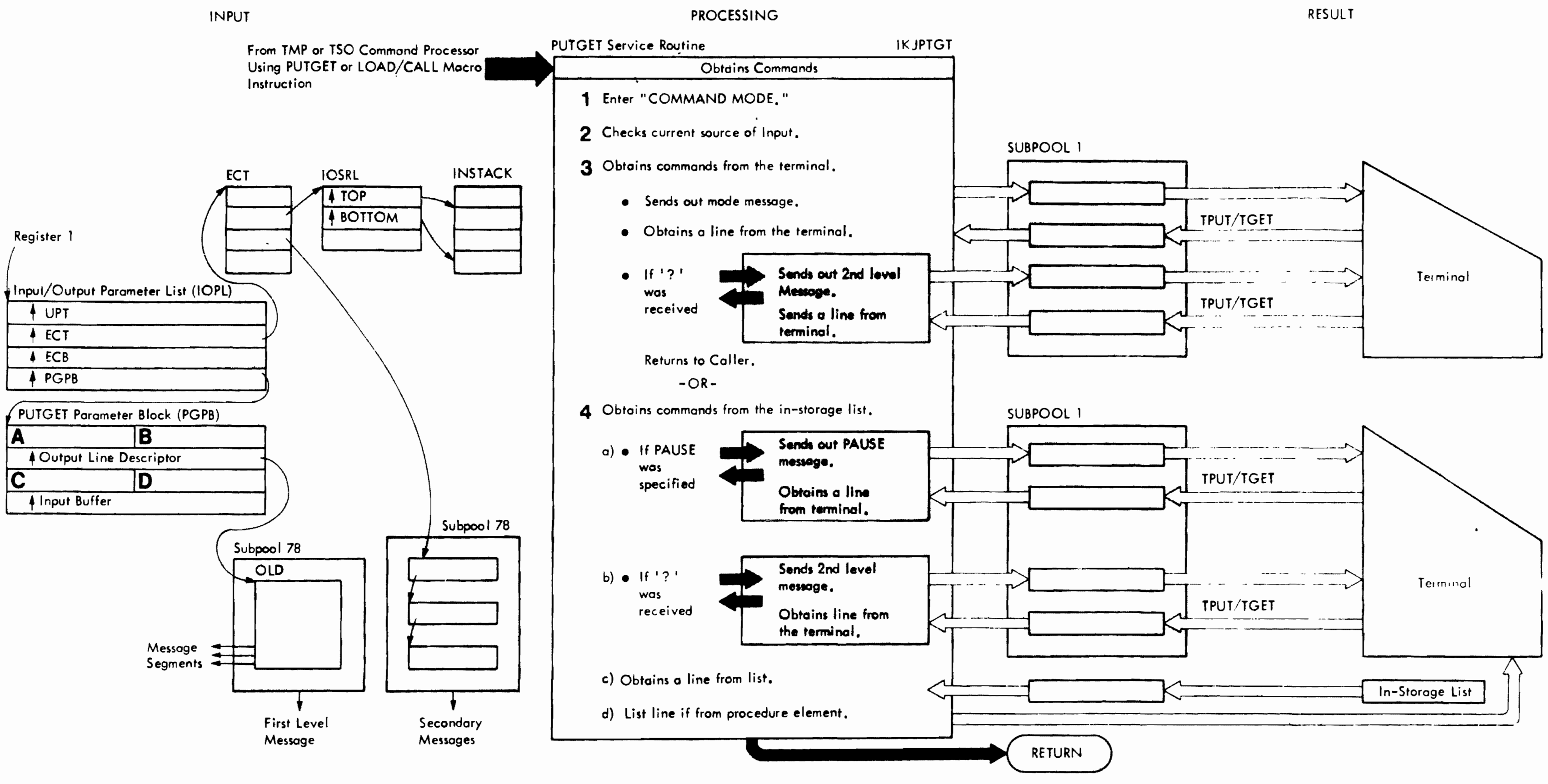
Bit	Meaning when set
0	Always zero for TPUT.
1-2	Reserved (0).
3	NOWAIT processing requested.
4	HOLD processing requested.
5	BREAKIN processing requested.
6	CONTROL processing requested.
7	ASIS processing requested.

Method of Operation Diagram 10. PUTLINE Service Routine (Part 1 of 2)

CROSS REFERENCE TABLE

Key Description	Routine	Label
1 The PUT Options field is checked to determine the function requested.	PUTLINE	IKJEFT40
2 If the line contains data (bit 2 is set), PUTLINE sends data to the terminal	TERMOUT	IKJEFT56
If the data consists of multiple lines (bit 4 is set) then all lines are sent to the terminal.		
3 If the line contains a message (bit 2 is not set) and format only is specified (bit 10 is set), PUTLINE formats the message but does not send it to the terminal, as follows:		
• Formats the message, if necessary.	TEXTIN	IKJEFT54
4 PUTLINE sends an informational message to the terminal if the line contains a message (bit 2 is not set), format only was not specified (bit 10 is not set), and the second word of the PTPB (PARMAOUT) contains an address. PUTLINE:	PUTLINE	IKJEFT40
• Formats the message, if necessary.	TEXTIN	IKJEFT54
• Sends it to the terminal.	TERMOUT	IKJEFT56
If bit 5 is set, PUTLINE constructs a second-level message chain, as follows:	PUTLINE	IKJEFT40
• Formats the message, if necessary.	TEXTIN	IKJEFT54
• Chains 2nd level message to the second-level message chain whose origin is the ECTMSG field of the ECT.	PUTLINE	IKJEFT40
5 PUTLINE sends second-level messages to the terminal if bit 2 is not set, format only was not specified (bit 10 is not set), and the second word of the PTPB contains zero. PUTLINE:	PUTLINE	IKJEFT40
• Sends all second-level messages to the terminal. If the ECTMSG field of the ECT contains the address of a second-level message chain.	CHAINOUT	IKJEFT52
• Sends a "NO INFORMATION AVAILABLE" message if the ECTMSG field of the ECT contains zero or if the ECTMSGF bit is set.	CHAINOUT	IKJEFT52
• Frees the storage obtained for the second-level message.	UNCHAIN	IKJEFT53
• Zeroes the ECTMSG field.	UNCHAIN	IKJEFT53

Method of Operation Diagram 10. PUTLINE Service Routine (Part 2 of 2)


INPUT
PROCESSING
RESULT
From TMP or TSO Command Processor Using PUTGET or LOAD/CALL Macro Instruction
PUTGET Service Routine
IKJPTGT
Obtains Commands
1 Enter "COMMAND MODE."
2 Checks current source of Input.
3 Obtains commands from the terminal.
• Sends out mode message.
• Obtains a line from the terminal.
• If '?' was received
Sends out 2nd level Message.
Sends a line from terminal.
Returns to Caller.
-OR-
4 Obtains commands from the in-storage list.
a) • If PAUSE was specified
Sends out PAUSE message.
Obtains a line from terminal.
b) • If '?' was received
Sends 2nd level message.
Obtains line from the terminal.
c) Obtains a line from list.
d) List line if from procedure element.
RETURN
Register 1
ECT
IOSRL
TOP
BOTTOM
INSTACK
Input/Output Parameter List (IOPL)
UPT
ECT
ECB
PGPB
PUTGET Parameter Block (PGPB)
A
B
Output Line Descriptor
C
D
Input Buffer
Subpool 78
OLD
Subpool 78
Message Segments
First Level Message
Secondary Messages
SUBPOOL 1
TPUT/TGET
TPUT/TGET
Terminal
SUBPOOL 1
TPUT/TGET
TPUT/TGET
Terminal
In-Storage List

A PUT Options

Bit settings that indicate the output operations to be performed by PUTGET, as follows:

Bit	Meaning when set
0	Reserved (0).
1	Reserved (0).
2	Always zero for PUTGET.
3	Single-level format.
4	Always zero for PUTGET.
5	Multi-level format.
6	Always zero for PUTGET.
7	Prompting message.
8	Mode message.
9	Reserved (0).

B TPUT Options

Bit settings that indicate the TPUT options requested, as follows:

Bit	Meaning when set
0	Always zero for TPUT.
1-2	Reserved (0).
3	NOWAIT processing requested.
4	HOLD processing requested.
5	BREAKIN processing requested.
6	CONTROL processing requested.
7	ASIS processing requested.
8-15	Reserved (0).

C GET Options

Bit settings that indicate the input operations to be performed by PUTGET, as follows:

Bit	Meaning when set
0	Reserved (0).
1	Not used in PUTGET.
2	Not used in PUTGET.
3	Demand from terminal.
4-15	Reserved (0).

D TGET Options

Bit settings that indicate the TGET options requested as follows:

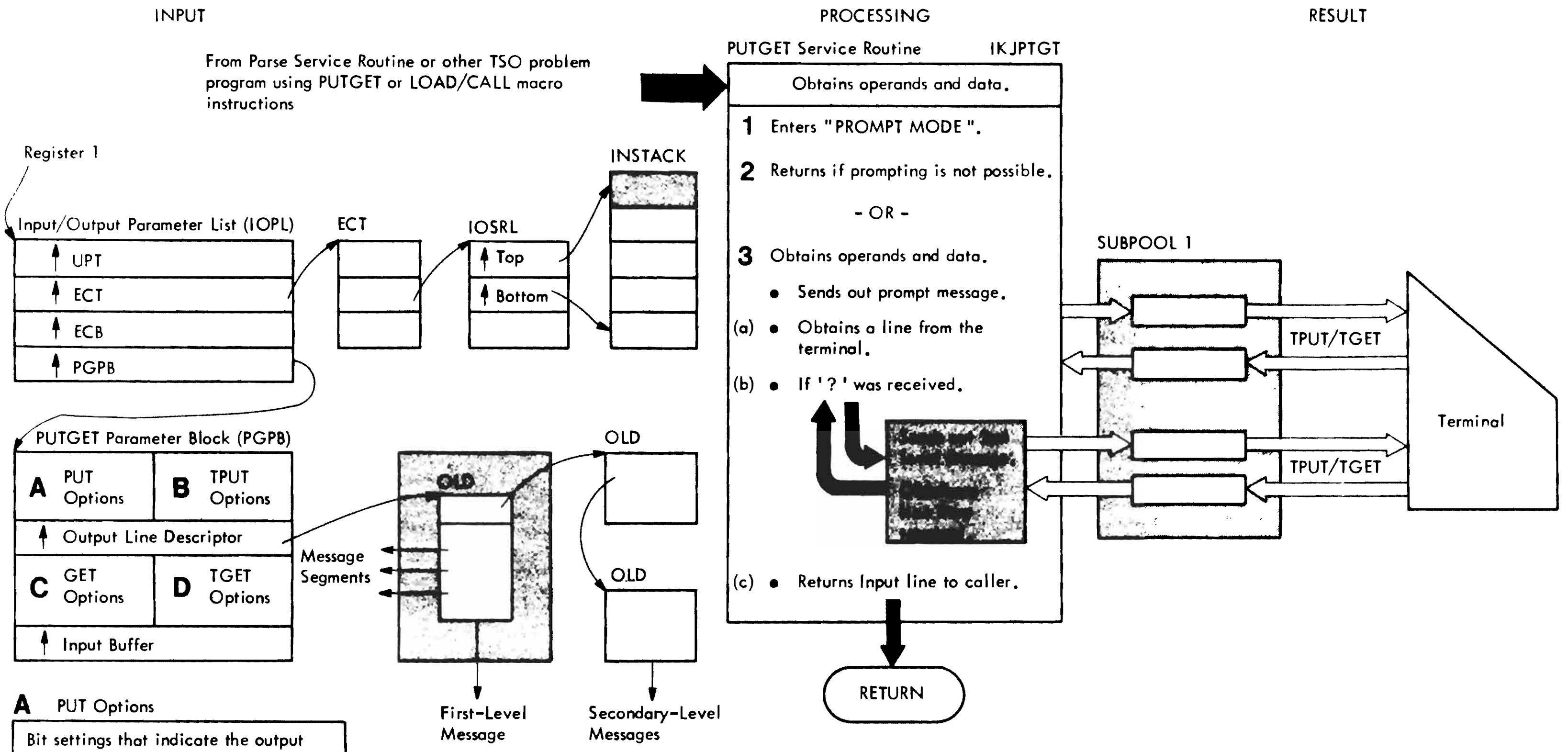
Bit	Meaning when set
0	Always set for TGET.
1-2	Reserved (0).
3	NOWAIT processing requested.
4-6	Reserved (0).
7	ASIS processing requested.
8-15	Reserved (0).

Method of Operation Diagram 11. PUTGET Service Routine (Command Mode) (Part 1 of 2)

CROSS REFERENCE TABLE

Key Description	Routine	Label
1 Command Mode is requested when bit 8 of the PUT Options field is set. (If bit 8 is zero, refer to Diagram 12 for Prompt Mode.)	PUTGET	IKJEFT45
2 Checks the IOSRL to determine source of input.		
3 If the terminal is the current source of input, PUTGET obtains a command from the terminal, as follows:		
• Formats the output line in a buffer, if necessary.	TEXTIN	IKJEFT54
• Sends the line to the terminal.	TERMOUT	IKJEFT56
• Gets a line from the terminal.	GETLINE	IKJEFT55
a) If the line begins with a question mark ("?") :		
• Sends second-level messages to the terminal if the ECTSMSG field of the ECT contains the address of a second-level message chain.	CHAINOUT	IKJEFT52
• Sends "NO INFORMATION AVAILABLE" message if the ECTSMSG field of the ECT contains zero.		
• Frees storage obtained for the second-level messages and zeroes the ECTSMSG field in the ECT.	UNCHAIN	IKJEFT53
• Sends a line to the terminal.	TERMOUT	IKJEFT56
b) If the line does not begin with a question mark ("?"), return to caller. (If the line begins with another question mark, repeat step 3a).		
4 If an in-storage list is the current source of input, PUTGET obtains commands from the list, as follows:	PUTGET	IKJEFT45
a) If the UPTPAUS bit is set in the UPT:		
• Sends PAUSE message to the terminal.	TERMOUT	IKJEFT56
• Gets a line from the terminal.	GETLINE	IKJEFT55
b) If the line begins with a question mark ("?"), sends second-level messages to the terminal as in step 3a. If line is a carriage return, deletes chain. Prompts user for proper message and repeats step 4a.		
c) • Gets a line from the in-storage list.	GETLINE	IKJEFT55
• If EOD is returned from the in-storage list, repeats step 2.	STACK	IKJEFT30
d) • Lists the line if from an in-storage procedure with LIST specified.		

Method of Operation Diagram 11. PUTGET Service Routine (Command Mode) (Part 2 of 2)


INPUT
PROCESSING
RESULT
From Parse Service Routine or other TSO problem program using PUTGET or LOAD/CALL macro instructions
Register 1
Input/Output Parameter List (IOPL)
UPT
ECT
ECB
PGPB
ECT
IOSRL
Top
Bottom
INSTACK
PUTGET Parameter Block (PGPB)
A PUT Options
B TPUT Options
Output Line Descriptor
C GET Options
D TGET Options
Input Buffer
Message Segments
OLD
OLD
OLD
First-Level Message
Secondary-Level Messages
PUTGET Service Routine IKJPTGT
Obtains operands and data.
1 Enters "PROMPT MODE".
2 Returns if prompting is not possible.
- OR -
3 Obtains operands and data.
• Sends out prompt message.
(a) • Obtains a line from the terminal.
(b) • If '?' was received.
(c) • Returns Input line to caller.
RETURN
SUBPOOL 1
TPUT/TGET
TPUT/TGET
Terminal

A PUT Options

Bit settings that indicate the output operations to be performed by PUTGET, as follows:

Bit	Meaning when set
0	Reserved (0).
1	Reserved (0).
2	Always zero for PUTGET.
3	Single-level format.
4	Always zero for PUTGET.
5	Multi-level format.
6	Always zero for PUTGET.
7	Prompting message.
8	Mode message.
9	Reserved (0).

B TPUT Options

Bit settings that indicate the TPUT options requested, as follows:

Bit	Meaning when set
0	Always zero for TPUT.
1-2	Reserved (0).
3	NOWAIT processing requested.
4	HOLD processing requested.
5	BREAKIN processing requested.
6	CONTROL processing requested.
7	ASIS processing requested.
8-15	Reserved (0).

C GET Options

Bit settings that indicate the input operations to be performed by PUTGET, as follows:

Bit	Meaning when set
0	Reserved (0).
1	Not used in PUTGET.
5	Not used in PUTGET.
3	Demand from terminal.
4-15	Reserved (0).

D TGET Options

Bit settings that indicate the TGET options requested, as follows:

Bit	Meaning When Set
0	Always set for TGET.
1-2	Reserved (0).
3	NOWAIT processing requested.
4-6	Reserved (0).
7	ASIS processing requested.
8-15	Reserved (0).

Method of Operation Diagram 12. PUTGET Service Routine (Prompting Mode) (Part 1 of 2)

CROSS REFERENCE TABLE

Key Description	Routine	Label
1 Prompt Mode is requested when bit 7 of the PUT Options field is set. (If bit 7 is zero, refer to Diagram 11 for Command Mode.)	PUTGET	IKJEFT45
2 If a command procedure is the current source of input, or if the UPTNPRM bit is set in the UPT, then PUTGET is unable to prompt and returns an error return code.	PUTGET	IKJEFT45
3 If the terminal (or an in-storage list containing source data) is the current source of input, PUTGET obtains operands or data from the terminal, as follows:		
• Formats the output line in a buffer, if necessary.	TEXTIN	IKJEFT54
• Sends the line to the terminal.	TERMOUT	IKJEFT56
a) • Gets a line from the terminal.	GETLINE	IKJEFT55
b) If the line begins with a questionmark ("?"):		
• Formats and next-level message to the terminal.	CHAINOUT	IKJEFT52
• Sends "NO INFORMATION AVAILABLE" message if no next-level message was provided.		
• Repeats step 3a.	GETLINE	IKJEFT55
c) If the line does not begin with a questionmark ("?"), return to caller. (If the line begins with a questionmark, repeat step 3b.)		

Method of Operation Diagram 12. PUTGET Service Routine (Prompting Mode) (Part 2 of 2)

Section 3: Program Organization

This section describes the organization of the terminal I/O service routines: STACK, GETLINE, PUTLINE, and PUTGET. It contains information about the hierarchy of the load module, the assembly modules, and the control sections that constitute the program. Figure 11 is a graphic representation of this hierarchy.

The module operation information briefly describes the processing operations that occur within each terminal I/O service routine module.

For a summary of the functions performed by subroutines, refer to the directory in Section 4.

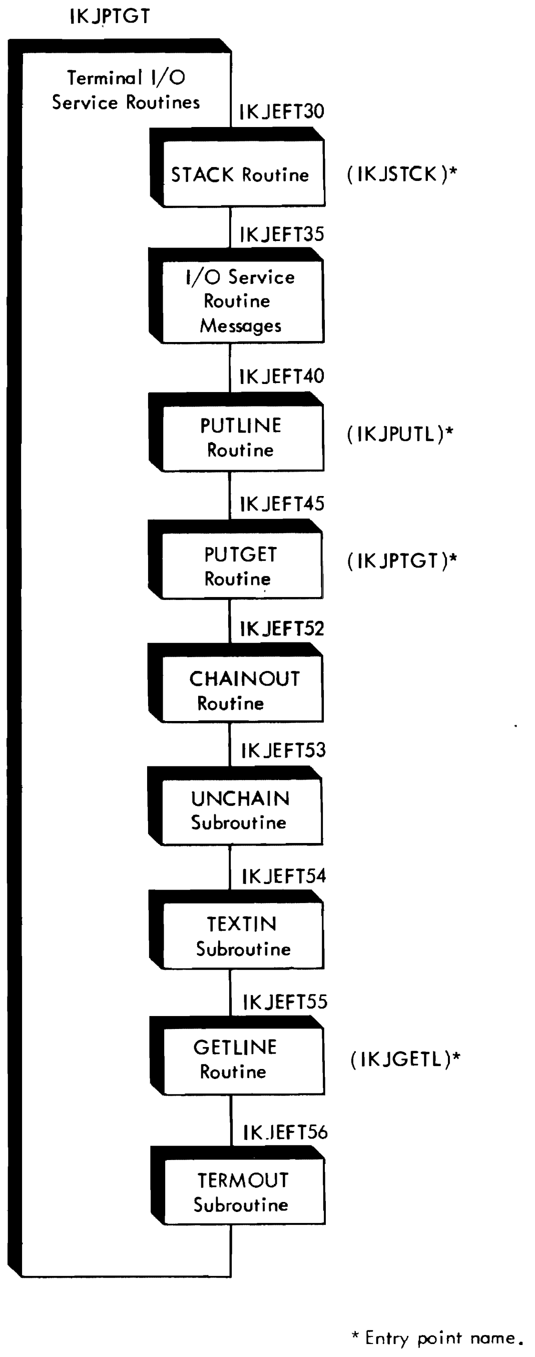
Program Hierarchy

The terminal I/O service routines are all parts of a single load module, IKJPTGT, as shown in Figure 11. Load module IKJPTGT has the following control sections:

IKJEFT30	--	STACK service routine (IKJSTCK)
IKJEFT35	--	I/O service routine messages
IKJEFT40	--	PUTLINE service routine (IKJPUTL)
IKJEFT45	--	PUTGET service routine (IKJPTGT)
IKJEFT52	--	CHAINOUT subroutine for PUTLINE and PUTGET
IKJEFT53	--	UNCHAIN subroutine for PUTLINE and PUTGET
IKJEFT54	--	TEXTIN subroutine for PUTLINE and PUTGET
IKJEFT55	--	GETLINE service routine (IKJGETL)
IKJEFT56	--	TERMOUT subroutine for PUTLINE and PUTGET

Note that load module IKJPTGT has four entry points:

IKJSTCK	--	for STACK
IKJGETL	--	for GETLINE
IKJPUTL	--	for PUTLINE
IKJPTGT	--	for PUTGET



Load Module Names	Normal Residence	Approximate Sizes
IKJPTGT	SYS1.LINKLIB	6.5K bytes

Figure 11. Program Hierarchy: Terminal I/O Service Routines

Module Operation

The following descriptions briefly describe the processing operations in each executable module of the terminal I/O service routines.

IKJEFT30 -- STACK SERVICE ROUTINE

Creates and updates the input stack (INSTACK) which defines the current source of input: either from the terminal or from an in-storage list.

- Creates the input stack. Initializes the first element (bottom element) to describe the terminal as a source of input.
- Adds an element to the top of the input stack to define a new source of input: either from the terminal or from an in-storage list.
- Deletes one or more elements from the top of the input stack, as follows:
  - top element only.
  - all elements down to and including the first element that describes an in-storage list containing a procedure.
  - all elements down to the first element (bottom element). The bottom element is never removed.

IKJEFT40 -- PUTLINE SERVICE ROUTINE

Processes line(s) of output and sends then to the terminal. Performs one of the following functions:

- Formats a message only. Does not send it to the terminal.
- Formats a single-level informational message and sends it to the terminal.
- Formats a first-level informational message, sends it to the terminal, or sends a first level informational message to the terminal and chains a second-level message to the ECTSMSG field of the ECT.
- Sends line(s) of data to the terminal.

IKJEFT45 -- PUTGET SERVICE ROUTINE

Entry module for the PUTGET service routine. Determines actions to be taken in PUTGET processing based on the input parameters, the user options specified in the UPT, the type of input source specified in the input stack, the type of response made by the terminal user, and the actions of subroutines.

PUTGET has two processing modes: command mode and prompting mode.

- In command mode, a mode message is sent to the terminal and a line of input is obtained from the terminal or from an in-storage list.
- In prompting mode, a prompting message is sent to the terminal (unless the user has specified NOPROMPT or the current source of input is an in-storage procedure) and a line of input is obtained from the terminal.

IKJEFT52 -- CHAINOUT SUBROUTINES OF PUTLINE AND PUTGET

Sends chained second-level messages, if available, or notifies the terminal that no second-level messages are available.

IKJEFT53 -- UNCHAIN SUBROUTINE OF PUTLINE AND PUTGET

Frees the storage allocated to a second-level message chain. Zeroes the ECTSMSG field in the ECT to indicate that no second-level messages are chained.

IKJEFT54 -- TEXTIN SUBROUTINE OF PUTLINE AND PUTGET

Obtains a buffer for an output message and moves the segment(s) of that message into the buffer to produce one contiguous character string.

TEXTIN processes four types of messages:

1. Single segment messages. These are validity checked and returned to the caller with no further processing.
2. Multiple segment messages. These are validity checked and segments are inserted to produce one contiguous character string.
3. Format only requests. These are validity checked and placed in a subpool 1 buffer.
4. Chained messages. Single segment or multiple segment messages that are to be forwarded-chained. These are validity checked and the segments are inserted, if necessary. A one-word forward-chaining element is built and the message is placed in a subpool 78 buffer.

2

IKJEFT55 -- GETLINE SERVICE ROUTINE

Obtains a line of input from the terminal or from the current source of input as determined by the input stack.

IKJEFT56 -- TERMOUT SUBROUTINE OF PUTLINE AND PUTGET

Sends line(s) of output to the terminal. The line(s) can contain messages or data.

- If the line(s) contain messages, TERMOUT optionally strips off message identifiers before sending line(s) to the terminal.
- If the line(s) contain data, TERMOUT sends them to terminal without further processing. If the data is multi-line, TERMOUT sends all the lines to the terminal.

Section 4: Directory

This chart contains information to help you find the appropriate program description, flowchart, or assembly listing. It correlates information from three sources:

- The source code.
- The executable load modules.
- This manual.

Label	Command Name	Load Module Name	Assembly Module Name	Control Section Name	Description	Diagram
IKJEFT30	STACK Service Routine	IKJPTGT	IKJSTCK	IKJEFT30	Creates and updates the IOSRL and input stack which determine the current source of input.	8
IKJEFT35	Terminal I/O Service Routine Message Module	IKJPTGT	IKJEFT35	IKJEFT35	Contains message segments used by terminal I/O service routines.	
IKJEFT40	PUTLINE Service Routine	IKJPTGT	IKJPUTL	IKJEFT40	Sends line(s) of data to the terminal; formats messages and sends them to the terminal.	10
IKJEFT45	PUTGET Service Routine	IKJPTGT	IKJPTGT	IKJEFT45	Sends messages to the terminal and obtain a line of input from the current source of input	11-12
IKJEFT52	CHAINOUT Subroutine	IKJPTGT	IKJEFT52	IKJEFT52	Sends chained second-level messages if available, or notifies terminal that no messages are available.	10-11
IKJEFT53	UNCHAIN Subroutine	IKJPTGT	IKJEFT53	IKJEFT53	Frees storage allocated to second-level message chain.	10-11
IKJEFT54	TEXTIN Subroutine	IKJPTGT	IKJEFT54	IKJEFT54	Obtains message buffer and inserts message segments.	10-12
IKJEFT55	GETLINE Service Routine	IKJPTGT	IKJGETL	IKJEFT55	Obtains line(s) of data from the terminal or from current source of input.	9,11-12
IKJEFT56	TERMOUT Subroutine	IKJPTGT	IKJEFT56	IKJEFT56	Sends line(s) to the terminal	10-12

Note: IKJPTGT is a reenterable and refreshable load module. Normal residence is SYS1.LINKLIB.

Section 5: Data Areas

This section describes the major data areas used by the terminal I/O service routines, including:

Environment control table (ECT)
GETLINE parameter block (GTPB)
Input stack (INSTACK)
I/O parameter list (IOPL)
I/O service routines parameter list (IOSRL)
List source descriptor (LSD)
Output line descriptor (OLD)
PUTGET parameter block (PGPB)
PUTLINE parameter block (PTPB)
STACK parameter block (STPB)
Text insertion parameter list (TXINPARM)
User profile table (UPT)

The following information appears for each data area:

- Size, in bytes.
- Name(s) of the routine(s) that creates it.
- Name(s) of the routine(s) that update and/or reference it.
- Field names, displacements, size, and contents.
- Cross-reference to method of operation diagrams and program flowcharts.

ENVIRONMENT CONTROL TABLE (ECT)

Size:	40 bytes.
Located in:	Subpool 1.
Created by:	Terminal monitor program (IKJEFT01).
Updated by:	IKJEFT45, IKJEFT52, IKJEFT53, IKJEFT55, IKJEFT56.
Referenced by:	IKJEFT30, IKJEFT45, IKJEFT52, IKJEFT53, IKJEFT55, IKJEFT56.
Contents:	Information about the user's environment in the foreground region.

Operation Diagrams
8-12

Displacement Dec.	Hex.	Field Name	Size Bytes	Contents
0	0	ECTRTCDF	1	ABEND Flags. Bit settings, as follows: Bit Meaning when on 0 ABEND in the command processor. 1-7 Reserved (0)
1	1	ECTRTCD	3	Return code from ABEND.
4	4	ECTIOWA	4	† I/O service routine list (IOSRL)
8	8	ECTMSGF	1	Message flags. Bit settings, as follows: Bit Meaning when on 0 Delete second-level messages. 1-7 Reserved (0)
9	9	ECTMSG	3	† Second-level message chain, or zero if no messages are chained.
12	C	ECTPCMD	8	Command name.
20	14	ECTSCMD	8	Subcommand name.
28	1C	ECTSWS	1	ECT switches. Bit settings, as follows:
		ECTNOPD		Bit Meaning when on 0 No parameters exist in command buffer. 1 Reserved (0).

(Continued)

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents	
		ECTATRM		2	Command processor being terminated by the TMP using a DETACH macro instruction with a STAE operand.
		ECTLOGF		3	LOGON or LOGOFF command processor has requested re-logon or logoff.
		ECTNMAL		4	No user messages at logon.
		ECTNNOT		5	No system messages at logon.
				6-7	Reserved (0).
29	1D	ECTDDNUM	3		Counter used by dynamic allocation SVC routines when assigning temporary ddnames.
32	20	ECTUSER	4		Reserved for installation.
36	24		4		Reserved (0).

2

GETLINE PARAMETER BLOCK (GTPB)

Size:	8 bytes.
Located in Subpool	Any.
Created by:	Caller of GETLINE service routine.
Updated by:	IKJEFT55.
Referenced by:	IKJEFT55.
Contents:	Control information to GETLINE.

Operation Diagrams
9

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	PARMCNTL	2	Control flags. Bit settings that indicate the operation to be performed by GETLINE, as follows: <u>Bit</u> <u>Meaning when on</u> 0-2 Reserved (0).
		PARMPHYS		3 The line of input is a physical line; if zero, it is a logical line.
		PARMTERM		4 The source of input is the terminal; if zero, it is as described by the top element on the input stack.
				5-15 Reserved (0)
2	2	PARMTGT	2	TGET options. Bit settings that indicate the operation to be performed by TGET, as follows: <u>Bit</u> <u>Meaning when on</u> 0 Always set for TGET. 1-2 Reserved (0). 3 NOWAIT was specified; if zero, WAIT was specified. 4-6 Reserved (0). 7 ASIS was specified; if zero, EDIT was specified. 8-15 Reserved (0). For further information about the TGFT macro instruction see <u>OS/VS2 TSO Guide to Writing a Terminal Monitor Program or a Command Processor</u>, GC28-0648.

(Continued)

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
4	4	PARMADIN	4	The address of the input buffer that contains the line. The format of the buffer is as follows: <u>Byte</u> <u>Contents</u> 0-1 The length, in bytes, of the input buffer. 2-3 The offset to the first character in the input line. 4-47 The line of input.

2

INPUT STACK (INSTACK)

Size:	Variable. A multiple of 32 bytes.
Located in:	Subpool 78.
Created by:	IKJEFT30.
Updated by:	IKJEFT30.
Referenced by:	IKJEFT45, IKJEFT55.
Contents:	A last in/first out (LIFO) queue of elements that describe sources of input.

Operation Diagrams
8

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	INSCODE	1	Bit settings that describe the type of element, as follows:
				Bit — Meaning when on
		INSTERM		0 Terminal element.
		INSSTOR		1 Storage element.
				2-5 Reserved (0).
		INSPROC		6 Procedure element.
		INSLIST		7 List each line of the in-storage list at the terminal when PUTGET obtains it.
1	1	INSADLSD	3	The address of the list source descriptor (LSD) or zero for a terminal element.
4	4	INSCODE	1	Same as above. 2nd stack element
5	5	INSADLSD	3	Same as above.
. . . .	. . . .	. . . .		. . . .
28	1C	INSCODE	1	Same as above. 8th stack element
29	1D	INSADLSD	3	Same as above.

Note: The initial size of the input stack is 8 words (32 bytes). If all eight available positions become active stack elements, then the stack is enlarged by 8 more words (32 more bytes).

INPUT/OUTPUT PARAMETER LIST (IOPL)

Size:	16 bytes.
Located in Subpool	Any.
Created by:	Caller of STACK, GETLINE, PUTLINE or PUTGET service routine.
Updated by:	None.
Referenced by:	IKJEFT30, IKJEFT40, IKJEFT45, IKJEFT55.

Contents:

Operation Diagrams
8-12

2

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	IOPLUPT	4	The address of the user profile table (UPT).
4	4	IOPLECT	4	The address of the environment control table (ECT).
8	8	IOPLECB	4	The address of the command processor's event control block (ECB).
12	C	IOPLIOPB	4	The address of the parameter block for one of the terminal I/O service routines, as follows: Service Routine — Parameter Block STACK — STPB GETLINE — GTPB PUTLINE — PTPB PUTGET — PGPB

I/O SERVICE ROUTINE LIST (IOSRL)

Size:	20 bytes.
Located in:	Subpool 78.
Created by:	IKJEFT30.
Updated by:	IKJEFT30.
Referenced by:	IKJEFT45, IKJEFT55.
Contents:	Information about the current status of the input stack.

Operation Diagrams
8-9,11-12

Displacement Dec.	Displacement Hex.	Field Name	Size in Bytes	Contents
0	0	IOSTELM	4	The address of the top element on the input stack which describes the current source of input.
4	4	IOSBELM	4	The address of the bottom element on the input stack which describes the terminal as the source of input.
8	8	IOSTLEN	2	The total length, in bytes, allocated for the input stack, as obtained by a GETMAIN.
12	C	IOSNELM	2	The number of elements that were on the input stack when the last GETMAIN was issued.
16	10	--	4	Reserved (0).

LIST SOURCE DESCRIPTOR (LSD)

Size:	16 bytes.
Located in:	Subpool 78.
Created by:	Caller of STACK service routine.
Updated by:	IKJEFT55 (as GETLINE or as the GETINPUT subroutine of PUTGET.
Referenced by:	IKJEFT30, IKJEFT55.
Contents:	Information about an in-storage list.

Operation Diagrams
8-9,11-12

2

Displacement Dec	Hex.	Field Name	Size in Bytes	Contents
0	0	LSDADATA	4	The address of an in-storage list. Set by the caller of STACK. Referred to by STACK, GETLINE and PUTGET.
4	4	LSDRCLEN	2	The length, in bytes, of each record in the in-storage list, or zero if the records are in variable-length format.
6	6	LSDTOTLN	2	The total length, in bytes, of the in-storage list. Set by the caller of STACK.
8	8	LSDANEXT	4	The address of the next record to be read. Set by the caller of STACK. Updated by GETLINE and PUTGET.
12	C	LSDRSVRD	4	Reserved (0).

OUTPUT LINE DESCRIPTOR (OLD)

Size:	Variable.
Located in Subpool	Any.
Created by:	Caller of PUTLINE or PUTGET service routine.
Updated by:	IKJEFT40.
Referenced by:	IKJEFT40, IKJEFT45, IKJEFT54, IKJEFT56.
Contents:	The addresses of message segments.

Operation Diagrams
10-12

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0		4	The address of the next OLD or zero for the last OLD in the chain. (This field is present only if the message pointed to is a multi-level message.)
4	4		4	The number of message segments in this OLD.
8	8		4	The address of the first message segment.
12	C		4	The address of the second message segment.
. . .	. . .			. . .
			4	The address of the last message segment.

PUTGET PARAMETER BLOCK (PGPB)

Size:	16 bytes.
Located in Subpool	Any.
Created by:	Caller of PUTGET service routine.
Updated by:	IKJEFT55.
Referenced by:	IKJEFT45, IKJEFT55, IKJEFT56.
Contents:	Control information for PUTGET.

Operation Diagrams
11-12

2

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	PTGPCNTL	2	PUT options. Bit settings that indicate the output operations to be performed by PUTGET, as follows:
				Bit — Meaning when on
		PTGPBTO		0 Reserved (0).
		PTGPPUT		1 Reserved (0).
		PTGPDTMS		2 Always zero for PUTGET.
		PTGPSNGL		3 Single-level format.
		PTGPMLIN		4 Always zero for PUTGET.
		PTGPMLEV		5 Multi-level format.
		PTGPIFOR		6 Always zero for PUTGET.
		PTGPPRMT		7 Prompting message.
		PTGPMODE		8 Mode message.
		PTGPDMND		9 Reserved (0).
				Bit — Meaning when on
		PTGFORM		10 Reserved (0).
		PTGPBYPS		11 Bypass processing. The terminal will send a message without printing it. (Used for secret communications such as passwords.)
		PTGPUNUS		12-15 Reserved (0)

(Continued)

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
2	2	PTGPTPUT	2	TPUT options field. Bit settings that indicate the TPUT options requested, as follows: <u>Bit</u> <u>Meaning when on</u> 0 Always zero for TPUT. 1-2 Reserved (0). 3 NOWAIT processing requested. 4 HOLD processing requested. 5 BREAKIN processing requested. 6 CONTROL processing requested. 7 ASIS processing requested. 8-15 Reserved (0). For further information about TPUT macro instruction, refer to the <u>OS/VS2 TSO Guide to Writing a Terminal Monitor Program or a Command Processor</u>, GC28-0648.
4	4	PTGPAOUT	4	The address of the output line descriptor (OLD) for the message.
8	8	PTGGCNTL	2	GET options. Bit settings that indicate the input operations to be performed by PUTGET, as follows: <u>Bit</u> <u>Meaning when on</u>
		PTGGBIT0		0 Reserved (0).
		PTGGGET		1 Not used in PUTGET.
		PTGGPHYS		5 Not used in PUTGET.
		PTGGTERM		3 Demand from terminal.
		PTGGBRSV		4-15 Reserved (0).
10	A	PTGGTGET	2	TGET options. Bit settings that indicate the TGET options requested, as follows: <u>Bit</u> <u>Meaning when on</u> 0 Always set for TGET. 1-2 Reserved (0). 3 NOWAIT processing requested. 4-6 Reserved (0). 7 ASIS processing requested. 8-15 Reserved (0). For further information about TGET macro instruction, refer to the <u>OS/VS2 TSO Guide to Writing a Terminal Monitor Program or a Command Processor</u>, GC28-0648.
12	C	PTGGADIN	4	The address of the input buffer receiving the line.

PUTLINE PARAMETER BLOCK (PTPB)

Size:	12 bytes.
Located in Subpool	Any.
Created by:	Caller of PUTLINE service routine.
Updated by:	IKJEFT56.
Referenced by:	IKJEFT40, IKJEFT54, IKJEFT56.
Contents:	Control information for PUTLINE.

Operation Diagrams
10

Displacement Dec.	Displacement Hex.	Field Name	Size in Bytes	Contents
0	0	PARMCNTL	2	Control Flags. Bit settings, as follows:
				Bit — **Meaning when on**
		PARMPBT0		0 Reserved (0)
		PARMPUT		1 Reserved (0)
		PARMDTMS		2 Line contains data. If 0, line contains message.
		PARMSNGL		3 Line is single-level or single-line.
		PARMMLIN		4 Multi-line format (data).
		PARMMLEV		5 Multi-level format (messages).
		PARMIFOR		6 Informational message.
		PARMPRMT		7 Reserved (0).
		PARMMODE		8 Reserved (0).
		PARMDMND		9 Reserved (0).
		PARMFORM		10 Format Only. (Do not send message to the terminal).
		PARMUNUS		11-15 Reserved (0).

(Continued)

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
2	2	PARMTPUT	2	TPUT options field. Bit settings that indicate the TPUT options requested, as follows: <u>Bit</u> <u>Meaning when on</u> 0 Always zero for TPUT. 1-2 Reserved (0). 3 NOWAIT processing requested. 4 HOLD processing requested. 5 BREAKIN processing requested. 6 CONTROL processing requested. 7 ASIS processing requested. For further information about TPUT macro instruction, refer to the <u>OS/VS2 TSO Guide to Writing a Terminal Monitor Program or a Command Processor</u>, GC28-0648. 8-15 Reserved (0).
4	4	PARMAOUT	4	The address of output line descriptor (OLD) for a message or the address of the fullword header for data.
8	8	PARMAFRM	4	The address of the formatted line if "format only" was specified. Otherwise, the field is not changed.

STACK PARAMETER BLOCK (STPB)

Size:	8 bytes.
Located in Subpool	Any.
Created by:	Caller of STACK service routine.
Updated by:	None.
Referenced by:	IKJEFT30.
Contents:	Control information for STACK.

Operation Diagrams
8

2

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	STBOPCOD	1	Operation code. Bit settings that indicate the operation to be performed, as follows:
				Bit — Meaning when on
		SPADD		0 — Add one element to the top of the stack.
		SPBDLCUR		1 — Delete one element from the top of the stack.
		SPDLPRC		2 — Delete the current procedure element from the stack. If the top element is not a procedure element, delete all elements down to and including the first procedure element.
		SPBDLALL		3 — Delete all elements down to the bottom element. (The bottom element is never removed.)
		SPBRSVRD		4-7 — Reserved (0).
1	1	STBCODE	1	Element code. Bit settings that indicate the type of element to be added to the stack, as follows:
				Bit — Meaning when on
		SPBTERM		0 — Terminal element.
		SPBSTOR		1 — In-storage element.
		SPBRSVD		2-5 — Reserved (0).
		SPBPROC		6 — Procedure element; if zero, source element.
		SPBLIST		7 — List each element at terminal as read. Used only by PUTGET.
2	2	SPBRESVD	2	Reserved (0).
4	4	STBADLSD	4	The address of the list source descriptor (LSD). If the input source is the terminal, or if the delete function was specified, the field will contain zeroes.

TEXT INSERTION PARAMETER LIST (TXINPARM)

Size:	12 bytes.
Located in Subpool	Any.
Created by:	IKJEFT40, IKJEFT45.
Updated by:	None.
Referenced by:	IKJEFT54.
Contents:	Information about output messages.

Operation Diagrams
10-12

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	TXISEGMT	4	The address of the output line descriptor (OLD).
4	4	TXIFRMSG	4	The address of a one-word field into which the TEXTIN subroutine places the address of a formatted message. The formatted message is constructed from variable-length segments in the OLD.
8	8	TXIFONLY	4	The address of a one-word field used as a switch, as follows: If set on entry to IKJEFT54, it indicates "Format Only" is requested. If set on exit from IKJEFT54, it indicates a PUTGET or PUTLINE buffer must be freed.

USER PROFILE TABLE (UPT)

Size:	16 bytes.
Located in:	Subpool 0.
Created by:	LOGON/LOGOFF scheduler.
Updated by:	PROFILE command processor.
Referenced by:	IKJEFT45, IKJEFT56.
Contents:	Information about terminal user.

Operation Diagrams
8-12

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	--------	2	Reserved (0).
2	2	UPTUSER	10	Reserved for installation use.
12	C	UPTSWS	1	User environment switches. Bit settings, as follows:
				<u>Bit</u> <u>Meaning when on</u>
				0 Reserved (0).
		UPTNPRM		1 Prompt; if zero, no prompt.
		UPTMID		2 Message identifiers; if zero, message identifiers will be removed.
		UPTNCOM		3 No user communication using the SEND command; if zero, user communication is permissible.
		UPTPAUS		4 PAUSE was specified; if zero, NOPAUSE was specified.
		UPTALD		5 Attention is a line delete character; if zero, attention is not a line delete character.
				6-7 Reserved (0).
13	D	UPTCDEL	1	Character delete character.
14	E	UPTLDEL	1	Line delete character.
15	F	--------	1	Reserved (0).

Section 6: Diagnostic Aids

This section contains the following information:

- Messages (Figure 12) -- a list of messages issued by terminal I/O service routines.
- Register Usage (Figure 13) -- a summary of the use of general registers 0-15.
- Return Codes (Figure 14) -- a summary of return codes and their meanings. Unless otherwise specified, return codes are contained in register 15.

Message ID	Message	Issued by
IKJ567601	NO INFORMATION AVAILABLE	IKJEFT56
IKJ56761A	INVALID RESPONSE. ENTER ? OR HIT CARRIER RETURN-	IKJEFT56
IKJ56762A	PAUSE	IKJEFT56

Figure 12. Messages: Terminal I/O Service Routines

	IKJEFT30		IKJEFT40	
Register	Name	Contents	Name	Contents
0	R0	Work Register	R0	Work Register
1	R1	↑ IOPL/Work Register	R1	↑ IOPL
2	R2	↑ ECT	--	Work Register
3	R3	↑ STPB	--	Work Register
4	R4	Work Register	R4	Work Register
5	R5	↑ IOSRL	R5	Work Register
6	R6	Work Register	--	Work Register
7	--	Work Register	--	Work Register
8	--	Work Register	--	Work Register
9	--	Work Register	--	Work Register
10	--	Work Register	--	Work Register
11	--	Base Register	--	Base Register
12	--	Work Register	--	Work Register
13	R13	↑ Register Save Area	R13	↑ Register Save Area
14	--	Work Register	--	Work Register
15	R15	Return Code	R15	Return Code

	IKJEFT45		IKJEFT52	
Register	Name	Contents	Name	Contents
0	R0	Work Register	R0	Work Register
1	R1	↑ Parm List	R1	↑ IOPL
2	UPTPTR	↑ UPT	ECTPTR	↑ ECT
3	ECTPTR	↑ ECT	PTPBPTR	↑ Parm Block
4	R4	Work Register	SAVEPARM	Users Old Address
5	R5	Work Register	SAVETPUT	Users Output Option
6	--	Work Register	--	Work Register
7	PGPBPTR	↑ PGPB	--	Work Register
8	--	Work Register	--	Work Register
9	--	Work Register	--	Work Register
10	--	Work Register	--	Work Register
11	--	Work Register	--	Work Register
12	--	Work Register	--	Work Register
13	R13	↑ Register Save Area	R13	↑ Register Save Area
14	--	Work Register	--	Work Register
15	R15	Return Code	R15	Return Code

Figure 13. Register Usage: Terminal I/O Service Routines (Part 1 of 2)

	IKJEFT53		IKJEFT54	
Register	Name	Contents	Name	Contents
0	R0	Work Register	R0	Work Register
1	R1	↑ IOPL	R1	↑ Parm List
2	R2	Work Register	R2	Work Register
3	--	Work Register	--	Work Register
4	--	Work Register	--	Work Register
5	--	Work Register	--	Work Register
6	--	Work Register	R6	↑ O.L.D.
7	--	Work Register	R7	↑ Area for format msg
8	--	Work Register	--	Work Register
9	--	Work Register	--	Work Register
10	--	Work Register	--	Work Register
11	--	Work Register	--	Work Register
12	--	Work Register	--	Work Register
13	--	Work Register	R13	↑ Register Save Area
14	--	Work Register	--	Work Register
15	--	Return Code	R15	Return Code

	IKJEFT55		IKJEFT56	
Register	Name	Contents	Name	Contents
0	R0	Work Register	R0	Work Register
1	R1	↑ Parm list or 'GET' info in PUTGET parm block	R1	↑ Copy of IOPL
2	R2	Work Register	PTPBPTR	↑ PTPB
3	R3	Work Register	--	Work Register
4	GTPBPTR	↑ GTPB	--	Work Register
5	STACKPTR	↑ STACK	--	Work Register
6	--	Work Register	--	Work Register
7	R7	Work Register	--	Work Register
8	--	Work Register	--	Work Register
9	--	Work Register	--	Work Register
10	--	Work Register	--	Work Register
11	--	Work Register	--	Work Register
12	--	Work Register	--	Work Register
13	R13	↑ Register Save Area	R13	↑ Register Save Area
14	--	Work Register	--	Work Register
15	R15	SVC Return Codes	--	Return Code

Figure 13. Register Usage: Terminal I/O Service Routines (Part 2 of 2)

Routine	Return Code Hexadecimal	Meaning
IKJEFT30	00	Normal.
	04	Operation code could not be interpreted, or an invalid record was found in an in-storage list for an add request.
IKJEFT35	None	This routine contains messages.
IKJEFT40	04	An attention interrupt occurred during PUTLINE processing.
	08	A 'nowait' option was specified and no line returned.
	0C	The write function could not be interpreted because of faulty input parameters.
	10	A conditional GETMAIN was executed and there was insufficient space.
IKJEFT45	00	Normal.
	04	Input line not received from terminal.
	08	No input returned and/or no output line put out. (An attention interrupt occurred and the communications ECB was posted.)
	0C	1) Mode message was specified, input is not from the terminal, second level messages are chained and user has specified no pause. 2) Prompt message was specified and either the user has specified no prompt or input is from a command procedure.
	10	Nowait was specified for the TPUT option and the output was not sent and input not received.
	14	Input was not received.
	18	Invalid parameters.
	1C	A conditional GETMAIN was issued and no space was available.
IKJEFT45	00	Normal.
	04	A line was successfully obtained from an in-storage list.
	08	The communications ECB was posted and there was no successful completion of a TGET.
	0C	1) For a prompt request -- either the UPT specified no prompt, or the current source of input was a procedure. 2) For a mode request -- the current source is non-terminal; no pause was specified in the UPT and a second level informational chain exists.
	10	The nowait option had been specified and no space was available in the TIOC output buffer.
	14	The nowait option had been specified and no line was available in the TIOC input buffer.
	18	PUTGET input parameters were invalid.

Figure 14. Return Codes: Terminal I/O Service Routines (Part 1 of 2)

Routine	Return Code Hexadecimal	Meaning
IKJEFT45 (continued)	1C	A GETMAIN was issued and no space was available.
IKJEFT52	00	Normal. The error codes provided by TERMOUT are returned to the caller by this routine.
IKJEFT53	00	Normal. No Error Codes.
IKJEFT54	00	Normal.
	0C	Invalid parameters.
	10	GETMAIN failure.
IKJEFT55	00	A line was obtained from the terminal.
	04	A line was obtained from a list.
	08	An ECB was posted as a result of an attention interrupt.
	0C	"nowait" was specified as a TGET option and a line could not be obtained from TGET.
	10	An EOD (End of Data) condition from a list.
	14	Invalid parameters were supplied to TGET or to the service routine.
	18	A conditional GETMAIN was executed and no space was available.
IKJEFT56	00	Normal.
	04	Attention interruption -- ECB posted.
	08	NOWAIT specified, no line put out.
	0C	Invalid parameters.

Figure 14. Return Codes: Terminal I/O Service Routines (Part 2 of 2)

Part 3: Command SCAN and PARSE Service Routines

3

Section 1: Introduction

Command scan and parse search the command buffer for TSO commands and their parameters. In general, command scan is invoked by the terminal monitor program while parse is invoked by TSO command processors. Command scan is also invoked by the TEST command processor and by other TSO command processors that process subcommands.

Command scan searches the command buffer for question mark, command name, or null line. If syntax checking is requested, command scan checks the command name to be sure that it starts with an alphabetic character and contains no more than eight alphameric characters. If syntax checking is not requested, command scan assumes that the first alphabetic character in the buffer is the start of a command name. Command scan translates the command name to uppercase and updates the buffer offset to point to the first parameter or to the end of the buffer.

Parse searches the command buffer for command parameters, checks them for correct syntax, translates them to uppercase, and presents them to a user-supplied exit routine for validity checking. If a parameter is invalid, or if a required parameter is missing, parse can either prompt the terminal user to enter the parameter or supply a default value.

As supplied with TSO, the command scan and parse service routines reside in SYS1.LINKLIB and execute in the user's foreground region with the protection key assigned to that region. The installation may choose to make command scan and parse resident in the TSO link pack area (TSLPA) in the region assigned to the Time Sharing Control Task (TSCT).

Section 2: Method of Operation

This section describes the method of operation of the command scan and parse service routines. It includes the following method of operation diagrams:

- Method of Operation Diagram 13: Command Scan and Parse Service Routines (Overview) -- which shows the basic functions performed by command scan and parse.
- Method of Operation Diagram 14: Command Scan Service Routine -- which shows how command scan searches the command buffer for TSO commands.
- Method of Operation Diagram 15: Parse Service Routine -- which shows how parse searches the command buffer for TSO command parameters.
- Method of Operation Diagram 16: Parse Initialization -- which shows how parse sets up the parameter descriptor list (PDL).
- Method of Operation Diagram 17: Searching for IKJPARS Positional Parameters -- which shows how parse searches the command buffer for IKJPARS positional parameters.
- Method of Operation Diagram 18: Searching for IKJPARS Positional Parameters -- which shows how parse searches the command buffer for IKJPARS2 positional parameters.
- Method of Operation Diagram 19: Searching for Keyword Parameters and Subfields -- which shows how parse searches the command buffer for keyword parameters and subfields consisting of positional and/or keyword parameters.

Each method of operation diagram includes a cross-reference table to help you find the appropriate program description or assembly listing.

Overview

Method of Operation Diagram 13 shows the shows the basic functions of command scan and parse and their use by the terminal monitor program, the TEST command processor, and other TSO command processors. Briefly, here is what happens:

- The terminal monitor program (or TEST command processor) obtains a line of input, which is assumed to contain a TSO command and its parameters.
- The terminal monitor program (or TEST command processor) links to command scan and passes it the address of the buffer.
- Command scan searches the buffer for a command name, updates the buffer offset to point to command parameters (if any), and returns control to the calling program.
- The terminal monitor program (or TEST command processor) receives the address of the command name and gives control to the appropriate TSO command processor.

- The TSO command processor links to parse and passes it the address of the buffer.
- Parse searches the buffer for parameters, builds a list of parameters found, and returns control to the calling program. If required parameters are missing, parse prompts the terminal user to enter the parameters or supplies default values.
- The TSO command processor processes the command according to the parameters received.
- When the TSO command processor completes, it returns control to the terminal monitor program (or TEST command processor) and the sequence is repeated.

Command Scan Service Routine

Method of Operation Diagram 14 shows how command scan searches the buffer for question mark, command name, or null line. It includes a cross-reference table to help you find the appropriate flowchart or assembly listing.

ENTRY TO COMMAND SCAN

The entry to command scan is by a LINK (or LOAD/CALL) macro instruction to entry point IKJSCAN in load module IKJSCAN.

INPUT TO COMMAND SCAN

On entry to command scan, register 1 points to the command scan parameter list (CSPL), which includes the following information:

- The address of the user profile table (UPT).
- The address of the environment control table (ECT).
- The address of the calling routine's event control block (ECB). (Command scan does not use the addresses of the UPT, ECT, and ECB.)
- The address of a flag word set as follows:

Flag	Meaning
X'00'	Check the syntax of the command name.
X'80'	Do not check the syntax of the command name.

(See "Checking Command Syntax" in this section.)

- The address of the command scan output area (CSOA) which is set by command scan to indicate the results of the search. (See "Output from command scan" in this section.)
- The address of the command buffer (CBUF).

SEARCHING FOR COMMANDS

Command scan searches the buffer for commands beginning at the current buffer offset. If the first character is a question mark, no further searching is necessary. Otherwise, command scan searches the buffer until it finds a command name or until it reaches the end of the buffer. It assumes the command name to start with the first non-separator character and to end just before the next separator character. See Figure 15 for a definition of character types recognized by command scan and parse.

Unless syntax checking is requested, command scan does no further checking and assumes the command name is correct.

CHECKING COMMAND SYNTAX

The calling program can request syntax checking by setting the flag word to X'00'. (See "Input to Command Scan" in this section.)

If syntax checking is requested, the command name must meet the following requirements:

- The first character must be an alphabetic or a national character.
- The other characters must be alphabetic or numeric.
- The length must not exceed eight characters.
- The delimiter must be a separator character.

See Figure 15 for a definition of character types recognized by command scan and parse.

TRANSLATING COMMANDS TO UPPERCASE

Command names are translated to uppercase. The other characters in the buffer are not translated.

OUTPUT FROM COMMAND SCAN

On return from command scan, the command scan output area (CSOA) contains the following information:

- The address of the command name in the buffer.
- The length of the command name.
- Flags set as follows:

Flag Setting	Result of Scan	Buffer Offset
X'80'	Command name is valid; buffer includes parameters.	Points to first parameter.
X'40'	Command name is valid; buffer does not include parameters.	Points to end of buffer.
X'20'	Command name is a question mark.	Unchanged.
X'10'	Buffer contains null line; no non-separator characters.	Points to end of buffer.
X'08'	Command name is syntactically invalid.	Unchanged.

CHARACTER		CHARACTER TYPE						
		Separator	National	Alphabetic	Numeric	Command Delimiter	Delimiter	Special
Horizontal Tab	HT	X				X		
Blank	ƀ	X				X		
Comma	,	X				X		
Dollar Sign	$		X					
Number Sign	#		X					
At Sign	@		X					
	a-z			X				
	A-Z			X				
	0-9				X			
New line	NL					X	X	
Period	.					X		X
Left parenthesis	(					X		X
Right parenthesis	)					X		X
Ampersand	&					X		X
Asterisk	*							X
Semicolon	;					X	X	
Minus sign, hyphen	-					X		X
Slash	/					X	X	
Apostrophe	'					X		X
Equal sign	=					X		X
Cent sign	¢							X
Less than	<							X
Greater than	>							X
Plus sign	+							X
Logical OR	|							X
Exclamation point	!							X
Logical NOT	¬							X
Percent sign	%							X
Dash	-							X
Question mark	?							X
Colon	:							X
Quotation Mark	"							X

Figure 15. Character Types Recognized by Command Scan and Parse

RETURN TO CALLING PROGRAM

Command scan issues a RETURN macro instruction to return control to the calling program. At exit, register 15 contains one of the following return codes.

Code	Meaning
X'0'	No errors found.
X'4'	Invalid input parameters.

Parse Service Routine

Method of Operation Diagram 15 shows the basic functions of parse and their use by the TSO command processors. It includes a cross-reference table to help you find the appropriate assembly listing.

Parse searches the command buffer for two main classes of parameters:

- Positional parameters -- which must appear in a certain order.
- Keyword parameters -- which can appear in any order but must follow all positional parameters. Keyword parameters may have subfields which include positional parameters and/or keyword parameters.

Briefly, here is what happens:

- The TSO command processor builds a parameter control list (PCL) that describes the parameters expected. That is, it describes acceptable values and defaults.
- The TSO command processor links to parse and passes it the address of the parse parameter list (PPL), which contains the address of the PCL, the address of the command buffer, and the address of a word where parse will put the address of the parameter descriptor list (PDL).
- Parse searches the command buffer for correct parameters and builds a PDL that describes the parameters found.
- Parse places the address of the PDL in the word pointed to by the PPL and returns control to the TSO command processor.

Other points made in the simplified method of operation diagram will be briefly mentioned here:

- The TSO command processor uses system macro instructions to build the PCL. (See Figure 16 for a brief description of these macro instructions.) Each macro instruction generates one parameter control entry (PCE).
- The TSO command processor uses system macro instruction IKJRLSA to free main storage obtained by parse for the PDL and for buffers used when prompting the terminal to re-enter parameters.

Macro Instruction	Description
IKJPARM	Marks the beginning of the PCL. Gives the length of PCL and length of PDL. Gives offset to next IKJKEYWD, IKJSUBF, or IKJENDP PCE.
IKJIDENT	Describes a positional parameter in the form of a character string with optional restrictions on the first character, other characters, and length.
IKJPOSIT	Describes a positional parameter which includes a delimiter as part of its syntax.
IKJTERM	Describes a positional parameter that may be a constant, variable or statement number.
IKJOPER	Describes an expression consisting of two operands and an operator.
IKJRSVWD	Is used with the IKJTERM macro to describe a figurative constant, or with the IKJOPER macro to describe the operator in an expression, or by itself to describe a reserved word parameter.
IKJKEYWD	Marks the beginning of a keyword field. Specifies a default, if there is one, for the field.
IKJNAME	Describes one of the eligible names for a keyword or a reserved word field. Specifies the options associated with this name.
IKJSUBF	Marks the beginning of a subfield and the end of a previous subfield. Gives offset to next IKJKEYWD, IKJSUBF, or IKJENDP PCE.
IKJENDP	Marks the end of the PCL.
Note: For a more complete discussion of the parse macros, refer to the publication, OS/VS2 TSO Guide to Writing a Terminal Monitor Program or a Command Processor, GC28-0648.	

Figure 16. Brief Description of Parse Macro Instructions

ENTRY TO PARSE

The entry to parse is by a LINK (or LOAD/CALL) macro instruction to entry point IKJPARS in load module IKJPARS.

INPUT TO PARSE

On entry to parse, register 1 points to the PPL, which includes the following information:

- The address of the user profile table (UPT).
- The address of the environment control table (ECT).
- The address of the calling routine's event control block (ECB).
- The address of the environment control table (ECT).
- The address of the calling routine's event control block (ECB).

(Parse does not use the address of the UPT, ECT, and ECB.

- The address of the parameter control list (PCL). (See below.)
- The address of a word where parse puts the address of the parameter descriptor list (PDL).

(See "Output from Parse" in this section.)

- The address of the command buffer (CBUF).
- The address of a user work area.

The PCL is the major input to parse. It consists of Parameter Control Entries (PCEs) that describe the parameters expected and determine the operations to be performed by parse.

SET UP AND INTIALIZATION

When parse locates the IKJPARM PCE, it performs various initialization functions.

Method of Operation Diagram 16 (foldout) shows what parse does during set up and initialization. It includes a cross-reference table to help you find the appropriate assembly listing.

Briefly, here is what happens:

- Parse gets main storage for the PDL and puts its address in the answer area provided by the calling routine. See "Input to Parse" in this section.

- Parse gets main storage for the parse work area (PWORK) and first recurse work area (RWORK) and initializes them. Parse gets another recurse work area each time it processes subfield.

- Parse loads the PUTLINE and PUTGET service routines.

- Parse loads IKJPARS2 if necessary.

When initialization is complete, parse is ready to search for positional parameters.

SEARCHING FOR POSITIONAL PARAMETERS

Parse recognizes two classes of positional parameters:

- Simple ones -- non-delimiter dependent parameters. These are described by IKJIDENT PCEs.
- Complicated ones -- delimiter dependent parameters. These are described by IKJPOSIT PCEs.

See Figure 17 for a description of each type of positional parameter.

Method of Operation Diagram 17 (foldout) shows how parse searches for positional parameters described by IKJIDENT, IKJPOSIT, IKJTERM, IKJOPER, or IKJRSVWD PCEs. It includes a cross-reference table to help you find the appropriate assembly listing.

Briefly, here is what happens:

- Parse locates an IKJIDENT, IKJPOSIT, IKJTERM, IKJOPER, or IKJRSVWD PCE and branches to the appropriate processing routine.
- Parse searches the buffer for the parameter. If the parameter is missing, Parse checks the PCE to see if the parameter is necessary, or if there is a default.
  - If the parameter is necessary, parse prompts the terminal for the parameter.
  - If there is a default, parse supplied the default.
  - If the parameter is not necessary, parse locates the next PCE.
- If the parameter is correct, parse builds a PDE for the parameter and translates the parameter to uppercase. If a list is being processed, parse builds a PDE for each element in the list.
- If the calling routine has specified a validity check exit, parse gives control to a validity check exit routine. If a range is being processed, the complete range is passed to the validity check exit routine. If a list is being processed, each element in the list is passed to the validity check exit routine.
  - If parse finds an error it prompts the terminal user to reenter the parameter, and erases the invalid PDE. the parameter, and erases the invalid PDE.

3

Macro Instruction	Type of Parameters	Description
IKJIDENT	Positional character string	The IKJIDENT PCE describes a positional parameter in the form of a character string with optional restrictions on the first character, other characters, and length. The IDENT parameter may be an asterisk. The following character types are recognized for the beginning chatacter and additional characters: ALPHA -- An alphabetic or national character. NONATABC -- Alphabetic characters, no national characters. NONATNUM -- Alphameric characters, no national characters. NUMERIC -- A digit, 0-9. ALPHANUM -- An alphabetic character, national character, or digit. ANY -- Any character other than a blank, coma, tab, semicolon, or carriage return character.
IKJPOSIT	Any of the following:	The IKJPOSIT PCE describes a positional parameter of one of the following types:
	DELIMITER	A delimiter parameter is a self-defining delimiter character used to delimit a string. It may be any character other than an asterisk, left parenthesis, right parenthesis, semicolon, blank, comma, tab, carriage return character, or digit. A self-defining delimiter character is represented by the symbol . The delimiter parameter is used only conjunction with the string parameter discussed below.
	STRING	A string is the group of characters between two alike self-defining characters, such as: Δstring Δ of the group of characters between a self-defining delimiter character and the end of a logical line, such as: Δstring

Figure 17. Types of Positional Parameters (Part 1 of 9)

Macro Instruction	Type of Parameters	Description
IKJPOSIT (cont.)	STRING (cont.)	The same self-defining delimiter character may be used to delimit two continuous strings, such as: Δstring Δstring Δ Δstring Δstring A null string is defined as two contiguous delimiters or a delimiter and the end of the logical line. If the string is required, a null string must be entered as two contiguous delimiters. If the next nonblank character scanned is not a valid self-defining character, the string is considered missing. A string received from a prompt or default must not include the delimiters. If the delimiter is a quote and the SQSTRING flag is on in the PCL, the string is processed as a quoted string (see QSTRING below).
	VALUE	A value consists of type-character followed by a string enclosed in quotes, such as: X'string' The type-character must be an alphabetic or national character. The string may consist of any combination of enterable characters of any length. The ending quote may be left off the string in which case the end of the string is the end of the logical line. A message is issued indicating the end quote is assumed. Two successive single quotes are considered part of the string, such as C'a''b'. The value is considered missing if the first character is not an alphabetic character of if the following character is not a quote. The type-character preceding the quoted string is always translated to uppercase.
	ADDRESS	There are several forms of the address parameter: an absolute address, a relative address, a general register address, a floating-point register address, a symbolic address, a qualified address, an indirect address, and an address expression. Absolute address -- One to six hexadecimal digits followed by a plus sign.

Figure 17. Types of Positional Parameters (Part 2 of 9)

3

Macro Instruction	Type of Parameters	Description
IKJPOSIT (cont.)	ADDRESS (cont.)	Relative address -- One to six hexadecimal digits preceded by a period. General register address -- A decimal integer in the range 0-15 followed by the letter "R". "R" may be upper case or lower case. Floating-point register address -- An even decimal integer in the range 0-6 followed by the letter "D" (for double precision) or the letter "E" (for single precision). The "E" and "D" may be upper case or lower case. Symbolic address -- Any 1-31 character combination of the alphameric characters and the break character, of which the first character is an alphabetic or national character. Qualified address -- A qualified address has the following format: [loadname].entryname .symbolic address .relative address [ ] - Optional loadname any combination of eight or fewer alphameric characters of which the first character is an alphabetic or national character. entryname same syntax as a load.name, (but must be preceded by a period as illustrated above). symbolic address defined above, (but must be preceded by a period as illustrated above). relative address defined above, (but must be preceeded by a period, as illustrated above). Indirect address -- An indirect address is an absolute, relative, symbolic, or general register address followed by from 1 to 255 percent signs, such as: A.%%% Address expression -- An address expression has the following format: address[%...]±expression value[%...] [±expression value[%...]...]

Figure 17. Types of Positional Parameters (Part 3 of 9)

Macro Instruction	Types of Parameters	Description
IKJPOSIT (cont.)	ADDRESS (cont.)	address an absolute, relative or symbolic address. address may be used but it must have indirect address notation, that is, it must be followed by at least one percent sign. expression value 1-6 hexadecimal digits or 1-6 decimal digits followed by the letter "N". "N" may be uppercase or lowercase. There is no limit to the number of expression values in the address expression. Blanks are not allowed within any form of the address parameter. [ ] - Optional
	PSTRING	A parenthesized string is a string of characters enclosed by a balanced set of parentheses, such as: (string) The string may consist of any combination of enterable characters of any length, with one restriction: If it includes parentheses, they must be balanced. The enclosing right parentheses may be eliminated if the string ends at the end of the logical line. A message is issued indicating it is assumed. A null string is defined as a left parenthesis followed by a right parentheses or a left parentheses at the end of the logical line. It may be entered in either form at all times. A parenthesized string received from a prompt or default must include at least the enclosing left parenthesis.
	USERID	A userid consists of an identification optionally followed by a slash and a password. The format is: identification [/password] identification any combination of seven or fewer alphameric characters, the first of which must be an alphabetic or national character. password any combination of eight or fewer alphameric characters.

Figure 17. Types of Positional Parameters (Part 4 of 9)

3

Macro Instruction	Type of Parameters	Description
IKJPOSIT (cont.)	ADDRESS (cont.)	Blanks, can be inserted between the identification and the slash and between the slash and the password. If only the identification is entered, no prompting for the password takes place, whether or not the userid parameter is necessary. If the identification is entered followed by a slash in bypass mode. The terminal user may enter a password or reply with a null line. See PUTGET service routine external specifications for an explanation of prompting in bypass mode.
	DSNAME	The data set name parameter has three possible formats: dsname[(membername)][/password] [dsname](membername)[/password] 'dsname[membername]'[/password [ ] dsname a qualified or unqualified name. An unqualified name is any combination of eight or fewer alphameric characters, the first of which must be an alphabetic or national character. A qualified name is made up of sevral unqualfied names; each name is separated by a period. A qualified name including the periods, can be as many as 44 characters long. membername 1-8 alphameric characters, the first of which must be an alphabetic or national character. The data set name parameter is considered missing if the first character is not a single quote, alphebetic or national character, or left parenthesis. The password may be any alphameric combination of eight or fewer characters. If the slash and password are not entered, no prompting for the password takes place whether or not the dsname parameter is required. If the slash is entered and not the password, a prompt for the password occurs in bypass mode. The terminal user may enter a password or reply with a null line. See PUTGET Service Routine for an explanation of prompting in bypass mode.

Figure 17. Types of Positional Parameters (Part 5 of 9)

Macro Instruction	Type of Parameters	Description
IKJPOSIT (cont.)	DSTHING	A data set thing is a dsname parameter as defined above except that an asterisk may be substituted for an unqualified name and for each qualifier of a qualified name.
	QSTRING	A quoted string is a string of characters 'string' The string may consist of any combination of enterable characters of any length, with one restriction: If the user wishes to enter quotes within the string, two successive quotes must be entered for every single quote desired. One of the quotes is removed during the parse. The ending quote may be eliminated if the string ends at the end of the logical line. A message is issued indicating the end quote is assumed. A null quoted string is defined as two contiguous quotes or a single quote at the end of the logical line. It may be entered in either form at all times. A quoted string received from a prompt must include at least the enclosing left quote.
	SPACE	This is a special purpose parameter for the TSO EDIT command. It allows a string parameter which directly follows a command name, to be entered without a preceding self-defining character. If the delimiter of the command name is a tab, the tab is the first character of the string. The string always ends at the end of the logical line. The space parameter must be followed by a string parameter.
IKJTERM	Any of the following	The IKJTERM PCE describes a positional parameter of one of the following types:
	CONSTANT	There are several forms of the constant parameter. <u>Fixed-point numeric literal</u> - Constants of a string of digits (0 - 9) preceded optional by a sign (+ or -), such as: +1234.43 This literal may contain a decimal point anywhere in the string except as the rightmost character. The total number of digits can not exceed 18. <u>Floating-point numeric literal</u> - has the form: +1234.56E+10

Figure 17. Types of Positional Parameters (Part 6 of 9)

Macro Instruction	Type Parameters	Description
IKJTERM (Cont.)	CONSTANT (Cont.)	This literal is a string of digits (0 - 9) preceded optionally by a sign (+ or os) and must contains a decimal point. This is immediately followed by the letter E and then a string of digits preceded optionally by a sign. Embedded blanks are not allowed. The string of digits preceding the E cannot be greater than 16 and the string following the E cannot be greater than 2. <u>Non-numeric literal</u> - Constants of a string of characters from the EBCDIC character set excluding the quote and enclosed in quotes such as: 'Numbers (123) and letters are OK)' The length of the string excluding apostrophes may be from 1 to 120 characters in length. <u>Figurative constant</u> - May be one of a set of keywords supplied by the caller of the parse routine such as: test123 A figurative constant consists of a string of characters up to a length of 255. Embedded blanks are not allowed. All characters of the EBCDIC set are allowed except the blank, comma, tab, semicolon, and carriage return.
	VARIABLE	A variable parameter has the form: [program-id.]data-name OF qualification IN (subscript) <u>Data-name</u> - consists of a maximum of 80 characters of the set: A through Z (alphabetic) 0 through 9 (numeric) - (hypen) such as: My-dataset-123 The data-name cannot begin or end with a hyphen and must contain at least one alphabetic character. <u>Program-id</u> - Consists of the first eight characters of a program identifier followed by a period. The first character must be alphabetic (A - Z) and the remaining characters alphameric (A - Z or 0 - 9) such as: Here55.My-dataset

Figure 17. Types of Positional Parameters (Part 7 of 9)

Macro Instruction	Type of Parameter	Description
IKJTERM (cont.)	VARIABLE (cont.)	Qualification - Is applied by placing after data-name one or more data-name(s) preceded by the reserved words IN or OF such as: My-dataset-123 OF Your-dataset-456 The number of qualifiers is limited to 255. Subscript - Consists of a data-name with subscripts enclosed in parentheses following the data-name such as: Your-dataset-456 (My-dataset-123) A separator between the data-name and subscripts is optional. Subscripts are a list of constants and/or variables. The number of subscripts is limited to 3.
	STATEMENT NUMBER	A statement number has the following form: [program-id.]line number[.verb number] An example is: Here.23.7 Program-id - Consists of the first eight characters of a program identifier followed by a period. The first character must be alphabetic (A - Z) and the rmeaining characters alphameric (A - Z or 0 - 9). Line number - Consists of a string of digits (0 - 9) and cannot exceed a length of 6 digits. Verb number - Consists of one digit (0 - 9) preceded by a period. Embedded blanks are not allowed.
IKJOPER	EXPRESSION	The IKJOPER PCE describes an expression that has the form: (operand1 operator operand2) The operator in an expression shows a relationship between the operands, such as: (abc equals 123) An expression must be enclosed in parentheses. An expression is defined by the IKJOPER macro. The operands are defined by the IKJTERM macro and the operator by the IKJRSVWD macro.

Figure 17. Types of Positional Parameters (Part 8 of 9)

3

Macro Instruction	Type of Parameter	Description
IKJRSVWD	RESERVED WORD	A reserved word has three uses depending on the presence or absence of operands on the IKJRSVWD macro. The uses are: a. When used with the RSVWD keyword on the IKJTERM macro, the IKJRSVWD macro identifies the beginning of a list of reserved words anyone of which can be entered as a constant. b. When used with the RSVWD keyword on the IKJOPER macro, the IKJRSVWD macro identifies the beginning of a list of reserved words anyone of which can be entered as an operator in an expression. c. When used by itself, the IKJRSVWD macro defines a positional reserved word parameter. Note: The IKJRSVWD macro is followed by a list of IKJNAME macros that contain all of the possible reserved words used as constants or operators.

Figure 17. Types of Positional Parameters (Part 9 of 9)

Lists and Ranges

Some positional parameters may be entered in the form of a list or a range or a list of ranges.

- A list is one or more of the same type of positional parameter enclosed in parentheses. For example: (parameter parameter)
- A range is two positional parameters separated by a colon. For example: parameter:parameter

The following positional parameters may be used in a list form: value, address, userid, dsname, dsthing, and positional character string. A list may not contain items with unmatched left and right parentheses except that the closing right parenthesis may be omitted at the end of a logical line.

The following positional parameters may be used in the form of a range or a list of ranges: address, value, and positional character string.

Validity Check

After the PDE is built, the PCE is checked to see if there was a validity check routine and, if so, the validity check routine is entered by a branch and link.

At entry to the validity check routine, register 1 points to the validity check parameter list (VCEPARAM) which contains the following information:

- The address of the PDE.
- The address of a work area built by the TSO command processor.
- The address of a second-level message (provided by the validity check routine) which is initialized to X'FF000000'.

On return to parse, register 15 contains the following return code:

Code	Meaning
X'00'	The parameter is valid, continue.
X'04'	The parameter is invalid, write an error message, prompt the terminal user to reenter it.
X'08'	The parameter is invalid, an error message was issued, prompt the terminal user to reenter it.
X'10'	The IKJTERM macro is chained under the IKJOPER macro, and the data field that follows is to be scanned as a subscript.
X'0C'	Error forces termination. Cleanup and return to TSO command processor.

SEARCHING FOR KEYWORD PARAMETERS AND SUBFIELDS

When parse locates an IKJKEYWD PCE, it searches the command buffer for a keyword field. The eligible names for a keyword are specified by IKJNAME PCEs. Keyword parameters may have subfields that contain positional and/or keyword parameters. If so, an IKJSUBF PCE marks the beginning of the subfield.

Method of Operation Diagram 18 shows how parse searches for keyword parameters described by IKJKEYWD and IKJNAME PCEs and for subfields specified by IKJSUBF PCEs. It includes a cross-reference table to help you to find the appropriate assembly listing.

Briefly, here is what happens:

- Parse locates an IKJKEYWD PCE and branches to the keyword processing routine (KEYWDP).
- The keyword processing routine searches the buffer for a keyword and compares it to each of the eligible names specified by IKJNAME PCEs.
- If keyword processing finds a march, parse builds an IKJKEYWD PDE and checks the IKJNAME PCE for a subfield. If no match is found, or if more than one match is found, parse prompts the terminal user to reenter the parameter.
- If the keyword has a subfield, parse searches for parameters in the subfield in exactly the same way that it searches for parameters in a field. When it reaches an IKJSUBF PCE or IKJENPD PCE, parse returns to processing the main part of PCL.
- After searching for all of the keyword fields, the keyword processing routine checks each IKJKEYWD PCE to see if it has been filled. If not, it supplies a default value. (Keyword parameters are never required.)
- When keyword processing reaches an IKJENDP PCE, parse returns to the calling routine.

Method of Operation Diagram 19 also illustrates the following additional points:

- Keyword parameters may be entered in any order. They must follow all positional parameters. The first character of the keyword must be alphabetic; all other characters must be alphameric. Maximum length is 31 characters. See Figure 15 for a definition of character types.
- Parse lets the terminal user enter the fewest number of characters required for uniqueness. If the abbreviation is not unique, parse writes an "ambiguous keyword" message and prompts the terminal user to reenter the parameter.

- When the terminal user enters conflicting keywords, the last keyword found overrides the previous ones. Note that the last keyword found may not be the last keyword entered. If the terminal user is prompted for one parameter and enters other parameters, parse checks these parameters before continuing to search for parameters in the command buffer.

Subfields

Keyword parameters may have other parameters associated with them. In this case, those parameters are enclosed in parentheses immediately following the keyword and for the purpose of syntax checking are known as a subfield. In the following subfield, positional1 and keyword2 are parameters in the subfield of keyword 1:

```
keyword1 (positional1 keyword 2)
```

The parameters in the subfield are searched for in exactly the same way as before. The enclosing right parenthesis can be omitted at the end of a line.

PROMPTING AND DEFAULTING

Parse searches the command buffer until it reaches the end of the buffer If it finds an error, parse gets the address of a default value or prompts the terminal user to reenter the parameter. Parse scans the new data (with further prompting or defaulting, if necessary) before continuing to search the command buffer.

Figure 18 shows the scanning sequence for a command buffer that contains two errors. Note that there was a third error in the data received from the second prompt. The search continued until end-of-buffer was reached.

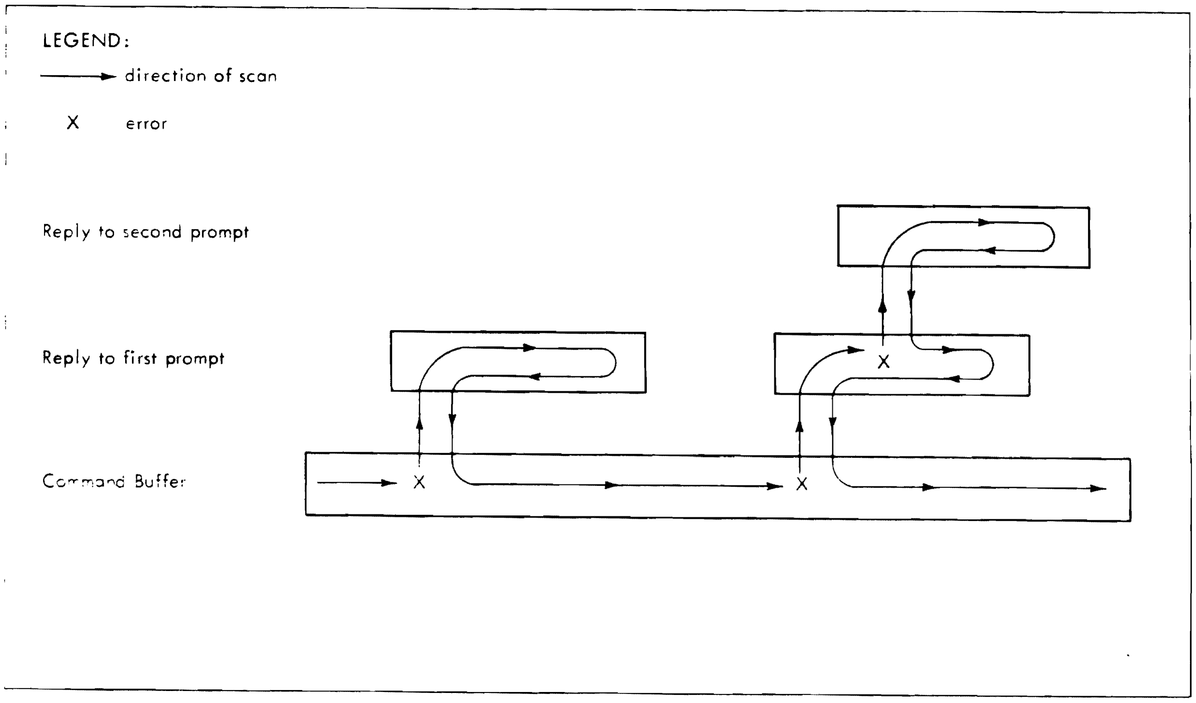


Figure 18. Scanning Sequence for Prompting and Defaulting

Parse keeps track of the data obtained by prompting or defaulting by means of an input stack as shown in Figure 19. Each element on the stack contains a pointer to the last character searched and a pointer to the end of the buffer.

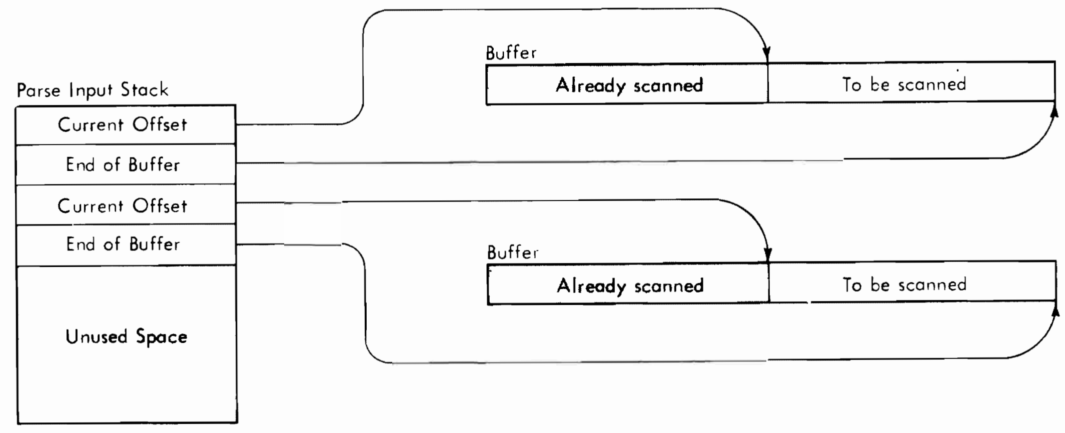


Figure 19. Keeping Track of Buffers with the Parse Input Stack

During initialization, parse sets up an input stack large enough to hold ten elements. If more elements are necessary, parse sets up new stacks and backward-chains them to the first stack. The current input pointer (PIPDLCUR) points to the current stack while the current index (PIPDLX) give the offset into the stack.

MESSAGES FROM PARSE

Parse uses the PUTLINE and PUTGET service routines to write messages to the terminal.

PUTLINE writes informational messages.
PUTGET prompts the terminal user for a line of input.

In most cases, PUTLINE and PUTGET are used in succession. For example:

entered from terminal	dataset%
issued by PUTLINE	INVALID DSNAME, dataset%
issued by PUTGET	REENTER -
entered from terminal	dataset1

OUTPUT FROM PARSE

Parse places the address of the PDL in the area provided by the calling routine. See "Input to Parse" in this section.

The PDL consists of PDEs that describe the parameters found in the command buffer or obtained by prompting or by supplying default values.

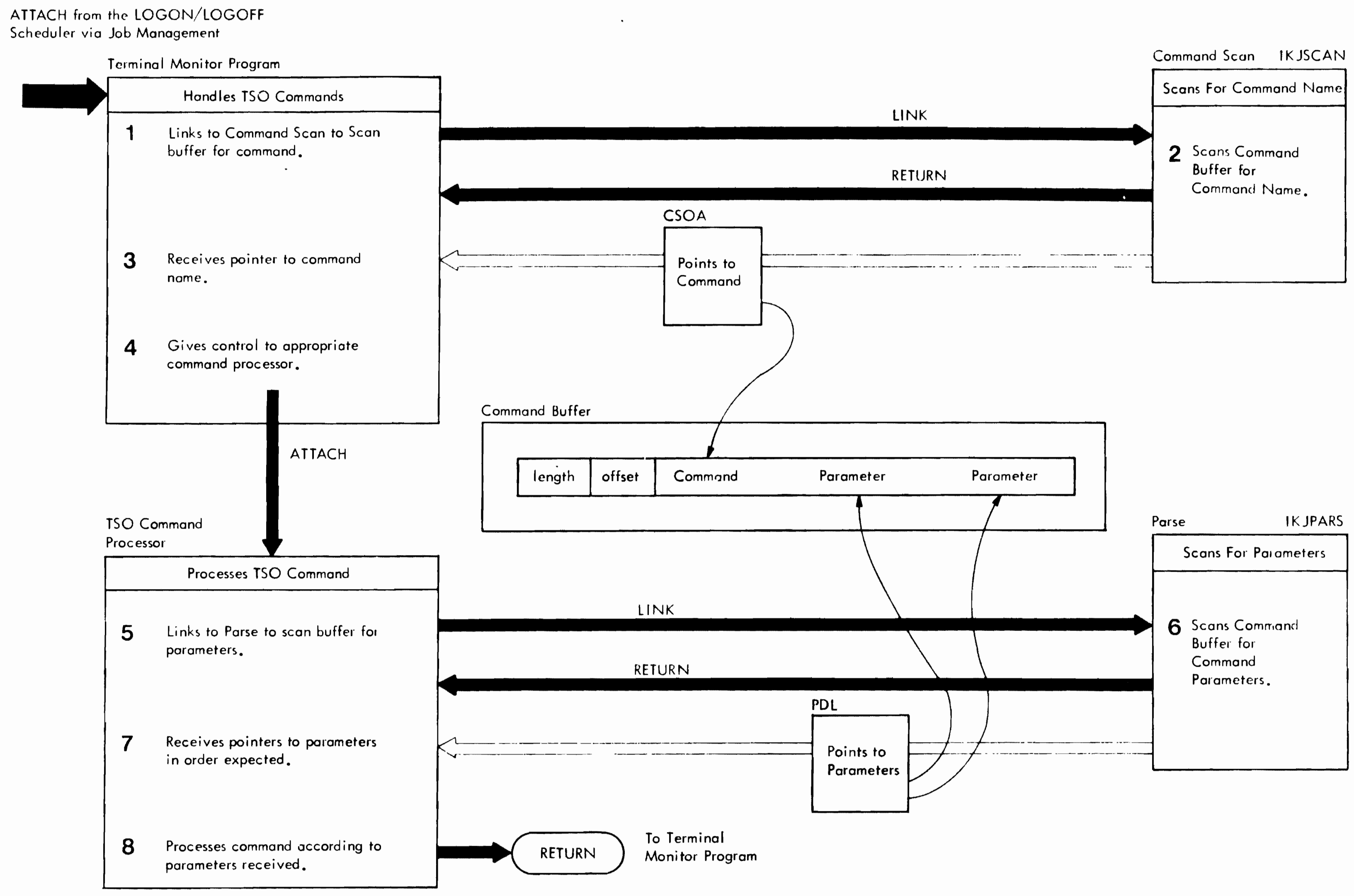
RETURN TO CALLING PROGRAM

Parse issues a RETURN macro instruction to return control to the calling program. At exit, register 15 contains one of the following return codes:

Code	Meaning
X'00'	Normal completion.
X'04'	Unable to prompt for valid parameter.
X'08'	Processing interrupted by attention interruption.
X'0C'	Invalid parameters from calling routine.
X'10'	No storage available for PDL
X'14'	Validity check routine requested termination.
X'18'	Invalid parameters passed to an IKJTERM, IKJOPER, or IKJRSVWD macro instruction.

Method of Operation Diagrams

3

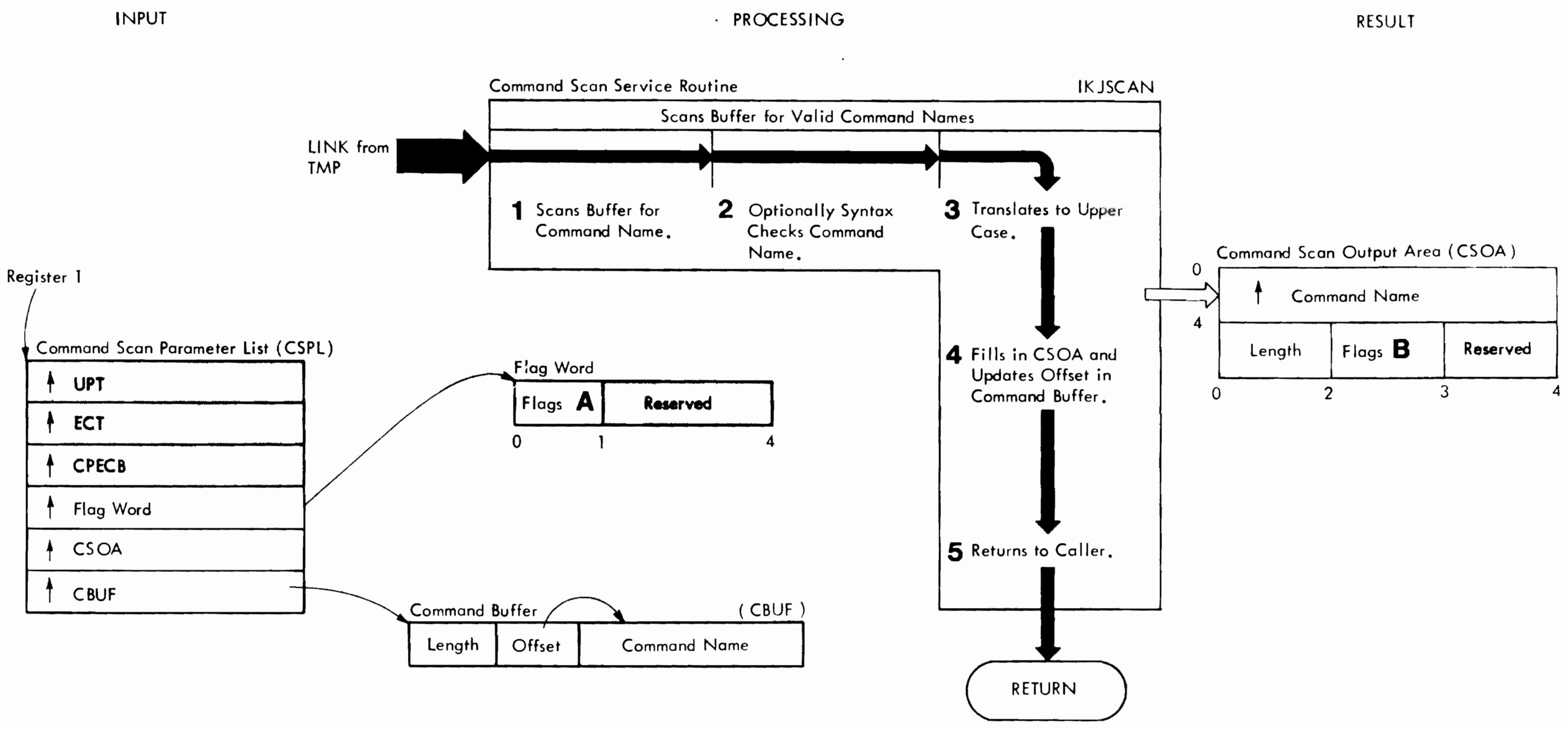

ATTACH from the LOGON/LOGOFF Scheduler via Job Management
Terminal Monitor Program
Handles TSO Commands
1 Links to Command Scan to Scan buffer for command.
3 Receives pointer to command name.
4 Gives control to appropriate command processor.
LINK
RETURN
Command Scan IKJSCAN
Scans For Command Name
2 Scans Command Buffer for Command Name.
CSOA
Points to Command
ATTACH
Command Buffer
length
offset
Command
Parameter
Parameter
TSO Command Processor
Processes TSO Command
5 Links to Parse to scan buffer for parameters.
7 Receives pointers to parameters in order expected.
8 Processes command according to parameters received.
LINK
RETURN
Parse IKJPARS
Scans For Parameters
6 Scans Command Buffer for Command Parameters.
PDL
Points to Parameters
RETURN
To Terminal Monitor Program

Method of Operation Diagram 13. Command Scan and Parse Service Routines (Overview) (Part 1 of 2)

CROSS REFERENCE TABLE

Key	Description	Routine	Label	Diagram
1	The Terminal Monitor Program gets a line of input from the terminal. The TMP links to the Commands Scan service routine and passes it the address of the Command Buffer.	Terminal Monitor Program	IKJEFT02	3
2	The Command Scan service routine scans the Command Buffer for a syntactically correct command name, updates the buffer offset field, and returns control to the TMP.	Command Scan service routine	IKJSCAN	14
3	The Terminal Monitor Program receives the address of the correct command name and gives control to the appropriate TSO command processor.	Terminal Monitor Program	IKJEFT02 Refer to part 1 of this book	3
4	The TSO command processor links to the Parse service routine and passes it the address of the Command Buffer and a Parameter Control List (PCL) that describes the parameters to be expected.		Refer to TSO command processors program logic manual	
5	The Parse service routine scans the Command Buffer for the parameters expected, builds a Parameter Descripror List (PDL) that describes the parameters found, updates the buffer offset, and returns control to the TSO command processor.	Parse service routine	IKJPARS	15
6	The TSO command processor processes the command according to the parameters received.		Refer to TSO command processors program logic manual	

Method of Operation Diagram 13. Command Scan and Parse Service Routines (Overview) (Part 2 of 2)


INPUT
PROCESSING
RESULT
Command Scan Service Routine
IKJSCAN
Scans Buffer for Valid Command Names
LINK from TMP
1 Scans Buffer for Command Name.
2 Optionally Syntax Checks Command Name.
3 Translates to Upper Case.
4 Fills in CSOA and Updates Offset in Command Buffer.
5 Returns to Caller.
RETURN
Register 1
Command Scan Parameter List (CSPL)
UPT
ECT
CPECB
Flag Word
CSOA
CBUF
Flag Word
Flags A
Reserved
0
1
4
Command Buffer
(CBUF)
Length
Offset
Command Name
Command Scan Output Area (CSOA)
0
Command Name
4
Length
Flags B
Reserved
0
2
3
4

A Input Flags

Flag	Meaning
X'00'	Syntax check
X'80'	No syntax check

B Output Flags

Flag	Meaning	Buffer Offset set to:
X'80'	Command name is valid; the remainder of the buffer contains non-separator characters.	First non-separator character following command name.
X'40'	Command name is valid; the remainder of the buffer is empty (contains only separator characters).	End of buffer.
X'20'	Command name is Questionmark.	Unchanged.
X'10'	Buffer is empty (contains only separator characters).	End of Buffer.
X'08'	Command name is syntactically invalid.	Unchanged.

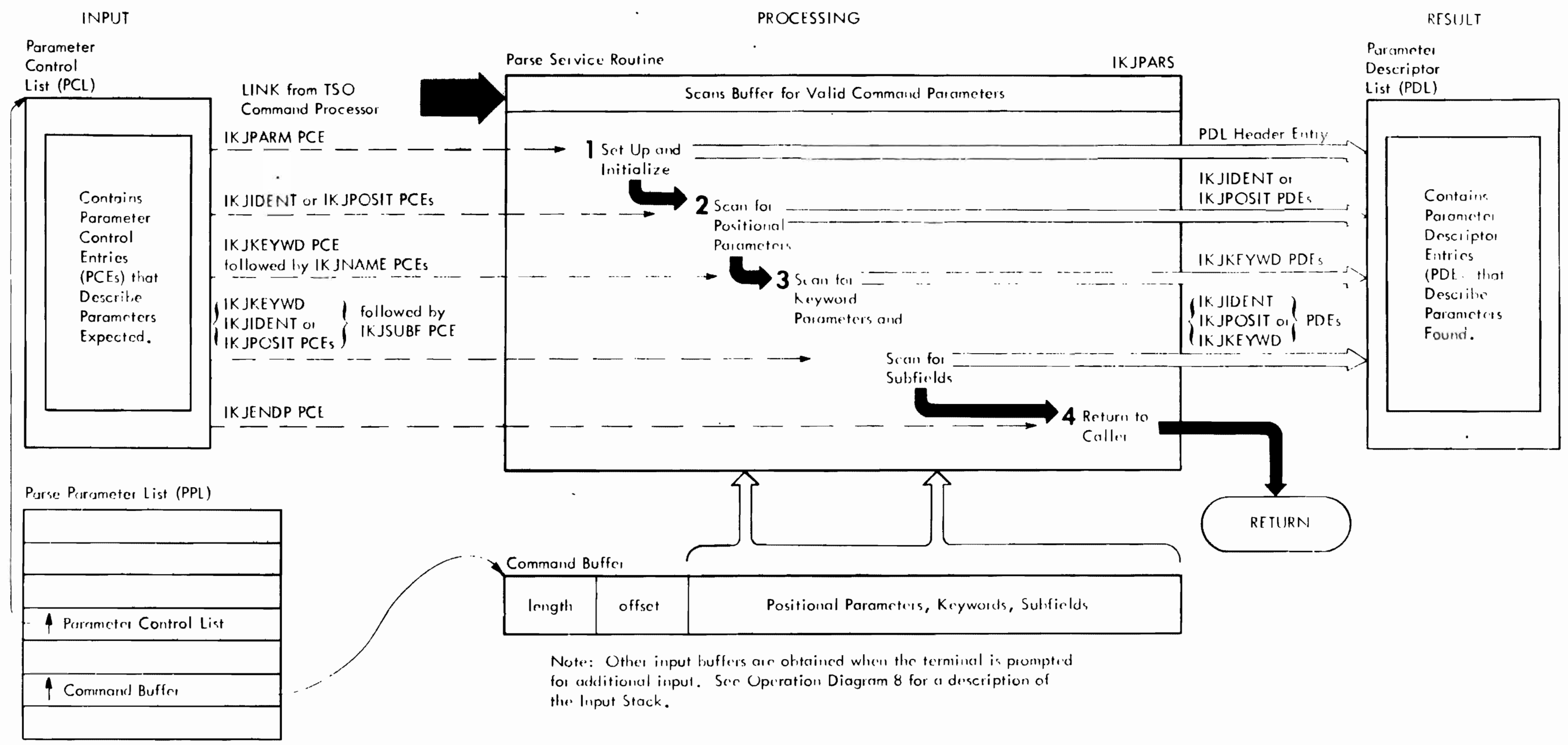
Method of Operation Diagram 14. Command Scan Service Routine

Method of Operation Diagram 14. Command Scan Service Routine (Part 1 of 2)

CROSS REFERENCE TABLE

Key	Description	Routine	Label
	During initialization, an unconditional GETMAIN is issued for a Command Scan Work Area (CSWORK).	IKJEFP30	IKJSCAN
1	Separators are skipped to the beginning of a command name. If the buffer is empty or if the first character is a questionmark, the program exits. Otherwise the scan continues to the next delimiter.	IKJEFP30	SKIPB
2	If the high order byte of the Flag Word is X'00' the command is syntax checked. Otherwise the command name must contain valid, enterable characters and end with a delimiter.	IKJEFP20 IKJEFP30	GENSCAN TYPETEST
3	Correct command names are translated to uppercase.	IKJEFP20	TRANSX
4	The CSOA is set to indicate the results of the scan and the buffer offset is updated as shown in **A**	IKJEFP30	CSEXIT80 CSEXIT40 CSEXIT20 CSEXIT10 CSEXIT08
5	Command Scan returns to caller.	IKJEFP30	CSEXIT

Method of Operation Diagram 14. Command Scan Service Routine (Part 2 of 2)


INPUT
PROCESSING
RESULT
Parameter Control List (PCL)
Contains Parameter Control Entries (PCEs) that Describe Parameters Expected.
LINK from TSO Command Processor
Parse Service Routine
IKJPARS
Scans Buffer for Valid Command Parameters
IKJPARM PCE
IKJIDENT or IKJPOSIT PCEs
IKJKEYWD PCE followed by IKJNAME PCEs
IKJKEYWD IKJIDENT or IKJPOSIT PCEs followed by IKJSUBF PCE
IKJENDP PCE
1 Set Up and Initialize
2 Scan for Positional Parameters
3 Scan for Keyword Parameters and
Scan for Subfields
4 Return to Caller
PDL Header Entry
IKJIDENT or IKJPOSIT PDEs
IKJKEYWD PDEs
IKJIDENT IKJPOSIT or IKJKEYWD PDEs
Parameter Descriptor List (PDL)
Contains Parameter Descriptor Entries (PDEs that Describe Parameters Found.
RETURN
Parse Parameter List (PPL)
Parameter Control List
Command Buffer
Command Buffer
length
offset
Positional Parameters, Keywords, Subfields

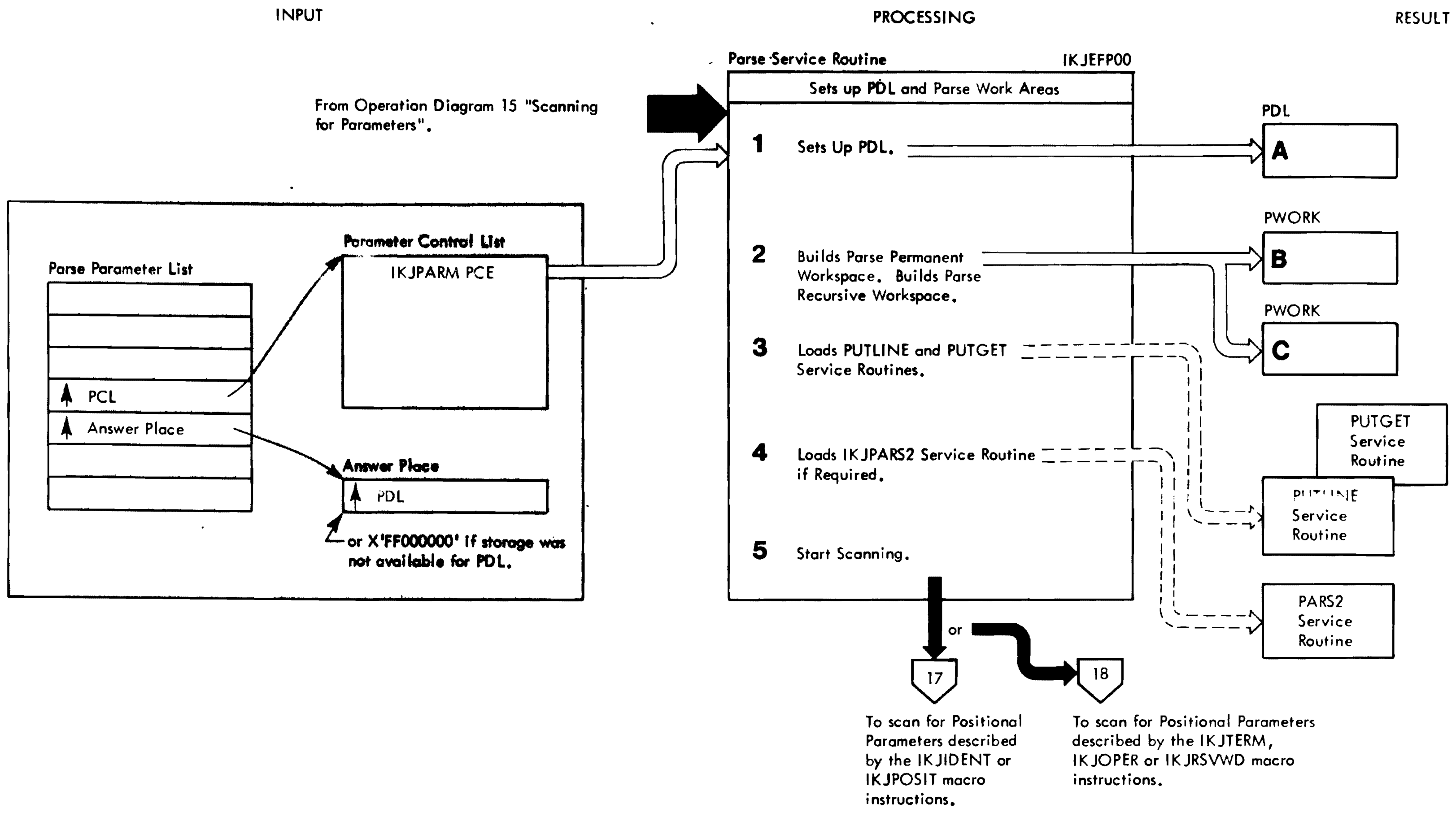
Note: Other input buffers are obtained when the terminal is prompted for additional input. See Operation Diagram 8 for a description of the Input Stack.

Method of Operation Diagram 15. Parse Service Routine (Part 1 of 2)

CROSS REFERENCE TABLE

Key	Description	Routine	Label	Diagram
·	Parse gets PCEs from the PCL and constructs PDEs in the PDL. The operations performed depend upon the type of PCE being processed.			
1	The IKJPARM PCE determines much of what is done during Parse initialization. It names the PCL and PDL (default name is IKJPARMD), gives the length of the PCL and PDL, and gives the offset in the PCL of the next IKJKEYWD, IKJSUBF, or IKJENDP PCE.	IKJEFP00	IKJPARS	16
2	The IKJIDENT and IKJPOSIT PCEs determine much of what is done during a scan for positional parameters. The parameters must appear in the command buffer in the same order that their PCEs appear in the PCL. All positional parameters must come before any keyword parameters.	IKJEFP01 IKJEFP00	IDENT POSIT	17
3	The IKJKEYWD and IKJNAME PCEs determine much of what is done during a scan for keyword parameters. The IKJKEYWD PCE marks the beginning of a keyword field while the following IKJNAME PCEs define eligible names for the keyword.	IKJEFP00	KEYWDP	19
	Keywords may have subfields that include both positional and keyword parameters. The IKJSUBF PCE marks the beginning of a subfield and the end of a previous field.			
4	The IKJENDP PCE marks the end of the PCL. Parse checks to see if the scan is complete before returning control to the calling routine.	IKJEFP00	ENDFIELD	19

Method of Operation Diagram 15. Parse Service Routine (Part 2 of 2)

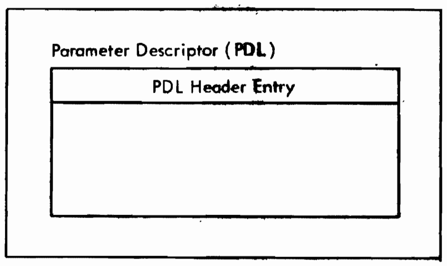

INPUT
PROCESSING
RESULT
From Operation Diagram 15 "Scanning for Parameters".
Parse Parameter List
PCL
Answer Place
Parameter Control List
IKJPARM PCE
Answer Place
PDL
or X'FF000000' if storage was not available for PDL.
Parse Service Routine
IKJEFP00
Sets up PDL and Parse Work Areas
1 Sets Up PDL.
2 Builds Parse Permanent Workspace. Builds Parse Recursive Workspace.
3 Loads PUTLINE and PUTGET Service Routines.
4 Loads IKJPARS2 Service Routine if Required.
5 Start Scanning.
PDL
A
PWORK
B
PWORK
C
PUTGET Service Routine
PUTLINE Service Routine
PARS2 Service Routine
or
17
18
To scan for Positional Parameters described by the IKJIDENT or IKJPOSIT macro instructions.
To scan for Positional Parameters described by the IKJTERM, IKJOPER or IKJRSVWD macro instructions.

Method of Operation Diagram 16. Parse Initialization (Part 1 of 2)

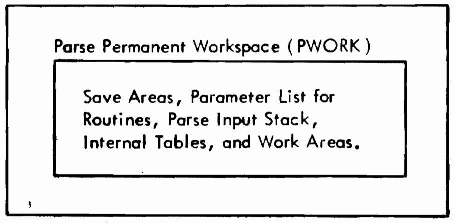
CROSS REFERENCE TABLE

Key	Description	Routine	Label
1	Parse gets main storage for the PDL from subpool 1. The DSECT for the PDL is named according to the value specified by the IKJPARM PCE. The default name is IKJPARMD.	IKJEFP00 IKJEFP02 IKJEFP00	IKJPARS STALOC
2	Storage is obtained for the Parse Work Area (PWORK) and first Recurse Work Area (RWORK) from subpool 0. Both work areas are initialized with information from the Parse Parameter List (PPL).	IKJEFP00 IKJEFP02	IKJPARS GETCORE
	• Additional Recurse Work Areas are obtained each time a subfield is processed.	IKJEFP00	RECURSE
3	The PUTLINE and PUTGET service routines are loaded.	IKJEFP00	IKJPARS
4	Parse loads IKJPARS2 Service Routine if the IKJTERM, IKJOPER or IKJRSVWD macro instruction is coded.	IKJEFP60	IKJPARS2
5	Parse is ready to get the next PCE and start the scan.	IKJEFP00	NEXTPCE

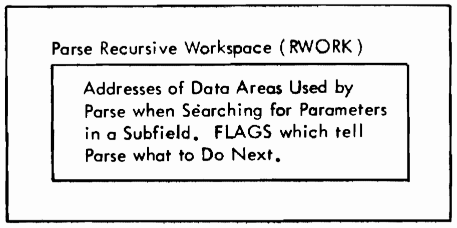
A



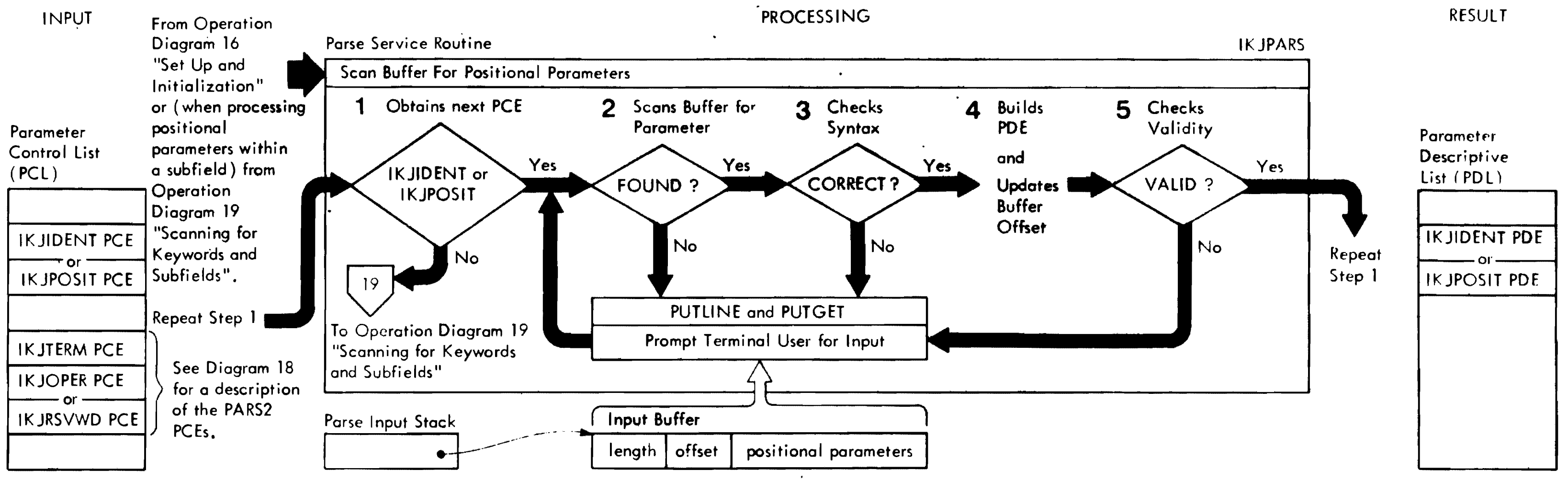
B


Parse Permanent Workspace (PWORK)
Save Areas, Parameter List for Routines, Parse Input Stack, Internal Tables, and Work Areas.

C


Parse Recursive Workspace (RWORK)
Addresses of Data Areas Used by Parse when Searching for Parameters in a Subfield. FLAGS which tell Parse what to Do Next.

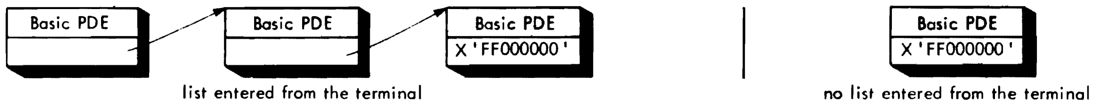
Method of Operation Diagram 16. Parse Initialization (Part 2 of 2)


INPUT
PROCESSING
RESULT
Parameter Control List (PCL)
IKJIDENT PCE
or
IKJPOSIT PCE
IKJTERM PCE
IKJOPER PCE
or
IKJRSVWD PCE
From Operation Diagram 16 "Set Up and Initialization" or (when processing positional parameters within a subfield) from Operation Diagram 19 "Scanning for Keywords and Subfields".
Repeat Step 1
See Diagram 18 for a description of the PARS2 PCEs.
Parse Service Routine
IKJPARS
Scan Buffer For Positional Parameters
1 Obtains next PCE
2 Scans Buffer for Parameter
3 Checks Syntax
4 Builds PDE and Updates Buffer Offset
5 Checks Validity
IKJIDENT or IKJPOSIT
Yes
No
FOUND ?
Yes
No
CORRECT ?
Yes
No
VALID ?
Yes
No
19
To Operation Diagram 19 "Scanning for Keywords and Subfields"
PUTLINE and PUTGET
Prompt Terminal User for Input
Repeat Step 1
Parse Input Stack
Input Buffer
length
offset
positional parameters
Parameter Descriptive List (PDL)
IKJIDENT PDE
or
IKJPOSIT PDE

LIST and RANGE Options

LIST OPTION

If the LIST option is specified, one word is added to the basic PDE. If a list is entered from the terminal, this word is used to chain the elements of the list together as shown below. The chain word for the last PDE is set to X'FF000000'. (If the list is not entered from the terminal, the chain word for the basic PDE will contain X'FF000000'.)


Basic PDE
Basic PDE
Basic PDE
X'FF000000'
list entered from the terminal
Basic PDE
X'FF000000'
no list entered from the terminal

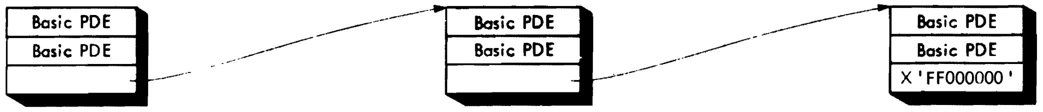
RANGE OPTION

If the RANGE option is specified, two identical PDEs are created. (If the range is not entered from the terminal, the second PDE contains zeroes.)

Basic PDE
Basic PDE
range entered from the terminal
Basic PDE
0
no range entered from the terminal

COMBINING THE LIST AND RANGE OPTIONS

If both the LIST and RANGE options are specified, two identical PDEs are created for each element in the list as shown below.



Method of Operation Diagram 17. Searching for IKJPARS Positional Parameters (Part 1 of 2)

CROSS - REFERENCE TABLE

Key Description

1 The IKJIDENT PCE describes a positional parameter in the form of a character string with optional restrictions on the first character, other characters, and length.
The IKJPOSIT PCE describes a positional parameter which includes a delimiter as part of its syntax. There are 10 kinds of IKJPOSIT PCEs and each has its own processing routine.

2 The buffer is scanned for the parameter.

- If the parameter is missing, the PCE is checked to see if it is required or if there is a default.
  - If the parameter is required, the terminal is prompted for the parameter.
  - If there is a default, the default is supplied.
  - If the parameter is not required, Parse gets the next PCE.

3 If the parameter is found (or defaulted) the parameter is checked for correct syntax.

- If an error is found, the user is prompted to reenter the parameter and step 2 is repeated.

4 If the parameter is correct, a PDE is built and the parameter is translated to upper case.
If a list is being processed, step 2 is repeated for each element in the list.

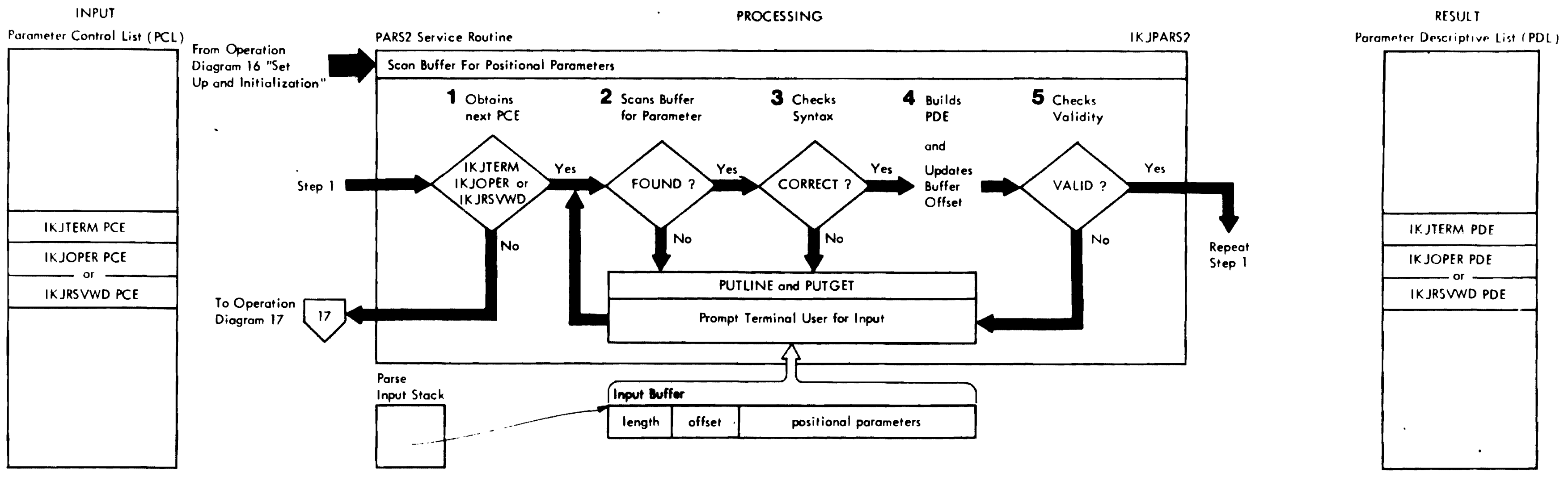
5 If the calling routine has specified a validity check exit, the validity check exit routine is entered. If a range is being processed, the complete range is passed to the exit routine. If a list is being processed, each element in the list is passed to the exit routine.

- If an error is found, the terminal is prompted to reenter the parameter and step 2 is repeated.

IKJIDENT PCEs		IKJPOSIT PCEs			
		DELIMITER		STRING	
Routine	Label	Routine	Label	Routine	Label
IKJEFP01	IDENT	IKJEFP00	DELIMITER	IKJEFP00	STRING
IKJEFP02	SKIPB	IKJEFP02	SKIPB	IKJEFP02	PROMPTQ
IKJEFP02	PROMPTQ	IKJEFP02	TYPETEST		
IKJEFP02	SCANF				
IKJEFP02	LISTT				
IKJEFP02	RANGE				
IKJEFP02	TYPETEST				
IKJEFP20	GENSCAN				
IKJEFP20	TRANSQ			IKJEFP20	TRANSQ
IKJEFP02	VCERTN				

IKJPOSIT PCEs													
VALUE		ADDRESS		PSTRING		USERID		DSNAME/DSTHING		QSTRING		SPACE	
Routine	Label	Routine	Label	Routine	Label	Routine	Label	Routine	Label	Routine	Label	Routine	Label
IKJEFP00	VALUE	IKJEFP02	ADDRESS	IKJEFP02	PSTRING	IKJEFP02	USERID	IKJEFP01	DSNAME	IKJEFP00	QSTRING	IKJEFP00	SPACE
IKJEFP02	SKIPB	IKJEFP02	SKIPB	IKJEFP02	SKIPB	IKJEFP02	SKIPB	IKJEFP02	SKIPB	IKJEFP00	SKIPB		
IKJEFP02	LISTT	IKJEFP02	LISTT	IKJEFP02	PROMPTQ	IKJEFP02	LISTT	IKJEFP02	LISTT	IKJEFP00	PROMPTQ		
IKJEFP02	TYPETEST	IKJEFP02	TYPETEST	IKJEFP02	GETCORE	IKJEFP02	TYPETEST	IKJEFP02	SCANF	IKJEFP00	GETCORE		
IKJEFP02	PROMPTQ	IKJEFP02	RANGE	IKJEFP02	WRITER1	IKJEFP02	SCANF	IKJEFP02	TYPETEST	IKJEFP00	WRITER1		
IKJEFP02	RANGE					IKJEFP02	GETCORE	IKJEFP02	PROMPTQ				
						IKJEFP02	PROMPTQ	IKJEFP02	GETCORE				
						IKJEFP02	WRITE2G	IKJEFP02	WRITER2G				
						IKJEFP20	GENSCAN	IKJEFP20	GENSCAN				
IKJEFP20	TRANSQ	IKJEFP02	TRANSQ	IKJEFP20	TRANSQ	IKJEFP20	TRANSQ	IKJEFP20	TRANSQ	IKJEFP20	TRANSQ		
IKJEFP02	VCERTN	IKJEFP02	VCERTN	IKJEFP02	VCERTN	IKJEFP02	VCERTN	IKJEFP02	VCERTN	IKJEFP00	VCERTN		

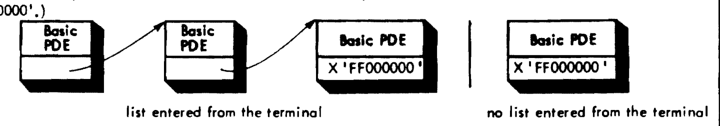
Method of Operation Diagram 17. Searching for IKJPARS Positional Parameters (Part 2 of 2)



LIST and RANGE Options

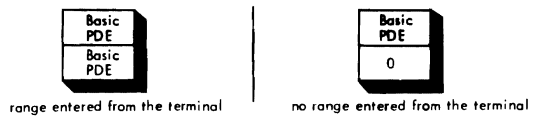
LIST OPTION

If the LIST option is specified, one word is added to the basic PDE. If a list is entered from the terminal, this word is used to chain the elements of the list together as shown below. The chain words for the last PDE is set to X'FF000000'. (If the list is not entered from the terminal, the chain word for the basic PDE will contain X'FF000000'.)


Basic PDE
Basic PDE
Basic PDE
X 'FF000000'
Basic PDE
X 'FF000000'
list entered from the terminal
no list entered from the terminal

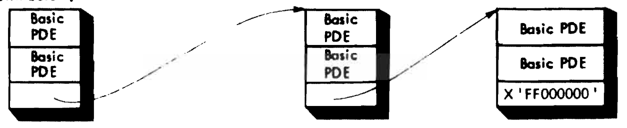
RANGE OPTION

If the RANGE option is specified, two identical PDEs are created. (If the range is not entered from the terminal, the second PDE contains zeroes.)

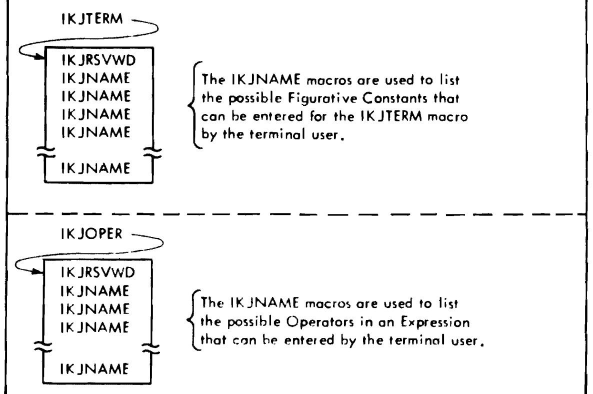

Basic PDE
Basic PDE
range entered from the terminal
Basic PDE
0
no range entered from the terminal

COMBINING THE LIST AND RANGE OPTIONS

If both the LIST and RANGE options are specified, two identical PDEs are created for each element in the list as shown below.



Note 1: The IKJRSVWD macro instruction can also be chained to the IKJTERM macro and the IKJOPER macro to define the beginning of a list of reserved words.


IKJTERM
IKJRSVWD
IKJNAME
IKJNAME
IKJNAME
IKJNAME
IKJNAME
The IKJNAME macros are used to list the possible Figurative Constants that can be entered for the IKJTERM macro by the terminal user.
IKJOPER
IKJRSVWD
IKJNAME
IKJNAME
IKJNAME
IKJNAME
The IKJNAME macros are used to list the possible Operators in an Expression that can be entered by the terminal user.

Method of Operation Diagram 18. Searching for IKJPARS2 Positional Parameters (Part 1 of 2)

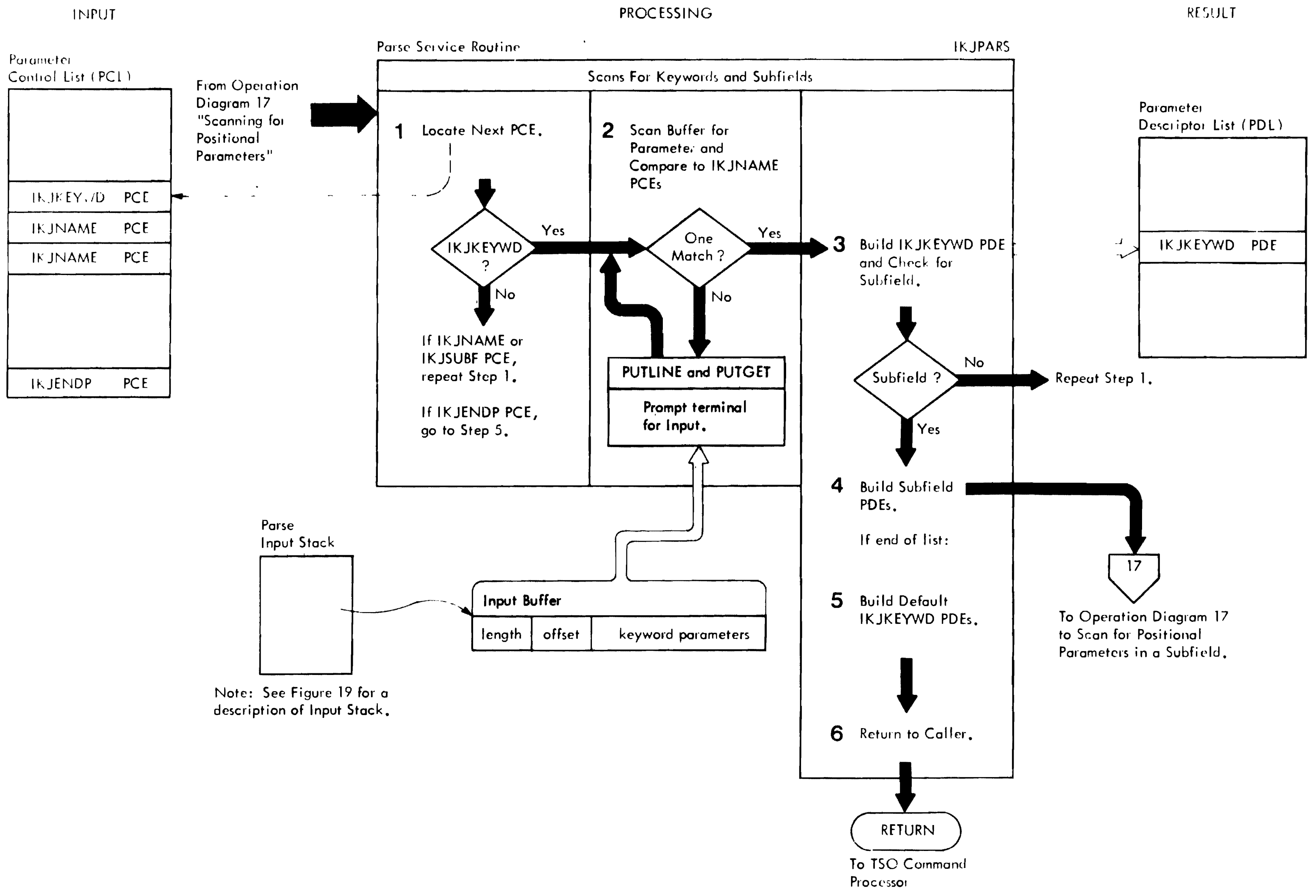
CROSS-REFERENCE TABLE

Key Description
1 • The IKJTERM PCE describes a positional parameter that may be entered as a Constant, Variable or Statement Number. • The IKJOPER PCE describes a positional parameter that may be entered as an EXPRESSION. • The IKJRSVWD PCE describes a positional parameter that may be entered as a Reserved Word. (Also see note 1).
2 The buffer is scanned for the parameter. • If the parameter is missing, the PCE is checked to see if it is required or if there is a default. – If the parameter is required, the terminal is prompted for the parameter. – If there is a default, the default is supplied. – If the parameter is not required, Parse gets the next PCE.
3 If the parameter is found (or defaulted) the parameter is checked for correct syntax. • If an error is found, the user is prompted to reenter the parameter and step 2 is repeated.
4 If the parameter is correct, a PDE is built and the parameter is translated to upper case. If a list is being processed, step 2 is repeated for each element in the list.
5 If the calling routine has specified a validity check exit, the validity check exit routine is entered. If a range is being processed, the complete range is passed to the exit routine. If a list is being processed, each element in the list is passed to the exit routine. • If an error is found, the terminal is prompted to reenter the parameter and step 2 is repeated.

CROSS-REFERENCE TABLE

IKJPARS Routines Used by IKJPARS2		Flowchart
CLEANUP	- Free core, delete modules, exit	EB
GENSCAN	- Parameter scan routine	EC
GETCORE	- Obtain storage (release before exit)	DU
LISTT	- Test for LIST entered	DT
NAMESKP3	- Skip to next PCE routine	CE
NEXTPCE	- Go to next PCE routine	CA
PARS2ENT	- Entry into IKJPARS from IKJPARS2 when subroutine functions are required	
POSITX	- Add PDE to PDL routine	EA
PROMPTQ	- Prompt with 'ENTER...' routine	DP
PSTRIMSG	- Ending parenthesis assumed message	CW
PUSHI	- Push the stack routine	DS
QSTRING	- Quoted string routine	DF
RANGE	- Test for a RANGE entered	DT
SCANF	- Pop the stack routine	DQ
SKIPB	- Skip blanks routine	DS
STALOC	- Allocate storage in Subpool 1	DR
SYSR1	- Write 'INVALID' message, then prompt	EA
TRANSQ	- Translate to uppercase routine	EE
TYPETEST	- Test for character type routine	DU

Method of Operation Diagram 18. Searching for IKJPARS2 Positional Parameters (Part 2 of 2)


INPUT
PROCESSING
RESULT
Parameter Control List (PCL)
IKJKEYWD PCE
IKJNAME PCE
IKJNAME PCE
IKJENDP PCE
From Operation Diagram 17 "Scanning for Positional Parameters"
Parse Service Routine
IKJPARS
Scans For Keywords and Subfields
1 Locate Next PCE.
IKJKEYWD ?
Yes
No
If IKJNAME or IKJSUBF PCE, repeat Step 1.
If IKJENDP PCE, go to Step 5.
2 Scan Buffer for Parameter and Compare to IKJNAME PCEs
One Match ?
Yes
No
PUTLINE and PUTGET
Prompt terminal for Input.
3 Build IKJKEYWD PDE and Check for Subfield.
Subfield ?
No
Yes
Repeat Step 1.
4 Build Subfield PDEs.
If end of list:
5 Build Default IKJKEYWD PDEs.
6 Return to Caller.
RETURN
To TSO Command Processor
Parameter Descriptor List (PDL)
IKJKEYWD PDE
17
To Operation Diagram 17 to Scan for Positional Parameters in a Subfield.
Parse Input Stack
Input Buffer
length
offset
keyword parameters
Note: See Figure 19 for a description of Input Stack.
Method of Operation Diagram 19. Searching for Keyword Parameters and Subfields

Method of Operation Diagram 19. Searching for Keyword Parameters and Subfields (Part 1 of 2)

CROSS REFERENCE TABLE

Key Description	Routine	Label
1 The IKJKEYWD PCE marks the beginning of a keyword field and specifies options for that field.	IKJEFP00	KEYWDP
2 The buffer is scanned for a keyword and the result is compared to the names specified by the IKJNAME PCEs.	IKJEFP02	KEYWD
3 If a match is found, the IKJKEYWD PDE is built and the IKJNAME PCE is checked for a subfield.	IKJEFP02	KEYWDNAM
• If no match is found, the terminal is prompted to reenter the parameter.	IKJEFP02	PROMPTQ
• If more than one match is found, all previous PDEs built for the keyword are erased, and the terminal is prompted to reenter the parameter. See "Erasing PDEs" in Section 1.	IKJEFP00 IKJEFP02	KEYWDER4 PROMPTQ
4 If the keyword name has a subfield, Parse interrupts the scan for the keyword field and processes the subfield in exactly the same way that it would process a field. When an IKJSUBF or IKJENDP PCE is reached, Parse resumes the scan for the keyword field.	IKJEFP02 IKJEFP00 IKJEFP00	KEYWDSUB ENDFIELD KEYWDP
5 When all of the keyword fields have been scanned, Parse checks each IKJKEYWD PCE to see if the corresponding PDE has been built.	IKJEFP00	KEYWDP
• If not, a default value is supplied. (Keyword parameters are never required.)	IKJEFP02	PROMPTQ
6 When an IKJENDP PCE is reached, Parse checks to see if the scan is complete.	IKJEFP00	ENDFIELD
• If the scan has reached the end of the buffer, Parse returns control to the calling routine.	IKJEFP00	ENDX1
• If the scan has not reached the end of the buffer, Parse writes an "extraneous data" message, or an "invalid keyword" message if there are keyword PCEs in the PCL, and returns control to the calling routine.	IKJEFP00	ENDX2

Method of Operation Diagram 19. Searching for Keyword Parameters and Subfields (Part 2 of 2)

Section 3: Program Organization

This section describes the organization of command scan and parse. It contains information about the hierarchy of the load modules, the assembly modules, and the control sections that constitute each program. Figure 20 is a graphic representation of this hierarchy.

The module operation information briefly describes the processing operations that occur within each parse and command scan module.

For a summary of the functions performed by each subroutine (routines below the control section level) refer to the Directory in Section 4.

Program Hierarchy

Command scan has two assembly modules while parse has three assembly modules as shown in Figure 20. One assembly module (IKJEFP20) is common to both parse and command scan so that the two programs, when combined, have only four assembly modules. They are:

IKJEFP00 -- Parse. This module has three control sections: IKJEFP00, IKJEFP01, and IKJEFP02.

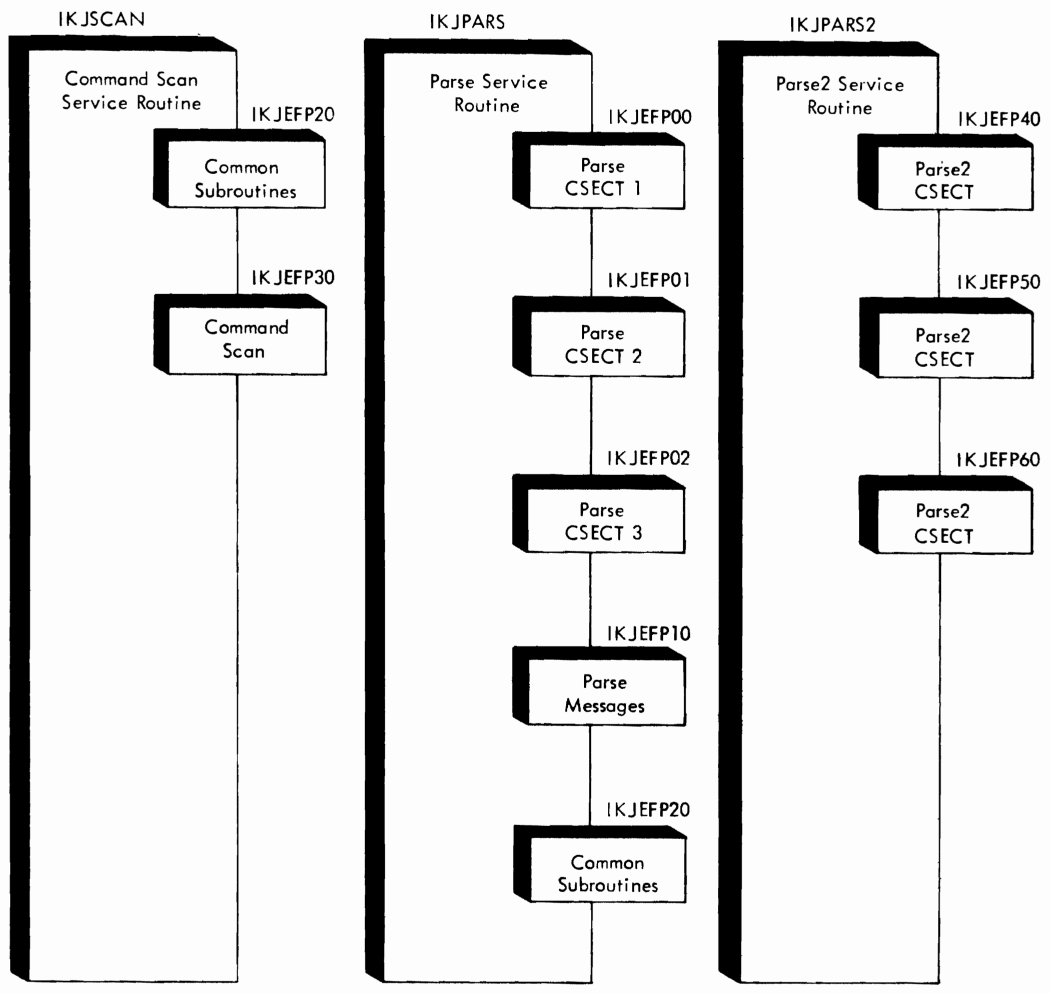
IKJEFP10 -- Parse messages

IKJEFP20 -- Common subroutines

IKJEFP30 -- Command scan

IKJPARS2 is an assembly module that is loaded by IKJPARS when an IKJTERM, IKJOPER or IKJRSVWD macro instruction is coded. IKJPARS2 includes three control sections; IKJEFP40, IKJEFP50 and IKJEFP60.

Figure 21 shows the linkage relationship between IKJPARS and IKJPARS2.



Load Module Names	Normal Residence	Approximate Sizes
IKJPARS	SYS1.LINKLIB	12K bytes
IKJSCAN	SYS1.LINKLIB	1.5K bytes
IKJPARS2	SYS1.LINKLIB	8K bytes

Note: IKJPARS2 is loaded by IKJPARS when an IKJTERM, IKJOPER or IKJRSVWD macro instruction is coded.

Figure 20. Program Hierarchy: Command Scan and Parse Service Routines

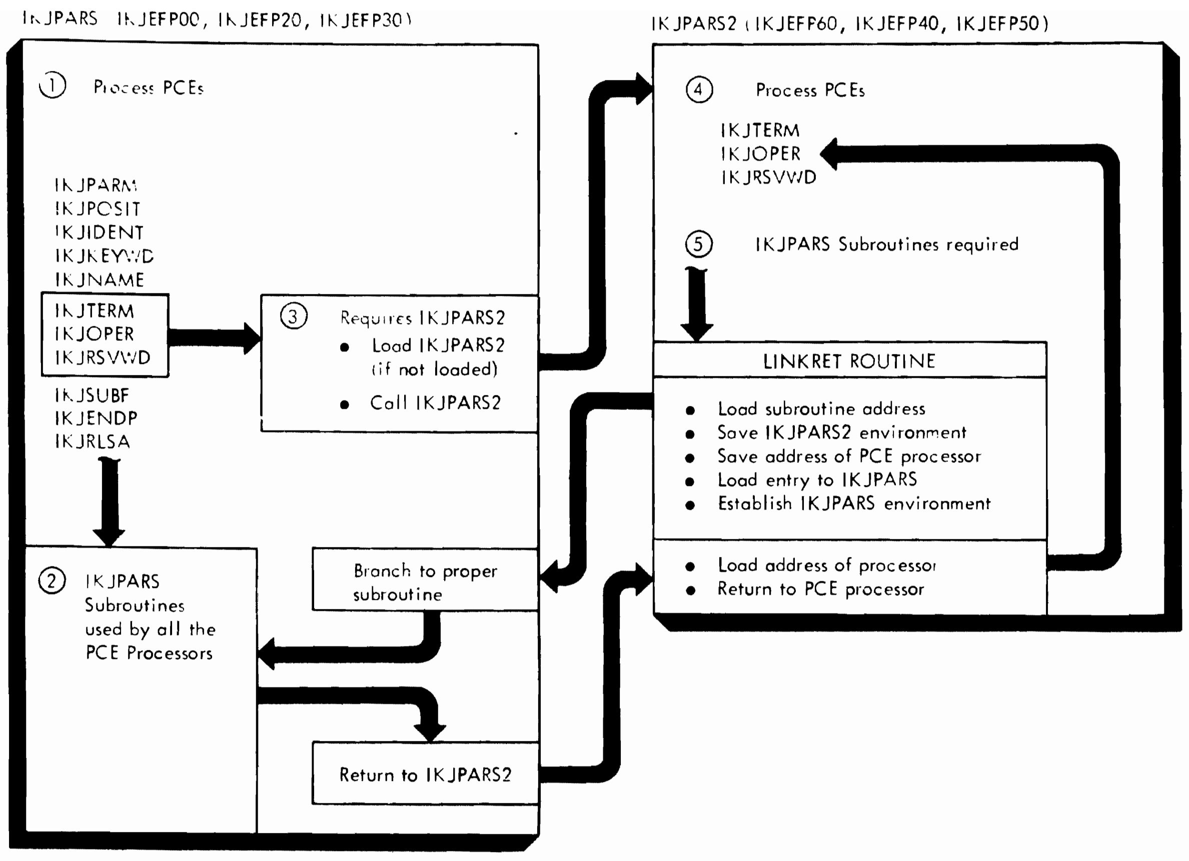

IKJPARS (IKJEFP00, IKJEFP20, IKJEFP30)
1 Process PCEs
IKJPARM
IKJPOSIT
IKJIDENT
IKJKEYWD
IKJNAME
IKJTERM
IKJOPER
IKJRSVWD
IKJSUBF
IKJENDP
IKJRLSA
3 Requires IKJPARS2
Load IKJPARS2 (if not loaded)
Call IKJPARS2
2 IKJPARS Subroutines used by all the PCE Processors
Branch to proper subroutine
Return to IKJPARS2
IKJPARS2 (IKJEFP60, IKJEFP40, IKJEFP50)
4 Process PCEs
IKJTERM
IKJOPER
IKJRSVWD
5 IKJPARS Subroutines required
LINKRET ROUTINE
Load subroutine address
Save IKJPARS2 environment
Save address of PCE processor
Load entry to IKJPARS
Establish IKJPARS environment
Load address of processor
Return to PCE processor

Figure 21. Linkage Between IKJPARS and IKJPARS2

Module Operation

The following descriptions briefly describe the processing operations in each executable module of the parse and command scan service routines.

IKJEFP00 -- PARSE SERVICE ROUTINE

This module has three control sections:

IKJEFP00
IKJEFP01
IKJEFP02

Scans buffer for command parameters, checks their syntax, optionally translates them to uppercase, optionally checks their validity, builds a PDL consisting of PDEs that describe the parameter found, returns to calling program.

IKJEFP20 -- COMMON MODULE (USED BY BOTH PARSE AND COMMAND SCAN)

Consists of the following routines:

GETSCAN is a generalized scan routine that checks the syntax of the command buffer parameter according to control information set up by the invoker of command scan or parse.

TRANSQ is a translate routine that translates lowercase alphabetic characters to uppercase.

TRANSX - Translates lowercase alphabetic characters to uppercase, if the parameter is known not to be defaulted.

IKJEFP30 -- COMMAND SCAN SERVICE ROUTINE

Searches the buffer for a valid command name, optionally checks the syntax of the command name, translates the command name to uppercase, indicates whether parameters follow the command name, updates buffer offset, returns to calling program.

IKJEFP60 -- IKJPARS2 SERVICE ROUTINE

1. Entry point IKJEFP40 provides syntax checking for Reserved Word parameters specified on the IKJRSVWD macro instruction.
2. Entry point IKJEFP50 provides syntax checking for Expression parameters, specified on the IKJOPER macro instruction.
3. Entry point IKJEFP60 provides syntax checking for Constant, Variable or Statement Number parameters specified on the IKJTERM macro instruction.

Section 4: Directory

This table contains information to help you find the appropriate module operation description or assembly listing. It correlates information from three sources:

- The source code.
- The executable load modules.
- This manual.

Label	Common Name	Load Module Name	Assembly Module Name	Control Section Name	Description	Diagram
ADDRESS	Positional Address	IKJPARS	IKJEFP00	IKJEFP01	Scans buffer for address - Prompts for missing data. Identifies address type. Builds temporary PDE. Checks for possible list. Branches to ILLADR if illegal address. Otherwise returns to NEXTPCE.	17
BUMP	Bump the Input Stack	IKJPARS2	----	IKJEFP60	Test for more data. Take off the last element put on the push-down stack, and continue processing, if more data. Return +0 on no more data, +4 on more data.	18
CLEANUP	Cleanup	IKJPARS	IKJEFP00	IKJEFP02	Frees all storage obtained by parse when terminal error occurs. Places X'FF000000' in answer place.	16,17 18
DELIMITR	Positional Delimiter	IKJPARS	IKJEFP00	IKJEFP00	Scans buffer for next self-defining delimiter and sets switch if invalid.	17
DSNAME	Positional Data Set Name	IKJPARS	IKJEFP00	IKJEFP01	Scans buffer for DSNAME or DSTHING. Builds temporary PDE.	17
ENDFIELD	End-of-Field Processing Routine	IKJPARS	IKJEFP00	IKJEFP00	Entered at end of field -- Checks for extraneous data in buffer of subfield and writes error message. Entered when erasing PDEs -- Releases current RWORK and gets next RWORK. Entered at end of subfields -- all functions executed for other entries.	19
EXIT	Final Exit	IKJPARS	IKJEFP00	IKJEFP00	Deletes PUTLINE and PUTGET. Frees PWORK and RWORK. Updates buffer offset to point one character past the last character scanned.	19

(Continued)

Label	Command Name	Load Module Name	Assembly Module Name	Control Section Name	Description	Diagram
FREECORE	Issue Freemain	IKJPARS2	----	IKJEFP40	Issues freemain to release storage. Returns to caller.	18
GENSCAN	General Scan	IKJPARS and IKJSCAN	IKJEFP20	IKJEFP20	Scans buffer for character string.	14,15
GETCORE	Get Core	IKJPARS	IKJEFP00	IKJEFP02	Gets more storage.	17,19
IDENT		IKJPARS	IKJEFP00	IKJEFP01	Scans for buffer positional parameter.	17
IKJEFP40	IKJRSVWD Processing	IKJPARS2	---	IKJEFP40	Processes the Reserved Word parameters.	18
IKJEFP50	IKJOPER Processing	IKJPARS2	---	IKJEFP50	Processes the expression parameter.	18
IKJEFP60	IKJTERM Processing	IKJPARS2	---	IKJEFP60	Processes a constant, Variable, or statement number parameter.	18
IKJPARS	Parse Service Routine	IKJPARS	IKJEFP00	IKJEFP00	Gets main storage for PWORK and RWORK. Loads PUTLINE and PUTGET.	16
IKJPARS2	Parse2 Service Routine	IKJPARS2	---	IKJEFP50	Processes an IKJTERM, IKJOPER, or IKJRSVWD macro Instruction when loaded by IKJPARS.	18
IKJSCAN	Command Scan	IKJSCAN	IKJEFP30	IKJEFP30	Scans buffer for command name.	14
KEYWD	Keyword Scan	IKJPARS	IKJEFP00	IKJEFP02	Scans buffer for keywords and subfields.	19
KEYWDP	Keyword Processor	IKJPARS	IKJEFP00	IKJEFP00	Entered before keyword field is scanned--Sets switch to indicate keyword scan and gives control to keyword scan routine. Entered after keyword field is scanned--Checks to see if PDE was built. It not, gets default and gives control to keyword scan. If so, gives control to Main Control. Entered when erasing PDEs--Zeros the keyword PDE, calculates the next PCE address.	19
LISTT	List	IKJPARS	IKJEFP00	IKJEFP02	Checks for a list and if one was entered. Gets address of first element.	17

(Continued)

Label	Command Name	Load Module Name	Assembly Module Name	Control Section Name	Description	Diagram
MSNGMSG	Missing Message	IKJPARS	IKJEFP00	IKJEFP02	Issues "MISSING" message when an "ENTER" or "ENTER PASSWORD" prompt message was attempted during no-prompt mode. If HELP messages were associated with the prompt, "MISSING" replaces "ENTER" on the HELP messages.	19
NAMESKP	Name Skip	IKJPARS	IKJEFP00	IKJEFP00	Entered after keyword field scan--Skips IKJNAME PCEs. When erasing IKJKEYWD PDEs--Checks IKJNAME PCE for subfield. If found, returns to RECURSE. If not found, returns to NEXTPCE.	19
NEXTPCE	PCE Locating Routine	IKJPARS	IKJEFP00	IKJEFP00	Locates the next PCE and branches to the appropriate PCE processing routine.	16
POSIT	Positional Processor	IKJPARS	IKJEFP00	IKJEFP00	Gives control to appropriate 2nd level positional parameter processor.	17
POSITERS	Positional PDE Erase	IKJPARS	IKJEFP00	IKJEFP01	Erases a PDE for a positional parameter.	17
POSITX	Positional Exit	IKJPARS	IKJEFP00	IKJEFP01	Checks for and processes a range. Checks for and processes a list. Translates to uppercase if string, pstring, gstring, value, or ident.	17
PROMPTQ	Prompt Default	IKJPARS	IKJEFP00	IKJEFP02	If prompt was specified--Prompts terminal. Saves address of command buffer. Saves message ID. If default was specified--Supplies default value.	19
PSTRIMSG	Parenthesis Assumed Message	IKJPARS	IKJEFP00	IKJEFP00	If a right parenthesis is not found, writes a message indicating it is assumed.	18
PSTRING	Positional Parenthe-sized String	IKJPARS	IKJEFP00	IKJEFP01	Scans buffer for parenthesized string-checks for embedded parentheses.	17
PUSHI	Push Input Stack	IKJPARS	IKJEFP00	IKJEFP02	Saves current buffer pointer and end-of-buffer pointer on input stack. Gets more storage when stack is full.	17,19

(Continued)

Label	Command Name	Load Module Name	Assembly Module Name	Control Section Name	Description	Diagram
QSTRING	Positional Quoted Stirng	IKJPARS	IKJEFP00	IKJEFP01	Scans buffer to next TAB or non-separator character.	17
RANGE	Range	IKJPARS	IKJEFP00	IKJEF02	Checks to see if a range is possible.	17
RECURSE	Subfield Processing Routine	IKJPARS	IKJEFP00	IKJEFP00	Gets main storage for RWORK.	16
SCANF	Pop the Stack	IKJPARS	IKJEFP00	IKJEFP00	Checks for end of stack. Removes last element put on the push-down stack. Issues freemain for previous stack. Returns to caller.	18
SCANPOP	Pop Input Stack	IKJPARS	IKJEFP00	IKJEFP02	Gets last element from input stack.	17,19
SKIPB	Skip Separators	IKJPARS	IKJEFP00	IKJEFP02	Updates buffer offset to first non-separator character.	19
STALOC	Storage Allocation	IKJPARS	IKJEFP00	IKJEFP02	Allocates main storage for PDL and prompt data from storage obtained by GETCORE.	16
STRING	Positional String	IKJPARS	IKJEFP00	IKJEFP00	Prompts for data if switch is set by delimiter routine. Scans buffer for character string.	17
SYSR1	Error Handling	IKJPARS	IKJEFP00	IKJEFP02	Calculates length of invalid data. Builds informational message segments.	19
TERMOCK	Check TERM PCE	IKJPARS2	---	IKJEFP40	Checks and test the IKJTERM PCE.	18
TRANSQ	Translate	IKJPARS and IKJSCAN	IKJEFP20	IKJEF20	Translates lowercase Alphabetic characters to uppercase.	14,15
TYPETEST	Character Type Test	IKJPARS	IKJEFP00	IKJEFP02	Tests current character against selected mask.	17,19
USERID	Positional Userid	IKJPARS	IKJEFP00	IKJEFP01	Scans buffer for userid/password.	17
VALUE	Positional Vaule	IKJPARS	IKJEFP00	IKJEFP00	Check type character.	17
VCERTN	Validity Check Exit	IKJPARS	IKJEFP00	IKJEFP02	Sets up linkage to user-supplied validity check exit routine.	17,19

(Continued)

3

Label	Command Name	Load Module Name	Assembly Module Name	Control Section Name	Description	Diagram
WRITER1	Informa-tional messages	IKJPARS	IKJEFP00	IKJEFP02	Writes informational messages.	17,19
WRITER2	Prompt Messages	IKJPARS	IKJEFP00	IKJEFP02	Prompts terminal for input if necessary or checks for default. Takes any new data and places it in storage allocated by STALOC.	

Section 5: Data Areas

This section describes the major data areas used by command scan and parse:

- Command scan parameter list (CSPL)
- Command scan output area (CSOA)
- Parameter control entry for IKJENDP macro instruction
- Parameter control entry for IKJIDENT macro instruction
- Parameter control entry for IKJKEYWD macro instruction
- Parameter control entry for IKJNAME macro instruction
- Parameter control entry for IKJPARM macro instruction
- Parameter control entry for IKJPOSIT macro instruction (All except string, pstring, and qstring)
- Parameter control entry for IKJPOSIT macro instruction (string, pstring, qstring)
- Parameter control entry for IKJTERM macro instruction
- Parameter control entry for IKJOPER macro instruction
- Parameter control entry for IKJRSVWD macro instruction
- Parameter control entry for IKJSUBF Macro Instruction
- Parameter descriptor entry for IKJIDENT macro instruction
- Parameter descriptor entry for IKJKEYWD macro instruction
- Parameter descriptor entry for IKJPARM macro instruction
- Parameter descriptor entry for IKJPOSIT macro instruction (address)
- Parameter descriptor entry for IKJPOSIT macro instruction (dsname, dsthing)
- Parameter descriptor entry for IKJPOSIT macro instruction (Expression/value)
- Parameter descriptor entry for IKJPOSIT macro instruction (userid)
- Parameter descriptor entry for IKJPOSIT macro instruction (value)
- Parameter descriptor entry for IKJTERM macro instruction (Constant)
- Parameter descriptor entry for IKJTERM macro instruction (Variable)
- Parameter descriptor entry for IKJTERM macro instruction (Variable, data name qualifier)
- Parameter descriptor entry for IKJTERM macro instruction (Statement Number)
- Parameter descriptor entry for IKJRSVWD macro (Reserved word)
- Parameter descriptor entry for IKJOPER macro (expression)
- Parse parameter list (PPL)
- Parse permanent workspace (PWORK)
- Parse recursive workspace (RWORK)
- Syntax checking mask area
- Validity check parameter list (VCEPARM)

The following information is included for each data area:

- Size, in bytes.
- Name(s) of the routine(s) that creates it.
- Name(s) of the routine(s) that update and/or reference it.
- Field names, displacements, size, and contents.
- Cross-references to method of operation diagrams.

3

COMMAND SCAN PARAMETER LIST (CSPL)

Size:	24 bytes.
Constructed by:	Terminal monitor program or any command processor using command scan.
Located in:	Subpool 78.
Updated by:	None.
Used by:	Command scan.
Contents:	Parameter list.

Operation Diagrams
14

Displacement Dec.	Displacement Hex.	Field Name	Size in Bytes	Contents
0	0	CSPLUPT	4	↑ User profile table (UPT).
4	4	CSPLECT	4	↑ Environment control table (ECT).
8	8	CSPLECB	4	↑ Event control block (ECB).
12	C	CSPLFLG	4	↑ Flag word set as follows:
				X'00' - syntax check command name.
				X'80' - do not syntax check command name.
16	10	CSPLOA	4	↑ Command scan output area (CSOA). (set by IKJEFP30)
20	14	CSPLCBUF	4	↑ Command buffer (CBUF).

COMMAND SCAN OUTPUT AREA (CSOA)

Size:	8 bytes.
Constructed by:	Calling routine
Located in:	Subpool 1.
Updated by:	Command scan.
Used by:	Terminal monitor program or command processors using command scan.
Contents:	Indicates the results of a scan for command name.

Operation Diagrams
14

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	CSOACNM	4	† command name (zero if invalid).
4	4	CSOALNM	2	Length of command name.
6	6	CSOAFLG	1	Flags set as follows:
		CSOAVWP		X'80' - valid with parameters.
		CSOAVNP		X'40' - valid, no parameters.
		CSOAOM		X'20' - questionmark.
		CSOANOC		X'10' - no command name.
		CSOABAD		X'08' - invalid command name
7	7	--------	1	Reserved (0).

3

COMMAND SCAN WORKSPACE (CSWORK)

Size:	91 bytes.
Constructed by:	Command scan.
Located in:	Subpool 0.
Updated by:	Command scan.
Used by:	Command scan.
Contents:	Register save area, internal storage.

Operation Diagrams
14

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	CSWORK	72	18-word register save area.
72	48	PDWORD	8	Scratch/save/convert area.
80	50	ENDINPUT	4	Last input character address used to determine end of data.
84	54	PPOINTER	4	Address of first character scanned.
90	5A	RETCODE	1	Return code.

Note: The DSECT for CSWORK is generated by the IKJEFPWA macro instruction, which also generates the DSECT for PWORK (Parse permanent work area).

PARAMETER CONTROL ENTRY FOR IKJENDP MACRO INSTRUCTION

Size:	1 Byte.
Located in:	Subpool 1.
Created by:	Command processor using IKJENDP macro instruction.
Updated by:	None.
Used by:	Parse.
Contents:	This PCE ends the PCL.

Operation Diagrams
19

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
1	0		1	Flags: <u>Bit</u> <u>Meaning</u> 0-2 B'000' -- end-of-field PCE. 3-7 Reserved (0).

3

PARAMETER CONTROL ENTRY FOR IKJIDENT MACRO INSTRUCTION

Size:	Variable.
Located in:	Subpool 1.
Created by:	Command processor using IKJIDENT macro instruction.
Updated by:	None.
Used by:	Parse.
Contents:	This PCE describes a positional parameter in the form of a character string with optional restrictions on the beginning character, additional characters, and length.

Operation Diagrams
17

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0		2	Flags: __Bit__ __Meaning when on__ 0-2 B'100' -- IKJPOSIT PCE 3 PROMPT 4 DEFAULT 5 Reserved (0) 6 HELP 7 VALIDCK 8 LIST 9 ASIS 10 RANGE 11-15 Reserved(0)
2	2		2	Length of PCE.
4	4		2	Offset in PDL at which PDE is found.
6	6		1	IDENT options specified. __Bit__ __Meaning when on__ 0 ASTERISK 1 MAXLNTH 2 PTBYPS 3-7 Reserved (0)
7	7		1	First Character. Hex __Number__ __Meaning__ 0 Any character accepted. 1 Alpha required. 2 Numeric required. 3 Alphameric required. 4-FF Not used.

(Continued)

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
8	8		1	Other Characters. Hex Number Meaning 0 Any character accepted. 1 Alpha required. 2 Numeric required. 3 Alphameric required. 4-FF Not used.
9	9		2	Length of parameter type including this and following field.
11	B		2	X'0012' -- message segment offset.
--	--		N*	Parameter type.
--	--		1**	Maximum length (MAXLNTH).
--	--		1**	Length -1 of default or prompt information.
--	--		N*	Default or prompt information.
--	--		2**	Length of total help data. Including this field and the following field.
--	--		1**	Number of HELP messages.
--	--		2**	Length of help information. Including this and the following two fields.***
--	--		2**	X'000' message segment offset.***
--	--		3**	Address of validity checking routine.
--	--		N*	HELP information.**
				*Optional field of length "N" bytes. **Optional field. ***These three fields are reproduced for each level of help information.

3

PARAMETER CONTROL ENTRY FOR IKJKEYWD MACRO INSTRUCTION

Size:	Variable.
Located in:	Subpool 1.
Created by:	Command processor using IKJEKYWD macro instruction.
Updated by:	None.
Used by:	Parse.
Contents:	This PCE begins a description of a keyword field. The eligible names for this keyword field are in the PCE's generated by subsequent IKJNAME macros. This PCE specifies the options for the keyword field.

Operation Diagrams
19

Displacement Dec.	Displacement Hex.	Field Name	Size in Bytes	Contents
0	0		2	Flags: <u>Bits</u> <u>Meaning when on</u> 0-2 B'010' -- IKJKEYWD PCE 3 Reserved (0) 4 DEFAULT 5-15 Reserved (0)
2	2		2	Length of PCE.
4	4		2	Offset in PDL at which PDE is found.
6	6		1*	Length -1 of default information.
7	7		N**	Default information *Optional Field. **Optional field of "N" length bytes.

PARAMETER CONTROL ENTRY FOR IKJNAME MACRO INSTRUCTION

Size:	Variable.
Located in:	Subpool 1.
Created by:	Command processor using IKJNAME macro instruction.
Updated by:	None.
Used by:	Parse.
Contents:	This PCE describes one of the eligible names for a keyword field and specifies the options associated with this name.

Operation Diagrams
19

Displacement Dec.	Displacement Hex.	Field Name	Size in Bytes	Contents
0	0		2	Flags: Bits / Meaning when on 0-2 B'011' -- IKJNAME PCE 3-4 Reserved (0) 5 SUBFLD 6-10 Reserved (0) 11 INSERT 12-15 Reserved (0)
2	2		2	Length of PCE.
4	4		1	Length -1 of the name specified.
5	5		N*	Name of acceptable keyword.
--	--		2**	Offset in PCL to subfield PCE +1.
--	--		1**	Length -1 of keyword string to be inserted.
--	--		N*	Keyword string to be inserted. *Optional field of "N" length bytes. **Optional field.

PARAMETER CONTROL ENTRY FOR IKJPARM MACRO INSTRUCTION

Size:	6 bytes.
Located in:	Subpool 1.
Created by:	Command processor using IKJPARM macro instruction.
Updated by:	None.
Used by:	Parse
Contents:	This PCE is at the beginning of the PCL.

Operation Diagrams
16

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0		2	Length of PCL.
2	2		2	Length of PDL.
4	4		2	Offset in PCL to next IKJKEYWD, IKJSUBF, or IKJENDP PCE.

PARAMETER CONTROL ENTRY FOR IKJPOSIT MACRO INSTRUCTION (ALL EXCEPT STRING, PSTRING, AND QSTRING)

Size:	Variable.
Located in:	Subpool 1.
Created by:	Command processor using IKJPOSIT macro instruction.
Used by:	Parse.
Contents:	This PCE describes a positional parameter which includes a delimiter as part of its syntax.

Operation Diagrams
17

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0		2	Flags: Bits / Meaning when on 0-2 B'001' -- IKJPOSIT PCE 3 PROMPT 4 DEFAULT 5 Reserved (0) 6 HELP 7 VALIDCK 8 LIST 9 ASIS 10 RANGE 11 Reserved (0) 12 SQSTRING 13-15 Reserved (0)
2	2		2	Length of PCE.
4	4		2	Offset in PDL at which PDE is found.
6	6		1	Type of positional parameter: Hex Number / Meaning 1 DELIMITER 2 STRING 3 VALUE 4 ADDRESS 5 PSTRING 6 USERID 7 DSNAME 8 DSTHING 9 QSTRING A SPACE B-FF Not used
7	7		1*	Length -1 of default or prompt information.
8	8		N**	Default or prompt information.

(Continued)

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
--	--		2*	Length of total help data including this field and the following field.
--	--		1*	Number of help messages.
--	--		2*	Length of help information including this and the following two fields.***
--	--		2*	X'000' message segment offset.***
--	--		N**	Help information.***
--	--		3*	Address of validity checking routine.***
				*Optional field. **Optional field of length "N" bytes. ***These three fields are reproduced for each level of help information.

PARAMETER CONTROL ENTRY FOR IKJPOSIT MACRO INSTRUCTION (STRING, PSTRING, QSTRING)

Size:	8 Bytes.
Located in:	Subpool 1.
Created by:	Command Processor using IKJPOSIT macro instruction.
Updated by:	None.
Used by:	Parse.
Contents:	This PCE describes a positional parameter which does not include a delimiter as part of its syntax.

Operation Diagrams
17

Displacement Dec.	Hex.	Field	Size in Bytes	Contents
0	0		4	↑Character string, zero if omitted. (The string begins at the first character past the enclosing left punctuation mark.)
4	4		2	Length of the string. (Any punctuation marks around the character is excluded. This field is zero if the string is omitted or if the string is null.)
6	6		1	Flags: <u>Bits</u> <u>Meaning when on</u> 0 Parameter is present. 1-7 Reserved (0).
7	7		1	Reserved (0). <u>Note</u>: If the string is null, the pointer is set, the length is zero and the flag bit is one.

3

PARAMETER CONTROL ENTRY FOR IKJTERM MACRO INSTRUCTION

Size:	Variable
Located in:	Subpool 1.
Created by:	Command processor using the IKJTERM macro instruction.
Used by:	Parse.
Contents:	This PCE describes a positional parameter that may be a Constant, Variable or Statement Number parameter.

Operation Diagrams
18

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0		2	Flags: Bits Meaning when on 0-2 B'110' - IKJTERM PCE 3 PROMPT 4 DEFAULT 5 Reserved (0) 6 HELP 7 VALIDCK 8 LIST 9 ASIS 10 RANGE 11 This term may be subscripted. 12 A Reserved Word is chained. 13-15 Reserved
2	2		2	Length of this PCE.
4	4		2	Offset in PDL to the PDE.
6	6		1	Type of Positional Parameter. Bits Meaning when on 0 STATEMENT NUMBER 1 VARIABLE 2 CONSTANT 3 ANY (Constant or Variable) 4 This is a Subscript term 5-7 Reserved
7	7		4	Bytes 1 - 2 Length of parameter-type field Bytes 3 - 4 Offset of parameter-type field (set to X'0012').

(Continued)

Displacement Dec.	Hex.	Field	Size in Bytes	Contents
11	B		N	Parameter-type field.
--	--		1*	Length of PROMPT or DEFAULT information.
--	--		N**	PROMPT or DEFAULT information.
--	--		2*	Offset into the PCL of Subscript PCE.
--	--		2*	Offset into the PCL of RSVWD PCE.
--	--		2*	Length of second-level message information including this and following field, specified by the HELP operand on the macro.
--	--		1*	Number of second-level messages.
--	--		2*	Length of second-level message including this and following two fields.***
--	--		2*	Message segment offset.***
--	--		N**	Second-level message information.***
--	--		3*	Address of Validity checking routine.
				N Field of variable length * Optional field. ** Optional field of "N" bytes. *** Repeated for each second-level message.

3

PARAMETER CONTROL ENTRY FOR IKJOPER MACRO INSTRUCTION

Size:	Variable.
Located in:	Subpool 1.
Created by:	Command Processor using the IKJOPER macro instruction.
Used by:	Parse.
Contents:	This PCE describes a positional parameter that is an Expression.

Operation Diagrams
18

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0		2	Flags: __Bits__ __Meaning when on__ 0-2 B'111' - IKJOPER PCE 3 PROMPT 4 DEFAULT 5 Reserved (0) 6 HELP 7 VALIDCK 8-15 Reserved
2	2		2	Length of this PCE.
4	4		2	Offset in PDL to the PDE.
6	6		4	__Bytes 1 - 2__ Length of parameter-type field. __Bytes 3 - 4__ Offset of parameter-type field (set to X'0012').
10	A		N	Parameter-type field.
--	--		2	Offset into the PCL of RSVWD PCE.
--	--		2	Offset into the PCL to OPERND1 PCE.
--	--		2	Offset into the PCL to OPERND2 PCE.
--	--		2*	Offset into the PCL TERM PCE. Zero if not present.
--	--		1*	Length of PROMPT or DEFAULT information.
--	--		N**	PROMPT or DEFAULT information.

(Continued)

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
--	--		1*	Number of second-level messages.
--	--		2*	Length of second-level message including this and following two fields.***
--	--		2*	Message segment offset.***
--	--		N**	Second-level message information.***
--	--		3*	Address of Validity checking routine.
				N Field of variable length. * Optional field. ** Optional field of "N" bytes. *** Repeated for each second-level message.

3

PARAMETER CONTROL ENTRY FOR IKJRSVWD MACRO INSTRUCTION

Size:	Variable.
Located in:	Subpool 1.
Created by:	Command processor using the IKJRSVWD macro instruction.
Used by:	Parse.
Contents:	This PCE describes a positional parameter that is a Reserved Word.

Operation Diagrams
18

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0		2	Flags: <u>Bits</u> <u>Meaning when on</u> 0-2 B'101' - IKJRSVWD PCE 3 PROMPT 4 DEFAULT 5 Reserved (0) 6 HELP 7 Reserved (0) 8 IKJTERM macro as a figurative constant. 9-15 Reserved
2	2		2	Length of this PCE.
4	4		2	Offset in PDL to the PDE. Zero when used with IKJTERM macro.
Note: The following fields are omitted if this PCE is used with the IKJTERM macro to describe a figurative constant.				
6	6		4	<u>Bytes 1 - 2</u> Length of parameter-type field. <u>Bytes 3 - 4</u> Offset of parameter-type field (set to X'0012').
10	A		N	Parameter type field.
--	--		1*	Length of PROMPT or DEFAULT information.
--	--		N**	PROMPT or DEFAULT information.

(Continued)

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
--	--		2*	Length of second-level message information including this and following field, specified by the HELP operand on the macro.
--	--		1*	Number of second-level messages.
--	--		2*	Length of second-level message including this and following two fields.***
--	--		2*	Message segment offset.***
--	--		N**	Second-level message information.***
				N Field of variable length. * Optional field. ** Optional field of "N" bytes. *** Repeated for each second-level message.

3

PARAMETER CONTROL ENTRY FOR IKJSUBF MACRO INSTRUCTION

Size:	3 Bytes.
Located in:	Subpool 1.
Created by:	Command processor using IKJSUBF macro instruction.
Updated by:	None.
Used by:	Parse.
Contents:	This PCE serves two purposes: It indicates the end of the previous subfield (or of the PCL itself), and it also indicates the beginning of a new subfield.

Operation Diagrams
19

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0		1	Flags: Bits Meaning when on 0-2 B'000' -- end-of-field PCE 3-7 Reserved (0).
1	1		2	Offset in the PCL to the first IKJKEYWD PCE, or to the next end-of-field indicator, if there are no keywords in this subfield. The next end-of-field indicator may be at an IKJSUBF or an IKJENDP PCE.

PARAMETER DESCRIPTOR ENTRY FOR IKJIDENT MACRO INSTRUCTION

Size:	8 Bytes.
Located in:	Subpool 1.
Created by:	Parse.
Updated by:	None.
Used by:	Command processor.
Contents:	Description built by parse upon finding an IKJIDENT parameter.

Operation Diagrams
17

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0		4	↑Character string, zero if omitted.
4	4		2	Length of character string.
6	6		1	flags: <u>Bits</u> <u>Meaning when on</u> 0 Parameter is present. 1-7 Reserved (0).
7	7		1	Reserved (0).

3

PARAMETER DESCRIPTOR ENTRY FOR IKJKEYWD MACRO INSTRUCTION

Size:	2 Bytes.
Located in:	Subpool 1.
Created by:	Parse.
Updated by:	None.
Used by:	Command processor.
Contents:	Description built by parse upon either finding an IKJKEYWD parameter or defaulting.

Operation Diagrams
19

Displacement Dec.	Displacement Hex.	Field Name	Size in Bytes	Contents
0	0		2	The number (binary) of the corresponding IKJNAME PCE in the PCL. For example, if the keyword found corresponds to the first IKJNAME PCE, this field will contain binary 1.

PARAMETER DESCRIPTOR ENTRY FOR IKJPARM MACRO INSTRUCTION

Size:	8 Bytes.
Located in:	Subpool 1.
Created by:	Parse.
Updated by:	Parse.
Used by:	Parse.
Contents:	This 8-byte header is at the beginning of the PDL.

Operation Diagrams
16

Displacement Dec.	Displacement Hex.	Field Name	Size in Bytes	Contents
0	0		4	↑Next block of storage.
4	4		2	Subpool number.
6	6		2	Length.

PARAMETER DESCRIPTOR ENTRY FOR IKJPOSIT MACRO INSTRUCTION (ADDRESS PARAMETERS)

Size:	36 Bytes.
Located in:	Subpool 1.
Created by:	Parse.
Updated by:	Parse.
Used by:	Command processor.
Contents:	Description built by parse upon finding an address parameter.

Operation Diagrams
17

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0		4	†Load module name -- zero if not specified.
4	4		2	Length1 -- Length of loadname, excluding the period.
6	6		1	Flags1: <u>Bits</u> <u>Meaning when on</u> 0 Loadname is present. 1-7 Reserved (0).
7	7		1	Reserved (0).
8	8		4	†Entry name (name of the CSECT) -- zero if not specified.
12	C		2	Length2 -- Length of entryname, excluding the period.
14	E		1	Flags2: <u>Bits</u> <u>Meaning when on</u> 0 Entryname is present. 1-7 Reserved (0).
15	F		1	Reserved (0).
16	10		4	†Address string -- zero if not specified.
20	14		2	Length3 -- Length of address string. 1. Relative address -- excludes the plus sign. 2. Register address -- excludes letters. 3. Absolute address -- excludes periods.

(Continued)

Displacement Dec.	Displacement Hex.	Field Name	Size in Bytes	Contents
22	16		1	Flags3: Bits / Meaning when on 0 Address is present. 1-7 Reserved (0).
23	17		1	Reserved (0).
24	18		1	Flags4: (Indicates type of address) Hex Number / Meaning X'00' Absolute address. X'80' Symbolic address. X'40' Relative address. X'20' General register. X'10' Double precision floating-point register. X'08' Single precision floating-point register.
25	19		1	Sign -- Arithmetic sign character used before an expression value. This field is zero if not an address expression.
26	1A		2	Indirect count -- The number of levels of indirect addressing.
28	1C		4	†First expression/value PDE -- If the address is in the term of an address expression, this is a pointer to the PDE for the first expression value. X'FF000000' if not an address expression.
32	20		4	User word for validity check exit routine.
32	20		1	Flag field for use by the command processor -- describes the user word.
33	21		3	Address field for use by the command processor.
36	24		4	†Next PDE in the list, if LIST was specified.

PARAMETER DESCRIPTOR ENTRY FOR IKJPOSIT MACRO INSTRUCTION (DSNAME, DSTHING PARAMETERS)

Size:	24 Bytes (30 Bytes if LIST was specified.)
Located in:	Subpool 1.
Created by:	Parse.
Updated by:	Parse.
Used by:	Command processor.
Contents:	Description built by Parse upon finding a dsname or dsthing parameter.

Operation Diagrams
17

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0		4	↑Dsname, zero if omitted. (If the dsname is quoted, the pointer points to the character after the quote.)
4	4		2	Length1 -- Length of dsname. Excluding quotation marks, if any.
6	6		1	Flags1: Bits Meaning when on 0 dsname is present. 1 dsname is quoted. 2-7 Reserved (0).
7	7		1	Reserved (0).
8	8		4	↑Member -- zero if member is omitted.
12	C		2	Length2 -- Length of member, excludes parentheses.
14	E		1	Flags2: Bits Meaning when on 0 Member is present. 1-7 Reserved (0).
15	F		1	Reserved (0).
16	10		4	↑Password -- zero if omitted.
20	14		2	Length3 -- Length of password.
22	16		1	Flags3: Bits Meaning when on 0 Password is present. 1-7 Reserved (0).
23	17		1	Reserved (0).
24	18		4	↑Next PDE in the list, if LIST was specified.

PARAMETER DESCRIPTOR ENTRY FOR IKJPOSIT MACRO INSTRUCTION (EXPRESSION/VALUE PARAMETER)

Size:	16 Bytes.
Located in:	Subpool 1.
Created by:	Parse.
Updated by:	Command processor.
Used by:	Command processor.
Contents:	Description built by Parse upon finding an expression/value parameter.

Operation Diagrams
17

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0		4	↑Address string.
4	4		2	Length of address string.
6	6		2	Reserved (0).
8	8		1	Flags: (Indicates type of expression value.) Hex <u>Number</u> <u>Meaning</u> X'04' Decimal expression value. X'02' Hexadecimal expression value.
9	9		1	Sign -- Arithmetic sign character used before an expression value. This field is zero if it is not an address expression.
10	A		2	Indirect count -- The number of levels of indirect addressing.
12	C		4	↑Next expression value. (The expression values are forward chained. The last element on the chain is indicated by X'FF000000'.)

PARAMETER DESCRIPTOR ENTRY FOR IKJPOSIT MACRO INSTRUCTION (USERID PARAMETER)

Size:	16 Bytes (20 Bytes if LIST was specified.
Located in:	Subpool 1.
Created by:	Parse.
Updated by:	Command processor.
Used by:	Command processor.
Contents:	Description built by parse upon finding a userid parameter.

Operation Diagrams
17

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0		4	↑Userid.
4	4		2	Length1 -- Length of userid.
6	6		1	Flags1: <u>Bits</u> <u>Meaning when on</u> 0 Userid is present. 1-7 Reserved (0).
7	7		1	Reserved (0).
8	8		4	↑Password.
12	C		2	Length2 -- Length of password, excluding slash.
14	E		1	Flags2: <u>Bits</u> <u>Meaning when on</u> 0 Password is present. 1-7 Reserved (0).
15	F		1	Reserved (0).
16	10		4	↑Next PDE in the list, if LIST was specified.

3

PARAMETER DESCRIPTOR ENTRY FOR IKJPOSIT MACRO INSTRUCTION (VALUE PARAMETER)

Size:	8 Bytes (12 Bytes if LIST was specified.)
Located in:	Subpool1.
Created by:	Parse.
Updated by:	Command processor.
Used by:	Command processor.
Contents:	Description built by Parse upon finding an IKJPOSIT parameter.

Operation Diagrams
17

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0		4	↑To character string. (The string begins at the first character past the quote. This field is zero if not specified.)
4	4		2	Length of character string, excluding the quotes.
6	6		1	Flags: __Bits__ __Meaning when on__ 0 Parameter is present. 1-7 Reserved (0).
7	7		1	Type-char (The letter preceding the quoted string.)
8	8		4	↑Next PDE in the list, if LIST was specified.

PARAMETER DESCRIPTOR ENTRY FOR IKJTERM MACRO INSTRUCTION (CONSTANT PARAMETER)

Size:	20 Bytes (24 if LIST is specified).
Located in:	Subpool 1.
Created by:	Parse.
Updated by:	Parse.
Used by:	Command Processor.
Contents:	Description built by parse upon finding a CONSTANT parameter.

Operation Diagrams
18

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0		1	Length1 - Length of term entered but not including signs, decimal points or apostrophes.
1	1		1	Length2 - For floating-point, length of digits following letter E.
2	2		2	Reserved.
4	4		2	Reserved word number - Number of IKJNAME macro that corresponds to the entered name.
6	6		2	Flags: Bit Meaning when on Byte 1 0 Parameter is present. 1 Constant. 2 Variable. 3 Statement number. 4 Fixed-point numeric lteral. 5 Non-numeric literal. 6 Figurative constant. 7 Floating-pt. numeric literal. Byte 2 0 Sign on constant is minus. 1 Sign on exponent of floating point is minus. 2 Decimal point is present. 3-7 Reserved.
8	8		4	Pointer to string of digits.
12	C		4	Pointer to the exponent.
16	10		4	Pointer to the decimal point. Note: Pointers are zero if not present.

3

PARAMETER DESCRIPTOR ENTRY FOR IKJTERM MACRO INSTRUCTION (VARIABLE PARAMETER)

Size:	20 Bytes (24 if LIST is specified).
Located in:	Subpool 1.
Created by:	Parse.
Updated by:	Parse.
Used by:	Command Processor.
Contents:	Description built by parse upon finding a Variable parameter.

Operation Diagrams
18

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0		4	Pointer to data-name.
4	4		1	Length1 - Length of data-name.
5	5		1	Reserved.
6	6		1	Flags: Bits Meaning when on 0 Parameter is present. 1 Constant. 2 Variable. 3 Statement number. 4-7 Reserved.
7	7		1	Reserved.
8	8		4	Pointer to PDE for first qualifier. Set to X'FF000000' if no qualifiers.
12	C		4	Pointer to program-id name.
16	10		1	Length2 - Length of program-id name.
17	11		1	Number of qualifiers.
18	12		1	Number of subscripts.
19	13		1	Reserved. Note: Pointer, length and number fields are zero if not present.

PARAMETER DESCRIPTOR ENTRY FOR IKJTERM MACRO INSTRUCTION (VARIABLE PARAMETER - DATA-NAME QUALIFIER)

Size:	12 Bytes.
Located in:	Subpool 1.
Created by:	Parse.
Updated by:	Parse.
Used by:	Command processor.
Contents:	Description built by parse upon finding a data-name qualifier on a VARIABLE parameter.

Operation Diagrams
18

Displacement Dec.	Displacement Hex.	Field Name	Size in Bytes	Contents
0	0		4	Pointer to data-name qualifier.
4	4		1	Length of data-name qualifier.
5	5		1	Reserved.
6	6		1	Flags: <u>Bits</u> <u>Meaning when on</u> 0-7 Reserved.
7	7		1	Reserved.
8	8		4	Pointer to PDE for next qualifier. Set to X'FF000000' on last qualifier.

3

PARAMETER DESCRIPTOR ENTRY FOR IKJTERM MACRO INSTRUCTION (STATEMENT NUMBER PARAMETER)

Size:	20 Bytes.
Located in:	Subpool 1.
Created by:	Parse.
Updated by:	Parse.
Used by:	Command processor.
Contents:	Description built by parse upon finding a Statement Number Parameter.

Operation Diagrams
18

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0		1	Length1 - Length of program-id.
1	1		1	Length2 - Length of line number.
2	2		1	Length3 - Length of verb number. Note: Lengths do not include periods and are set to zero if not entered.
3	3		1	Reserved.
4	4		2	Reserved.
6	6		2	Flags: Bits Meaning when on 0 Parameter is present. 1 Constant. 2 Variable. 3 Statement number. 4-15 Reserved.
8	8		4	Pointer to program-id.
12	C		4	Pointer to line number.
16	10		4	Pointer to verb number. Note: Pointers are set to zero if not entered.

PARAMETER DESCRIPTOR ENTRY FOR IKJRSVWD MACRO INSTRUCTION (RESERVED WORD PARAMETER)

Size:	8 Bytes.
Located in:	Subpool 1.
Created by:	Parse.
Updated by:	Parse.
Used by:	Command processor.
Contents:	Description built by parse upon finding a reserved word Parameter.

Operation Diagrams
18

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0		2	Reserved.
2	2		2	Reserved word number - Number of the IKJNAME macro that corresponds to the entered name.
4	4		2	Reserved.
6	6		1	Flags: Bits Meaning when on 0 Parameter is present. 1-7 Reserved.
7	7		1	Reserved.

3

PARAMETER DESCRIPTOR ENTRY FOR IKJOPER MACRO INSTRUCTION (EXPRESSION PARAMETER)

Size:	8 Bytes.
Located in:	Subpool 1.
Created by:	Parse.
Updated by:	Parse.
Used by:	Command processor.
Contents:	Description built by parse upon finding an Expression Parameter.

Operation Diagrams
18

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0		4	Reserved.
4	4		2	Reserved.
6	6		1	Flags: <u>Bits</u> <u>Meaning when on</u> 0 Parameter is present. 1-7 Reserved.
7	7		1	Reserved.

PARSE PARAMETER LIST (PPL)

Size:	28 bytes.
Constructed by:	Command processor.
Located in:	Subpool 1.
Updated by:	Parse.
Used by:	Parse.
Contents:	Parameter list for parse.

Operation Diagrams
15

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	PPLUPT	4	↑ User profile table (UPT).
4	4	PPLECT	4	↑ Environment control table (ECT).
8	8	PPLECB	4	↑ Event control block.
12	C	PPLPCL	4	↑ Parameter control list (PCL).
16	10	PPLANS	4	↑ fullword area where parse will place the address of the PDL.
20	14	PPLCBUF	4	↑ Command buffer (CBUF).
24	18	PPLUWA	4	↑ user work area (for validity check routines).

PARSE PERMANENT WORKSPACE (PWORK)

Size:	440 bytes.
Constructed by:	Parse.
Located in:	Subpool 0.
Updated by:	Parse and IKJPARS2.
Used by:	Parse and IKJPARS2.
Contents:	Parameter lists for routines, input stack, temporary PDE for positional parameters, parse save area, internal tables and work areas.

Operation Diagrams
16

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	PWORK	72	18-word register save area.
72	48	PDWORD	8	Scratch/save/convert area.
80	50	ENDINPUT	4	↑ end of command buffer. Used to determine end-of-field.
84	54	PPOINTR	4	↑ first character scanned.
88	58	PLENGTH	2	Length of field scanned.
90	5A	RETCODE	1	Return code.
91	5B	--------	3	Not used.
96	60	SUBRWORK	8	Scratch/save area for GETCORE routine.
104	68	XPDL	4	↑ Parameter descriptor list (PDL).
108	6C	TEMPSAVE	4	Used to temporarily store register 1 before linking to ATRANQ routine.
112	70	PFLAGS	1	Permanent workspace flags.
				<u>Setting</u> <u>Meaning</u>
		PFLIST		X'80' List is being processed.
		PFDEFLT		X'40' Default has been supplied.
		PFENDF		X'20' End of buffer has been reached.
		PFKEYFND		X'10' Keyword match has been found.
		HEXBIT		X'08' Address expression contains a hexadecimal character.

(Continued)

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
		PFNEW		X'02' Used by address routine to indicate a new valid address entryname (with or without loadname qualification).
		DECBIT		X'01' Address expression is decimal.
113	71	PFLAGS2	1	Second flag byte.
				Setting Meaning
		PFSKPINV		X'80' Validity check routine requested a reenter message only.
		RNGEVAL1		X'40' Address routine processed first value of range parameter.
		ONERBIT		X'20' Control bit used during scan by address routine.
		TWORBIT		X'10' Control bit used during scan by address routine.
		RNGEVAL2		X'08' Address routine processed second value of range parameter.
		REGBIT		X'04' Control bit used during scan by address routine.
		FLTERBIT		X'02' Control bit used during scan by address routine.
		BREAKBIT		X'01' Used by address routine to indicate a break character in the parameter.
114	72	PFLAGS3	1	Third flag byte.
				Setting Meaning
		PFSTPRMT		X'80' Prompt for string.
		PFONE		X'40' At least one PDE has been built.
		LOADBIT		X'20' Control bit used by address routine to indicate loadname data.
		ENTRYBIT		X'10' Control bit used by address routine to indicate entryname data.
		PFNULL		X'08' A null line was entered after a prompt.
		LPRNFND		X'04' A left parenthesis was found by the error routine.

(Continued)

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
		PFSPACE		X'02' A positional space parameter was found; the positional string routine knows when to end the string.
		PFMORE		X'01' The left parenthesis of a subfield was also used as the left parenthesis of the list within the subfield.
115	73	PFLAGS4	1	Fourth flag byte.
				<u>Setting</u> <u>Meaning</u>
		PFENDLIM		X'80' End of self-delimiting string found.
		PFLSTEND		X'40' End of LIST#.
		PFVCMSG		X'20' Validity check routine message.
		PFPDDATA		X'10' Processing prompt or default data.
		PFSLASH		X'08' Password for DSNAME/USERID.
		PFENDSET		X'04' Backup pointer for ENDINPUT.
		PFNOPOP		X'02' Do not remove the top element from the stack.
		CKRANGE		X'01' Check for RANGE.
116	74	PFLAGS5	1	Fifth Flag Byte.
120	78	PANCHOR		† last area of main storage space.
124	7C	PANCHORT	4	† internal main storage space anchor.
128	80	PGETLIST		Parameter list for GETCORE routine.
128	80	PGETLNTH	4	Number of bytes requested.
132	84	PGETADDR	4	† place in which address of storage is to be placed.
136	88	PGETMDSP	2	Subpool number from which to get storage.
140	8C			First input stack.
140	8C	PIPDLCUR	4	† Current stack (Initially points to PIPDLCHN).
144	90	PIPDLCHN	4	† Next lower stack (0).
148	94		80	Storage for first stack.
228	E4	PIPDLX	1	Index to next free space in this input stack (Initially points to PIPDLCHN + 4)

(Continued)

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
229		--------	3	Not used.
232	E8	PLINKSV1	4	Save area for return addresses internal subroutines which use other internal subroutines. The latter may use still other subroutines.
236	EC	INVPSAVE	4	Address of invalid parameter.
240	F0			Addresses for keyword scans.
240	F0	PKEYWDPS	4	Address of current IKJNAME PCE.
244	F4	PKEYWDPC	4	Address of current IKJKEYWD PCE.
248	F8	PKEYWDPX	4	Save area for IKJKEYWD PCE address when erasing the corresponding PDE.
252	FC	PKEYWDTB	4	Address of keyword PCE for which a name PCE has been found.
256	100	PKEYWDPM	4	Address of IKJKEYWD PDE.
260	104	PTABLEAD	4	Address of the start of the PCL.
264	108	PTABLEND	4	Address of the end of the PCL.
268	10C	TEMPPDE		Temporary PDE for positional parameters. (80 Bytes)
268	10C	TEMPPDE2		IKJPARS only (36 Bytes)
268	10C	DATAPTR1	4	Address of string, pstring qstring, password, dsname, loadname, or value.
272	110	DATALEN1	2	Length of string, pstring, qstring, password, dsname, loadname, or value.
274	112	DATAFLA1	1	X'80'--above parameter present X'00'--above parameter not present
275	113	DATAFLB1	1	Type code for value.
276	114	DATAPTR2	4	Address of member or entry name.
280	118	DATALEN2	2	Length of member or entry name.
282	11A	DATAFLA2	1	X'80'--member or entry name present. X'00'--member or entry name not present.
283	11B	DATAFLB2	1	Reserved (0).
284	11C	DATAPTR	4	Address of password or address.
288	120	DATALEN3	2	Length of password or address.
290	122	DATAFLA3	1	X'80'--Password or address present X'00'--Password or address not present
291	123	DATAFLB3	1	Reserved.

(Continued)

3

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
292	124	DATAFLAC	1	Following four fields used by address routine. Register notation.
293	125	DATASGN'	1	Sign of first value.
294	126	DATAICT	2	Indirect addressing count.
296	128	DATAEXP	4	↑ next expression/value PDE.
300	12C	DATAUSER	4	User word.
304	130	ENDBACKUP	4	IKJPARS2 additions to temporary PDE.
348	15C	ENDBAKUP	4	Backup for ENDINPUT.
352	160	PDELIM	1	Self-defined delimiter stored by DELIMITER routine.
353	161	PPCOUNT	1	Length of the PCE.
354	162	PPDE. SIZE	1	Length of the PDE.
355	163	PERRCODE	1	Index to address table for rescan address after prompt or default.
356	164	PKEYWDVL	4	Number of IKJNAME in PCE for keyword found.
360	168	RNG2ADDR	4	↑ second PDE for a range.
364	16C	SEGLIST	20	List of message segments for PUTLINE and PUTGET service routines.
384	180	PREVPDEL	4	↑ previous PDE. Used by the validity check exit routine.
388	184	VCEPARAM	12	Parameter list for validity check exit routine.
388	184	PDEADR	4	↑ PDE just constructed.
392	188	USERWORD	4	Seventh word from PPL.
396	18C	VALMSG	4	↑ second level message. Initialized to X'FF000000' by parse.
400	190	MSGCODE	1	Index to message address.
				Message segment containing the last primary message ID. This is used as the first segment in a HELP message passed to PUTLINE and PUTGET. It contains the byte header required by PUTLINE and PUTGET and the word "ENTER".
401	191	PRIMSGID	21	Primary message segment.
422	1A6	SAVLSLEN	2	Save area for storing main storage size of HELP message routine.

(Continued)

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
424	1A8	PLUSSEG	8	Second-level message segment.
432	1B0	PUTLPTR	4	↑ PUTLINE service routine.
436	1B4	PUTGPTR	4	↑ PUTGET service routine.
440	1B8	SRPARAMS		I/O parameter list used when linking to PUTLINE or PUTGET.
440	1B8	UPTADDR	4	↑ User profile table (UPT).
444	1B5	ECTADDR	4	↑ Environment control table (ECT).
448	1C0	ECBADDR	4	↑ Command processor's ECB.
452	1C4	A00000	4	↑ Parameter block.
456	1C8	OPERAND	4	Pointer to last PCE under an IKJOPER PCE.
460	1CC	RSVWDPCE	4	Pointer to IKJRSVWD PCE.
464	1D0	TERMXPCE	4	Pointer to IKJTERM PCE.
468	1D4	OPERPCE	4	Pointer to IKJOPER PCE.
472	1D8	OPERSVE	4	Pointer to left parenthesis of expression in process.
476	1D6	RSVWDSV1	4	LINK reg. SAVE AREA
480	1E0	RSVWDSV2	4	LINK reg. SAVE AREA
484	1E4	CBLNKSV1	4	LINK reg. SAVE AREA
488	1E8	CBLNKSV2	4	LINK reg. SAVE AREA
492	1EC	ENDNMPTR	4	Pointer to end of last data-name scanned.
496	1F0	CHAINPTR	4	Pointer to chain word for data-name qualifier PDEs.
500	1F4	PDEPTR	4	Pointer to next available space in the temporary PDE.
504	1F8	AANC	4	Storage anchors used to control
508	1FC	TANC	4	allocation of data-name qualifier
512	200	OANC	4	PDEs in storage obtained by
516	204	ENDANC	4	the STALOC routine
520	208	PRMTPTR	4	Pointer to start of invalid data for special message format.
524	20C	OPERLL	2	Length of all PDE fields under the IKJOPER and associated PCEs.

(Continued)

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
526	20E	MSGAREA		Special message (6 Bytes)
526	20E	MSGLEN	2	Length of first segment for special message format.
528	210	MSGADDR	4	Pointer to first segment for special message format.
532	214	DIGITCT	1	Counter for length of digit strings.
533	215	ELEMNCT	1	Number of data name qualifiers.
534	216	QUALCT	1	Number of data-name qualifiers.
535	217	CBFLGS1	1	Flags set as follows:
		COBOLMOD		X'80' - IKJPARS2 has been entered.
		OPERMODE		X'40' - IKJOPER has been entered.
		SUBSMODE		X'20' - IKJTERM is processing a subscript.
		NAMEREQD		X'10' - IKJTERM expects a data-name to follow in the buffer.
		ERRORBIT		X'08' - IKJTERM encounters an error.
		RSVDPRMT		X'04' - A reserved word has been prompted for.
		OPERPRMT		X'02' - An expression has been prompted for by IKJOPER.
		RC16		X'01' - A 16 return code is returned.
536	218	CBFLAGS2	1	Flags set as follows:
		SPECMSG		X'80' - A special format error message is necessary.
		LFTPAREN		X'40' - A left paren is to be inserted in the special message buffer.
		RHTPAREN		X'20' - A right paren is to be inserted in special message buffer.
		CHAINTRM		X'10' - A chained IKJTERM macro is in process.
		PARS2IN		X'08' - PARS2 is loaded.
		PRMTSCAN		X'04' - Prompt data being scanned.
		BUFPOPED		X'02' - Buffer popped in SCANF routine.
		RNGADDED		X'01' - First value of RANGE is added.

(Continued)

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
537	219	CBFLAGS3	1	Flags set as follows:
		FIRSTNAM		X'80' - First Variable data-name.
		CTFOUND		X'40' - Beginning of subscript found.
		BLNKFLAG		X'20' - Blank in invalid message format.
538	21A	CBFLAGS4	1	Reserved.
539	21B		1	Unused.
540	21C	TRANAREA	2	IKJTERM compares for OF or IN.
542	21E	CORELEN	2	Reserved.
544	220	PARS2ADR	4	Address of IKJPARS2 load module.
548	224	VCONAD	4	Address of VCON table in IKJPARS.
552	228	GOREGSV	4	Return address from subroutine.
556	22C	TERMBASE	4	IKJTERM base reg save area.
560	230	OPERBASE	4	IKJOPER base reg save area.
564	234	BASE3SV	4	PARSE REG3 save area.
568	238	BASE2SV	4	PARSE REG2 save area.
572	23C	BASE1SV	4	PARSE REG1 save area.
576	240	RBASESV	4	PARSE RBASE reg save area.
580	244	CBLRET	4	Pointer to IKJPARS2 after using subroutine in IKJPARS.
584	248	COREADDR	4	Address of storage for message.
588	24C	AUTOBASE	4	DATAREG save area.
592	250	WORKSAVE	16	WORK registers save areas.
608	260	PLINKSV2	4	Return address from Validity Check routine.
612	264	PUTLINE		Allocate space in which to move the List form of the PUTLINE and PUTGET macro instructions. (32 Bytes)
612	264	PUTLINE	4	Zeroes (0) (set control and output fields).
616	268	-------	4	Zeroes (0) (will contain address of output line).
620	26C	-------	4	Zeroes (0) (will contain address of formatted output).

(Continued)

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
624	270	PUTGET	4	Zeroes (0) (set control and output fields).
628	274	-------	4	Zeroes (0) (will contain address of output).
632	278	-------	4	X'00008000' (set control and output field).
636	27C	-------	4	Zeroes (0) (will contain input buffer address).
640	280			Aligned on doubleword boundary for FREEMAIN.

PARSE RECURSIVE WORKSPACE (RWORK)

Size:	25 bytes.
Constructed by:	Parse.
Located in:	Subpool 0.
Updated by:	Parse.
Used by:	Parse.
Contents:	Internal work areas and pointers.

Operation Diagrams
16

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	RPCEAD	4	↑ IKJSUBF PCE.
4	4	RBASESV	4	↑ previous recursive workspace.
8	8	RXPCESV	4	↑ resume PCE.
12	C	RLINKSV	4	Return address from RECURSE routine.
16	10	RKEYSV	4	↑ first IKJKEYWD PCE.
20	14	RLINKSV1	4	Return address from erase mode.
24	18	RFLAGS	1	Flags set as follows:
		RFKYPRSE		X'80' - keywords have been scanned once.
		RFQDSNM		X'40' - a quoted data set name is being processed.
		RFERASE		X'20' - a PDE is being erased.
		RFPRES		X'10' - a keyword PCE was found in the PCL.
		RFKEYWDS		X'08' - the next recursive processing will be for a keyword.
		RFMEMB		X'04' - a member name is being processed.
		RFNOTQ1		X'02' - the first qualifier is not being processed.
		RFNOSKIP		X'01' - blanks should not be skipped.

SYNTAX CHECKING MASK AREA

Size:	20 bytes.
Located in:	Subpool 1.
Created by:	IKJEFP00 creates all five words of masks. IKEJEP30 creates (and uses) only the first four words.
Used by:	GENSCAN in IKJEFP20.
Contents:	Masks used to control the syntax checking of command name and parameters.

Operation Diagrams
17

Displacement Dec.	Displacement Hex.	Field Name	Size in Bytes	Contents
0	0	DSNCNTL MEMBCNTL	4	A word of one-byte masks, as follows: X'40' - asterisk not allowed. X'01' - first character must be alphabetic. X'03' - other characters must be alphameric. X'08' - maximum permissible dsname or membername length.
4	4	PWSYNTAX	4	A word of one-byte masks, as follows: X'40' - asterisk not allowed. X'03' - first character must be alphameric. X'03' - other characters must be alphameric. X'08' - maximum permissible password length.
8	8	USIDCNTL	4	A word of one-byte masks, as follows: X'40' - asterisk not allowed. X'01' - first character must be alphabetic. X'03' - other characters must be alphameric. X'07' - maximum permissible userid length.

(Continued)

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
12	C	DSTCNTL	4	A word of one-byte masks, as follows: X'C0' - asterisk is allowed. X'01' - first character must be alphabetic. X'03' - other characters must be alphameric. X'08' - maximum permissible DSTHING length.
16	10	KEYSNTX	4	A word of one-byte masks, as follows: X'40' - asterisk not allowed. X'01' - first character must be alphabetic. X'03' - other characters must be alphameric. X'1F' - maximum permissible length.

3

VALIDITY CHECK PARAMETER LIST (VCPARM)

Size:	12 bytes.
Constructed by:	Parse.
Located in:	Subpool 1.
Updated by:	Command processor.
Used by:	Command processor.
Contents:	Address of the PDE, address of user work area, address of second-level message (set by validity check routine).

Operation Diagrams
17

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	PDEADR	4	↑ PDE.
4	4	USEWORD	4	↑ user work area (seventh word in the parse parameter list).
8	8	VALMSG	4	↑ Second-level message (initialized to X 'FF000000' by parse).

3

Section 6: Diagnostic Aids

This section contains the following information:

- Messages (Figure 22) -- a list of messages issued by command scan and parse routines.
- Register Usage (Figure 23) -- a summary of the use of general registers 0-15.
- Return Codes (Figure 24) -- a summary of return codes and their meanings. unless otherwise specified, return codes are contained in register 15.

Other useful diagnostic information is contained in the parse permanent workspace (PWORD) and the parse recursive workspace (RWORK). Descriptions for these data areas are in Section 5.

Message ID	Message	Issued by
IKJ56700A	ENTER xxx	IKJEFP00
IKJ56701	MISSING xxx +	IKJEFP00
IKJ56702I	INVALID,xxx,yyy	IKJEFP00
IKJ56708I	INVALID PASSWORD	IKJEFP00
IKJ56709I	INVALID DATA SET NAME,yyy	IKJEFP00
IKJ56710I	INVALID USERID,yyy	IKJEFP00
IKJ56711I	INVALID ADDRESS,yyy	IKJEFP00
IKJ56712I	INVALID KEYWORK,yyy	IKJEFP00
IKJ56713I	INVALID VALUE,yyy	IKJEFP00
IKJ56715I	INVALID STRING,yyy	IKJEFP00
IKJ56704I	xxx AMBIGUOUS	IKJEFP00
IKJ56714A	ENTER PASSWORD FOR xxx	IKJEFP00
IKJ56703A	REENTER	IKJEFP00
IKJ56706I	ENDING QUOTE ASSUMED,xxx	IKJEFP00
IKJ56707I	RIGHT PARENTHESIS ASSUMED,xxx	IKJEFP00
IKJ56716I	EXTRANEOUS INFORMATION - IGNORED,xxx	IKJEFP00
IKJ56705	MISSING PASSWORD FOR xxx	IKJEFP00
IKJ56717I	INVALID xxx	IKJEFP00

Figure 22. Messages: Command Scan and Parse Service Routines

	IKJEFP00		IKJEFP20		IKJEFP30		IKJEFP60	
Register	Name	Contents	Name	Contents	Name	Contents	Name	Contents
0	R0	Work/Parameter register -- must be zero.	R0	Work/Parameter register -- must be zero.	R0	Work/Parameter register -- must be zero.	R0	Work/Parameter register.
1	R1	Work/Parameter register -- must be one.	R1	Work/Parameter register -- must be one.	R1	Work/Parameter register -- must be one.	R1	Work/Parameter register.
2	R2	Work register.	R2	Work register.	R2	Work register.	R2	Base1 register.
3	R3	Work register.	R3	Work register.	R3	Work register.	R3	Base2 register.
4	XINPUT	↑ Next character to scan.	XINPUT	↑ Parameter to be scanned.	XINPUT	↑ Next character to scan.	XINPUT	Pointer to next character to scan.
5	XINPUTB	↑ Last character scanned -- used to compute length of scanned data.	XINPUTB	↑ Last character scanned -- used to compute length of scanned data.	XINPUTB	↑ Last character scanned -- used to compute length of scanned data.	XINPUTB	Pointer to last character scanned.
6	XPCE	↑ Current PCE.	XPCE	If called by Parse--points to current PCE. If called by command scan--not used.	XFLAGS	↑ Input flag word.	XPCE	Pointer to current parameter control entry (PCE)
7	BASE3	Additional base register for first CSECT.	R7	Not used.	CSOAPTR	↑ Output area.	R7	Work register.
8	LINK2	Second level linkage register.	LINK2	Second level return register.	LINK2	Second level linkage register for linkage between subroutines.	LINK2	Linkage register.
9	LINK1	First level linkage register.	LINK1	First level return register.	LINK1	First level linkage register for linkage between mainline and subroutines.	LINK1	Linkage register.
10	BASE2	Base register for second CSECT.	R10	Not used.	CSPLPTR	↑ Command scan parameter list.	R10	Work register.
11	BASE1	Mainline base register.	R11	Not used.	CBUFPTR	↑ Command buffer.	PWAREG	Base register for command workspace.
12	RBASE	Base register for current recursive workspace.	R12	Not used.	BASE	Base register for command scan.	R12	Base register for Getmain (automatic storage allocation)
13	PBASE	Base register for command workspace.	WORKBASE	Base register for common workspace -- must be 13.	WORKBASE	Base register for command scan.	R13	Work register.
14	R14	Return register.	R14	Work register.	R14	Scratch/call register.	GOREG	Linkage register (between IKJEFP00-PARS2ENT and IKJEFP60-LINKRET)
15	R15	Call register.	BASE	Linkage register used as base register.	R15	Scratch/call register.	R15	Return index for return to IKJPARS2 (contains 0, +4, +8, +12, or +16)

Figure 23. Register Usage: Command Scan and Parse Service Routines

Routine	Return Code Decimal	Meaning
IKJEFP00	00	Success.
	04	Unable to prompt, parameter missing.
	08	Processing interrupted by attention.
	12	Invalid parameters passed to parse by command processor.
	16	No space available for PDL.
	20	Validity check routine requested termination.
	24	Invalid parameters passed to the IKJTERM, IKJOPER, or IKJRSVWD macros.
IKJEFP20	--	Return codes are not used. The address returned to in the calling program is determined by the results of this program.
IKJEFP30	00	Success.
	04	The CSPL contains invalid parameters. (The output area and command buffer offset are unchanged.)

Figure 24. Return Codes: Command Scan and Parse Service Routines

Part 4: Dynamic Allocation

4

Section 1: Introduction

The dynamic allocation routines consist of the dynamic allocation interface routine (DAIR), the special TSO SVC 99 routines, and the DAIR error code analyser (DAIRFAIL). DAIR handles the allocation and freeing of data sets needed by the terminal monitor program, the TSO command processors, and other TSO problem programs. In general, DAIR obtains information about a data set and, if necessary, invokes the SVC 99 routines to perform the requested operation. DAIRFAIL examines error return codes and issues appropriate informational messages when independently invoked to do so by a command processor.

DAIR and the SVC 99 routines provide the services of:

- Obtaining the current status of a data set.
- Allocating a data set.
- Building and storing lists of data set attributes.
- Assigning attributes to data sets.
- Freeing a data set or attribute list.
- Concatenating data sets.
- Deconcatenating data sets.
- Updating the DCB and/or DSE blocks as necessary to conform with the change of status in allocation.

Usually the data sets a terminal user will need are resident before he logs onto the system, and he has included dd statements in his logon procedure to reserve space for them in the Task Input/Output Table (TIOT). The user may, however, allocate data sets during a terminal session if the data set resides on a volume already mounted at the time of the request for allocation.

As supplied with TSO, DAIR resides in SYS1.LINKLIB, and DAIRFAIL resides in SYS1.CMDLIB. The both execute in the user's foreground region with the protection key assigned to that region. SVC 99 resides in SYS1.SVCLIB. The installation may choose to make DAIR and DAIRFAIL resident in the TSO Link Pack Area (TSLPA) in the region assigned to the Time Sharing Control Task (TSCT).

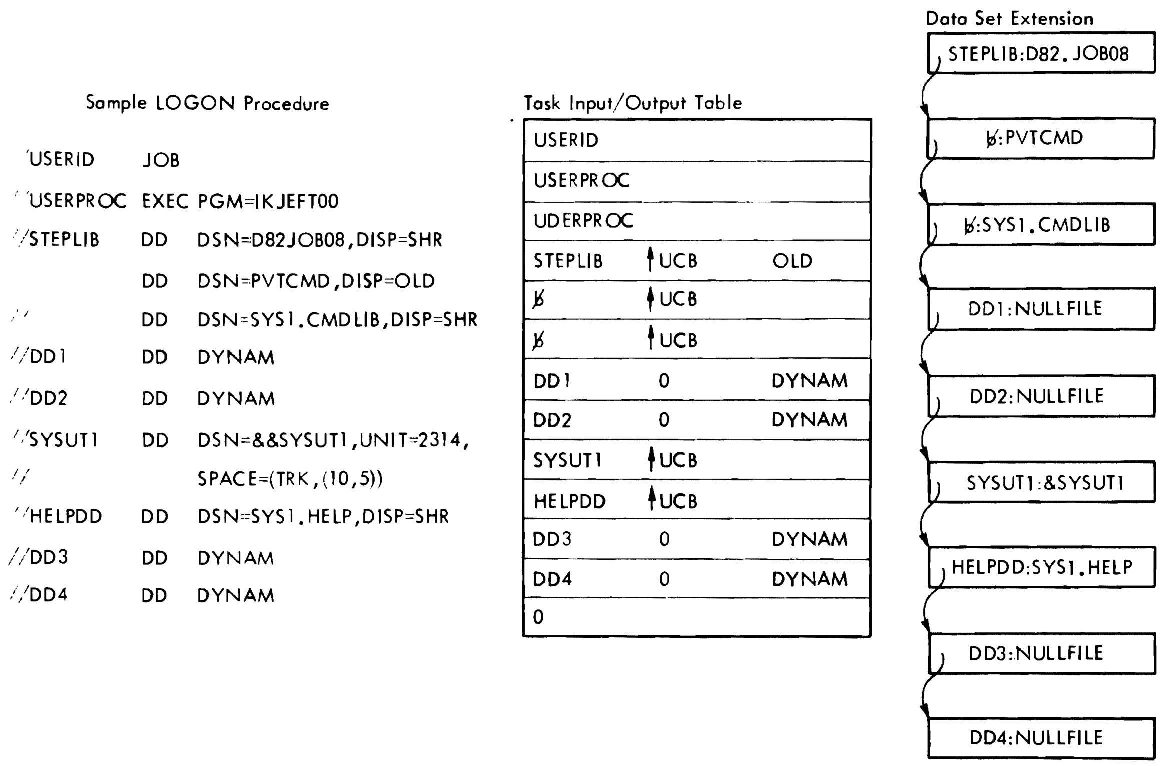
4

Specifying Data Sets at LOGON

When the terminal user logs on to the system, the LOGON scheduler invokes the user LOGON procedure, which contains data definition (dd) statements that define the data sets for use during the session. The job management routines read and interpret the user LOGON procedure and construct standard system control blocks, including the TIOT and the special data set extension (DSE) for TSO, as shown in Figure 25.

The TIOT plays a central role in dynamic allocation. It contains allocation information that includes the relationship between ddnames and devices; it is an interface between the SVC 99 routines and job management.

The data set extension (DSE) is the TSO control block that includes the relationship between ddnames and dsnames; it therefore provides an interface between DAIR and the SVC 99 routines.



```
//USERID    JOB
//USERPROC  EXEC PGM=IKJEFT00
//STEPLIB   DD   DSN=D82JOB08,DISP=SHR
            DD   DSN=PVTCMD,DISP=OLD
//          DD   DSN=SYS1.CMDLIB,DISP=SHR
//DD1       DD   DYNAM
//DD2       DD   DYNAM
//SYSUT1    DD   DSN=&&SYSUT1,UNIT=2314,
//               SPACE=(TRK,(10,5))
//HELPDD    DD   DSN=SYS1.HELP,DISP=SHR
//DD3       DD   DYNAM
//DD4       DD   DYNAM
```

USERID		
USERPROC		
UDERPROC		
STEPLIB	↑UCB	OLD
⊬	↑UCB	
⊬	↑UCB	
DD1	0	DYNAM
DD2	0	DYNAM
SYSUT1	↑UCB	
HELPDD	↑UCB	
DD3	0	DYNAM
DD4	0	DYNAM
0		

When the terminal user logs onto the system, the LOGON scheduler invokes the user logon procedure which contains DD statements that define the data sets to be used during the session. MVT job management routines read and interpret the user logon procedure and construct standard system control blocks including the Task Input/Output Table (TIOT), and the special Data Set Extension (DSE) for TSO.

Figure 25. Specifying Data Sets at Logon

Section 2: Method of Operation

This section describes the method of operation of the dynamic allocation routines, which include the dynamic allocation interface routine (DAIR) and the special TSO SVC 99 routines. DAIR handles the changes to allocation of data sets needed by TSO problem programs, including the terminal monitor program and the TSO command processors. The SVC 99 routines provide the actual movements of data that effect these changes, which include allocating and freeing data sets, concatenating and deconatenating them, listaing and changing their attributes, and updating the DSE to reflect the changes made.

Method of Operation Diagram 20 shows how the DAIR service routine obtains information about data sets and, if necessary, invokes the SVC 99 routines to perform the requested service.

Briefly, here is what happens:

- When the terminal monitor program, a TSO command processor, or TSO problem program needs to use a data set, it links to DAIR and passes it the address of a DAIR parameter block.
- The first two bytes of the DAIR parameter block contain an entry code that defines the service requested. (For example, DAIR entry code X'0008' is a request to allocate a data set by dsname.)
- The DAIR control routine (DAIRCTRL) sets up the DAIR work area (DAIRWA) and branches and links to the appropriate DAIR subroutine. DAIR subroutine names are of the form DAIRnn, where nn is the entry code.
- The DAIR subroutine performs the requested service. If this service requires one of the functions of SVC 99, the subroutine sets up a special entry code to denote which of the functions is necessary and invokes the SVC. Upon return from the SVC, the DAIR subroutine returns control to the calling program.

4

The TIOT plays a central role in dynamic allocation. It contains allocation information that includes the relationship between ddnames and I/O devices; it is an interface between SVC 99 and job management.

The DSE contains allocation information that includes the relationship between ddnames and dsnames; it provides information for correlation between DAIR and SVC 99.

Method of Operation Diagram 20 shows the DSE, the task input/output table (TIOT), and other system control blocks.

DAIR Service Routine

The following discussions describe how DAIR provides an interface between the TSO problem programs that need functions of dynamic allocation, and the SVC 99 routines that provide them. As Figure 26 shows, however, some requested DAIR functions do not require the execution of SVC 99.

ENTRY TO DAIR

DAIR is invoked by a LINK macro instruction to entry point IKJEFD00. At entry, register 1 points to the DAIR parameter list (DAPL).

The DAIR Parameter List contains:

- The address of the user profile table (UPT).
- The address of the environment control table (ECT).
- The address of the calling program's event control block (DCB).
- The address of the protected step control block (PSCB).
- The address of the DAIR parameter block. DAIR parameter block names are of the form DAPBnn, where nn is the DAIR entry code.

DAIR uses the second and fourth fields when initializing the DAIR work area (DAIRWA); the region control task (RCT) uses the first four fields during swap in and swap out.

The DAIR parameter block is the major input to IKJDAIR. The first two bytes contain an entry code (for example X'0008') which defines the operation requested. The remaining bytes contain such things as the ddname for the data set, the dsname for the data set, and whether the userid must be prefixed to the dsname.

SET UP AND INITIALIZATION

At entry, DAIR issues the STAE macro instruction to provide for handling ABEND conditions through the DAIR STAE routine. Then DAIR gets main storage for the DAIR work area (DAIRWA) and initializes it, and gets the address of the DSE.

DAIR then gets the DAIR parameter block and branches to the appropriate DAIR subroutine.

PREFIXING USERID TO DSNAME

If bit 2 of DAnnCTRL is on, DAIR employs the USERID subroutine to prefix the userid to the specified dsname. The format of the dsname buffer is as follows:

Byte	Contents
0-1	The length, in bytes, of the dsname.
2-45	The dsname, left justified, -- the buffer is only as long as necessary to contain the dsname occupying it.

The userid subroutine gets the userid from the DAIR work area and prefixes it to the dsname.

SEARCHING THE DSE CHAIN

Usually, DAIR needs to get some kind of information about a data set. If so, it invokes the SEARCH subroutine to search the DSE chain for a specified dsname or ddname or for an available entry. The DSE contains the following information:

- The ddname.
- The dsname.
- The member name, for a partitioned data set.
- The condition of the data set, as follows:
  - Allocated dynamically.
  - In use.
  - A member of a partitioned data set.
  - Available for dynamic allocation.
  - Dynamically concatenated.
  - Permanently allocated.
- The status of the data set at the time of allocation (NEW, OLD, MOD, or SHR).
- The normal and abnormal disposition for the data set.
- The organization of the data set, as follows:
  - Indexed sequential (IS).
  - Physical sequential (PS).
  - Direct organization (DO).
  - Partitioned organization (PO).
- The address of the TCB for the routine for which the data set is allocated. (Zero, if the data set was allocated during LOGON.)

FUNCTIONS PERFORMED BY DAIR

DAIR gets information from the DSE and, if necessary, invokes one of the SVC 99 routines to perform the requested operation. In general, the DAIR calling routine may specify data sets by either ddname or dsname. An entry code specifies the function requested, as shown in Figure 26.

Entry Code	Function Performed by DAIR
X'00'	Searches the DSE for information about a data set by ddname or dsname.
X'04'	Searches the DSE for information about a data set by dsname. If not found, searches the system catalog.
X'08'*	Allocates a data set by dsname.
X'0C'*	Concatenates data sets by ddname.
X'10'*	Deconcatenates data sets by ddname.
X'14'	Searches the system catalog for all qualifiers for a dsname.
X'18'*	Frees a data set.
X'1C'*	Allocates a data set to a terminal.
X'24'*	Allocates a data set by ddname, or dsname if necessary.
X'28'	Performs a list of operations indicated by other DAIR entry codes.
X'2C'*	Marks DSE entries not available for the specified task.
X'30'*	Allocates a SYSOUT data set.
X'34'*	Builds and frees attribute lists (ATRCBs) and ATRCB chain.
*Requires execution of some SVC 99 Routines	

Figure 26. Functions Performed by DAIR

4

SVC 99 Dynamic Allocation Routines

DAIR invokes SVC 99 to perform the following functions:

- Allocate data sets.
- Free data sets.
- Concatenate data sets.
- Deconcatenate data sets.
- Change the attributes of an allocated data set.
- Update the DSE.
- Build and maintain attribute lists (ATRCBs) and the ATRCB chains.

In general, SVC 99 must have the ddname for the data set. DAIR makes it possible for the calling program to specify data sets by dsname rather than by ddname. An entry code specifies the function requested as shown in Figure 27.

Entry Code	Function Performed by SVC 99
X'00'	Update the data set extension (DSE).
X'01'	Define or allocate a data set by ddname.
X'02'	Free a data set by ddname.
X'03'	Concatenate a data set by ddname.
X'04'	Deconcatenate a data set by ddname.
X'05'	Clean up any outstanding DDRs in this task after ABEND, and return.
X'06'	Change the attributes of a currently allocated data set.
X'07'	Build and free attribute lists (ATRCBs) and ATRCB chains.

Figure 27. Functions Performed by SVC 99

SVC 99 always receives control from DAIR at entry point IGC00099, the beginning of the allocation control routine. This routine performs a control routing function. It examines the entry code in the dynamic allocation parameter block from DAIR to determine the requested function, then performs initialization appropriate for the routines that process the request. Before transferring control to the processing routines, IGC00099 obtains and partially initializes main storage for the dynamic allocation work table (DAWT), a common work area for all the SVC 99 routines. See Section 5 for a description of the DAWT and how the SVC 99 routines use it, according to the function under execution.

Figure 28 shows the method of operation diagrams for dynamic allocation and a legend for using them.

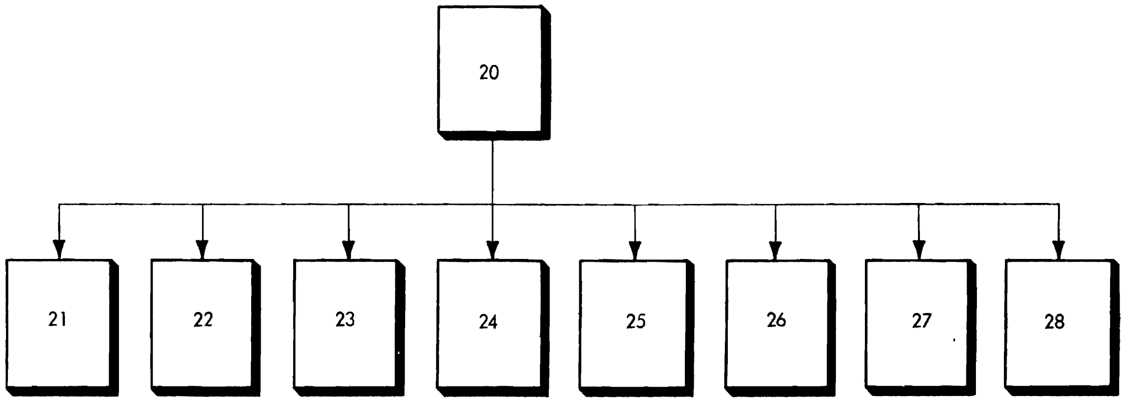


TABLE OF CONTENTS

M.O. Diagram		Diagram Title
20	--	Dynamic Allocation Overview
21	--	Allocating Data Sets
22	--	Freeing Data Sets
23	--	Converting Data Set Attributes
24	--	Concatenating Data Sets
25	--	Deconcatenating Data Sets
26	--	Updating the DSE and DCB
27	--	Building and Freeing ATRCBs
28	--	Analyzing Dynamic Allocation Error Codes

LEGEND

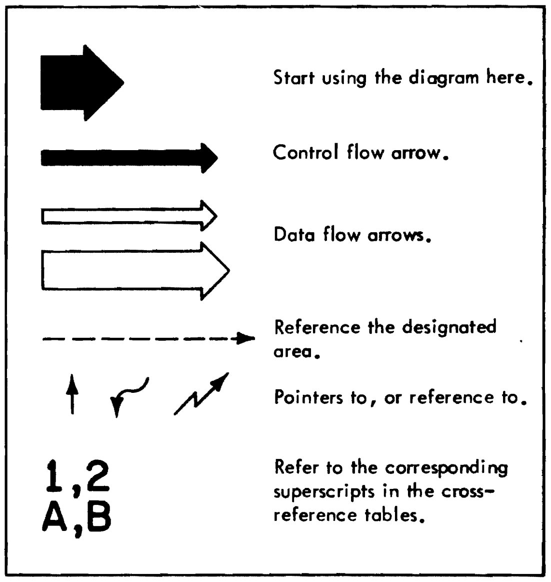

Start using the diagram here.
Control flow arrow.
Data flow arrows.
Reference the designated area.
Pointers to, or reference to.
1,2
A,B
Refer to the corresponding superscripts in the cross-reference tables.

Figure 28. The Method of Operation Diagrams for Dynamic Allocation

ALLOCATING DATA SETS

Method of Operation Diagram 21 shows how SVC 99 performs the steps of data set allocation and invokes the direct access device space management (DADSM) routines -- SVC 32 -- to secure the secondary storage space for the data set. The user requests this function through DAIR, which passes a parameter block that contains the ddname for the data set to be allocated, and the function code (X'01') to designate the requested function to the allocation control routine.

Allocating a data set is the process of initializing a set of system control blocks to describe it according to user specifications, and securing space for it on direct access secondary storage.

The principal descriptors for the data set reside in the job file control block (JFCB), so the dynamic allocation routines construct an entire new JFCB for it.

The correlation between the data set name (dsname) and the ddname of the data set is in the SIOT; the dynamic allocation routines provide this correlation appropriately.

The place-holder for the data set in the task is the dd entry with the ddname and other appropriate information. The dd entry, and others like it for each data set, are the principal parts of the TIOT.

Additional information about the TIOT and JFCB appears in the <u>OS/VS2 System Data Areas</u>, SY28-0606. Additional information about the SIOT appears in the <u>OS/VS2 Job Management Logic</u>, SY28-0620.

FREEING DATA SETS

Method of Operation Diagram 22 shows how SVC 99 frees -- or unallocates -- data sets. The user requests this function through DAIR, which passes a parameter block that contains the ddnames for each data set to be freed, and the function code (X'02') to designate the requested function to the allocation control routine.

Freeing data sets is the process of releasing the operating system resources in use to maintain them; this includes disposing of the data sets according to a set of user-defined or default specifications.

Releasing the resources is a general way of saying that the unallocation routines may (but do not in all cases have to) dequeue the data set from the system task, nullify or free a number of system control blocks as appropriate, and free the direct access storage space that the data set occupies.

Disposing of the data sets means doing some discrete processing to free each data set in some specially defined way. For example, the disposition processing may direct the data set to SYSOUT for printing, or for storing on tape.

In any event, when all those operating system facilities that constitute the resources in use by the data set become free, they are in turn available for use by other data sets.

The system control blocks principally involved in freeing data sets are the same as those for allocating them: the JFCB, the SIOT, and the TIOT. Most of the processing of these control blocks occurs in the unallocate work area, a 584 byte area of main storage that does not appear on Method of Operation Diagram 22. The unallocation routines secure this area for various operations at the beginning of processing and release it at the end.

Additional information about the JFCB and the TIOT appears in the OS/VS2 System Data Areas, SY28-0606. Additional information about the SIOT appears in the OS/VS2 Job Management Logic, SY28-0620.

CONVERTING DATA SET ATTRIBUTES

Method of Operation Diagram 23 shows how SVC 99 converts data set attributes. DAIR passes a parameter block that contains the ddname of the data set whose attributes need changing, and the function code (X'06') to designate the function to the allocation control routine.

Converting data set attributes is the process of changing those fields that record the nature of a data set -- its characteristics, or attributes -- in the system control blocks. The system control blocks principally involved are the job file control block (JFCB) and the step input/output table (SIOT). ddname changes in the SIOT, however, also require corresponding changes to the ddname in the TIOT associated with the same data set.

Additional information about the JFCB and the TIOT appears in the OS/VS2 System Data Areas, SY28-0606. Additional information about the SIOT appears in the OS/VS2 Job Management Logic, SY28-0620.

CONCATENATING DATA SETS

Method of Operation Diagram 24 shows how SVC 99 concatenates data sets. The user requests this function through DAIR, which passes a parameter block that contains the ddnames for each data set to be concatenated, and the function code (X'03') to designate the requested function to the allocation control routine.

Concatenating data sets is the process of grouping them together relative to a single ddname. This grouping shows up internally as contiguous data definition (dd) entries in the TIOT, one entry for each data set. Only the first of these TIOT dd entries has a nonblank ddname field. The ddname in this field is the one, therefore, that identifies the concatenated group of data sets.

To prepare the grouping of dd entries that results in concatenated data sets, the concatenation routines build a new TIOT, with the dd entries correctly rearranged, and move it over the old one. Then they update the chain of SIOTTTRs, if necessary, to re-order it so that the SIOTTTRs still correspond to the rearranged dd entries. If the concatenation results in the relocation of any TIOT dd entries for open data sets, data control block (DCB) updating is necessary; for this, one of the concatenation routines prepares a parameter list for the DCB update routine.

Additional information about the TIOT appears in the OS/VS2 System Data Areas, SY28-0606.

DECONCATENATING DATA SETS

Method of Operation Diagram 25 shows how SVC 99 deconcatenates data sets. The user requests the function through DAIR, which passes a parameter block that contains the ddname for the concatenated group, and the function code (X'04') to designate the requested function to the allocation control routine.

Deconcatenating data sets is the process of reassigning each data set in the concatenated group to its own unique ddname. This is done by retrieving the ddname from the respective SIOT for the data set and replacing the name in the proper TIOT dd entry, which was previously blanked out to concatenate the data set.

Additional information about the TIOT appears in the OS/VS2 System Data Areas, SY28-0606. Additional information about the SIOT appears in OS/VS2 Job Management Logic, SY28-0620.

UPDATING THE DSE AND DCB

Method of Operation Diagram 26 shows how SVC 99 updates both the data set extension (DSE) and the data control block (DCB).

Updating the DSE is the process of changing the information in it to conform with the most current status of allocation. Usually this updating occurs at the conclusion of some other operation in SVC 99, and the routines in the update function provide the normal common exit to DAIR from any of the other functions in the SVC. DAIR may, however, enter the update function directly (that is, through the allocation control routine -- IGC00099 -- only, rather than through another SVC function) to mark a data set as not in use; this happens every time any command processor returns control to the terminal monitor program, which in turn invokes DAIR for the function.

Updating the DCB is a related, but separate operation that sometimes provides the entry into the update function from the concatenation function. It is necessary only when rearrangement of the TIOT dd entries for open data sets has taken place as a result of concatenating other data sets. It consists of making the TIOT dd entry offset information for any rearranged open data set coincide in its respective DCB with the new rearranged position of the entry in the TIOT.

A full description of the DSE appears in "Section 5: Data Areas"; a full description of the DCB appears in the OS/VS2 System Data Areas, SY28-0606.

MANAGING USER-SELECTED DATA SET ATTRIBUTES

Method of Operation Diagram 27 shows how SVC 99 builds and maintains the lists of data set attributes that the user may select to override the default attributes provided by TSO. The user requests this function through DAIR, which passes a parameter block with data for the requested attribute management operation, and the function code (X'07') to designate the attribute management function to the allocation control routine.

Assigning a data set its attribute is the process of defining a set of descriptors for it in portions of the job file control block (JFCB). Ordinarily, TSO provides a set of default descriptors for each data set allocated, so that the JFCB will be complete at allocation time. The operating system refers to these descriptors to complete data control block (DCB) information that defines the data set attributes.

When the TSO user enters his own choice of data attributes at the terminal, however, the ATTRIB command processor and DAIR format his choices into data that SVC 99 can process. This resultant DAIR attribute control block (DAIRACB) provides the information for SVC 99 to build its own attribute list (ATRCB) in the format of corresponding default attribute information in the JFCB. SVC 99 incorporates the ATRCB into a chain of ATRCBs for later use in allocating a data set (or data sets) with the listed attributes.

When the user allocates a data set with his previously listed choice of attributes, the ALLOCATE command processor invokes DAIR and SVC 99 to overlay the TSO default attributes in the JFCB with the proper ATRCB from the chain. This separate operation makes the user attributes available to the operating system.

Conversely, when the user attributes are no longer desirable, SVC 99 unchains the ATRCB that lists them. This occurs when the user logs off, or when he uses the FREE command to abrogate his selection of attributes during a terminal session. The subpool storage made available by the unchaining operation is subsequently reusable.

Additional information about the JFCB appears in OS/VS2 System Data Areas, SY28-0606.

EXIT FROM SVC 99 AND FROM DAIR

The SVC 99 routines return to DAIR using an EXIT macro instruction from the DSE update routines after successful execution of the requested dynamic allocation function. Upon encountering an error condition, however, any one of the SVC 99 routines immediately returns to DAIR via SVC 3 (the EXIT macro) with an appropriate return code in register 15 as shown in Figure 36.

DAIR, in turn, returns to the calling program using a RETURN macro instruction and restoring all registers except 15. At exit, register 15 contains the DAIR return code as described in Figure 35. DAIR uses this return procedure whether or not it invoked the dynamic allocation routines to perform the caller's request.

CLEARING A DYNAMIC DEVICE REQUEST AFTER ABEND

A dynamic device request (DDR) is an event that results from a user's request for dynamic data set allocation. It is represented in the operating system task by fields of control information in several system data areas. Three fields in particular may need clearing if a failure occurs during dynamic allocation, as follows:

- TCB address for the current task.
- TJID address for the terminal job identifier related to the request.
- ECB address for the current DDR event.

These fields are all address in the IORMSCOM system data area, which is addressed by the CVTDCBA field of the communication vector table (CVT).

Clearing a DDR is an SVC 99 operation (function code X'05') performed entirely by the allocation control routine (IGC00099). Since clearing a DDR is a function of DAIR ABEND processing, it is the DAIR STAE routine that invokes SVC 99, and it is the need for supervisory operations on system data areas that makes the use of the SVC mandatory.

Clearing a DDR consists of the following phases, or operations in the SVC 99 allocation control routine.

- Determining whether there is an outstanding DDR in the system.
- Determining whether such a DDR is in the current task.
- Clearing the three address fields related to the DDR.
- Determining whether the DDR is in progress and, if so, posting it.

To determine whether there is an outstanding DDR in the system, SVC 99 checks the CVT CVTDDR field for nonzero contents. A number in CVTDDR means that there is a DDR outstanding in the system.

To determine subsequently whetl er the outstanding DDR is in the current task, SVC 99 compares the ALLOTJID field in IORMSCOM for equality with the TJID field in the JFCB. Equality indicates that the DDR is for the current task.

To clear the three IORMSCOM address fields listed above, SVC 99 fills them with zeros.

To determine whether the DDR is in progress, SVC 99 checks the DDRBUSY bit in IORMSCOM. The bit is on if the DDR is in progress, in which case SVC 99 turns on the DDRPOST bit in the zeroed-out ECB address field of IORMSCOM.

When SVC 99 has cleared the IORMSCOM address fields and posted the DDR-in-progress event (if necessary), it returns to the DAIR STAE routine which called it. The clearing of the outstanding DDR is at this time complete.

DAIR Error Code Analyser

The DAIR error code analyser (IKJEFF18 -- DAIRFAIL) examines dynamic allocation and related catalog management failure codes and issues appropriate informational messages concerning them, when invoked by a command processor to do so. This processing provides the user, at his option, with meaningful diagnostic information about failures that occur during functions of dynamic allocation.

Method of Operation Diagram 27 shows how DAIRFAIL analyses errors and issues messages. Because DAIRFAIL can access the return codes that SVC 99 or the catalog management routines supply to DAIR upon an error return, DAIRFAIL can determine the root of the failure in dynamic allocation.

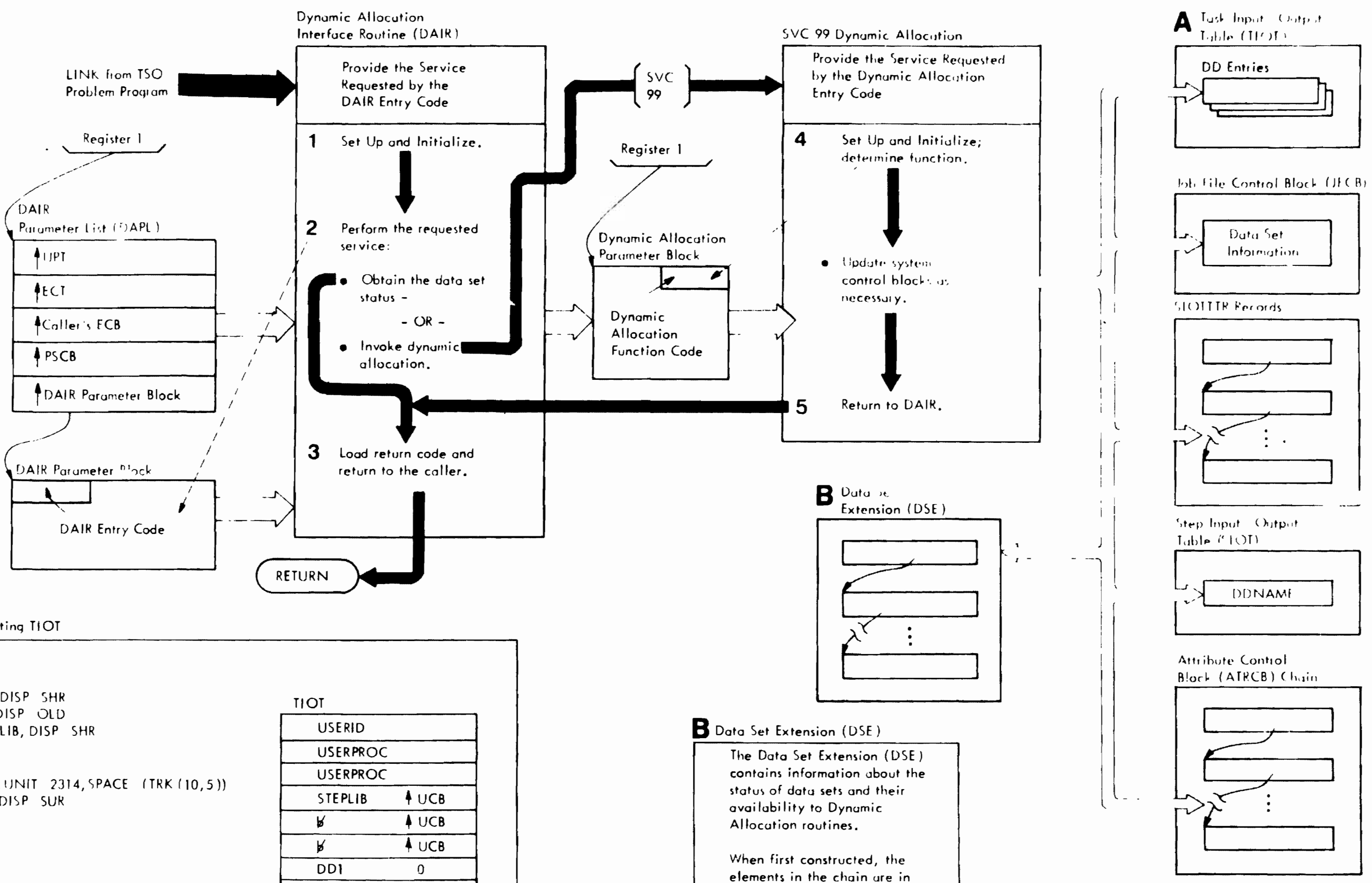
Briefly here is what happens:

- When the user invokes DAIRFAIL through a command processor, DAIRFAIL receives a parameter list of pointers to --
  - The DAIR parameter list (and hence to the DAIR parameter block, which may contain a catalog management or SVC 99 return code).
  - A word of storage containing the applicable DAIR return code.
  - A word of storage that contains either the address of the DAIRFAIL message writer or a direction to load it.
  - Two bytes of storage that identify the calling command processor.
- DAIRFAIL exaimes the DAIR entry code to get the means for initializing and inserting such variables as ddname or dsname into the message text.
- DAIRFAIL examines the applicable return code from DAIR, SVC 99, or the catalog management routines to select the proper informational message.
- DAIRFAIL then prepares a parameter list that contains --
  - The message identifier -
  - The variables for insertion into the message -
  - The address of the message control section -

  -- and passes control to its message writer (IKJEFF02) for completion and issuing of the message.
- DAIRFAIL terminates by zeroing the standard return code register and returning to the calling command processor.

Method of Operation Diagrams

4


Dynamic Allocation Interface Routine (DAIR)
Provide the Service Requested by the DAIR Entry Code
LINK from TSO Problem Program
Register 1
DAIR Parameter List (DAPL)
UPT
ECT
Caller's ECB
PSCB
DAIR Parameter Block
DAIR Parameter Block
DAIR Entry Code
1 Set Up and Initialize.
2 Perform the requested service:
• Obtain the data set status -
- OR -
• Invoke dynamic allocation.
3 Load return code and return to the caller.
RETURN
SVC 99
Register 1
Dynamic Allocation Parameter Block
Dynamic Allocation Function Code
SVC 99 Dynamic Allocation
Provide the Service Requested by the Dynamic Allocation Entry Code
4 Set Up and Initialize; determine function.
• Update system control blocks as necessary.
5 Return to DAIR.
B Data Set Extension (DSE)
A Task Input Output Table (TIOT)
DD Entries
Job File Control Block (JFCB)
Data Set Information
TTR Records
Step Input Output Table (SIOT)
DDNAME
Attribute Control Block (ATRCB) Chain

A Sample LOGON Procedure and Resulting TIOT

```
USERID
USERPROC    EXEC PGM IKJTMP
STEPLIB     DD DSN D82J0B08, DISP SHR
            DD DSN PUTCMD, DISP OLD
//          DD DSN SYS1.CMDLIB, DISP SHR
DD1         DD DYNAM
DD2         DD DYNAM
SYSUT1      DD DSN &&SYSUT1, UNIT 2314, SPACE (TRK (10,5))
HELPDD      DD DSN SYS1.HELP, DISP SHR
DD3         DD DYNAM
//DD4       DD DYNAM
```

TIOT

USERID	
USERPROC	
USERPROC	
STEPLIB	↑ UCB
ƀ	↑ UCB
ƀ	↑ UCB
DD1	0
DD2	0
SYSUT1	↑ UCB
HELPDD	↑ UCB
DD3	0
DD4	0
	0

For a more detailed description of the TIOT, refer to the OS/VS2 System Data Areas, SY28-0606.

B Data Set Extension (DSE)

The Data Set Extension (DSE) contains information about the status of data sets and their availability to Dynamic Allocation routines.

When first constructed, the elements in the chain are in the same order as the corresponding DD statements in the user LOGON procedure.

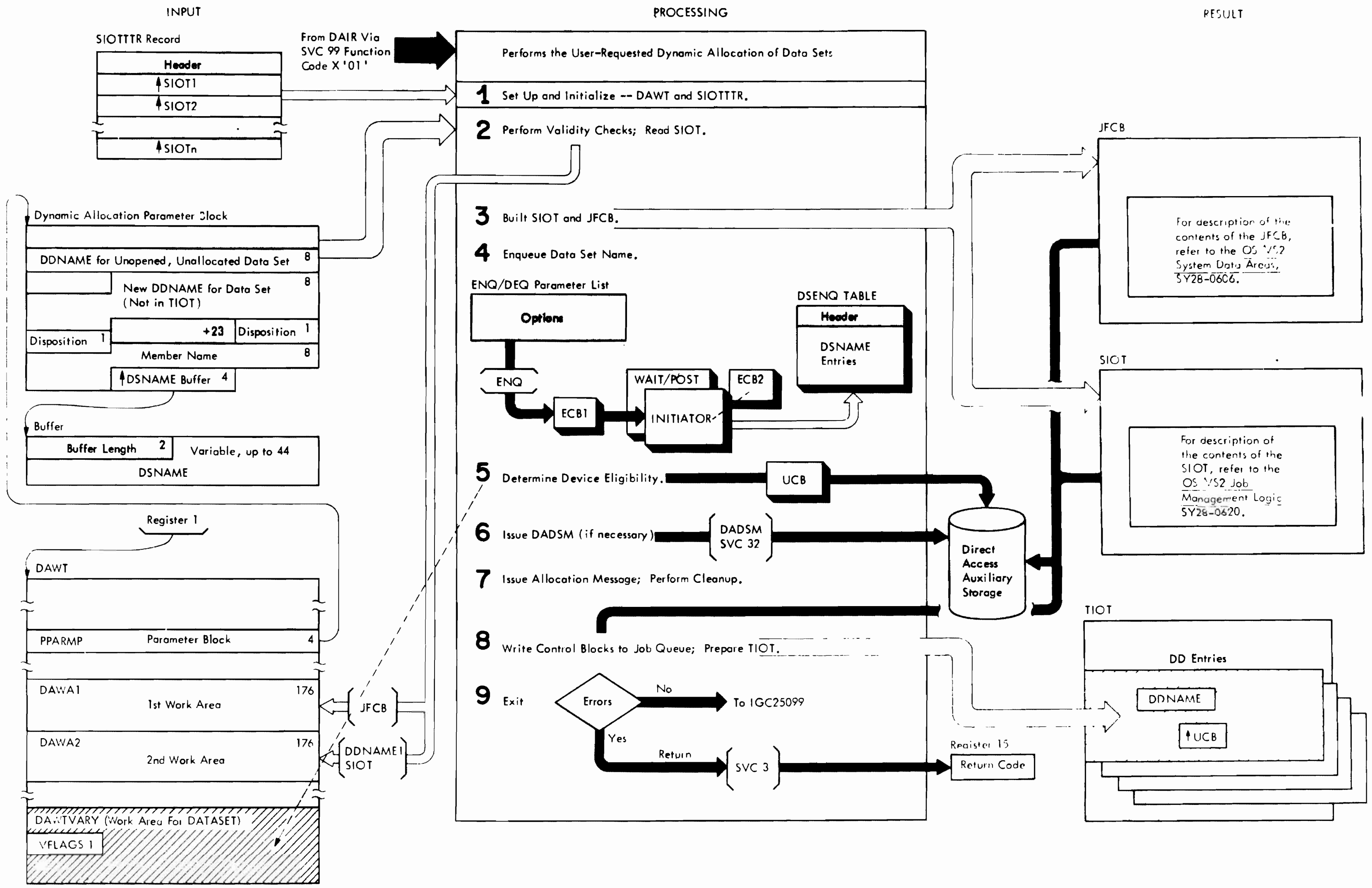
Thereafter, the top element represents the most recently allocated data set, while the bottom element represents the most recently freed data set.

Method of Operation Diagram 20. Dynamic Allocation Interface Routine (Part 1 of 2)

CROSS REFERENCE TABLE FOR DAIR

Key Description			Routine	Label	M.O.
1	At entry, DAIR gets main storage for the DAIR Work Area (DAIRWA) and initializes it, gets the address of the DSE, gets a DAIR Parameter Block (DAPBnn, where nn is the DAIR entry code) and branches and links to the corresponding DAIR subroutine.		IKJEFD00	DAIRCTRL	
2	The DAIR entry code specifies the requested service as follows:				
	Code	Operation			
	X'00'	Search the DSE chain for a data set by DDNAME or DSNAME.	IKJEFD00	DAIR00	
	X'04'	Search the DSE chain for a data set by DSNAME; if not found, search the system catalog.	IKJEFD00	DAIR04	
	X'08'	Allocate a data set by DSNAME -- invoke SVC 99 DATASET function.	IKJEFD00	DAIR08	
	X'0C'	Concatenate data sets by DDNAME -- invoke SVC 99 CONCAT function.	IKJEFD00	DAIR0C	
	X'10'	Deconcatenate data sets by DDNAME -- invoke SVC 99 DECONCAT function.	IKJEFD00	DAIR10	
	X'14'	Search the system catalog for all qualifiers for a DSNAME.	IKJEFD00	DAIR14	
	X'18'	Free a data set -- invoke SVC 99 UNALLOC function.	IKJEFD00	DAIR18	
	X'1C'	Allocate a data set to the terminal -- invoke SVC 99 DATASET function.	IKJEFD00	DAIR1C	
	X'24'	Allocate a data set by DDNAME (first) or DSNAME -- invoke SVC 99 DATASET function.	IKJEFD00	DAIR24	
	X'28'	Perform a list of operations indicated by other DAIR entry codes.	IKJEFD00	DAIR28	
	X'2C'	Free DSE entries for the specified task -- invoke SVC 99 UPDATE function.	IKJEFD00	DAIR2C	
	X'30'	Allocate a SYSOUT data set -- invoke SVC 99 DATASET function.	IKJEFD00	DAIR30	
	X'34'	Build and maintain attribute control blocks (ATRCBs).	IKJEFD00	DAIR34	
3	Returns to the calling program with return code in register 15. See Figure 37 for the return codes and their meanings.		IKJEFD00	EXITCODE	
4	At entry, SVC 99 gets control at entry point IGC00099, the beginning of the allocation control routine. This routine gets main storage for and partially initializes the dynamic allocation work table (DAWT) and invokes the queue management routines to read the queue address (SIOTTTR) associated with the first DDNAME in the dynamic allocation parameter block. Also, by referring to the function code in the second two bytes of the parameter block, the routine determines which of the dynamic allocation functions will subsequently receive control, as follows:		IGC00099	CONTROL	
	Code	Function Performed by SVC 99			
	X'00'	Update the Data Set Extension (DSE)	IGC25099	UPDATE	26
	X'01'	Define or allocate a data set by DDNAME.	IGC07099	DATASET	21
	X'02'	Free a data set by DDNAME.	IGC01099	UNALLOC	22
	X'03'	Concatenate a data set by DDNAME.	IGC18099	CONCAT	24
	X'04'	Deconcatenate a data set by DDNAME.	IGC23099	DECONCAT	25
	X'06'	Change the attribute of a currently allocated data set.	IGC16099	CONVERT	23
	X'07'	Perform attribute function.	IGC30099	ATTRIB	
5	Returns to DAIR. Normally this return is from one of the update routines at the conclusion of processing for any specified function. In case of error, any routine in SVC 99 can return control with an error return code in register 15. See Figure 38 for these return codes.		IGC25099 IGC26099 IGC27099 IGC29099	UPDATE	26

Method of Operation Diagram 20. Dynamic Allocation Interface Routine (Part 2 of 2)

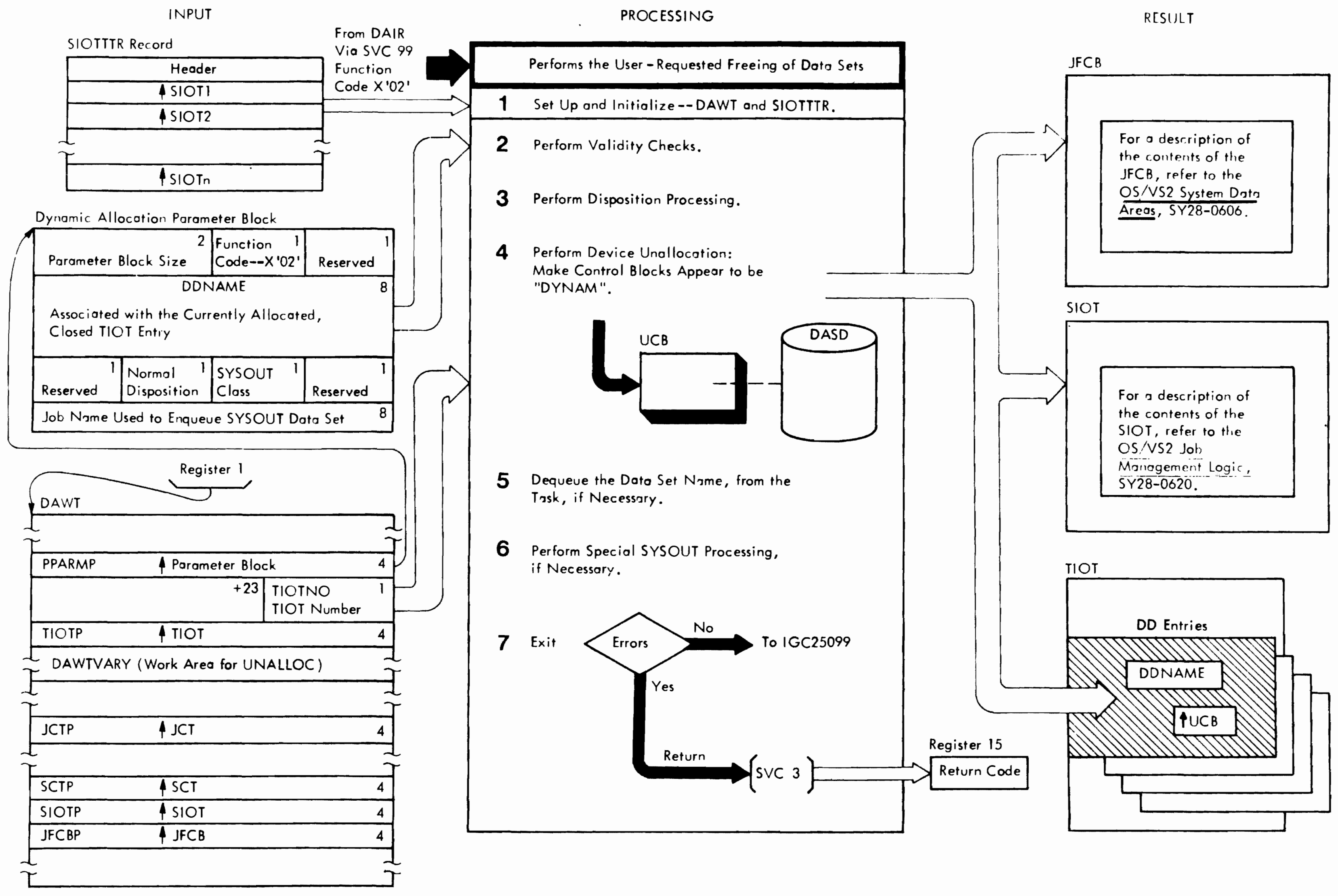

INPUT
PROCESSING
RESULT
SIOTTTR Record
Header
SIOT1
SIOT2
SIOTn
From DAIR Via SVC 99 Function Code X '01'
Performs the User-Requested Dynamic Allocation of Data Sets
1 Set Up and Initialize -- DAWT and SIOTTTR.
2 Perform Validity Checks; Read SIOT.
3 Built SIOT and JFCB.
4 Enqueue Data Set Name.
ENQ/DEQ Parameter List
Options
ENQ
ECB1
WAIT/POST
ECB2
INITIATOR
DSENQ TABLE
Header
DSNAME Entries
5 Determine Device Eligibility.
UCB
6 Issue DADSM (if necessary)
DADSM SVC 32
7 Issue Allocation Message; Perform Cleanup.
Direct Access Auxiliary Storage
8 Write Control Blocks to Job Queue; Prepare TIOT.
9 Exit
Errors
No
To IGC25099
Yes
Return
SVC 3
Register 15
Return Code
Dynamic Allocation Parameter Block
DDNAME for Unopened, Unallocated Data Set 8
New DDNAME for Data Set (Not in TIOT) 8
+23
Disposition 1
Disposition 1
Member Name 8
DSNAME Buffer 4
Buffer
Buffer Length 2
Variable, up to 44
DSNAME
Register 1
DAWT
PPARMP
Parameter Block 4
DAWA1 176
1st Work Area
DAWA2 176
2nd Work Area
DAWTVARY (Work Area For DATASET)
VFLAGS 1
JFCB
DDNAME1 SIOT
JFCB
For description of the contents of the JFCB, refer to the OS/VS2 System Data Areas, SY28-0606.
SIOT
For description of the contents of the SIOT, refer to the OS/VS2 Job Management Logic SY28-0620.
TIOT
DD Entries
DDNAME
UCB

Method of Operation Diagram 21. Allocating Data Sets (Part 1 of 2)

CROSS-REFERENCE TABLE FOR DATASET

Key	Description	Routine	Label
1	• The input SIOTTTR (relative track address record) contains the pointer (TTR) to the SIOT containing the first user-supplied DDNAME in the dynamic allocation parameter block from DAIR. • Preparing the DAWT includes inserting the parameter block address for use by the allocation routines, reading key information from the TIOT extension into DAWA1, and reading the input SIOTTTR into DAWA2. • The dynamic allocation parameter block contains the address and control information necessary for the DATASET function to allocate a data set dynamically. Refer to 'Section 5: Data Areas" for a complete description of the parameter block contents.	Allocation Control Routine	IGC00099
2	Validity checks include the following: • Name checking (NAMECK subroutine). • DDNAME1 TIOT DD entry is -- - Available for dynamic allocation (a DYNAM entry). - Associated with neither a concatenated data group nor a multi-volume data set. • DDNAME2, if specified -- valid characters and not already in TIOT. • DSNAME, if specified -- proper length and valid characters. • Membername, if specified -- valid characters. • SYSOUT information, if specified -- valid characters. • No mutually exclusive parameters -- DSNAME, DUMMY, TERM, or SYSOUT appear alone. Moves key TIOT extension information from DAWA1 to DAWTVARY; reads the SIOT that contains DDNAME1 into DAWA2.	Validity Checking Routine	IGC07099
3	Initializes SIOT in DAWA2 by zeroing and blanking out some fields, loading others from the parameter block, and setting bits, as appropriate. Moves it into a specially obtained holding buffer to await movement to the job queue data set (SYS1.SYSJOBQE). • Initializes JFCB in DAWA1 similarly.	SIOT and JFCB Building Routine	IGC09099
	• Processing for SYSOUT data sets -- Obtains storage for a SYSOUT work area, places its address in the DAWT, reads the SCT into DAWA1, obtains storage for an enqueue/dequeue parameter list, and issues an ENQ macro instruction for the WTPCB; assigns space in the job queue for a new SMB, a new DSB, and the JFCB and chains them into the message class output chain. Generating a temporary data set name -- Generates name in JFCB in DAWA1 (entry from IGC09099) or in the SYSOUT work area (entry from IGC07099); obtains storage for an enqueue/dequeue parameter list for the allocation/termination (Q4/Q5) resource and initializes it.	SYSOUT Processing Routine	IGC08099
	Additional SIOT and JFCB processing: • If DUMMY or TERM=TS is specified -- places a DSNAME of NULLFILE in the JFCB; turns on the DUMMY or TERM DSNAME bit in the VFLAGS field of DAWTVARY; sets the disposition bits in the SIOT and JFCB as specified in the input parameter block. • If the DSNAME is omitted or begins with "&" -- turns on the VGENDS bit in VFLAGS to indicate that a DSNAME must be generated; turns on the STATUS = NEW bit in VFLAGS and sets a disposition of NEW, DELETE in both the JFCB and SIOT. • If the data set is a SYSOUT data set -- moves the generated DSNAME in the SYSOUT work area into the JFCB and frees the storage occupied by that work area; turns on the STATUS=NEW bit in VFLAGS, sets a disposition of NEW, DELETE in the SIOT, and a status of MOD in the JFCB. • If the DSNAME is specified but does NOT begin with "&" -- moves the name from the input parameter block into the JFCB; turns on STATUS=NEW bit in VFLAGS if disposition is NEW; issues LOCATE macro to get the volume serial number from the volume list block if disposition is not NEW and volume serial number not specified; turns on STATUS=NEW bit in VFLAGS if volume serial number cannot be found. • If an attribute list name is specified, the ATRCB by that name is merged into the JFCB. • Saves the unit type field in the SIOT, places the SIOT in its buffer, transfers control to one of four routines, as follows: - To IGC15099 if the DUMMY or TERM data set bit is on in VFLAGS. - To IGC08099 if the VGENDS bit is on in VFLAGS -- DSNAME needs generating. - To IGC10099 if a DSNAME is specified in the input parameter block. - To IGC11099 in other cases with the VNOENQ bit turned on in VFLAGS to signal IGC11099 not to perform data set enqueue processing.	SIOT and JFCB Building Routine	IGC09099
4	Searches for a DSENQ table entry for this data set to see if it has already been enqueued; transfers control to IGC11099 if it finds a DSNAME entry with compatible use attributes.	Data Set Enqueuing Routine.	IGC10099
	• Changes use attribute from shared to exclusive (RET=CHANGE option on ENQ macro) if necessary for a currently enqueued data set; enqueues data set if there is no DSENQ table entry for it (see IGC10099 module description in Section 3); builds ENQ/DEQ parameter list for allocation/termination (Q4/Q5) resource and passes it to IGC11099 for any further necessary enqueue processing.	SIOT and JFCB Update Routine	IGC15099
5	The device, or unit, information in the DEVTYP field of DAWTVARY is in one of three forms, processed accordingly, as follows: • Converts EBCDIC information -- - If the unit name is either generic or esoteric, searches the device name table (DNT) for the unit entry and saves it in DEVTYP. - If the unit name is a specific UCB, saves a pointer to it in the UCBP field of DAWTVARY and turns on the specific UCB flag (VSPFCUCB) in VFLAGS. • Considers all direct access devices as eligible if unit not specified. • Performs no processing if unit information already is in device type form.	DSENQ Update Routine	IGC11099
	Uses device mask table (DMT) and/or system UCBs to build a bit pattern that represents devices that may be eligible to satisfy the allocation request; turns on bit pattern construction flag (VBPCONST) in VFLAGS to indicate initialization of the pattern construction area (PCA -- see Section 5) for the bit pattern.	Bit Pattern Con-struction Routine	IGC12099
	Uses specific UCB designation or PCA bit pattern to furnish device candidates for eligibility validation; performs a comprehensive series of checks on the device and the volume mounted on it to ensure that the allocation request can be satisfied; puts addresses of eligible candidates in UCBLIST (see Section 5) in DAWA2 and frees the PCA or DMT.	Device Reduction Routine	IGC13099
6	Puts default space parameters in the JFCB; for a single candidate, issues direct access device space management (DADSM -- SVC 32) to allocate the necessary space on the volume; otherwise, builds and searches the candidate list for the best candidate and invokes I/O load balancing to select a candidate for allocation; puts the selected UCB (candidate) address in the UCBP field in DAWTVARY or returns error code if DADSM fails for all.	DADSM Routine	IGC14099
7	Issues the message through the WTP facility.	TIOT and JFCB Update Routine	IGC15099
8	Writes the JFCB and SIOT to SYS1.SYSJOBQE and moves the DDNAME to the TIOT, updating the TIOT as necessary, if control is from IGC09099. For other than a DUMMY or TERM request: places the volume serial number of the allocated device in the JFCB and writes the SIOT and JFCB to SYS1.SYSJOBQE; updates the UCB, issues a DEQ macro for the allocation/termination resource, frees the ENQ/DEQ parameter list, and issues any requested allocation message (see 7 above).		
9	Errors during dynamic allocation cause an immediate return to DAIR by SVC 3 with return code as shown in Figure 38.		

Method of Operation Diagram 21. Allocating Data Sets (Part 2 of 2)

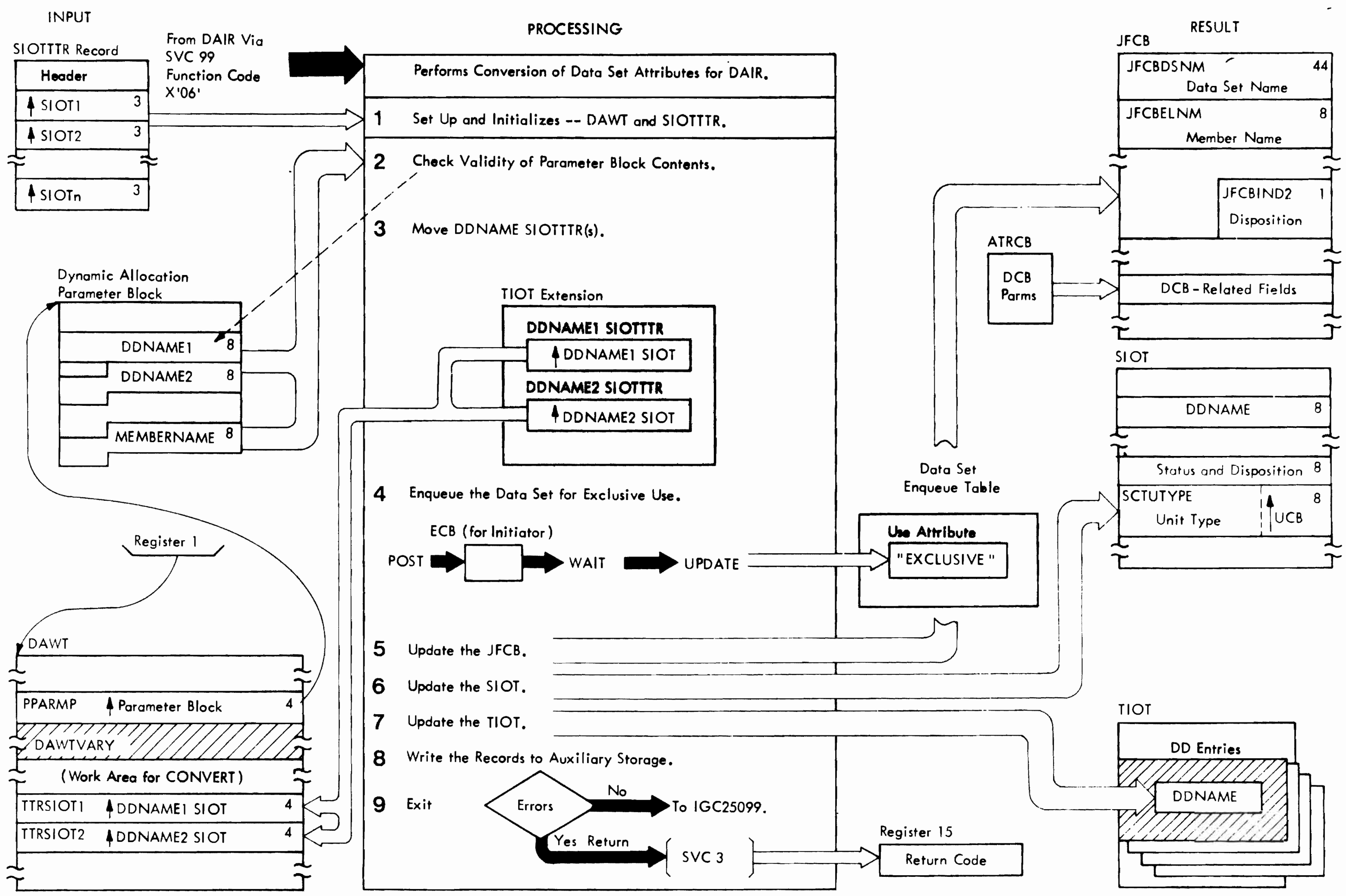

INPUT
PROCESSING
RESULT
SIOTTTR Record
Header
SIOT1
SIOT2
SIOTn
From DAIR Via SVC 99 Function Code X'02'
Performs the User-Requested Freeing of Data Sets
1 Set Up and Initialize--DAWT and SIOTTTR.
2 Perform Validity Checks.
3 Perform Disposition Processing.
4 Perform Device Unallocation: Make Control Blocks Appear to be "DYNAM".
UCB
DASD
5 Dequeue the Data Set Name, from the Task, if Necessary.
6 Perform Special SYSOUT Processing, if Necessary.
7 Exit
Errors
No
To IGC25099
Yes
Return
SVC 3
Register 15
Return Code
Dynamic Allocation Parameter Block
Parameter Block Size 2
Function Code--X'02' 1
Reserved 1
DDNAME 8
Associated with the Currently Allocated, Closed TIOT Entry
Reserved 1
Normal Disposition 1
SYSOUT Class 1
Reserved 1
Job Name Used to Enqueue SYSOUT Data Set 8
Register 1
DAWT
PPARMP Parameter Block 4
+23 TIOTNO TIOT Number 1
TIOTP TIOT 4
DAWTVARY (Work Area for UNALLOC)
JCTP JCT 4
SCTP SCT 4
SIOTP SIOT 4
JFCBP JFCB 4
JFCB
For a description of the contents of the JFCB, refer to the OS/VS2 System Data Areas, SY28-0606.
SIOT
For a description of the contents of the SIOT, refer to the OS/VS2 Job Management Logic, SY28-0620.
TIOT
DD Entries
DDNAME
UCB

Method of Operation Diagram 22. Freeing Data Sets (Part 1 of 2)

CROSS-REFERENCE TABLE FOR UNALLOC

Key Description	Routine	Label
1 • The input SIOTTTR (relative track address record) contains the pointer (TTR) to the SIOT containing the user supplied DDNAME in the dynamic allocation parameter block from DAIR. • Preparing the DAWT for freeing data sets includes inserting the parameter block address for use by the unallocation routines and initializing the pointers in DAWTVARY as shown on the diagram. See "Section 5: Data Areas" for a full description of DAWTVARY. • The dynamic allocation parameter block contains the address and control information necessary for the UNALLOC function to free a data set. Refer to "Section 5: Data Areas" for a description of the parameter block contents.	Allocation Control Routine	IGC00099
2 Returns to DAIR as soon as the reading of the input SIOTTTR is complete if the data set is already free; otherwise, checks the contents of the parameter block as follows: • That the data set is not associated with multiple volumes or units. • That the device involved is direct access. • That the disposition is either valid or omitted. • That the DDNAME is associated with neither an open data set nor with a concatenated data group. For valid checks, secures the unallocate work area, initializes pointers into it, and reads the SIOT and JFCB into it; checks data sets other than DUMMY or terminal to ensure that they are not part of a generation data group, and are not passed, suballocated, or shared, with a disposition of DELETE; initializes parameter and volume lists for the CATALOG and SCRATCH macros, and gets storage for and initializes an enqueue parameter list, if these operations and checks produce no errors.	Validity Checking Routine	IGC01099
3 Checks operator and/or terminal requests for disposition messages and makes appropriate indications; prepares and writes disposition messages IEF283I, IEF285I, and IEF287I as appropriate (see the OS/VS Message Library: OS/VS2 System Messages, GC38-1002); if the message is to the operator or user, issues the WTO/WTP macro; if any terminals are to receive the message, searches the TJB chain and invokes the TPUT macro to issue the message. May also receive entries from SYSOUT processing routines (see 6 below): • Control from IGC05099 -- initializes the WTO buffer and the disposition message SYSOUT. • Control from IGC04099 -- initializes the WTO buffer and scratches the data set.	Disposition Processing Routine	IGC02099
4 Decrements the user count; cleans up the UCB so that it no longer appears allocated, if last user. • For terminal or DUMMY data sets -- turns off the terminal bit in the SIOT and TIOT; zeros out the UCB address part of the SCTUTYPE field of the SIOT; turns on the DYNAM bits in the SIOT and TIOT; turns off the DUMMY bit in the SIOT; puts "NULLFILE" in the DSNAME field of the JFCB; writes the SIOT and JFCB to the job queue; frees the storage for the unallocate work area; passes control to IGC25099 for DSE updating. • For SYSOUT data sets -- zeros out the SIOT DSB pointer and turns off the SYSOUT bit; if the data set is to be enqueued for the output writer task immediately (that is, when its disposition is not DELETE), rather than waiting for the completion of LOGOFF, continues processing as though the data set were DUMMY; in this case, zeros out the UCB pointer in the TIOT. • For other data sets -- issues an ENQ macro for termination resources and decrements the user count; if the user count is zero, frees the device by turning off the allocated bit in its UCB; turns off the nonsharable and data management count bits; if the volume is private, not permanently resident, or reserved, and if it does not have retain-specified or passed data sets, passes control to IGC06099 for KEEP message processing.	Device Freeing Routine	IGC03099
5 Issues a DEQ macro for the termination resource; searches the DSENQ table records for the DSNAME if the data set is neither SYSOUT nor temporary; deletes name from table and dequeues it by using the POST macro against the ECB for the initiator (which in turn issues the DEQ macro against the DSNAME); issues the WAIT macro for the dequeuing operation to complete. Frees the storage occuped by the ENQ parameter list, zeros out the UCB pointer in the TIOT, and continues processing as though for a DUMMY data set (see 4 above).	Device Freeing Routine	IGC03099
6 For SYSOUT dispositions other than DELETE, checks validity of job name and SYSOUT class; obtains storage for the JCF if the disposition is not DELETE and for the SCT if the DSB is in the message class, and reads the tables into storage; if the disposition is not DELETE, invokes the transient queue management routines to assign records for the DSB and JFCB associated with the data set; if the data set is to be enqueued for a SYSOUT writer (that is, when the disposition is not DELETE), passes control to IGC05099; for cases where the DSB is in the message class, writes the SCT to the job queue and frees the SCT storage; transfers control to IGC02099, which deletes the data set.	SYSOUT Chain DSB Processing Routine	IGC04099
Initializes a DSB using information from the SIOT, TIOT and input parameter block; for SMF, places appropriate SMF information from the JMR into the DSB; writes the DSB and JFCB to the job queue using the records assigned by IGC04099; if the DSB is in the message class, writes the SCT to the job queue and frees the JCT and SCT storage; invokes the transient queue management routines to enqueue the data set on the proper output class; transfers control -- to IGC02099 to issue the disposition message if MSGLEVEL=1, to IGC03099 otherwise.	SYSOUT Data Set Enqueuing Routine	IGC05099
7 Normally, transfers control to IGC25099 for DSE updating; however, errors cause a return to DAIR via SVC 3 with a return code as shown in Figure 38.		

Method of Operation Diagram 22. Freeing Data Sets (Part 2 of 2)

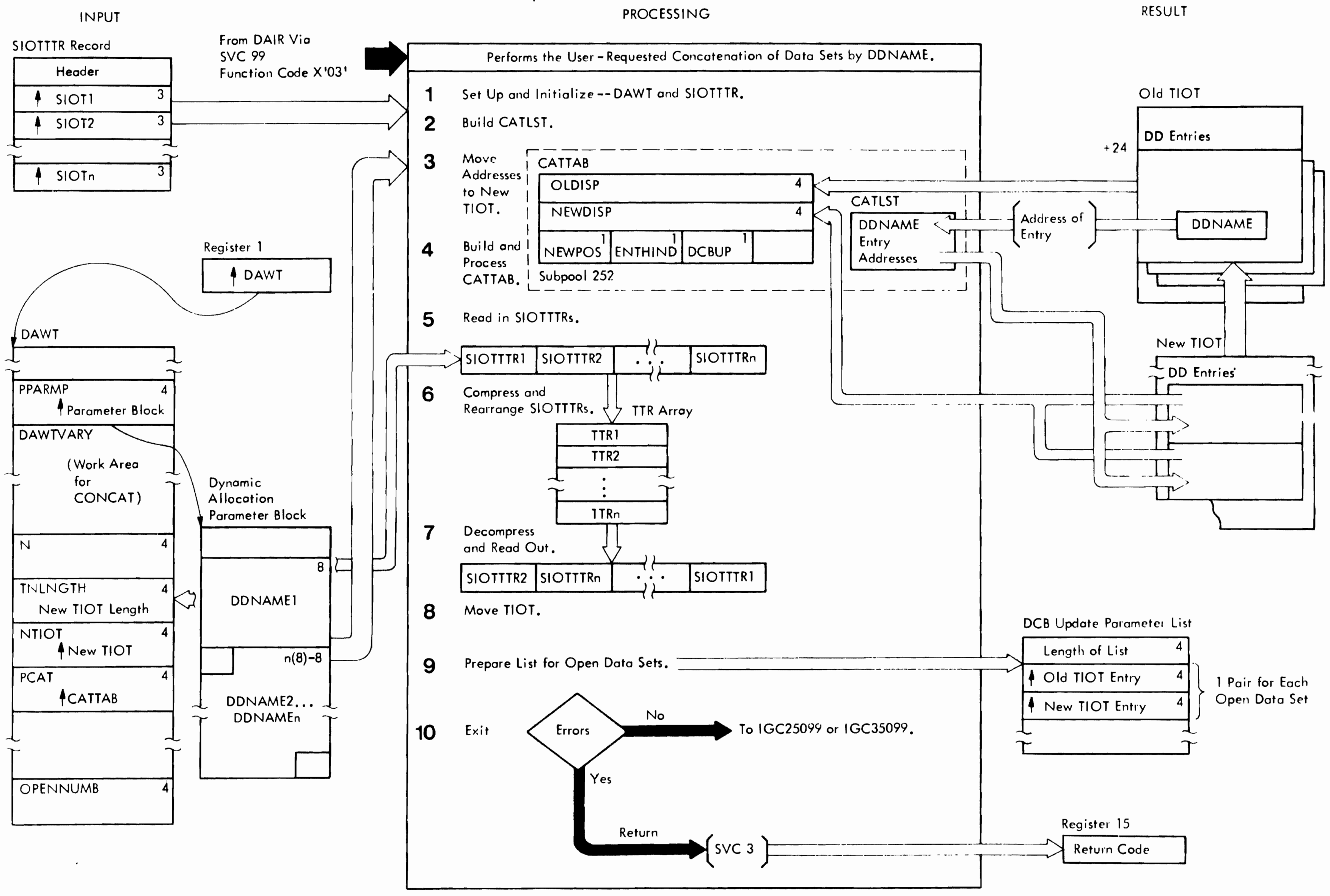


Method of Operation Diagram 23. Converting Data Set Attributes (Part 1 of 2)

CROSS-REFERENCE TABLE FOR CONVERT

Key	Description	Routine	Label
1	• The input SIOTTTR (relative track address record) contains the pointer (TTR) to the SIOT containing the first user-supplied DDNAME in the dynamic allocation parameter block from DAIR. See the description for the parameter block below. • Preparing the DAWT for attribute conversion includes inserting the parameter block address for use by the conversion routines. • The dynamic allocation parameter block contains address and control information for the CONVERT function. It includes: - DDNAME1 -- the DDNAME currently associated with the unopened data set whose attribute needs changing. - DDNAME2 -- the new DDNAME to be associated with the data set if, for example, DAIR has specified the exchange option. - MEMBERNAME -- the member name associated with the data set if it is part of a partitioned data set.	Allocation Control Routine	IGC00099
2	Checking the validity ensures that: • DDNAME1 data set is currently closed. • DDNAME2, if specified, contains no invalid characters, is in the TIOT if the EXCHANGE option was selected, or, that is not in the TIOT if the exchange option was not selected. • Membername, if specified, contains no invalid characters. An internal subroutine performs the name checking.	Validity Checking Routine	IGC16099 NAMECK
3	Saves the SIOTTR associated with DDNAME1 in the TTRSIOT1 field of DAWTVARY for later use in finding the SIOT to be updated; saves DDNAME2 SIOTTTR in TTRSIOT2 field in DAWTVARY for use in processing the EXCHANGE option, if specified -- this SIOTTTR for DDNAME2 points to the SIOT for DDNAME2 and can come from either the same or a different blocked SIOTTTR record as the DDNAME1 SIOTTTR.	Validity Checking Routine	IGC16099
4	Posts an event control block so that the initiator will try to enqueue the data set for exclusive use, if the user requests this; for a successful enqueue, updates the use attribute in the data set enqueue (DSENQ) table entry for this data set to show its change in status.	Validity Checking Routine	IGC16099
5	Necessary for changes to the status or member name: if its DCB-related fields are to be cleared, turns on the bit to prevent forward merge and makes its blocksize field equal to the blocksize parameter passed in the dynamic allocation parameter block. Merges ATRCB, if specified, into JFCB.	SIOT and JFCB Updating Routine	IGC17099
6	Necessary for changes to the status, disposition, or DDNAME. (Updating two SIOTs is necessary to process the EXCHANGE option.) For warmstart processing, places the UCB address in the low-order two bytes of the SCTUTYPE field. This UCB is the one associated with the data set involved in the attribute conversion request.	SIOT and JFCB Updating Routine	IGC17099
7	Necessary if the updating of the SIOT included a change of DDNAME; updates the DDNAME in the TIOT DD entry that corresponds to the same data set whose SIOT DDNAME entry was updated (see 6 above).	SIOT and JFCB Updating Routine	IGC17099
8	Invokes the queue management routines to write the updated records to the SYS1.SYSJOBQE data set.	SIOT and JFCB Updating Routine	IGC17099
9	Normally, passes control to IGC25099 for DSE updating; errors during processing cause an immediate return to DAIR by SVC 3 with a return code as shown in Figure 38.	SIOT and JFCB Updating Routine	IGC17099

Method of Operation Diagram 23. Converting Data Set Attributes (Part 2 of 2)

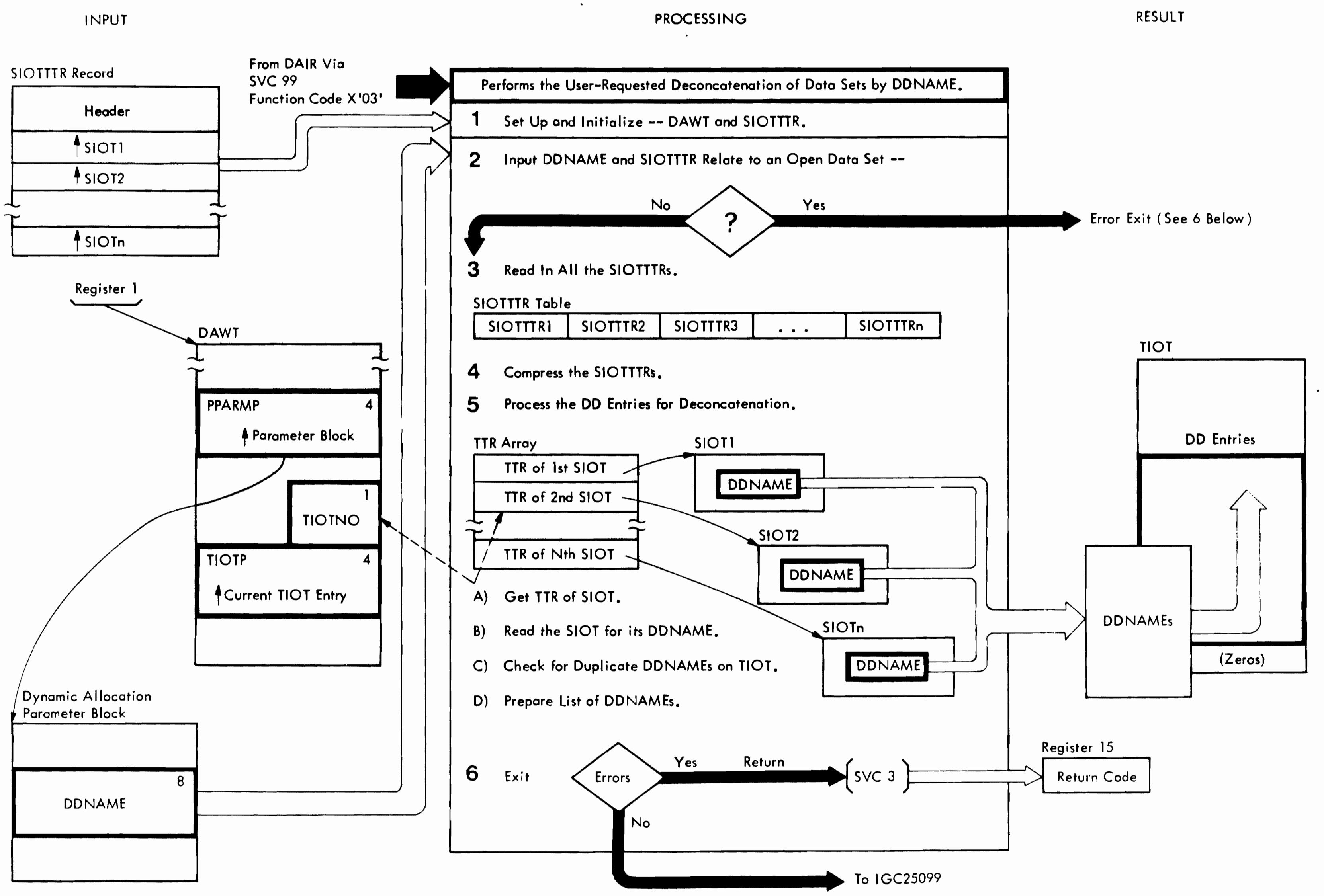


Method of Operation Diagram 24. Concatenating Data Sets (Part 1 of 2)

CROSS - REFERENCE FOR CONCAT

Key	Description	Routine	Label
1	• The input SIOTTTR (relative track address record) is associated with the SIOT containing the first user-supplied DDNAME in the dynamic allocation parameter block from DAIR. See the description for the parameter block below. • Preparing the DAWT for concatenation includes inserting the parameter block address for use by the concatenation routines. • The dynamic allocation parameter block includes an eight-byte DDNAME for every DD statement that refers to a data set to be concatenated.	Allocation Control Routine	IGC00099
2	CATLST is obtained to contain the addresses of the TIOT entries corresponding to the input DDNAMEs in the parameter block. These addresses provide access to the appropriate SIOTTTR (see 6 below). Uses each DDNAME to locate the proper old TIOT entry; moves the entry address to CATLST in the requested concatenation order.	CATLIST Construction Routine	IGC18099
3	The contents of the addresses in CATLST become the first entries in the new TIOT; the N field in DAWTVARY records the number of these new entries; moves these entries in conjunction with the use of CATTAB, as described in 4 below.	CATLST Construction and TIOT Building Routines	IGC18099
4	CATTAB correlates the building of the new TIOT with the possible need to update the SIOTTTR chain to reflect the new concatenation order, and to prepare for updating a data control block, if necessary. It contains a twelve-byte entry, as illustrated, for each entry in CATLST (and hence for each TIOT DD entry and each data set). Fields are as follows: • ENTHND -- A flag turned on to show that movement of the associated CATLST entry to the new TIOT is complete. • OLDISP and NEWDISP -- Addresses of the associated DD entry in the old and new TIOTs, respectively; emplacing these addresses occurs during movement of the CATLST entry for this data set; inequality between them indicates (see 5 and 6 below) that SIOTTTR rearrangement is necessary to reflect the concatenation order in the new TIOT. • NEWPOS -- Position of the data set in the new TIOT. • DCBUP -- Turned on when the data set associated with this entry is open and the OLDISP and NEWDISP fields are unequal; signals that preparation of a DCB update parameter list is necessary for the data set; OPENNUMB in DAWTVARY is incremented for each such data set marked. During the initialization of CATTAB, its location is recorded in the PCAT pointer in DAWTVARY. The routine moves the CATLST entries into the new TIOT; marks NEWPOS in each associated CATTAB entry; each of the DD entries is rounded out by using the address and control information in CATTAB to move the remainder of the DD entries to the new TIOT.	TIOT Building Routine	IGC19099
5	Invokes transient queue management routines to read in SIOTTTRs for necessary rearrangement, as indicated by unequal OLDISP and NEWDISP fields in the corresponding CATTAB entry (see 4 above).	TIOT Building Routine	IGC19099
6	(Each SIOTTTR contains the TTR for a single SIOT; each SIOT is related by DDNAME to a specific TIOT DD entry, and therefore to the corresponding data set and CATTAB entry; hence, each TTR available via the read-in relates directly to a specific data set in the concatenated group and must therefore reflect the order within the group.) Provides an array of TTRs that can easily be accessed, indexed, and rearranged, excerpting the SIOTTTR headers and compressing the residual TTRs into contiguous storage; rearranges the TTRs to reflect the concatenation order.	TIOT Moving Routine	IGC20099
7	Spreads the array to decompress it and moves the SIOTTTR headers back in; invokes the transient queue management routines to read the records back to the job queue (SYS1.SYSJOBQE data set).	TIOT Moving Routine	IGC20099
8	Moving the new TIOT over the old one completes concatenation.		
9	Rearrangement of any open data set DD entries in the TIOT during concatenation requires the construction of a parameter list for the DCB update routine; the open data sets are earmarked by the DCBUP flag turned on in CATTAB entries during processing of their CATLST entries (see 4 above) and by their respective unequal OLDISP and NEWDISP entries; preparation of the list is as illustrated, with the old and new TIOT entry offsets taken respectively from the OLDISP and NEWDISP fields in the appropriate CATTAB entry.	TIOT Moving Routine	IGC20099
10	Normally, passes control to IGC25099 for DSE updating (via the DCB update routine, IGC35099, if the concatenation moved the TIOT DD entries for open data sets); errors during processing cause an immediate return to DAIR by SVC 3 with a return code as shown in Figure 38.	TIOT Moving Routine	IGC20099

Method of Operation Diagram 24. Concatenating Data Sets (Part 2 of 2)

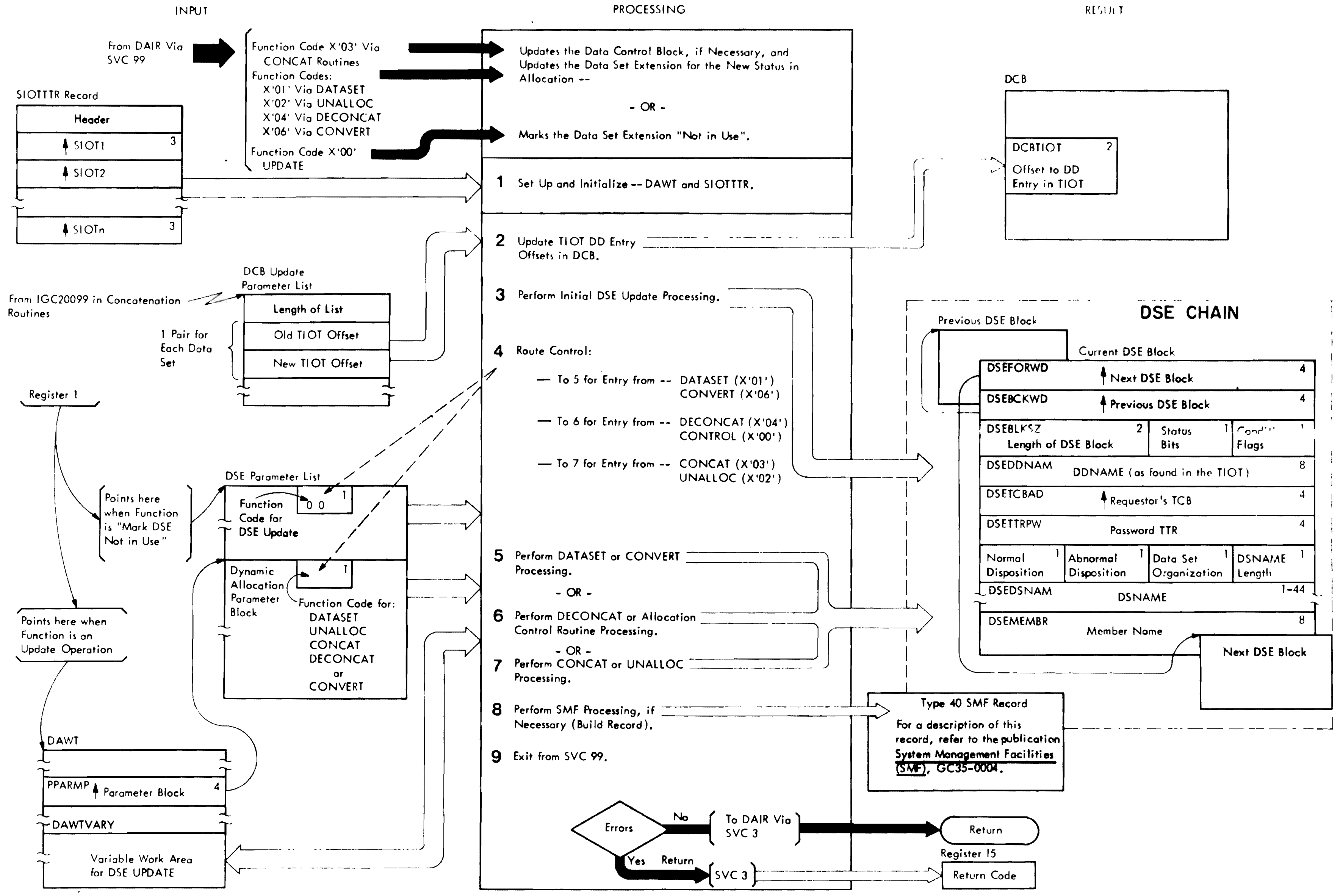


Method of Operation Diagram 25. Deconcatenating Data Sets (Part 1 of 2)

CROSS-REFERENCE TABLE FOR DECONCAT

Key Description	Routine	Label
1 • The input SIOTTTR (relative track address record) is associated with the SIOT containing the user-supplied DDNAME (see 2 below) in the dynamic allocation parameter block from DAIR.	Allocation Control Routine	IGC00099
• Preparing the DAWT for deconatenation includes inserting the parameter block address and the TIOT entry address associated with the input DDNAME.		
• The TIOTP and TIOTNO fields in DAWT relate to the input DDNAME as follows --		
- TIOTP -- TIOT address for the entry containing the DDNAME of the concatenated group of data sets.		
- TIOTNO -- Denotes the SIOTTR position for the SIOT containing the input DDNAME; this SIOT is associated with the TIOT DD entry that contains the same DDNAME; the SIOTTTR position is where the TTR for the SIOT resides in a table of SIOTTTRs.		
2 • This DDNAME is associated with the group of concatenated data sets that the user wishes to deconcatenate; it occupies the first TIOT DD entry for all the data sets in the group; the DDNAME fields in the DD entries for the succeeding data sets in the group are blank. At the end of processing, there is a separate, named DD entry for each data set; this new structure relates each data set to a single DD entry, thereby providing the requested deconcatenation.	Deconcatenation Routine	IGC23099
• The input SIOTTTR points to the SIOT associated with the input DDNAME (that is, with the first of the data sets in the concatenated group). The data set associated with this SIOT must not be open because allocation changes to open data sets can cause errors -- checks each new entry during processing (via the reading of the SIOTs -- see 5 below) to make sure it is not open.		
3 Invokes the queue management routines to read in the SIOTTTRs after getting the main storage to contain them.		
4 Compresses the SIOTTTRs by excerpting their headers and squeezing the remaining TTRs together into contiguous space -- results in a single table of TTRs that can be easily indexed by using the incrementable value in TIOTNO.		
5 Deconcatenation consists of putting DDNAMEs from the SIOT into their respective blank DD entries in the TIOT. Processing for each DD entry includes:	Deconcatenation Routine	IGC23099
a) Indexing the array of TTRs for the appropriate SIOT; TIOTNO provides the indexing factor for using the compressed TTR table to locate the SIOT.		
b) Reading the SIOT for the DDNAME it contains; a particular SIOT correlates with each DD entry in the TIOT and provides the DDNAME that deconcatenates the entry from the group.		
c) Checking to see that the SIOT DDNAME is not the same as one already in the TIOT (duplicate DDNAMEs in the TIOT is an error situation).		
d) Preparing a list of DDNAMEs valid for movement into the DD entries. This processing constitutes a loop in which one circuit processes one DD entry and its associated SIOT. Within this loop there is another small loop (see C above) to check the TIOT entries for a duplicate DDNAME. This possibility arises each time a SIOT turns up with a non-blank DDNAME; a duplicate DDNAME is invalid for entry into the list of DDNAMEs to be moved to the TIOT. The termination of the big loop is at either:		
• The next non-blank DD entry (the end of the concatenated group).		
• The end of the TIOT (a word of zeros).		
6 Normally, passes control to IGC25099 for DSE updating; errors cause a return to DAIR by SVC 3 with a return code in register 15, as shown in Figure 38.		

Method of Operation Diagram 25. Deconcatenating Data Sets (Part 2 of 2)

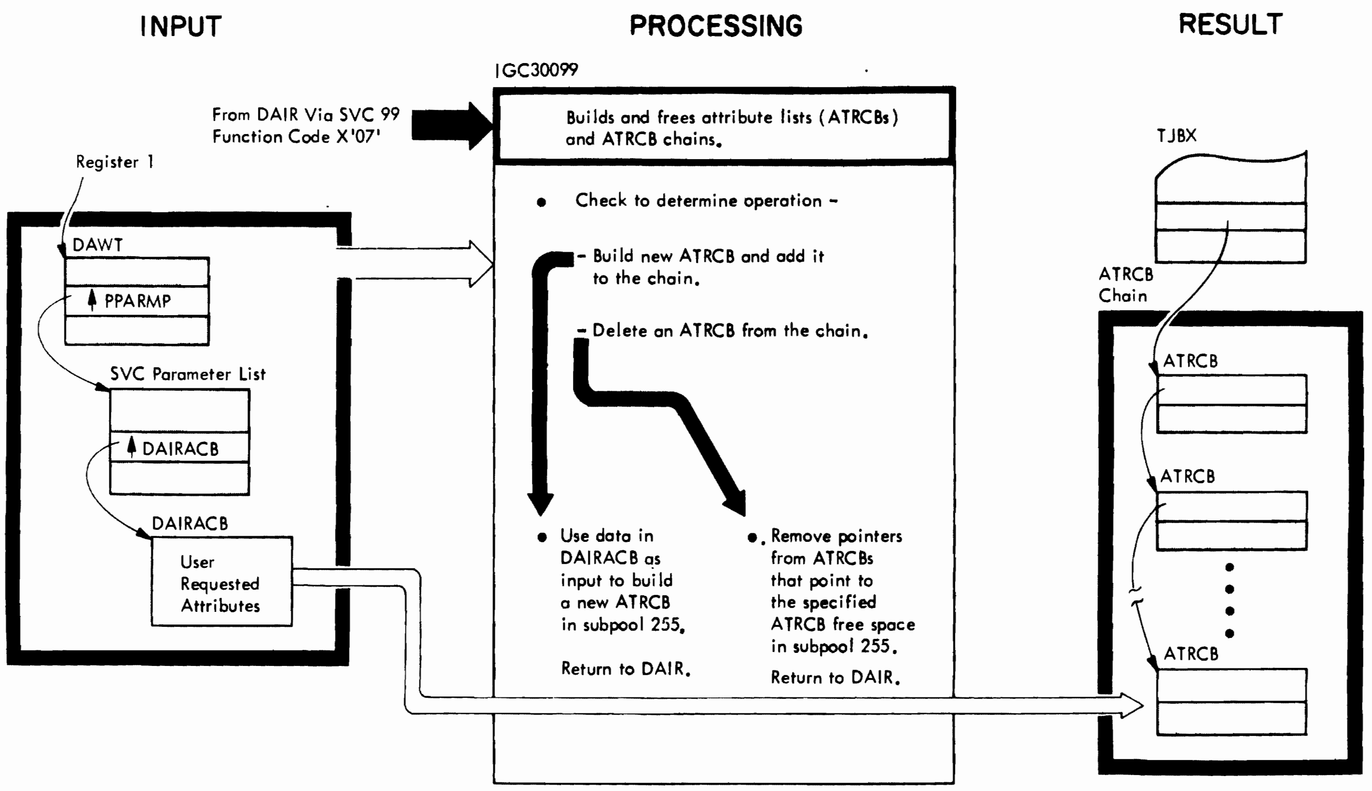

INPUT
PROCESSING
RESULT
From DAIR Via SVC 99
Function Code X'03' Via CONCAT Routines
Function Codes:
X'01' Via DATASET
X'02' Via UNALLOC
X'04' Via DECONCAT
X'06' Via CONVERT
Function Code X'00' UPDATE
Updates the Data Control Block, if Necessary, and Updates the Data Set Extension for the New Status in Allocation --
- OR -
Marks the Data Set Extension "Not in Use".
1 Set Up and Initialize -- DAWT and SIOTTTR.
2 Update TIOT DD Entry Offsets in DCB.
3 Perform Initial DSE Update Processing.
4 Route Control:
— To 5 for Entry from -- DATASET (X'01') CONVERT (X'06')
— To 6 for Entry from -- DECONCAT (X'04') CONTROL (X'00')
— To 7 for Entry from -- CONCAT (X'03') UNALLOC (X'02')
5 Perform DATASET or CONVERT Processing.
- OR -
6 Perform DECONCAT or Allocation Control Routine Processing.
- OR -
7 Perform CONCAT or UNALLOC Processing.
8 Perform SMF Processing, if Necessary (Build Record).
9 Exit from SVC 99.
Errors
No
Yes
To DAIR Via SVC 3
Return
SVC 3
Return
Register 15
Return Code
SIOTTTR Record
Header
SIOT1 3
SIOT2
SIOTn 3
DCB Update Parameter List
From IGC20099 in Concatenation Routines
Length of List
1 Pair for Each Data Set
Old TIOT Offset
New TIOT Offset
Register 1
Points here when Function is "Mark DSE Not in Use"
DSE Parameter List
Function Code for DSE Update
0 0 1
Dynamic Allocation Parameter Block
1
Function Code for:
DATASET
UNALLOC
CONCAT
DECONCAT
or
CONVERT
Points here when Function is an Update Operation
DAWT
PPARMP Parameter Block 4
DAWTVARY
Variable Work Area for DSE UPDATE
DCB
DCBTIOT 2
Offset to DD Entry in TIOT
DSE CHAIN
Previous DSE Block
Current DSE Block
DSEFORWD Next DSE Block 4
DSEBCKWD Previous DSE Block 4
DSEBLKSZ 2 Length of DSE Block
Status 1 Bits
Cond'l 1 Flags
DSEDDNAM DDNAME (as found in the TIOT) 8
DSETCBAD Requestor's TCB 4
DSETTRPW Password TTR 4
Normal 1 Disposition
Abnormal 1 Disposition
Data Set 1 Organization
DSNAME 1 Length
DSEDSNAM DSNAME 1-44
DSEMEMBR Member Name 8
Next DSE Block
Type 40 SMF Record
For a description of this record, refer to the publication System Management Facilities (SMF), GC35-0004.

Method of Operation Diagram 26. Updating the DCB and DSE (Part 1 of 2)

CROSS REFERENCE TABLE FOR UPDATE

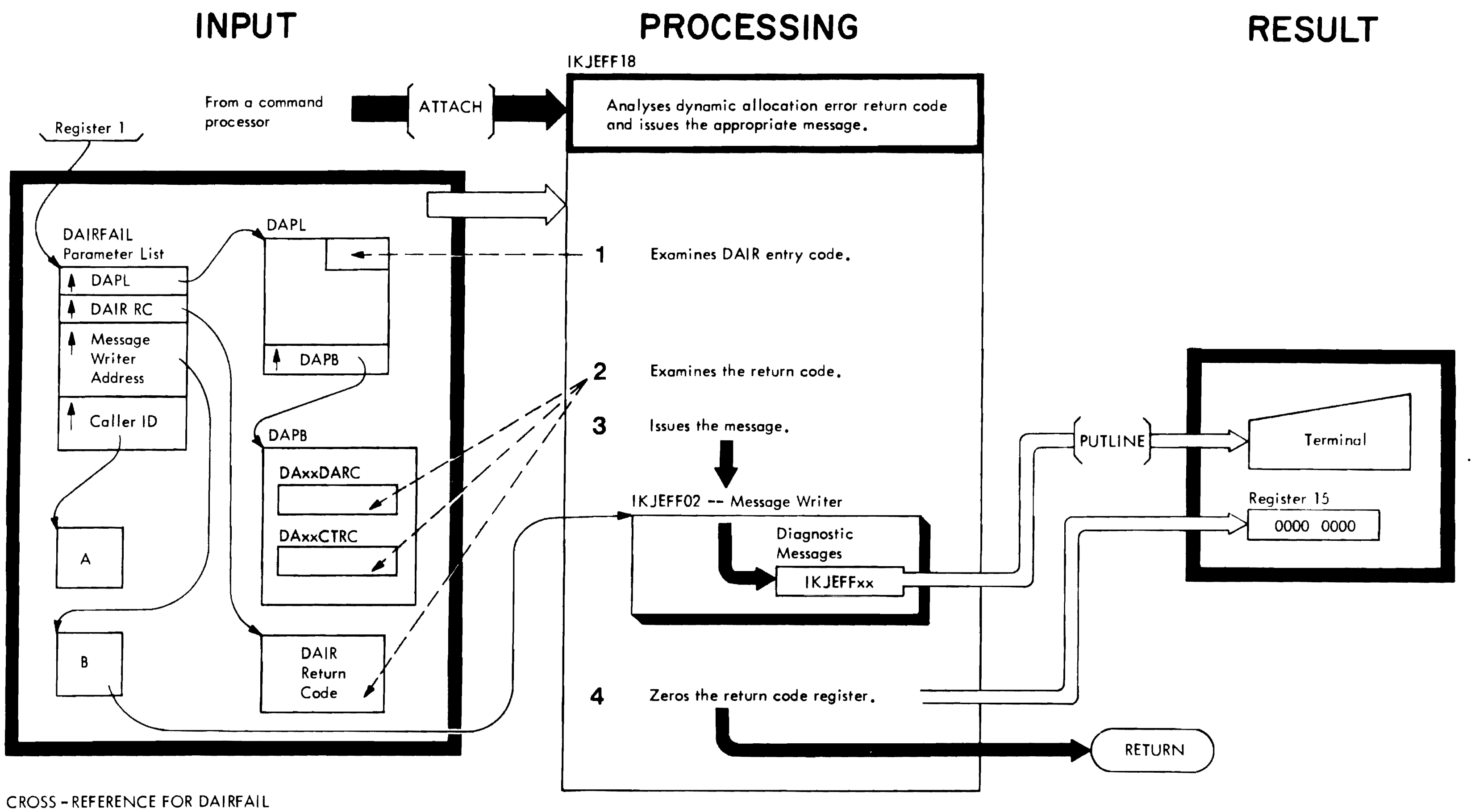
Key Description	Routine	Label
1 • The input SIOTTTR (relative track address record) contains the pointer (TTR) to the SIOT containing the first user-supplied DDNAME in the dynamic allocation parameter block from DAIR. • Preparing the DAWT includes inserting the parameter block address for use by the update routines and reading the input SIOTTTR (above) into DAWA2. • The dynamic allocation parameter block contains the address and control information necessary for the UPDATE function to perform the requested operations on the data set extension (DSE) for the user-specified data set. See "Section 5: Data Areas" for the complete description of the UPDATE (function code X'00') parameter block. • When the function code is X'00', marking the DSE "not in use" is the only updating necessary. In this case - - Register 1 points to the DSE parameter list. - There is no dynamic allocation parameter block for any other function. - Control passes directly from DAIR through the allocation control routine to IGC25099 for updating.	Allocation Control Routine	IGC00099
2 Optional operation performed when concatenation of data sets has resulted in rearrangement of TIOT DD entries for open data sets. DCB update parameter list furnished by concatenation routines provides the old and new offsets into the TIOT for each such rearranged entry. DCB update routine uses old offset of each pair to find the proper DCB to update, then updates with the new offset.	DCB Update Routine	IGC35099
3 • Determines which function passed control. • Searches the DSE chain for the input DDNAME(s) in the parameter block. • Partially initializes a new DSE block for a newly allocated data set, if control is from the DATASET function; this includes - - Obtaining the space for it. - Calculating its size and annotating DSEBLKSZ accordingly. - Setting the DSEFORWD and DSEBCKWD chain pointers. - Moving in the DDNAME from DAWTVARY. - Setting the status and disposition fields from the DATASET parameter list. - Moving the DSNAME and its length from the parameter block to DSEDSLNG and DSEDSNAM fields, respectively.	DSE Update Routine	IGC25099
4 For a function code other than X'00', control transfers to other routines in the UPDATE function as shown.	DSE Update Routine	IGC25099
5 Refer to the module operation description for IGC26099 for an outline of this processing. sing.	DATASET and CONVERT Update Routine	IGC26099
6 Refer to the module operation description for IGC27099 for an outline of this processing. ssing.	DECONCAT and Control Routine Update Routine	IGC27099
7 Refer to the module operation description for IGC29099 for an outline of this processing. ssing.	UNALLOC and CONCAT Update Routine	IGC29099
8 Necessary only if SMF is active in the system; refer to the module operation description for IGC28099 for an outline of this processing. ssing.	SMF Exit Routine	IGC28099
7 The normal and error exits from the SVC 99 routines is to DAIR by the EXIT macro instruction (SVC 3) at the conclusion of update processing. See Figure 38 for applicable return codes.	(Update routines)	IGC26099 IGC27099 IGC28099 IGC29099

Method of Operation Diagram 26. Updating the DCB and DSE (Part 2 of 2)


INPUT
PROCESSING
RESULT
IGC30099
From DAIR Via SVC 99
Function Code X'07'
Builds and frees attribute lists (ATRCBs) and ATRCB chains.
Register 1
DAWT
PPARMP
SVC Parameter List
DAIRACB
DAIRACB
User Requested Attributes
• Check to determine operation –
– Build new ATRCB and add it to the chain.
– Delete an ATRCB from the chain.
• Use data in DAIRACB as input to build a new ATRCB in subpool 255.
Return to DAIR.
• Remove pointers from ATRCBs that point to the specified ATRCB free space in subpool 255.
Return to DAIR.
TJBX
ATRCB Chain
ATRCB
ATRCB
ATRCB

Method of Operation Diagram 27. Building and Freeing ATRCBs

4



CROSS-REFERENCE FOR DAIRFAIL

A -- Caller identification: (Halfword)	ID = 2 for the FREE Command Processor. ID = 1 for any other command processor.
B -- Message writer address: (Fullword)	This field contains either the address of the message writer control section (IKJEFF02) or the address of a word of zeros; if the word contains zeros, the message writer is not currently in main storage, and the DAIRFAIL analyser (IKJEFF18) must load it.

Method of Operation Diagram 28. Analyzing Dynamic Allocation Error Codes (Part 1 of 2)

CROSS - REFERENCE TABLE FOR DAIRFAIL

Key	Description	Routine	Label
1	The DAIR entry code indicates the function that produced the error return, for example, a failure in allocating a data set by dsname.	DAIRFAIL Analyser	IKJEFF18
2	The applicable return code may be one of three types, as follows: • DAIR return code -- indicates the DAIR function failure only; is meaningful only when DAIR does not invoke SVC 99 or the catalog management routines to perform the function. • Dynamic allocation (SVC 99) return code -- located in the DAxxDARC field of the applicable DAIR parameter block (DAPB) created initially for DAIR by the calling command processor. Each DAIR function has its own unique DAPB, whose field names are prefaced by DAxx, where xx is the entry code for that function: for example, DA08DARC is the SVC 99 return code field for dynamic allocation of a data set by dsname. • Catalog management return code -- located in the DAxxCTRC field of the applicable DAPB. The naming conventions for this field are similar to those for the DAxxDARC field (which immediately precedes it in the DAPB), as explained above. DAIR invokes catalog management after dynamic allocation functions that affect the cataloging of data sets.		
3	The return code indexes a table of message frameworks in the DAIRFAIL TSMSG control section. It also provides the means for selecting applicable information to insert in the appropriate message framework.		
	When the message is complete, the message writer issues it to the terminal by the PUTLINE service routine. The DAIRFAIL message identifications are IKJEFFxx, where xx is the specific message ID.	DAIRFAIL Message Writer	IKJEFF02
4	Before returning control to the calling command processor, DAIRFAIL zeros the standard return code register to show normal completion with no outstanding error return codes for the request.	DAIRFAIL Analyser	IKJEFF18

Method of Operation Diagram 28. Analyzing Dynamic Allocation Error Codes (Part 2 of 2)

Section 3: Program Organizati)n

This section describes the organization of the dynamic allocation interface routine (DAIR), the SVC 99 routines, and the DAIR error code analyser (DAIRFAIL). It contains information about the hierarchy of the load modules, the assembly modules, and the control sections that constitute each of these groups of routines. Figures 29, 30, and 31 are graphic representations of this hierarchy.

The module operation information briefly describes the processing operations that occur within the individual routines in each group -- DAIR, SVC 99, and DAIRFAIL.

For a summary of the functions of each DAIR subroutine, SVC 99 routine, and DAIRFAIL routine, refer to the Directory in Section 4.

Program Hierarchy

The DAIR service routine has only one load module, IKJEFD00, as shown in Figure 29. The load module includes the following major routines:

DAIRCTRL	- Initializes the DAIRWA, routes control to the appropriate DAIR routine.
DAIR00	- Searches the DSE chain for information about a data set.
DAIR04	- Searches the DSE chain and system catalog, if necessary, for information about a data set.
DAIR08	- Allocates a data set by dsname.
DAIR0C	- Concatenates data sets by ddname.
DAIR10	- Deconcatenates data sets by ddname.
DAIR14	- Searches the system catalog for qualifiers for a dsname.
DAIR18	- Frees a data set.
DAIR1C	- Allocates a data set to a terminal.
DAIR24	- Allocates a data set by ddname or dsname.
DAIR28	- Performs a list of operations.
DAIR2C	- Marks DSE entries not in use for the specified task.
DAIR30	- Allocates a SYSOUT data set.
DAIR34	- Builds and frees attribute lists (ATRCBs) and ATRCB chain.

ATTRSRCH -	Searches the ATRCB chain for invalid duplicate attr-list-names.
SEARCH -	Searches the DSE Chain for information about a data set.
GENDDN -	Generates a ddname of the form 'SYSnnnnn', where nnnnn is a count in a TSO control block, the environment control table (ECT).
USERID -	Prefixes userid to dsname.
EXITCODE -	Routes control from one DAIR routine to another. Loads return code and returns control to calling program.

Figure 30 shows the organizational interrelationship of the SVC 99 routines, while Figure 33 shows their functional grouping and lists the common name for each.

In response to requests for dynamic allocation, DAIR gets information from the DSE and, if necessary, issues SVC 99 to invoke dynamic allocation. DAIR uses register 1 to pass dynamic allocation the address of a parameter block that contains one of the dynamic allocation entry codes. This entry code tells the allocation control routine what the requested function is. The control routine, in turn, routes control to the routine appropriate for beginning the operations necessary to perform the requested function. Figure 32 illustrates this flow of control.

Each routine constitutes a single load module of 1024 (1K) bytes or less, according to the conventions for Type 4 SVC routines. Because of this physical size limitation, each request for SVC function may require the loading of successive modular routines. The normal passage of control among the routines is via XCTL macro instruction, while error conditions encountered during execution result in a return to the caller via the EXIT macro instruction (SVC 3.)

Figure 31 shows the hierarchy of the DAIR error code analyser (DAIRFAIL), which a user may invoke to get a diagnostic message related to a dynamic allocation failure. DAIRFAIL consists of the following routines:

IKJEFF18 -	the DAIRFAIL load module, which contains one CSECT (IKJEFF18) to control and execute the analysis of error return codes, and another CSECT (TSMSG) that contains the pertinent message segments for the construction of meaningful diagnostic messages.
IKJEFF02 -	the DAIRFAIL message writer, which formats and issues the diagnostic messages under the control of IKJEFF18.

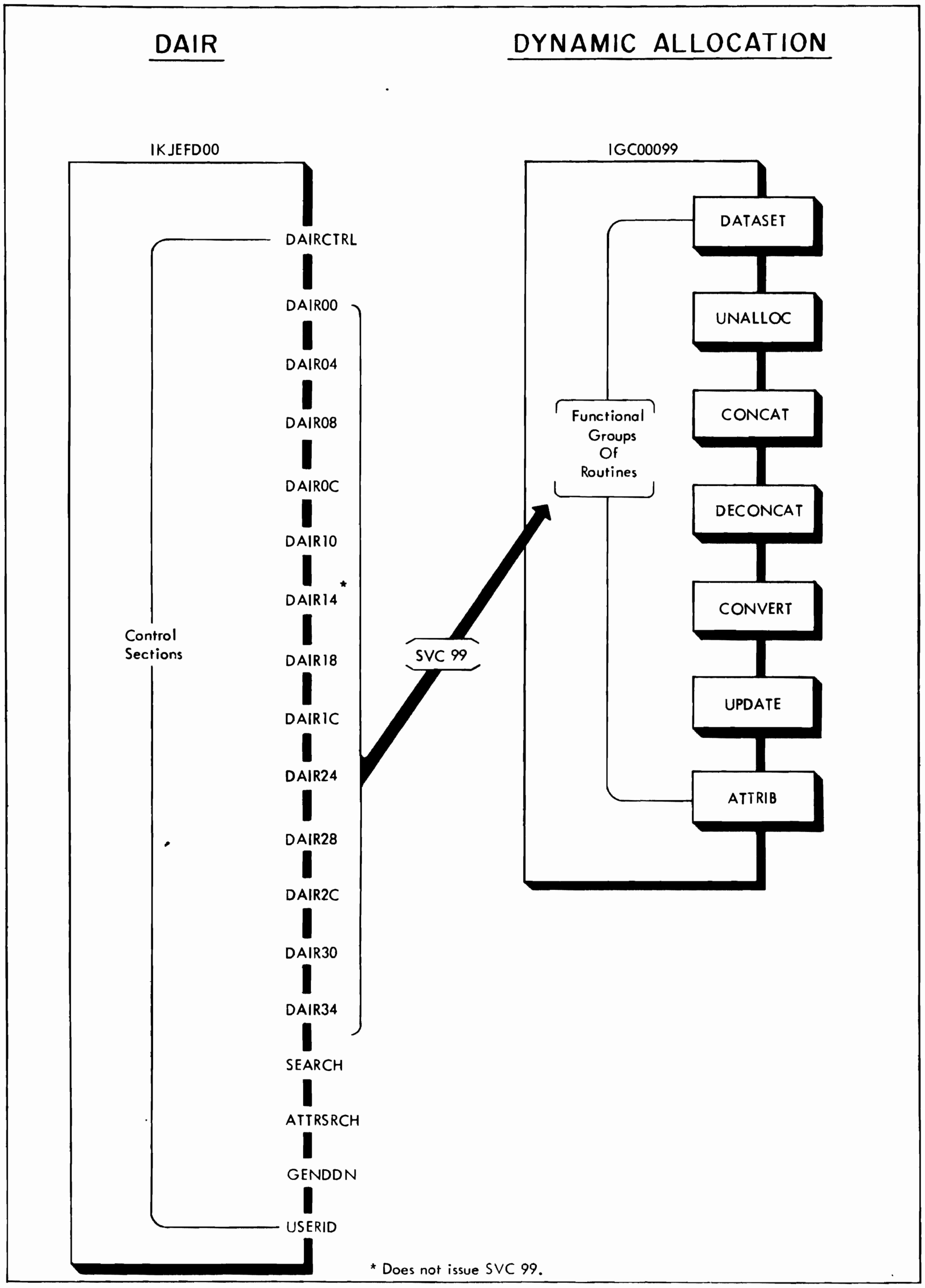


Figure 29. Program Hierarchy: Relationship between DAIR and the SVC 99 Routines

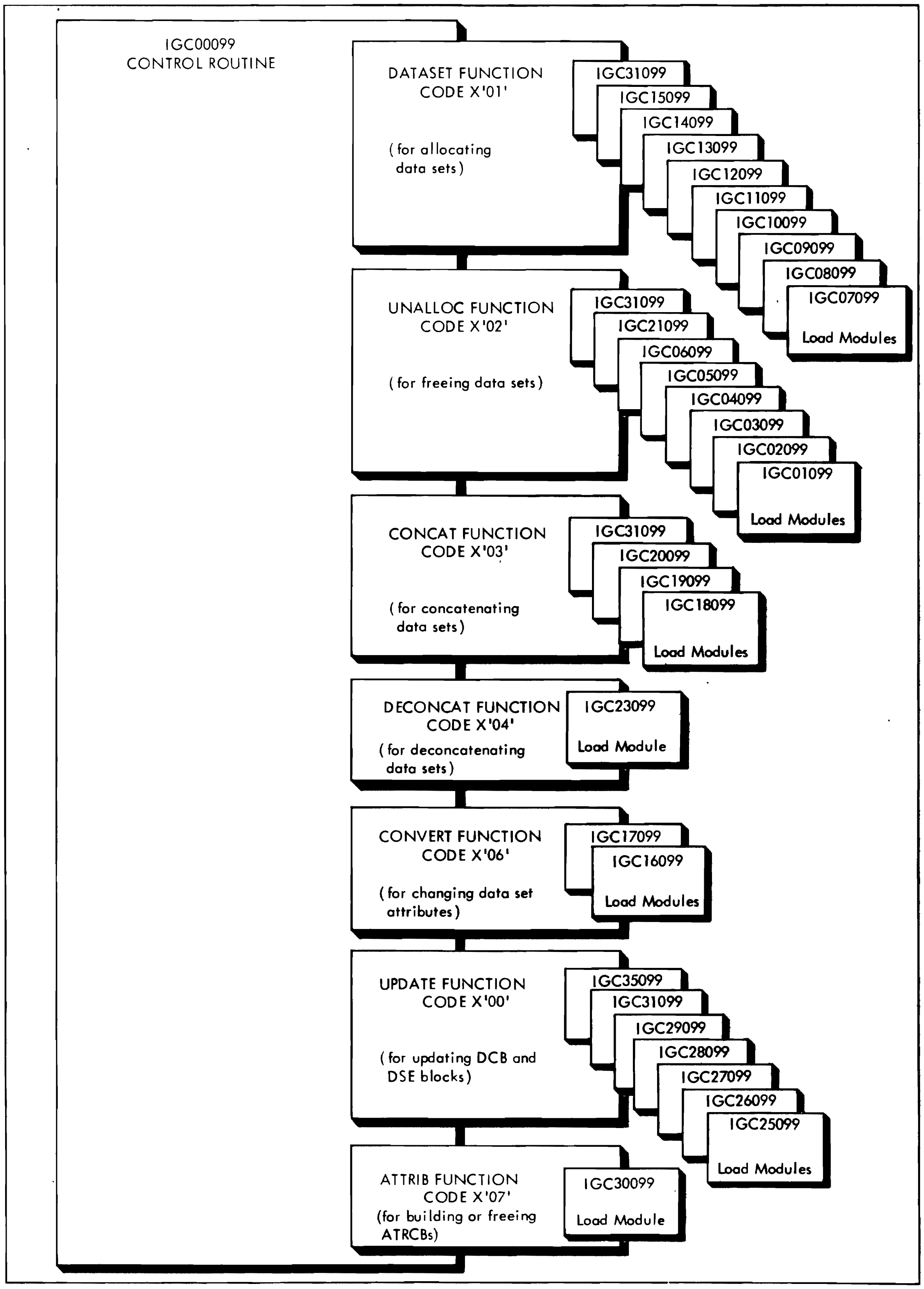


Figure 30. Program Hierarchy: SVC 99 Routines

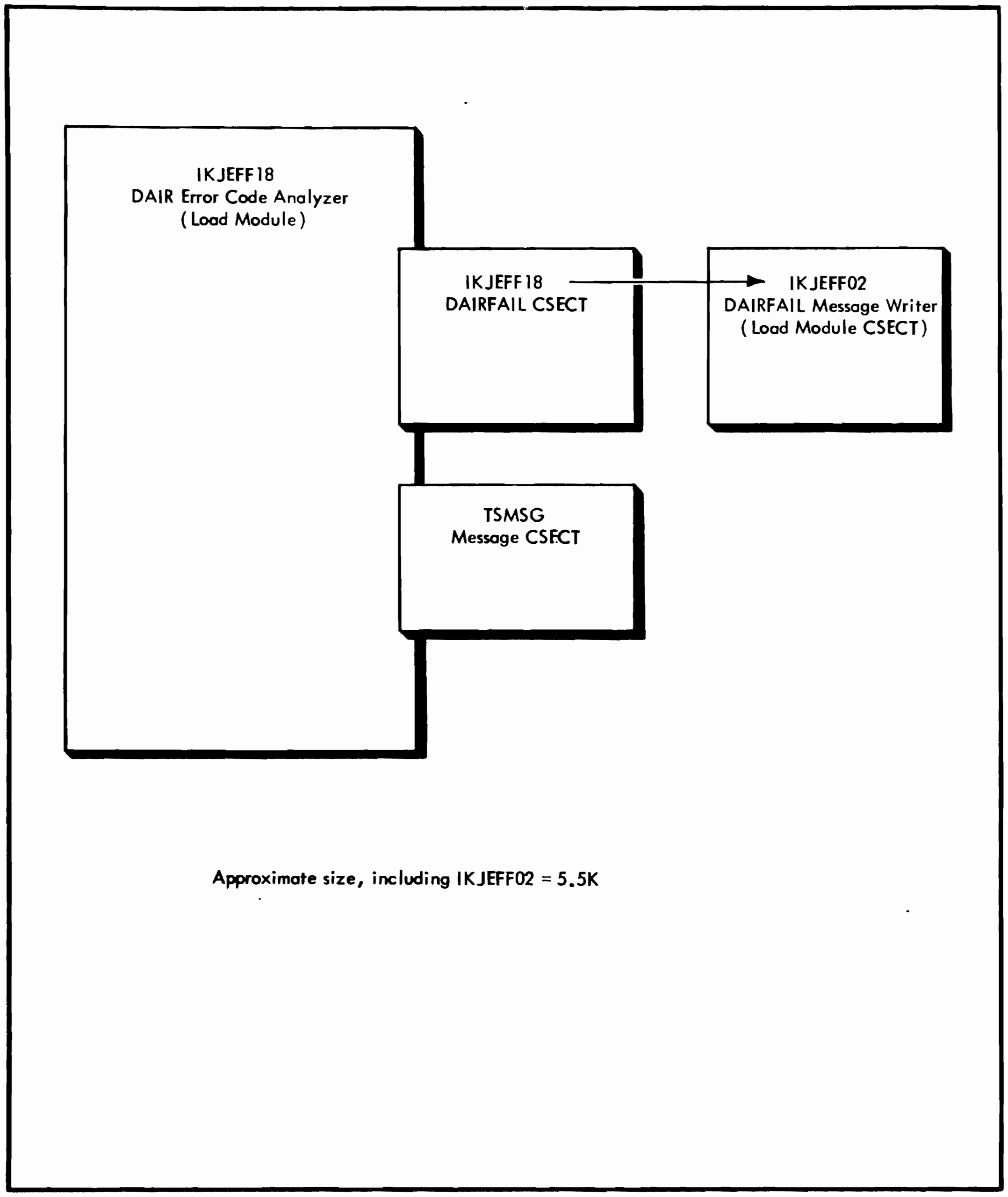


Figure 31. Program Hierarchy: DAIR Error Code Analyser

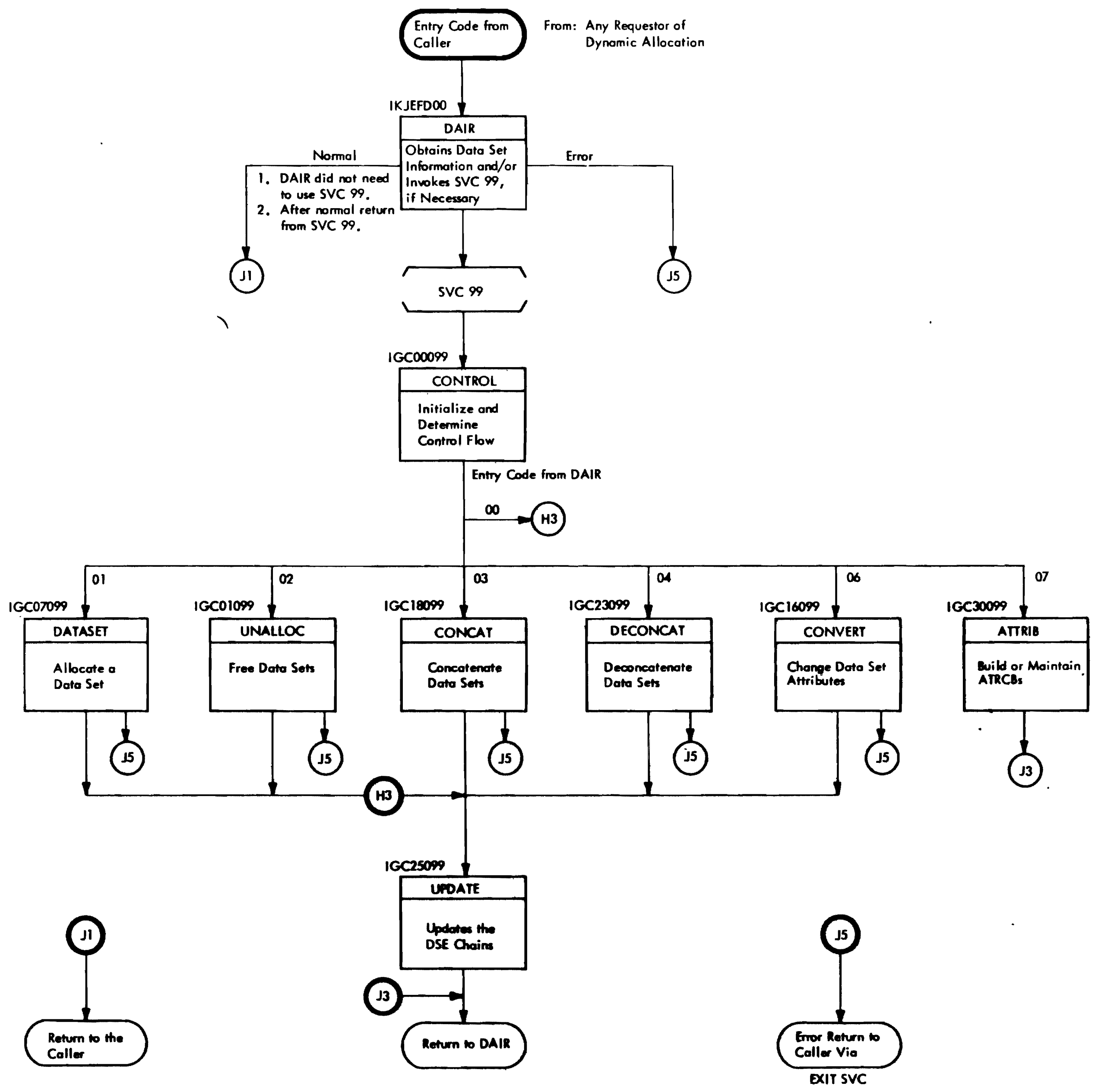


Figure 32. Flow of Control in Dynamic Allocation

Parent Function	Routine Designation	Common Routine Name
CONTROL	IGC00099	Allocation Control routine
UNALLOC	IGC01099	Validity checking routine
	IGC02099	Disposition processing routine
	IGC03099	Device freeing routine
	IGC04099	SYSOUT chain DSB processing routine
	IGC05099	SYSOUT enqueuing routine
	IGC06099	KEEP message processing routine
	IGC21099	TSO terminal KEEP message processing routine
DATASET	IGC07099	Validation and initialization routine
	IGC08099	SYSOUT processing routine
	IGC09099	SIOT and JFCB processing routine
	IGC10099	Data set enqueuing routine
	IGC11099	DSENQ update and device name table load routine
	IGC12099	Bit pattern construction routine
	IGC13099	Device reduction routine
	IGC14099	Direct access device space management routine
	IGC15099	TIOT and JFCB update routine
	--------	I/O load balancing routines (in scheduler -- IEFABxxx)
	IGC31099	TIOT open bit verification routine.
CONVERT	IGC16099	Validity checking routine
	IGC17099	SIOT and JFCB updating routine
	IGC31099	TIOT open bit verification routine
CONCAT	IGC18099	Concatenation list construction routine
	IGC19099	TIOT building routine
	IGC20099	TIOT moving routine
DECONCAT	IGC23099	Deconcatenation routine
UPDATE	IGC25099	DSE update routine
	IGC26099	DATASET and CONVERT update routine
	IGC27099	DECONCAT and control routine handling routine
	IGC28099	SMF Exit routine
	IGC29099	UNALLOC and CONCAT Update routine
	IGC31099	TIOT open bit verification routine
	IGC35099	DCB update routine
ATTRIB	IGC30099	Chain or unchain an attribute list (ATRCB).

Figure 33. Functional Grouping of the SVC 99 Routines

Module Operation

The following descriptions briefly describe the processing operations in each executable module of DAIR and SVC 99.

IKJEFD00 -- DAIR

Obtain the status of a data set by searching the DSE Chain for the appropriate information and, if necessary, invoke the SVC 99 routines to perform the following functions:

- Allocate a data set.
- Free a data set.
- Concatenate a group of data sets.
- Deconcatenate a data set from a group of data sets.
- Convert a data set from one use to another.
- Update the information in the DSE chain.
- Build or free an attribute list.

ATTRSRCH SUBROUTINE OF IKJEFD00

Calls the SEARCH routine to search the DSE chain for a ddname that matches a specified name, if necessary. Searches the ATRCB chain for an attr-list-name that matches a specified name.

GENDDN SUBROUTINE OF IKJEFD00

Generates a ddname of the form "SYSnnnnn", where nnnnn is a count in the environment control table (ECT).

SEARCH SUBROUTINE OF IKJEFD00

Searches the DSE chain for a specified ddname or a dsname. The search starts at the bottom of the chain or at the address specified in BLKPTR. The search ends at the top of the chain or at the first occurrence of the dsname if bit 9 of STATUS1 is set.

USERID SUBROUTINE OF IKJEFD00

Prefixes the userid to a specified dsname.

IGC00099 -- ALLOCATION CONTROL ROUTINE

Creates the dynamic allocation work table (DAWT), performs initialization common to each function of the SVC, and exits to the function requested with the address of the DAWT in register 1.

IGC01099 -- UNALLOC VALIDITY CHECKING ROUTINE

Obtains space for an partially fills in the 584-byte unallocation work area. Gets a 28-byte work space for enqueuing the Q5 termination resources of the operating system, if necessary. Reads in the SIOT and the JFCB. Invokes IGC31099 if the TIOTOPEN bit is on for the current unallocation request; when IGC31099 returns control, continues normal processing if TIOTOPEN is off, or starts abnormal exit processing if the bit is on.

IGC02099 -- DISPOSITION PROCESSING ROUTINE

Initializes the message and WTO buffers and makes appropriate change to the catalog for dispositions of CATALOG, UNCATALOG, and DELETE. Issues disposition message IEF283I, IEF285I, and IEF287I, as necessary.

IGC03099 -- DEVICE FREEING ROUTINE

Determines the type of data set being freed, then, appropriately, performs some combination of the following operations:

- Modifies the appropriate bits and fields of the SIOT, TIOT, and JFCB to free the data set.
- Enqueues the termination resource (Q5), modifies the UCB, and dequeues the termination resources.
- Updates the DSENQ (ampersand in the first character of the data set name for non system-generated data set names only.
- Uses the transient queue manager to maintain YSY1.SYSJOBQE and to dequeue the dsname, as necessary.
- Frees the unallocate work area and transfers control to the UPDATE function.
- Monitors any of these operation for errors, frees all main storage devoted to work areas in error situations, and subsequently returns to the user with an error code via the EXIT SVC 3.

IGC04099 -- SYSOUT CHAIN DSB PROCESSING ROUTINE

Provides interim processing for SYSOUT data sets between initial validity check (IGC01099) and disposition processing for the data set (IGC02099). Processing includes validity checks of jobname and SYSOUT class, obtaining storage for and reading in the SCT and/or JCT, as appropriate for the disposition and message class, using the transient queue management routines to place a null SMB on SYS1.SYSJOBQE, and assigning two records from SYS1.SYSJOBQE when the disposition parameter is not DELETE.

IGC05099 -- SYSOUT ENQUEUING ROUTINE

Places a complete KEEP message (IEF280E) in the message buffer for direction to SYSOUT and the operator console only, and issue it; or partially constructs the TSO terminal KEEP message depending upon user requirements, for completion by IGC21099. Clears fields in the UCB so that the volume may be demounted.

IGC06099 -- KEEP MESSAGE PROCESSING ROUTINE

Initializes a DSB using information from the SIOT, TIOT, user parameter list, and SMF information from the JMR when SMF is active for the user's job. Writes the DSB and the JFCB using the records assigned in IGC04099 to SYS1.SYSJOBQE and enqueues the data set on the proper output class by using the jobname from the parameter list, if available, or from the TIOT.

IGC07099 -- DATASET VALIDATION AND INITIALIZATION ROUTINE

Checks the validity of the input to the function of allocating data sets.

Contains the NAMECK subroutine, which validates the character strings used as input as the table of allowable characters shows below:

Position of the Character	DSNAME	DDNAME MEMBERNAME PROGRAMNAME	SYSOUT Class
First Character	A - Z & # $	A - Z @ $	A - Z 0 - 9
Not First Character	A - Z 0 - 9 . @ # $ - 12-0 punch	A - Z 0 - 9 @ # $	N/A

IGC08099 -- SYSOUT PROCESSING ROUTINE

Processes SYSOUT data sets, and generates a dsname whenever necessary

IGC09099 -- SIOT AND JFCB CONSTRUCTION ROUTINE

Assigns a record for the JFCB, if necessary. Initializes the SIOT and saves it in the SIOT buffer. Initializes the JFCB from parameters in the parameter list or in an ATRCB.

IGC10099 -- DATA SET ENQUEUING ROUTINE

Searches the records of the DSENQ table to see if it si necessary to enqueue the data set name to the task. Performs one of the following operations based upon the results of the table search:

- Dsname already appears on the table -
  - <u>With exclusive use attribute</u> -- ENQ is not necessary, and the subsequent exit is to IGC11099.
  - <u>With share attribute</u> -
    1) Request is for shared use -- ENQ is not necessary, and the subsequent exit is to IGC11099.
    2) Request is for exclusive use -- Builds the dsname ENQ parameter list with the RFT=CHNGE option; passes control to IGC11099.

- Dsname not found -- builds E Q parameter list and performs the enqueue by posting an ECB (E B15 to start the initiator. Posting ECB1 includes passing the in tiator the address of the ENQ/DEQ parameter list. When posted by SVC 99, the initiator issues the enqueue macro instruction while SVC 99 waits. When the macro instruction returns control, the initiator posts an ECB (ECB2) with the macro return code. This code indicates to SVC 99 and the WAIT macro instruction it issued whether or not enqueuing the data set was successful. This processing is called "the WAIT/POST mechanism with the initiator"; SVC 99 acts upon the results as follows:

  - Successful execution -- Adds a new dsname entry to the DSENQ table and passes control to IGC11099.
  - Error -- Returns control to the user by EXIT SVC 3.

Also obtains and initializes main storage for the allocation/termination dequeue parameter list before passing control to IGC11099.

IGC11099 -- DSENQ UPDATE AND DEVICE NAME TABLE LOAD ROUTINE

Writes the updated DSENQ record to the SYS1.JOBQE data set and performs any necessary chaining. Loads the device name table to convert the unit name to a device type. Deletes the device name table and transfers control to IGC12099.

IGC12099 -- BIT PATTERN CONSTRUCTION ROUTINE

Loads the device mask table (DMT) and scans it for an entry with a device type that matches the one in the DAWT. Passed the address of the entry that it may find by the scan to IGC13099. Otherwise, deletes the DMT and obtains space for, initializes, and builds the bit pattern in the pattern construction area (PCA). (See Section 5 for a description of the PCA.) Passes the completed bit pattern to IGC13099 if it finds any UCBs that meet the device type requirements for the request. Finding no suitable UCBs results in an error return.

IGC13099 -- DEVICE REDUCTION ROUTINE

Builds a list of UCBs (the I/O load balancing candidate list) that can satisfy the request for allocation in --

- DAWA2 field of DAWTVARY, for 43 or fewer potentially eligible candidates.
- Subpool 253 for more than 43 candidates.

The candidates may be those represented by either -

- The specific unit address passed by IGC11099 to indicate which IOS lookup table entries will yield potential candidates.
- The bit pattern passed by IGC12099 to represent eligible candidates.

IGC14099 -- DIRECT ACCESS DEVICE SPACE MANAGEMENT ROUTINE

Invokes the scheduler I/O load balancing routines to optimize the I/O load with respect to the UCB candidates for the current allocation request. (The load balancing routines read the candidate list received from IGC13099 and select the best candidate from it.)

Invokes DADSM by SVC 32 to allocate direct access space for the best device candidate, as determined by load balancing. If DADSM fails for the current candidate, invokes load balancing to select another candidate. This interaction between load balancing and DADSM continues until DADSM is successful, or until either the candidate list is exhausted or load balancing returns a code for WTO/WTP to indicate insufficient storage for load balancing to operate.

With load balancing inoperative, invokes DADSM to allocate space directly from the candidate list. This process continues until DADSM is successful, or until the candidate list is exhausted, in which case, returns a code that reflects the first candidate for which DADSM was unsuccessful. This return code reaches the user through IGC15099, the exit routine from the DATASET function.

IGC15099 -- TIOT AND JFCB UPDATE ROUTINE

Upon entry from IGC09099 (DUMMY or TERM data sets):

Writes the JFCB and TIOT to SYS1.SYSJOBQE and moves the new ddname to the TIOT. Updates fields in the TIOT as follows --

- TIOTEDYNM = '0'B
- TIOESTTA = '00'x
- TIOESTTB = '00'X
- TIOESTTC = '00'X
- TIOSUSED = '0'B
- TIOEFSRT = '000000'X
- TIOTTERM = '1'B (if this is a terminal request)

Passes control to IGC25099 by XCTL when TIOT updating is complete.

Upon normal entry from IGC13099 or IGC14099:

For a request to change the enqueue use attribute from "share" to "exclusive", enqueues the data set by using the WAIT/POST mechanism with the initiator. (See the description for this under "Operation" in the module description for IGC10099.)

- Successful enqueue -
  - Reads the correct record of the DSENQ table.
  - Updates the dsname use attribute.
  - Writes the updated table to SYS1.SYSJOBQE.

- Unsuccessful enqueue - error return to the user with return code X'20C'.

Places the volume serial number of the allocated device into the JFCB, then writes the JFCB and SIOT to the SYS1.SYSJOBQE data set. Updates the UCB or subUCB, in the case of 2311 devices) as follows --

- Increments the user count (SRTEUSER/DCELUSER).
- Sets the public volume attribute if it is not already set.
- Turns on the "allocated" bit (SRTEALOC).

Dequeues the allocation/termination resource (Q4 and Q5) and frees the ENQ/DEQ parameter list. Updates the TIOT as describes above except that it sets TIOEFSRT with the address of the correct UCB. Issues message IEF237I if allocation messages are requested and transfer control to IGC25099.

4

Upon entry from IGC13099 or IGC14099 for error processing:

- If the data set name has been enqueued -
  - Dequeue the data set name using the WAIT/POST mechanism with the initiator. (See module description for IGC10099 -- initiator issues DEQ macro instruction rather than ENQ.)
  - Reads the last record of the DSENQ table and overlays the first byte of the dsname with an ampersand.
  - Writes the updates DSENQ record.

- Frees the main storage obtained for the data set name enqueue parameter list, if any storage was obtained previously.

- Dequeues the allocation/termination resources (Q4 and Q5) and frees the corresponding parameter list.

- Frees the SIOT buffer and the DAWT.

- Places the return code in register 15 and exits to the user.

IGC16099 -- CONVERT VALIDITY CHECKING ROUTINE

- Waits for the reading of the SIOTTTR initiated in IGC00099.
- Checks the validity of ddnames and member names by using a subroutine.
- Saves the TTR of the SIOT for the first ddname (ddname1 from the TIOT extension) in the DAWTVARY portion of the DAWT.
- Determines whether saving a second TTR is necessary. (This is the case when the user specifies the "exchange" option and the TTRs for the ddname1 and ddname2 SIOTs are not in the same TTR record.)
- Saves either this second SIOTTTR or the SIOTTTR of the SIOT for ddname2 in DAWTVARY.
- Change the use attribute for the data set to "exclusive", if necessary, and updates the DSENQ table to reflect this change.
- Invokes IGC31099 if the TIOTOPEN bit is on for the current attribute conversion request; when IGC31099 returns control, continues normal processing if TIOTOPEN is off, or starts abnormal exit processing if the bit is on.

IGC17099 -- SIOT AND JFCB UPDATING ROUTINE

Determines which control blcoks are necessary to satisfy the requests on the caller's parameter list, and obtains them. Updating the control blocks may include the following operations:

JFCB - Changing the status of the member name.

- Clearing the DCB-related fields -- This operation also involves turning on the bit that prevents a forward merge, and setting the blocksize to the user's specifications. If the input parameter list contains the address of an ATRCB, the routine overlays the DCB parameters with the user's DCB parameter selections in the ATRCB.

SIOT - Changing the status, disposition, or ddname.

- Placing the UCB address in the low-order two bytes of the unit type field (SCTUTYPE).

- Obtaining, and updating as above, a second SIOT - this one for the second ddname supplied via the user's selection of the "exchange" option.

TIOT - Changing the ddname(s) to agree with corresponding changes in the SIOT(s).

Writes the updated records to SYS1.SYSJOBQE and transfers control to IGC25099.

IGC18099 -- CONCATENTATION LIST CONSTRUCTION ROUTINE

- Waits for the reading of the SIOTTTR initiated in OGC00099.
- Gets space for and builds the concatenation list (CATLST), which consists of the addresses of the entries to be concatenated in the order that they are to be concatenated.
- Determines the number of entries for, and the length of, the new TIOT.
- Gets main storage for the TIOT and the concatenation table (CATTAB -- see Section 5) based on the numbers determined above.
- Prepares CATTAB by recording the old addresses of each TIOT entry.
- Begins constructing the new TIOT using the addresses of the entries from CATLST, and by turning on the "entry handled" flag (ENTHND) in CATTAB for each entry it processes.
- Invokes IGC31099 if the TIOTOPEN bit is on for the current concatenation request; when IGC31099 returns control, continues normal processing if TIOTOPEN is off, or starts abnormal exit processing if the bit is on.

IGC19099 -- TIOT BUILDING ROUTINE

- Examines each CATTAB entry and moves those not previously moved to the TIOT.
- Turns on the DCBUP flag in the CATTAB entry if the data set associated with the corresponding TIOT entry is open. This indicates that updating the DCB for this data set may be necessary after its TIOT entry is moved. (See the description of operation for IGC35099.)
- Increments OPENNUMB in DAWTVARY each time it earmarks an open data set for updating to show the number of such data sets.
- Checks the OLDISP and NEWDISP entries in CATTAB for any inequality when the new TIOT is complete. The inequalities, if there are any, reflect changes to the position of the entries in the new TIOT, if there are no changes, and transfers control to IGC25099.
- • Obtains main storage for and calls the transient queue manager to read in the SIOTTTRs, if any rearrangement of the entries has taken place. Then it transfers control to IGC20099.

4

IGC20099 -- TIOT MOVING ROUTINE

- Compresses the SIOTTTRs into a single array for indexing and rearranges them to reflect the new structure of the TIOT by using the old position (OLDPOS) and new position (NEWPOS) entries in the CATTAB.
- Decompresses the SIOTTTRs, moves the header back, and writes them to the SYS1.SYSJOBQE data set.
- Determines by examining the DCBUP bit whether updating the DCB is necessary (IGC19099 turned the bit on if updating may be necessary).

- Moves the new TIOT over he old one, frees main storage, and transfers control to IGC 5099 if the bit is off in all CATTAB entries.
- Gets main storage for and prepares the DCB update parameter list, whose address it p.aces in register 1. Moves the new TIOT over the old one. Frees main storage (except for the DAWT and the update parameter list), places the address of the DAWT into register 0, and transfers control to IGC35099 if the bit is on in any CATTAB entry.

IGC21099 -- TSO TERMINAL KEEP MESSAGE PROCESSING ROUTINE

Constructs the KEEP message (IEF280E) for TSO terminals according to requirements passed from IGC06099, determines which TSO terminals are to receive the message, and directs the message to the terminals as well as to the SYSOUT device and to the operator's console.

IGC23099 -- DECONCATENATION ROUTINE

- Waits for the reading of the SIOTTTTR that IGC00099 (the control routine) initiated, and checks for reading errors.
- Reads in all the SIOTTTRs and compresses then into a single array that it can index. Points to the TIOT entries specified for deconcatenation in the array, and performs processing to create a list of ddnames from the associated SIOTs and to check for errors. The resultant list comprises ddnames that are valid for deconcatenation.
- Verifies that the TIOTs involved in the deconcatenation are not for open data sets.
- Moves the ddnames from the list to the appropriate TIOT entries to accomplish the specified deconcatenation operations.
- Transfers control to IGC25099 for DSE updating.

IGC25099 -- DSE UPDATE ROUTINE

- Places the JFCB address contained in the DD1ADD field of DAWTVARY into the DAW1P field also.
- Transfers control to other routines as follows:

  - To IGC27099 if the entry is via the allocation control routine or the deconcatenation routine.
  - To IGC29099 if the entry is via the freeing or the concatenation routines.

- Otherwise, processing continues via the DSESERCH subroutine, which looks for ddname(s) that the user supplies via either the DATASET or the CONVERT function parameter lists. Reaction to the search is as follows:

  - <u>Unsuccessful search</u> (no ddname found) -- causes control to transfer to IGC26099 with a return code of X'504' in the DAWT.
  - <u>Successful search</u>:

    1) When entry is from the attribute conversion routines (CONVERT function), control passes to IGC26099.
    2) When entry is from the allocation routines (DATASET function), the routine calculates the size of the new DSE block, obtains storage for it, initializes it, and then transfers control to IGC26099.

IGC26099 -- DATA SET AND CONVERT UPDATE ROUTINE

- Moves the TCB address in from the parameter list and turns on the "dynamically allocated" bit; invokes the DSECHAIN subroutine to place the DSE to the top of the chain of DSEs; moves in the status; determines whether entry is from the DATASET or the CONVERT function (through IGC25099) and performs one of the following operations, accordingly:

  - DATASET - Places the device type and the volume serial number on which the allocated data set resides into the DATASET parameter list.

  - CONVERT -

    1) Exchange option specified -- Swap the positions of ddname1 and ddname2 in their respective DSEs.
    2) Normal and conditional dispositions specified -- Moves the specified dispositions from the CONVERT parameter list to the ddname1 DSE.
- Processing the DSE for permanent allocation: turns on the "permanently allocated" bit.
- Processing the DSORG field in the DSE if the field is zero:
  - For old data sets - Moves the DSORG field in from the DSE update parameter list.
  - For new data sets -
    1) Sets the field "PO" if there is a directory quantity for the data set. Moves in the membername, if supplied.
    2) Sets the field "PS", otherwise.

- Processing the DSE when the user specifies a password:
  If there is no TTR in the DSE (DSETTRPW field), the routine issues the PROTECT SVC and places the returned TTR in the DSE. Then it releases the old DSE (for the DATASET function) and exits to the caller.

4

IGC27099 -- DECONCAT AND CONTROL ROUTINE HANDLING ROUTINE

- Entry from the allocation control routine (IGC00099):
  - Marking the DSE entries as not in use -
    1) Indicate by a DSLNGTH of zero in the input parameter list.
    2) Bit 31 in the input parameter list indicates that the TCB address field contains the address of the single DSE that is to be marked not in use.
    3) Bit 30 in the input parameter list indicates that all DSE's except those allocated to the current task and its ancestors are to be marked not in use.
    4) Otherwise the TCB address becomes the search argument for the DSESERCH subroutine, which determines the proper DSE to mark.

  - Processing for dsname updating -
    1) Places the address of the dsname in the DSNADD field of DAWTVARY in response to a nonzero indication in DSLENGTH.
    2) Uses DSNADD as the search argument for the DSESERCH subroutine, which looks for the proper dsname.
    3) For password data sets, ensures that any available password TTR, or password itself, is in both the DSE and the DSEUPDATE parameter list. Uses the password via the PROTECT SVC 98 to attempt to obtain the password TTR for the parameter list if it is otherwise unavailable.

• Exit from the Allocation Control routine entry is always a return to the caller by SVC 3 after freeing the DAWT.

- Entry from the deconcatenation routines (DECONCAT):
  • Restores the dds of the concatententated ddnames -
    1) Uses the DSESERCH subroutine to look for matches to the search argument ddname in each successive DSE block on the chain, until it finds a block with a mismatch.
    2) Moves the ddname from the TIOT entry corresponding with the next succeeding DSE block on the chain into each DSE block that produced a match. This operation performs the deconcatenation annotation.

  • Turns off the dynamically concatenated bit for each ddname deconcatedated as above.

  • Makes the exit determination depending upon whether SMF is active in the system.

IGC28099 -- SMF EXIT ROUTINE

Two operations:

1) Dynamically maintains the TCTIOT after dynamic allocation processing of either data set allocation or concatenation functions.

2) Builds a Type 40 SMF record after the freeing concatenation, or deconcatenation functions have executed.

IGC29099 -- UNALLOC AND CONCAT UPDATE ROUTINE

• Invokes the DSESERCH subroutine to look for the DSEs in inverse order from which their associated ddnames appear in the input parameter list.
• Invokes the DSECHAIN subroutine to attach each such DSE as it is found to the beginning of the DSE chain.
• Marks each such DSE as dynamically concatenated.
• Propagates the first ddname (ddname1) through the ddname fields of each such marked DSE.

- Entry from the freeing routine (UNALLOCe -

• Invokes the DSESERCH subroutine to look for the freed ddname.
• Fills all but the first ten bytes of the DSE with zeros, marks it as "DYNAM", moves in the ddname, and sets the dsname field to "NULLFILE". dsname
• Invokes the DSECHAIN subroutine to attach the DSE to the end of the chain.

Determines whether SMF is in the system, and exits appropriately.

IGC30099 -- ATTRIBUTE LIST HANDLING ROUTINE

Creates ATRCBs.
Places ATRCBs on the ATRCB chain.
Removes ATRCBs from the ATRCB chain.

IGC31099 -- TIOT OPEN BIT VERIFICATION ROUTINE

Checks the bit which indicates in the TIOT whether or not the control block is for a currently open data set. Prohibits, by means of this verification, the execution of SVC 99 functions on open data sets.

IGC35099 -- DCB UPDATE ROUTINE

Updates the DCB TIOT offset after the dynamic concatenation routines have rearranged the TIOT entries for open data sets.

IKJEFF18 -- DAIR ERROR CODE ANALYSER (DAIRFAIL)

- Examines error return codes from dynamic allocation --
  - From DAIR,
  - From SVC 99, or
  - From the catalog management routines.
- Constructs an appropriate informational message based on the error code.
- Invokes a special message writer to issue the message to the terminal by a PUTLINE macro instruction.

Module Overview

The following illustrations show the flow of control among the modules in each function of SVC 99.

4

Chart 01. DATASET Function Control Flow Overview

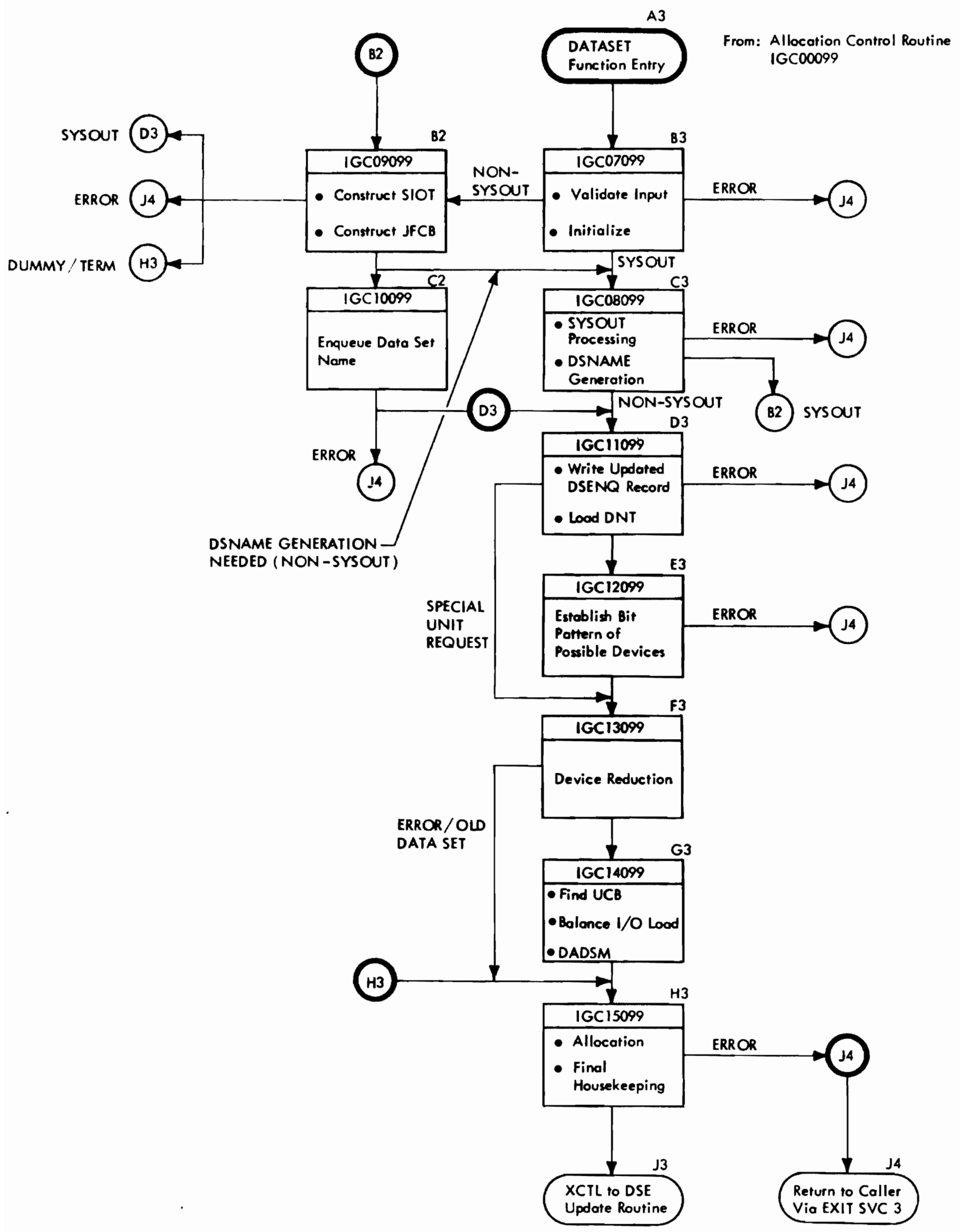
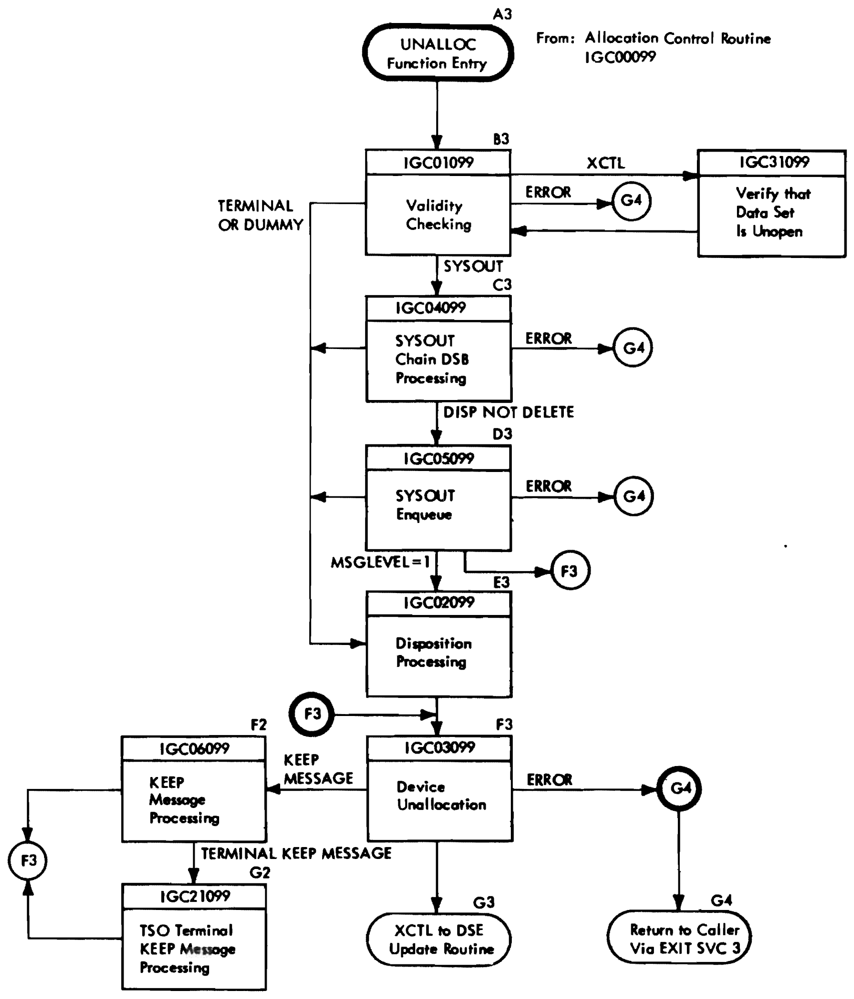


Chart 02. UNALLOC Function Control Flow Overview



4

Chart 03. Convert - Control Flow Overview

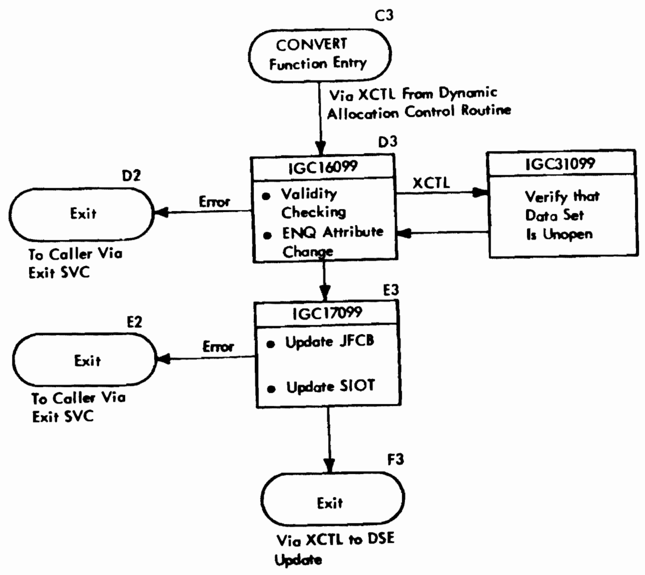


Chart 04. Concatenation Function Control Flow Overview

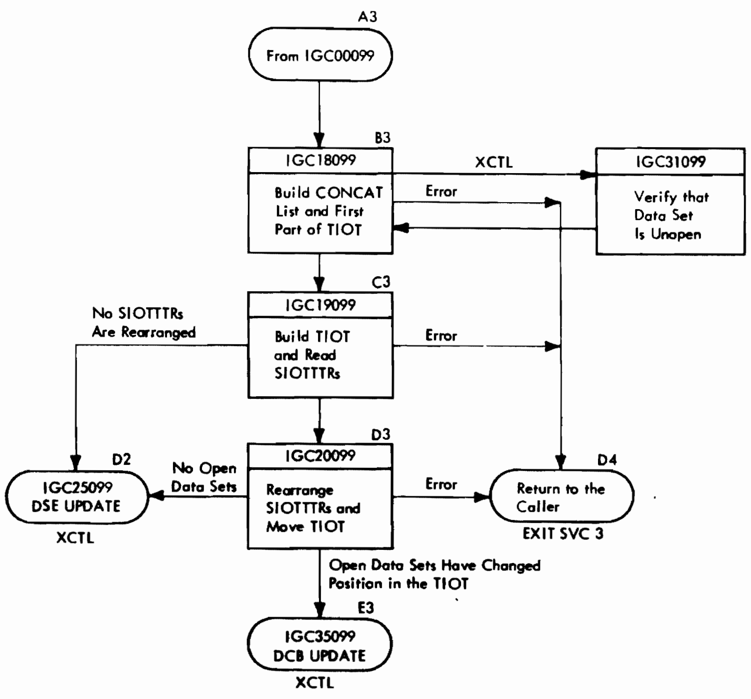

A3
From IGC00099
B3
IGC18099
Build CONCAT List and First Part of TIOT
XCTL
Error
IGC31099
Verify that Data Set Is Unopen
C3
IGC19099
Build TIOT and Read SIOTTTRs
No SIOTTTRs Are Rearranged
Error
D3
IGC20099
Rearrange SIOTTTRs and Move TIOT
D2
IGC25099 DSE UPDATE
XCTL
No Open Data Sets
Error
D4
Return to the Caller
EXIT SVC 3
Open Data Sets Have Changed Position in the TIOT
E3
IGC35099 DCB UPDATE
XCTL

Chart 05. DECONCAT and ATTRIB Control Flow Overview

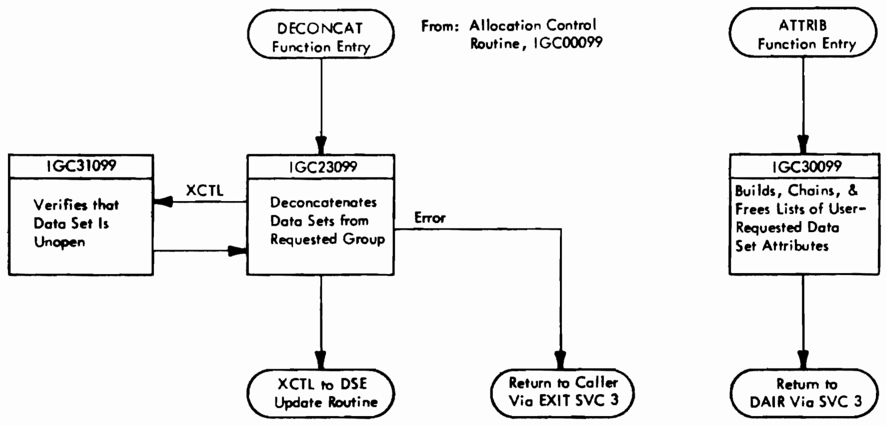
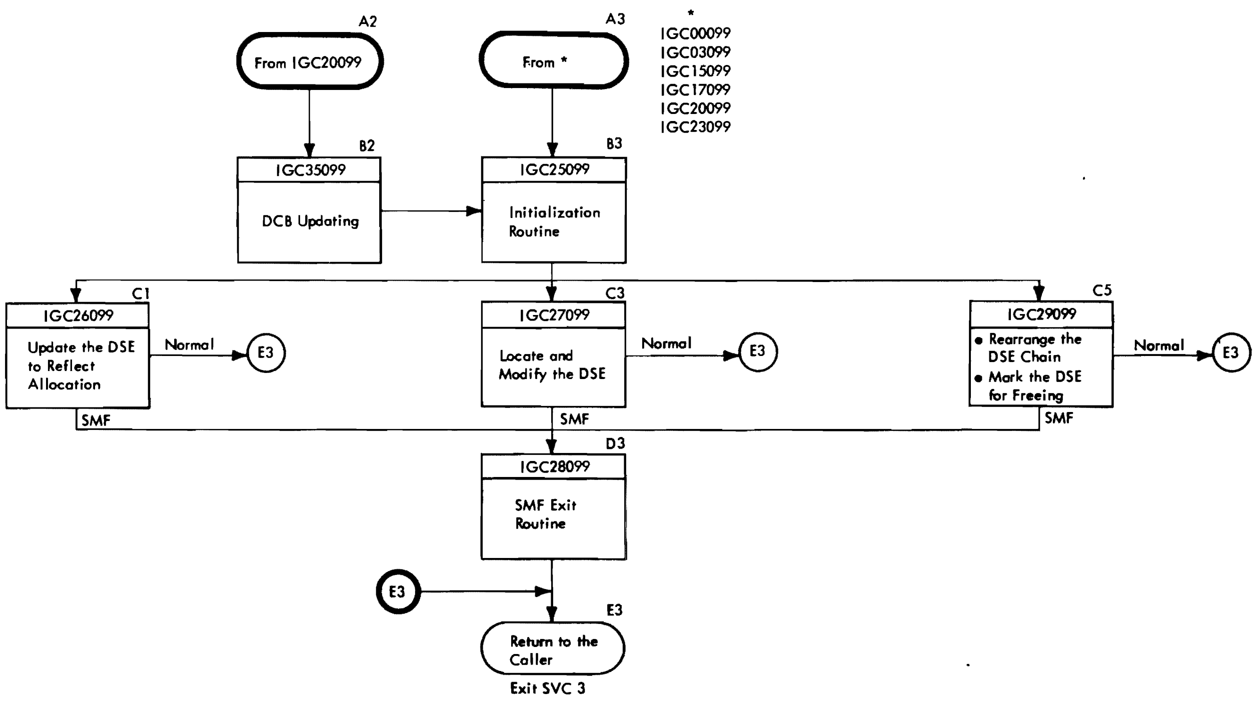

DECONCAT Function Entry
From: Allocation Control Routine, IGC00099
ATTRIB Function Entry
IGC31099
Verifies that Data Set Is Unopen
XCTL
IGC23099
Deconcatenates Data Sets from Requested Group
Error
IGC30099
Builds, Chains, & Frees Lists of User-Requested Data Set Attributes
XCTL to DSE Update Routine
Return to Caller Via EXIT SVC 3
Return to DAIR Via SVC 3

Chart 06. Update Function Control Flow Overview



4

Section 4: Directory

These directories contain information to help you find the appropriate program description, or assembler listing. They correlate information from three sources:

- The source code.
- The executable load modules.
- This manual.

DAIR DIRECTORY

Label	Common Name	Load Module Name	Assembly Module Name	Control Section Name	Description	Diagram
ATTRSRCH	-------	IKJEFD00	IKJEFD00	IKJEFD00	Search ATRCB chain for name.	
CHECKON	-------	IKJEFD00	IKJEFD00	IKJEFD00	Flowchart label.	20
CKMEM	-------	IKJEFD00	IKJEFD00	IKJEFD00	Flowchart label.	20
CKMEMORG	-------	IKJEFD00	IKJEFD00	IKJEFD00	Flowchart label.	20
CONVTEST	-------	IKJEFD00	IKJEFD00	IKJEFD00	Flowchart label.	20
DAIRCTRL	Control Routine	IKJEFD00	IKJEFD00	IKJEFD00	Initializes DAIRWA, routes control to DAIR routine corresponding to entry code.	20
DAIR00	Code X'00' Routine	IKJEFD00	IKJEFD00	IKJEFD00	Search for data set in DSE Chain.	20
DAIR04	Code X'04' Routine	IKJEFD00	IKJEFD00	IKJEFD00	Search for data set in DSE chain and system catalog.	20
DAIR08	Code X'08' Routine	IKJEFD00	IKJEFD00	IKJEFD00	Allocate data set by dsname	20
DAIR0C	Code X'0C' Routine	IKJEFD00	IKJEFD00	IKJEFD00	Concatenate data sets by ddname.	20
DAIR10	Code X'10' Routine	IKJEFD00	IKJEFD00	IKJEFD00	Deconcatenate data sets by ddname.	20
DAIR14	Code X'14' Routine	IKJEFD00	IKJEFD00	IKJEFD00	Search for qualifiers for dsname.	20
DAIR18	Code X'18' Routine	IKJEFD00	IKJEFD00	IKJEFD00	Unallocate a data set.	20
DAIR1C	Code X'1C' Routine	IKJEFD00	IKJEFD00	IKJEFD00	Allocate a data set to the terminal.	20
DAIR24	Code X'24' Routine	IKJEFD00	IKJEFD00	IKJEFD00	Allocate a data set by ddname or dsname.	20
DAIR30	Code X'30'	IKJEFD00	IKJEFD00	IKJEFD00	Allocate a SYSOUT data set.	20
DAIR34	Code X'34' Routine	IKJEFD00	IKJEFD00	IKJEFD00	Perform attribute list operation.	

(continued)

4

Label	Common Name	Load Module Name	Assemk .y Module Name	Control Section Name	Description	Diagram
EXITCODE	Exit Processing Routine	IKJEFD00	IKJEFD00	IKJEFD00	Routes control from one DAIR routine to another Loads return code and returns to caller.	20
GENDDN	Generate ddname	IKJEFD00	IKJEFD00	IKJEFD00	Generates a ddname of the form 'SYSnnnnn' where nnnnn is a count in the ECT.	
USERID	Userid Routine	IKJEFD00	IKJEFD00	IKJEFD00	Prefixes userid to data set name.	20
SEARCH	Search Routine	IKJEFD00	IKJEFD00	IKJEFD00	Search the DSE chain for information about a data set.	20

Label	Common Name	Load Module Name	Assembly Module Name	Control Section Name	Description	Diagram
IGC00099	Allocation Control Routine	IGC0009I	IGC00099	IGC00099	Creates DAWT, performs appropriate initialization, and routes control to the requested function.	20
IGC01099	Validity Checking Routine	IGC0109I	IGC01099	IGC01099	Performs validity checks for the freeing function, gets and initializes the UNALLOC work area.	22
IGC02099	Disposition Processing routine	IGC0209I	IGC02099	IGC02099	Initializes message buffers, makes appropriate disposi-changes to catalog, and issues disposition messages.	22
IGC03099	Device Freeing routine	IGC0309I	IGC03099	IGC03099	Performs the operations necessary to free the data set requested.	22
IGC04099	SYSOUT Chain DSB Processing routine	IGC0409I	IGC04099	IGC04099	Provides interim processing for SYSOUT data sets between initial validity checks and disposition processing.	22
IGC05099	SYSOUT Enqueuing routine	IGC0509I	IGC05099	IGC05099	Performs the necessary operations to enqueue the data set on the proper output class.	22
IGC06099	KEEP Msg Processing routine	IGC0609I	IGC06099	IGC06099	Completes KEEP message IEF280E for the SYSOUT device and the console, and/or partially prepares it for TSO terminal.	22
IGC07099	Validation an Initial-ization routine	IGC0709I	IGC07099	IGC07099	Performs a series of checks upon the validity of the input to the function of allocating data sets.	21
IGC08099	SYSOUT Processing routine	IGC0809I	IGC08099	IGC08099	Processes SYSOUT data sets, and generates a dsname if necessary.	21
IGC09099	SIOT and JFCB Con-struction routine	IGC0909I	IGC09099	IGC09099	Assigns a record for the JFCB, if necessary, init-ializes and saves the SIOT in a buffer, and initializes the JFCB.	21

(Continued)

Label	Common Name	Load Module Name	Assembly Module Name	Control Section Name	Description	Diagram
IGC10099	Data Set Enqueuing routine	IGC1009I	IGC10099	IGC10099	Searches the DSENQ table to ensure that the data set name is properly enqueued. If necessary, performs an ENQ.	21
IGC11099	DSENQ Update and DNT Load routine	IGC1109I	IGC11099	IGC11099	Writes and chains the updated DSENQ record, if necessary, and converts the unit name parameter to a device type using the device name table (DNT).	21
IGC12099	Bit Pattern Construction routine	IGC1209I	IGC12099	IGC12099	Uses the device mask table (DMT) or searches the UCB's to convert the device type to a bit pattern representing eligible devices.	21
IGC13099	Device Reduction routine	IGC1309I	IGC13099	IGC13099	Prepares a list of UCBs for valid allocation candidates from the bit pattern derived from processing in IGC12099.	21
IGC14099	DADSM routine	IGC1409I	IGC14099	IGC14099	Prepares a candidate attribute list and searches it for the best logical candidate for allocation; invokes DADSM to allocate the required space.	21
IGC15099	TIOT and JFCB Update routine	IGC1509I	IGC15099	IGC15099	Updates the JFCB and TIOT as necessary to complete allocation.	21
IGC16099	Validity Checking routine	IGC1609I	IGC16099	IGC16099	Checks the validity of ddnames and member names specified for the attribute conversion function.	23
IGC17099	SIOT and JFCB Updating routine	IGC1709I	IGC17099	IGC17099	Updates the JFCB, the SIOT, and the TIOT as necessary to reflect changes to the data set attributes, as specified.	23
IGC18099	Concatenation List Construction routine	IGC1809I	IGC18099	IGC18099	Builds the CATLST and CATTAB and uses them to begin modifying the TIOT.	24
IGC19099	TIOT Building routine	IGC1909I	IGC19099	IGC19099	Performs remaining necessary TIOT updating to complete the requested concatenation.	24

(Continued)

Label	Common Name	Load Module Name	Assembly Module Name	Control Section Name	Description	Diagram
IGC20099	TIOT Moving routine	IGC2009I	IGC20099	IGC20099	Determines whether DCB updating is necessary and moves the new TIOT over the old one.	24
IGC21099	TSO Terminal KEEP Processing	IGC2109I	IGC21099	IGC21099	Constructs message IEF280E and directs it to the proper terminals.	22
IGC23099	Deconcatenation routine	IGC2309I	IGC23099	IGC23099	Modifies ddnames in TIOT entries as necessary to accomplish the requested deconcatenation operation.	25
IGC25099	DSE Update routine	IGC2509I	IGC25099	IGC25099	Initializes updating operations and transfers control to appropriate modules for the specific updating required. The kind of updating required depends upon the function code.	26
IGC26099	DATASET and CONVERT Update routine	IGC2609I	IGC26099	IGC26099	Updates the DSE appropriately for entry from DATASET or CONVERT functions.	26
IGC27099	DECONCAT and Control Routine	IGC2709I	IGC27099	IGC27099	Updates the DSE appropriately for the entry from DECONCAT or the allocation control routine.	26
IGC28099	SMF Exit routine	IGC2809I	IGC28099	IGC28099	Builds a Type 40 SMF record for SMF to reflect the updating of the DSE.	26
IGC29099	UNALLOC and CONCAT Update routine	IGC2909I	IGC29099	IGC29099	Updates the DSE appropriately for the entry from the UNALLOC or CONCAT functions.	26
IGC30099	ATRCB Update routine	IGC3009I	IGC30099	IGC30099	Builds, chains, and frees ATRCBs	
IGC31099	TIOT Open Bit Verification Routine	IGC31099	IGC31099	IGC31099	Checks TIOTOPEN bit in TIOT to determine whether the request is for an operation on an open data set.	
IGC35099	DCB Update routine	IGC3509I	IGC35099	IGC35099	Updates the DCB after concatenation operations, if necessary, prior to DSE updating.	26
DSECHAIN	DSE Chain subroutine	IGC2609I	IGC25099	IGC25099	Chains DSEs appropriately for permanently allocated data sets.	25
DSESERCH	DSE Search subroutine	IGC2509I	IGC25099	IGC25099	Looks for the ddnames specified in the user's parameter list for the update function.	26

4

DAIRFAIL DIRECTORY

Label	Common Name	Load Module Name	Assembly Module Name	Control Section Name	Description	Diagram
IKJEFF18	DAIR Error Code Analyser (DAIRFAIL)	IKJEFF18	IKJEFF18	IKJEFF18	Analyses dynamic allocation error codes and prepares appropriate informational messages.	28
TSMSG	DAIRFAIL Message CSECT	IKJEFF18	IKJEFF18	IKJEFF18	Contains DAIRFAIL message frameworks.	28
IKJEFF02	DAIRFAIL Message Writer	IKJEFF02	IKJEFF02	IKJEFF02	Issues the DAIRFAIL messages.	28

Section 5: Data Areas

This section describes the major data areas that DAIR and the SVC 99 routines use, including:

Attribute control block (ATRCB)

Candidate list

Concatenation list (CATLST)

Concatenation table (CATTAB)

DAIR attribute control block (DAIRACB)

DAIR work area (DAIRWA)

DAIR parameter block, code X'00' (DAPB00)

DAIR parameter block, code X'04' (DAPB04)

DAIR parameter block, code X'08' (DAPB08)

DAIR parameter block, code X'0C' (DAPB0C)

DAIR parameter block, code X'10' (DAPB10)

DAIR parameter block, code X'14' (DAPB14)

DAIR parameter block, code X'18' (DAPB18)

DAIR parameter block, code X'1C'(DAPB1C)

DAIR parameter block, code X'24' (DAPB24)

DAIR parameter block, code X'28' (DAPB28)

DAIR parameter block, code X'2C' (DAPB2C)

DAIR parameter block, code X'30' (DAPB30)

DAIR parameter block, code X'34' (DAPB34)

DAIR parameter list (DAPL)

Data set extension block (DSE)

Dynamic allocation parameter block, code X'00'

Dynamic allocation parameter block, code X'01'

Dynamic allocation parameter block, code X'02'

Dynamic allocation parameter block, code X'03'

Dynamic allocation parameter Block, Code X'04'

Dynamic allocation parameter block, code X'05'

Dynamic allocation parameter block, code X'06'

4

Dynamic allocation parameter block, code X'07'

Dynamic allocation work table (DAWT)

Dynamic allocation work table, variable area (DAWTVARY) for CONVERT

Dyanmic allocation work table, variable area (DAWTVARY) for CONCAT

Dynamic allocation work table, variable area (DAWTVARY) for UNALLOC

Dynamic allocation work table, variable area (DAWTVARY) for UPDATE

ENQUEUE/DEQUEUE parameter list for allocation/termination resource

ENQUEUE work area (EQA) for enqueuing dsname

Pattern construction area (PCA)

SYSOUT work area (SWA)

Unallocate work area

The following information is included for each data area:

- Size in bytes.
- Name of the routine that creates it.
- Name(s) of the routine(s) that use and/or update it.
- Field names, displacements, size, and contents.
- Cross-reference to method of operation diagrams.

ATTRIBUTE CONTROL BLOCK (ATRCB)

Size:	61 bytes.
Located in:	Subpool 255.
Created by:	IGC30099 (SVC 99).
Used by:	IKJEFD00, IGG09099, IGC17099, IGC30099.
Updated by:	None.
Contents:	User requested data set attributes.

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	ATRFORWD	4	Address of next ATRCB.
4	4	ATRBCKWD	4	Address of previous ATRCB.
8	8	-------	4	Reserved.
12	C	ATRNAME	8	Attr-list-name.
20	14	ATRLNGH	2	Length of ATRCB.
22	16	ATRMASK	8	INOUT/OUTIN options of OPEN.
22	16	-------	6	Reserved.
28	1C	ATRLABEL	1	Specifies OPEN type:
		1...		INOUT.
		.1..		OUTIN.
29	1D	-------	1	Reserved.
30	1E	-------	3	Reserved.
33	21	ATREXPDT	3	Data set expiration date.
36	24	-------	2	Reserved.
38	26	ATRBUFNO	1	No. of buffers required.
39	27	ATRBFTEK	1	BFTEK, BFALN:
		.1..		Simple buffering "S"
		.11.		Automatic record area construction "A"
		..1.		Record buffering "R"
		...1		Exchange buffering "E"
		.... ..1.		Doubleword boundary "D"
		1		Fullword boundary "F"
40	28	ATRBUFL	2	Buffer length.
42	2A	ATREROPT	1	Error option:
		1...		Accept the error record.
		.1..		Skip the error record.
		..1.		Abnormal end of task.

(continued)

4

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents	
43	2B	ATRKEYLE	1	Key length.	
44	2C	-------	6	Reserved.	
50	32	ATRECFM	1	Record format:	
		1...		Fixed	"F"
		.1..		Variable	"V"
		11..		Undefined	"U"
		..1.		Track overflow	"T"
		...1		Blocked	"B"
		.... 1...		Standard blocks	"S"
		.... .1..		ASA printer chars.	"A"
		.... ..1.		Machine control character	"M"
51	33	ATROPTCD	1	Option code:	
		1...		Write validity check	"W"
		..1.		Chained scheduling	"C"
		.... 1...		ANSI translate	"O"
		.... ..1.		User totaling	"T"
52	34	ATRBLKSI	2	Maximum block size.	
54	36	ATRLRECL	2	Logical record length.	
56	38	ATRNCP	1	Max # of READ/WRITE macros before CHECK.	
57	39	-------	4	Reserved.	

CANDIDATE LIST (CANDLIST)

Size:	Variable -- 4 bytes of control information plus 4 bytes for each UCB address in the list.
Located in:	Subpool 253.
Created by:	IGC13099.
Used by:	IGC14099.
Contents:	Address and control information for allocation I/O load balancing.

Operaion Diagrams
21

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	LBFCNDE	1	Bit settings, as follows: Bit Meaning when on 0 Retry at candicate selection; if zero, the first attempt at candidate selection. 1-7 Reserved.
1	1	-------	1	Reserved.
2	2	LBNBRCDT	2	Number of candidates (more than one) in the list.
4	4	LBCNDAT	Variable	Addresses of UCBs of candidates; refer to the number of candidates in LBNBRCDT (above) to determine the length of this field.

4

CONCATENATION LIST (CATLST)

Size:	Variable -- 4 bytes for each address in the list.
Located in:	Subpool 252.
Created by:	IGC18099.
Used by:	IGC18099.
Contents:	A list of identically formatted four-byte addresses of the TIOT entries to be concatenated as illustrated below.

Operation Diagrams
24

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	--	4	Address of a TIOT entry to be concatenated.

CONCATENATE TABLE (CATTAB)

Size:	Variable -- 12 bytes for each TIOT entry.
Located in:	Subpool 252.
Created by:	IGC18099.
Used By:	IGC19099, IGC20099.
Updated by:	IGC19099.
Contents:	A table consisting of one entry for each entry of the TIOT, in the format shown below.

Flowcharts	Operation Diagram
JD	24

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	OLDISP	4	Old address of the entry.
4	4	NEWDISP	4	New address of the entry.
8	8	NEWPOS	1	New relative number of the entry.
9	9	ENTHND	1	Flag that indicates completed processing of the entry.
10	A	DCBUP	1	Flag that indicates the potential requirement for the updating of the DCB.
11	B	--	1	Reserved.

4

DAIR ATTRIBUTE CONTROL BLOCK (DAIRACB)

Size:	47 bytes.
Located in:	Subpool 1.
Created by:	IKJEFFAT.
Used by:	IGC30099 (SVC99).
Updated by:	None.
Contents:	DCB parameters to go into an ATRCB.

Operation Diagrams
27

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	-------	8	Reserved.
8	8	DAIMASK	8	INOUT/OUTIN options of Open.
8	8	-------	6	Reserved.
14	E	DAILABEL	1	Specifies Open type:
		1...		INOUT.
		.1..		OUTIN.
15	F	....	1	Reserved.
16	10	-------	3	Reserved.
19	13	DAIEXPDT	3	Data set expiration date:
		DAIYEAR	1	Year.
		DAIDAY	2	Day.
22	16	-------	2	Reserved.
24	18	DAIBUFNO	1	No. of buffers required.
25	19	DAIBFTEK	1	BFTEK/BFALN:
		.1..		simple buffering "S"
		.11.		Automatic record area construction "A"
		..1.		Record buffering "R"
		...1		Exchange buffering "E"
		.... ..1.		Doubleword boundary "D"
		1		Fullword boundary "F"
26	1A	DAIBUFL	2	Buffer length.
28	1C	DAIEROPT	1	Error option:
		1...		Accept.
		.1..		Skip.
		..1.		Abnormal end of task.
29	1D	DAIKEYLE	1	Key length.
30	1E	-------	6	Reserved.

(Continued)

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents	
36	24	DAIRECFM	1	Record format:	
		1...		Fixed	"F"
		.1..		Variable	"V"
		11..		Undefined	"U"
		..1.		Track overflow	"T"
		...1		Blocked	"B"
		.... 1...		Standard blocks	"S"
		.... .1..		ASA printer chars.	"A"
		.... ..1.		Machine control character	"M"
37	25	DAIOPTCD	1	Option code:	
		1...		Write validity check	"W"
		..1.		Chained scheduling	"C"
		.... 1...		ANSI translate	"O"
		.... ..1.		User totaling	"T"
38	26	DAIBLKSI	2	Maximum block size.	
40	28	DAILRECL	2	Logical record length.	
42	2A	DAINCP	1	Max # of READ/WRITE macros before check.	
43	2B	-------	4	Reserved.	

4

DAIR WORK AREA (DAIRWA)

Size:	116 bytes.
Located in:	Subpool 1.
Constructed by:	DAIRCTRL.
Updated by:	All DAIR Routines.
Used by:	All DAIR Routines.
Contents:	Addresses and control information for use of all DAIR routines.

Operation Diagrams
20

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	DSEADDR	4	Address of first DSE block.
4	4	SAVPTR	4	Address of register save area.
8	8	TCBADD	4	Address of TCB.
12	C	USERID	8	Userid (from the PSCB).
20	14	UNITDEF	8	Default unit name (from the PSCB).
28	1C	PARMPTR	4	Address of DAIR Parameter List.
32	20	RTCODE	4	Return code.
36	24	RTCODE15	4	Return code.
40	28	LIST99	4	Address of Alloc and DSE Parameter List.
44	2C	LIST99SZ	4	Size of Alloc and DSE Parameter List.
	30	CATLIST	4	Address of Catalog List.
	34	CATLSTSZ	4	Size of Catalog List.
56	38	SEGSAVAR	4	Address of secondary save area.
60	3C	SECSAVSZ	4	Size of secondary save area.
64	40	DDNPTR	4	Address of needed ddname.
68	44	DSNPTR	4	Address of needed dsname.
72	48	MEMPTR	4	Address of needed membername.
76	4C	BLKPTR	4	Address of next DSE block.
80	50	DDNADDR	4	Address of DSE block for ddname.

(continued)

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
84	54	DSNADDR	4	Address of DSE block for dsname.
88	58	NOTUADDR	4	Address of DSE block for data set marked not in use.
92	5C	DYNMADDR	4	Address of DSE block for data set marked for dynamic allocation.
96	60	STATUS1	1	Bit settings as follows: <u>Bit</u> <u>Meaning when on</u> 0 Catalog search is required. 1 Use DSE block to build parameter list. 2 Stop at first occurrence of dsname in DSE. 3 dsname found at least once. 4 dsname is a member of a dynamically concatenated group. 5 dsname is a member of a concatenated group. 6 dsname appears more than once in DSE. 7 Default status to OLD.
97	61	STATUS2	1	Bit settings as follows: <u>Bit</u> <u>Meaning when on</u> 0 Key on ddname. 1 Attr-list given. 2 Attr-name found. 3 Attr search DSES. 4-7 Used as follows: B'0001'-control returned to DAIR08 from DAIR18. B'0010'-control returned to DAIR08 from DAIR10. B'0011'-control returned to DAIR18 from DAIR10. B'0100'-control returned to DAIR1C from DAIR18. B'0101'-control returned to DAIR1C from DAIR10. 0110-control returned to DAIR28. 0111-control returned to DAIR30 from DAIR18. B'1000'-control returned to DAIR30 from DAIR10.

4

Displacement DEC.	Hex.	Field Name	Size in Bytes	Contents
98	62	SWITCHES	1	Bit settings as follows: Bit Meaning when set 0 This is a CONVERT data set. 1 A MOD as NEW data set. 2 A LOCATE was performed. 3-7 Reserved (0).
99	63	--------	1	Reserved (0).
100	64	CNCAADDR	4	Address of concatenated data set marked not in use in its DSE.
104	68	ATRLNAME	8	Name of the attribute list.
112	70	ATTRADDR	4	↑ Attribute control block (ATRCB).

DAIR PARAMETER BLOCK, CODE X'00' (DAPB00)

Size:	20 bytes.
Located in:	Subpool 1.
Created by:	Calling program.
Updated by:	DAIR00, USERID, SEARCH.
Used by:	DAIR00, USERID, SEARCH.
Contents:	Addresses and control information for DAIR00.

Operation Diagrams
20

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	DA00CD	2	Entry Code X'0000'.
2	2	DA00FLG	2	Flags set on return, as follows:
				<u>Bit</u> <u>Meaning when on</u>
				0-3 Reserved(0).
		DA00PERM		4 ddname or dsname is permanently allocated.
		DA00DYNM		5 ddname is a DYNAM entry.
		DA00DSE		6 dsname currently allocated (in DSE).
		DA00TERM		7 ddname currently allocated to the terminal.
				8-15 Reserved(0).
4	4	DA00PDSN	4	Address of dsname Buffer.
				The format of the dsname Buffer is as follows:
				<u>Byte</u> <u>Contents</u>
				0-1 The length, in bytes, of the dsname.
				2-45 The dsname, left justified, and padded to the right with blanks.
8	8	DA00DDN	8	ddname for the data set to be searched for. (If dsname is present, this field is ignored.)

(Continued)

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
16	10	DA00CTL	1	Bit settings, as follows:
				<u>Bit</u> <u>Meaning when on</u>
				0-1 Reserved(0).
		DA00UID		2 Prefix userid to dsname.
				3-7 Reserved(0).
17	11	--------	2	Reserved(0).
19	13	DA00DSO	1	Bit settings that indicate the organization of the data set, as follows.
				<u>Bit</u> <u>Meaning when set</u>
				0 Indexed sequential (IS).
				1 Physical sequential (PS).
				2 Direct organization (DO).
				3-5 Reserved(0).
				6 Partitioned organization (PO).
				7 Unmoveable.

DAIR PARAMETER BLOCK, CODE X'04' (DAPB04)

Size:	16 bytes.
Located in:	Subpool 1.
Constructed by:	Calling program.
Updated by:	DAIR04, USERID, SEARCH.
Used by:	DAIR04, USERID, SEARCH.
Contents:	Addresses and control information for DAIR04.

Operation Diagrams
20

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	DA04CD	2	Entry Code X'0004'.
2	2	DA04FLG		Flags set on return as follows:
				Bit / Meaning when on
				0-4 Reserved(0).
		DA04CAT		5 dsname found in catalog.
		DA04DSE		6 dsname currently allocated in DSE.
				7-15 Reserved(0).
4	4		2	Reserved(0).
6	6	DA04CTRC	2	Error return code for catalog management routines.
8	8	DA04PDSN	4	Address of dsname buffer.
				The format of the dsname buffer is as follows:
				Byte / Contents
				0-1 The length, in bytes, of the dsname.
				2-45 The dsname, left justified, and padded to the right with blanks.
12	C	DA04CTL	1	Bit settings as follows:
				Bit / Meaning when on
				0-1 Reserved(0).
		DA04UID		2 Prefix userid to dsname.
				3-7 Reserved(0).
13	D		2	Reserved(0).

(Continued)

4

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
15	F	DA04DSO	1	Returned when dsname is currently allocated in DSE bit settings that indicate the organization of the data set, as follows: __Bit__ __Meaning when on__ 0 Indexed sequential (IS). 1 Physical sequential (PS). 2 Direct organization (DO). 3-5 Reserved(0). 6 Partitioned organization (PO). 7 Unmoveable.

DAIR PARAMETER BLOCK, CODE X'08' (DAPB08)

Size:	84 bytes.
Located in:	Subpool 1
Constructed By:	Calling program.
Updated by:	DAIR08, USERID, SEARCH.
Used by:	DAIR08, USERID, SEARCH.
Contents:	Addresses and control information for DAIR08.

Operation Diagrams
20

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	DA08CD	2	Entry Code X'008'.
2	2	DA08FLG	2	Flags set on return, as follows: __Bit__ __Meaning when on__ 0 Data set allocated but secondary error occurred. Register 15 contains error return code. 1-15 Reserved (0)
4	4	DA08DARC	2	Error return code from dynamic allocation routines
6	6	DA08CTRC	2	Error return code from catalog management routines
8	8	DA08PDSN	4	Address of dsname Buffer. The format of the dsname Buffer is as follows: __Byte__ __Contents__ 0-1 The length, in bytes, of the dsname. 2-45 The dsname, left justified, and padded to the right with blanks.
12	C	DA08DDN	8	ddname for the data set, padded to the right with blanks. If a specific ddname is not required, this field must contain 8 blanks; the ddname to which the data is allocated will be placed in this field.
20	14	DA08UNIT	8	Unit Name desired.

(Continued)

4

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
28	1C	DA08SER	8	Serial number desired. Only the first 6 bytes are significant. If the serial number is less than 6 bytes it must be padded to the right with blanks. If the serial number is omitted, the entire field must contain blanks.
36	24	DA08BLK	4	Block size requested. The average record length desired.
40	28	DA08PQTY	4	Primary space quantity desired. The high order byte must be zero, the low order three bytes contain the space quantity desired. If the quanity is omitted, the entire field must be set to zero.
44	2C	DA08SQTY	4	Secondary space quantity desired. The high order byte must be zero, the low order three bytes contain the space quantity desired. If the quantity is omitted, the entire field must be set to zero.
48	30	DA08DQTY	4	Directory quantity desired. The high order byte must be zero, the low order three bytes contain the number of Directory blocks desired. If the quantity is omitted, the entire block field must be set to zero.
52	34	DA08MNM	8	Member name of a partitioned data set. If the name has less than 8 characters, it must be padded to the right with blanks. If the name is omitted, the entire field must contain blanks.
60	3C	DA08PSWD	8	Password for the data set. If the password has less than 8 characters, it must be padded to the right with blanks. If the password is omitted, the entire field must contain blanks.
68	44	DA08DSP1	1	Bit settings that indicate the status of the data set, as follows: <u>Bit</u> <u>Meaning when on</u> 0-3 Reserved(0)
		DA08SHR		4 SHR
		DA08NEW		5 NEW
		DA08MOD		6 MOD
		DA08OLD		7 OLD
69	45	DA08DSP2	1	Bit settings that indicate the normal disposition of the data set, as follows:

(Continued)

Displacement		Field	Size in	
Dec.	Hex.	Name	Bytes	Contents
69 (cont.)	45			<u>Bit</u> <u>Meaning when on</u>
				0-3 Reserved(0)
		DA08KEEP		4 KEEP
		DA08DEL		5 DELETE
		DA08CAT		6 CATLG
				7 UNCATLG
70	46	DA08DSP3	1	Bit settings that indicate the abnormal disposition of the data set, as follows:
				<u>Bit</u> <u>Meaning when on</u>
				0-3 Reserved(0)
		DA08KEP		4 KEEP
		DA08DELE		5 DELETE
		DA08CATL		6 CATLG
		DA08UNCT		7 UNCATLG
71	47	DA08CTL	1	Bit settings that control the operations to be performed by DAIR, as follows:
				<u>Bit</u> <u>Meaning when on</u>
				0-1 Specifies the type of units desired for the space parameters, as follows:
		DA08BLK		'01'B units are in average block length
		DA08TRKS		'10'B units are in TRKS
		DA08CYLS		'11'B units are in CYLS
		DA08UID		2 Prefix userid to dsname
		DA08RLSE		3 RLSE is desired
		DA08PERM		4 Data set is to be permanently allocated; not be unallocated until specifically requested.
		DA08DMMY		5 DUMMY data set is desired.
		DA08ATRL		6 Attribute list is supplied.
			7	Reserved(0).
72	48	--------	3	Reserved(0)
75	4B	DA08DSO	1	Bit settings on return that indicatethe organization of the data set, as follows:
				<u>Bit</u> <u>Meaning when on</u>
				0 Indexed sequential (IS)
				1 Physical sequential (PS)
				2 Direct organization (DO)
				3-5 Reserved (0)
				6 Partitioned organization (PO)
76	4C	DA08ALN	8	Attribute list name.

4

DAIR PARAMETER BLOCK, CODE X'0C' (DAPB0C)

Size:	Variable -- 12 bytes as shown below, plus 8 bytes for each DDNAME to be concatenated.
Located in:	Subpool 1.
Constructed by:	Calling program.
Updated by:	DAIR0C.
Used by:	DAIR0C.
Contents:	Addresses and control information for DAIR0C.

Operation Diagrams
20

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	DA0CCD	2	Entry Code X'0C'.
2	2	DA0CFLG	2	Reserved (0).
4	4	DA0CDARC	2	Error return code from dynamic allocation.
6	6	--------	2	Reserved (0).
8	8	DA0CNUMB	2	Number of data sets to be concatenated.
10	A		2	Reserved (0).
12	C	DA0CDDN	8	ddname of first data set to be concatenated.
.		.		
.		.		
.		.	.	ddname of last data set to be concatenated.
.		.	.	

DAIR PARAMETER BLOCK, CODE X'10' (DAPB10)

Size:	16 bytes.
Located in:	Subpool 1.
Constructed by:	Calling program.
Updated by:	DAIR10.
Used by:	DAIR10.
Contents	Addresses and control information for DAIR10.

Operation Diagrams
20

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	DAIOCD	2	Entry code X'0010'.
2	2	DAIOFLG	2	Reserved (0).
4	4	DAIODARC	2	Error return code from dynamic allocation routines.
6	6		2	Reserved (0)
8	8	DAIODDN	8	ddname of data set to be deconcatenated.

4

DAIR PARAMETER BLOCK, CODE X'14' (DAPB14)

Size:	16 bytes.
Located in:	Subpool 1.
Created by:	Calling program.
Updated by:	DAIR14, USERID, IKJEHCIR.
Used by:	DAIR14, USERID, IKJEHCIR.
Contents:	Address and control information for DAIR14.

Operation Diagrams
20

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	DA14CD	2	Entry Code X'0014'.
2	2	DA14FLG	2	Reserved (0).
4	4	DA14DSN	4	Address of the dsname Buffer. The format of the dsname Buffer is as follows: __Bit__ __Contents__ 0-1 The length, in bytes, of the dsname. 2-45 The dsname, left justified, and padded to the right with blanks.
8	8	DA14PRET	4	Address of the return area in which are qualifiers for the dsname are placed. The format of the return area is as follows: __Byte__ __Contents__ 0-1 The length, in bytes, of the return area. 2-3 Reserved (0). 4-11 First Qualifier (or 0, if none) 12-19 Second Qualifier (or 0, if none). Etc.
12	C	DA14CTL	1	Bit settings that control DAIR operation, as follows: __Bit__ __Meaning when on__ 0-1 Reserved (0).
		DA14UID		2 Prefix userid to dsname. 3-7 Reserved (0).
13	D	--------	3	Reserved (0).

DAIR PARAMETER BLOCK, CODE X'18' (DAPB18)

Size:	40 bytes.
Located in:	Subpool 1.
Created by:	Calling program.
Updated by:	DAIR18, USERID, SEARCH.
Used by:	DAIR18, USERID, SEARCH.
Contents:	Address and control information for DAIR18.

Operation Diagrams
20

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	DA18CD	2	Entry Code X'0018'.
2	2	DA18FLG	2	Flags set on return, as follows: Bit Meaning when on 0 Data set freed but secondary error occurred -- return code is in RTCODE of DAWT. 1-15 Reserved (0).
4	4	DA18DARC	2	Error return code from dynamic allocation routines.
6	6	DA18CTRC	2	Error return code from catalog management routines.
8	8	DA18PDSN	4	Address of the dsname Buffer. The format of the dsname Buffer is as follows: Byte Contents 0-1 The length, in bytes, of the dsname. 2-45 The dsname, left justified, and padded to the right with blanks.
12	C	DA18DDN	8	ddname of the data set to be unallocated.
20	14	DA18MNM	8	Membername of a partitioned data set. If the name has less than 8 characters, it must be padded to the right with blanks. If the password is omitted, the entire field must contain blanks.
28	16	DA18CLS	2	SYSOUT class. An alphabetic or numeric character. If SYSOUT is not specified, this field must contain blanks.

(Continued)

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
30	18	DA18DSP2	1	Bit settings that indicate the normal disposition of the data set, as follows: Bit / Meaning when on 0-3 Reserved (0) 4 KEEP 5 DELETE 6 CATLG 7 UNCATLG
31	19	DA18CTL	1	Bit settings that control DAIR operations, as follows: Bit / Meaning when on 0-1 Reserved (0). 2 Prefix userid to dsname. 3 Free permanently allocated data sets. (Mark "not in use" if the bit is off.) 4-7 Reserved (0).
32	20	DA18JBNM	8	The jobname for enqueuing SYSOUT data sets. If the jobname is omitted, the jobname will be taken from the TIOT.

DAIR PARAMETER BLOCK, CODE X'1C' (DAPB1C)

Size:	24 bytes.
Located in:	Subpool 1.
Constructed by:	Calling program.
Updated by:	DAIR1C, SEARCH.
Used by:	DAIR1C, SEARCH.
Contents:	Address and control information for DAIR1C.

Operation Diagrams
20

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	DA1CCD	2	Entry Code X'001C'.
2	2	DA1CFLG	2	Reserved (0).
4	4	DA1CDARC	2	Error return code from dynamic allocation routines.
6	6		1	Reserved (0).
7	7	DA1CCTL	1	Bit settings, as follows: Bit Meaning when on 0-3 Reserved (0).
		DA1CPERM		4 Allocate permanently.
		DA1CATRL		5 Attribute list is supplied.
				6-7 Reserved (0).
8	8	DA1CDDN	8	ddname for the data set to be allocated to the terminal
16	10	DA1CALN	8	Attribute list name.

4

DAIR PARAMETER BLOCK, CODE X'24' (DAPB24)

Size:	84 bytes.
Located in:	Subpool 1.
Created by:	Calling program.
Updated by:	DAIR24, DAIR08, USERID, SEARCH.
Used by:	DAIR24, DAIR08, USERID, SEARCH.
Contents:	Addresses and control information for DAIR24.

Operation Diagrams
20

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	DA24CD	2	Entry Code X'0024'.
2	2	DA24FLG	2	Flags set on return as follows: <u>Bit</u> <u>Meaning when on</u> 0 Data set allocated but secondary error occurred. Register 15 contains the error return code. 1-3 Reserved (0). 4 ddname requested is allocated as DUMMY. 5-15 Reserved (0).
4	4	DA24DARC	2	Error return code from dynamic allocation routines.
6	6	DA24CTRC	2	Error return code from catalog management routines.
8	8	DA24PDSN	4	Address of dsname buffer. The format of the dsname buffer is as follows: <u>Byte</u> <u>Contents</u> 0-1 The length, in bytes, of the dsname. 2-45 The dsname, left justified, and passed to the right with blanks.
12	C	DA24DDN	8	ddname of data set to be allocated. (Required)
20	14	DA24UNIT	8	Unit name. If the unit name is less than 8 bytes, it is padded to the right with blanks.

(Continued)

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
28	1C	DA24SER	8	Serial number. Only the first 6 bytes are significant. If the serial number is less than 6 bytes, it must be padded to the right with blanks. If the serial number is omitted, the entire field must contain blanks.
36	24	DA24BLK	4	Block size requested. The average record length desired.
40	28	DA24PQTY	4	Primary space quantity desired. The high order byte must be zero, the low order three bytes contains the space quantity desired. If the quantity is omitted, the entire field must be set to zero.
44	2C	DA24SQTY	4	Secondary space quantity desired. The high order byte must be zero, the low order three bytes contain the space quantity desired. If the quantity is omitted, the entire field must be set to zero.
48	30	DA24DQTY	4	Directory quantity desired. The high order byte must be zero, the low order three bytes contain the number of Directory blocks desired. If the quantity is omitted, the entire block field must be set to zero.
52	34	DA24MNM	8	Membername of a partitioned data set. If the name has less than 8 characters, it must be padded to the right with blanks. If the name is omitted, the entire field must contain blanks.
60	3C	DA24PSWD	8	Password for the data set. If the password has less than 8 characters, it must be padded to the right with blanks. If the password is omitted, the entire field must contain blanks.
68	44	DA24DSP1	1	Bit settings that indicate the status of the data set, as follows:
				<u>Bit</u> <u>Meaning when on</u>
				0-3 Reserved (0).
		DA24SHR		4 SHR
		DA24NEW		5 NEW
		DA24MOD		6 MOD
		DA24OLD		7 OLD

(Continued)

4

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
69	45	DA24DSP2	1	Bit settings that indicate the normal disposition of the data set, as follows:
				<u>Bit</u> <u>Meaning when on</u>
				0-3 Reserved (0).
		DA24KEEP		44 KEEP
		DA24DEL		5 DELETE
		DA24CAT		6 CATLG
		DA24UCAT		7 UNCATLG
70	46	DA24DSP3		Bit settings that indicate the abnormal disposition of the data set, as follows:
				<u>Bit</u> <u>Meaning when on</u>
				0-3 Reserved (0).
		DA24KEP		4 KEEP
		DA24DELE		5 DELETE
		DA24CATL		6 CATLG
		DA24UNCT		7 UNCATLG
71	47	DA24CTL	1	Bit settings that control the operations to be performed by DAIR, as follows:
				<u>Bit</u> <u>Meaning when on</u>
				0-1 Specific the type of units. desired for the space parameters, as follows:
		DA24BLK		'01'B units are in average block length.
		DA24TRKS		'10'B units are in TRKS
		DA24CYLS		'11'B units are in CYLS
		DA24UID		2 Prefix userid to dsname
		DA24RLSE		3 RLSE is desired
		DA24PERM		4 Data set is to be permanently allocated; not be unallocated until specifically requested.
		DA24DMMY		5 DUMMY data set is desired.
		DA24ATL		6 Attribute list supplied.
			7	Reserved (0).
72	48	--------	3	Reserved (0).
75	4B	DA24DSO	1	Bit settings on return that indicate the organization of the data set, as follows:
				<u>Bit</u> <u>Meaning when on</u>
				0 Indexed Sequential (IS)
				1 Physical Sequential (PS)
				2 Direct Organization (DO)
				3-5 Reserved (0)
				6 Partitioned Organization (PO)
				7 Unmoveable
76	4C	DA24ALN	8	Attribute list name.

DAIR PARAMETER BLOCK, CODE X'28' (DAPB28)

Size:	Variable -- 8 bytes as shown below, plus 4 bytes for each operation to be performed.
Located in:	Subpool 1.
Created by:	Calling program.
Updated by:	DAIR28.
Used by:	DAIR28.
Contents:	Addresses and control information for DAIR28.

Operation Diagrams
20

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	DA28CD	2	Entry code X'0028'.
2	2	DA28NOP	2	Number of operations to be performed.
4	4	DA28PFOP	4	Address of DAPB for first operation that fails. (Zero if all operations are successful.) Register 15 contains error return code.
8	8	DA28OPTR	4	Address of DAPB for first operation.
		.		.
		.		.
		.		.
		DA28OPTR	4	Address of DAPB for last operation.

4

DAIR PARAMETER BLOCK, CODE X'2C' (DAPB2C)

Size:	16 bytes.
Located in:	Subpool 1.
Constructed by:	Calling program.
Updated by:	DAIR2C.
Used by:	DAIR2C.
Contents:	Addresses and control information for DAIR2C.

Operation Diagrams
20

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	DA2CCD	2	Entry code X'002C'
2	2	DA2CFLG	2	Bit settings as follows: Setting Meaning X'02' Mark all DSEs allocated by the ancestors of the current task, and by the initiator, as "not in use". X'01' Mark the DSE for the specified ddname as "not in use". X'00' Mark all the DSEs that have a supplied TCB address as "not in use".
4	4	DA2CTCB	4	Address of the TCB for the routine whose data sets are to be marked "not in use".
8	8	DA2CDDN	8	ddname for DSE entry to be marked "not in use".

DAIR PARAMETER BLOCK, CODE X'30' (DAPB30)

Size:	72 bytes.
Located in:	Subpool 1.
Constructed by:	Calling program.
Updated by:	DAIR30.
Used by:	DAIR30.
Contents:	Addresses and control information for DAIR30.

Operation Diagrams
20

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	DA30CD	2	Entry code X'0030'.
2	2	DA30FLG	2	Flags set on return, as follows: Bit Meaning when on 0 Data set allocated by secondary error occurred. Register 15 contains error return code. 1-15 Reserved (0).
4	4	DA30DARC	2	Error return code from Dynamic Allocation routines.
6	6	--------	2	Reserved (0).
8	8	DA30PDSN	4	Address of dsname buffer. The format of the dsname buffer is as follows: Byte Contents 0-1 The length, in bytes, of the dsname. 2-45 The dsname, left justified, and padded to the right with blanks.
12	C	DA30DDN	8	ddname for the data set. If a specific ddname is not required, this field must contain 8 blanks; the ddname to which the data is allocated will be placed in this field.
20	14	DA30UNIT	8	Unit Name desired.
28	1C	DA30SER	8	Serial number desired. Only the first 6 bytes are significant. If the serial number is less than 6 bytes, it must be padded to the right with blanks. If the serial number is omitted, the entire field must contain blanks.

(Continued)

4

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
36	24	DA30BLK	4	Block size requested. The average record length desired.
40	28	DA30PQTY	4	Primary space quantity desired. The high order byte must be zero, the low order three bytes contain the space quantity desired. If the quantity is omitted, the entire field must be set to zero.
44	2C	DA30SQTY	4	Secondary space quantity desired. The high order byte must be zero, the low order three bytes contain the space quantity desired. If the quantity is omitted, the entire field must be set to zero.
48	30	DA30PGNM	8	Programname-The member name of a special user program to handle SYSOUT operations. This field must be blanks if this parameter is omitted.
56	38	DA30FORM	4	Form Number-indicates that the output should be printed or punched on a specific output form. It is a four character form number. This field must be blanks if this parameter is omitted.
60	3C	DA30CLS	2	SYSOUT class - A single alphameric character (0-9, A-Z). <u>Note</u>: The SUBMIT command processor passes the SUBMIT queue number rather than a SYSOUT class to have its data sets placed in the proper queue.
62	3E	--------	1	Reserved (0)
63	3F	DA30CTL	1	Bit settings that control the operations to be performed by DAIR, as follows: <u>Bit</u> <u>Meaning when set</u> 0-1 Specifies the type of units desired for the space parameters, as follows:
		DA30ABLK		'01'B units are in average block length
		DA30TRKS		'10'B units are in TRKS
		DA30CYLS		'11'B units are in CYLS
		DA30UID		2 Prefix userid to DSNAME
		DA30RLSE		3 RLSE is desired
		DA30PERM		4 Data set is to be permanently allocated; not be unallocated until specifically requested.
		DA30DMMY		5 DUMMY data set is desired.
		DA30ATRL		6 Atribute list name supplied.
				7 Reserved.
64	40	DA30ALN	8	Atribute list name.

DAPB34 - DAIR PARAMETER LIST

Size:	20 bytes.
Located in:	Subpool 1.
Constructed by:	IKJEFATT; NAMECK Validity Checking Routine.
Used by:	DAIR34.
Contents:	Addresses & control information for DAIR34.

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	DA34CD	2	Entry code X'0034'.
2	2	DA34FLG	2	Flags.
		DA34FIND		Bit 0 on, attr-list-name found.
				Bits 1-7 reserved.
4	4	DA34DARC	2	Return code from dynamic allocation.
6	6	DA34CTRL	1	Function to perform:
		DA34SRCH		Bit 0 on, search for name only.
		DA34CHN		Bit 1 on, chain an ATRCB.
		DA34UNCH		Bit 2 on, unchain ATRCB.
				Bits 3-7 reserved.
7	7	-------	1	Reserved.
8	8	DA34NAME	8	Attr-list-name.
16	10	DA34ADDR	4	Address of DAIRACB.

4

DATA SET EXTENSION BLOCK (DSE)

Size:	Varialbe.
Location in Subpool	225.
Created by:	MVT Job Management.
Updated by:	MVT Dynamic Allocation routines -- SVC 99.
Used by:	SEARCH.
Contents:	Information about data sets including ddnames and dsnames.

Operation Diagrams
20

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	DSEFORWD	4	Address of next DSE. (0 in the last DSE.)
4	4	DSEBCKWD	4	Address of last DSE. (0 in the first DSE.)
8	8	DSEBLKSZ	2	Length in bytes of DSE
10	A	DSESTAT	1	Allocated status of the data set, as follows:
				Bit Meaning when on
				0-3 Reserved
				4 DSE SHR
				5 DSE NEW
				6 DSE MOD
				7 DSE OLD
11	B	DSECNTRL	1	Bit settings that indicate the status of the data set, as follows:
				Bit Meaning when on
		DSEDYN		0 Data set allocated dynamically.
		DSENUSED		1 Data set not in use.
		DSEMEM		2 The DSEMEMBER field is present in this DSE.
		DSEDTIOT		3 TIOT entry is DYNAM.
		DSECON		4 Data set was concatenated dynamically.
		--------		5 Reserved (0) use.
		DSEPERM		6 Data set is permanently allocated.
		--------		7 Reserved (0).

(Continued)

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
12	C	DSEDDNAM		8 ddname as found in TIOT
20	14	DSETCBAD		4 Address of the TCB for the routine which requested dynamic allocation.
24	18	DSETTRPW	4	The relative TTR of the password for this password protected data set within the password data set (0 if data set is not password protected or unit password TS entered from the terminal).
28	1C	DSENDISP	1	Bit settings that indicate the normal disposition of the data set, as follows:
				<u>Bit</u> <u>Meaning when on</u>
				0-3 Reserved (0).
		DSEKEP		4 KEEP
		DSEDEL		5 DELETE
		DSECAT		6 CATLG
		DSEUCAT		7 UNCATLG
29	ID	DSEADISP	1	Bit settings that indicate the abnormal disposition of the data set, as follows:
				<u>Bit</u> <u>Meaning when on</u>
				0-3 Reserved (0).
				4 KEEP
				5 DELETE
				6 CATLG
				7 UNCATLG
30	1E	DSEDSORG	1	Bit settings that indicate the organization of the data set, as follows:
				<u>Bit</u> <u>Meaning when on</u>
		DSEIS		0 Indexed Sequential (IS).
		DSEPS		1 Physical Sequential (PS).
		DSEDO		2 Direct Organization (DO).
				3-5 Reserved (0).
		DSEPO		6 Partitioned Organization (PO).
		DSEU		7 Unmoveable.

(Continued)

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
31	1F	DSEDSLNG	1	Length in bytes of the dsname.
32	20	DSEDSNAM	1-44	dsname for the data set.
		DSEMEMBR	8	The name of a member of a partitioned data set. (This field is present only for partitioned data sets and only when the DSEMEM bit is set in DSECNTRL.)

DAIR PARAMETER LIST (DAPL)

Size:	20 bytes.
Located in:	Subpool 1.
Created by:	Calling program.
Updated by:	None
Used by:	IKJEFD00.
Contents:	Parameter List for DAIR -- IKJEFD00.

Operation Diagrams
20

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	DAPLUPT	4	Address of user profile table (UPT).
4	4	DAPLECT	4	Address of environment control table (ECT).
8	8	DAPLECB	4	Address of calling program's event control block (ECB).
12	C	DAPLPSCB	4	Address of protected step control block (PSCB).
16	10	DAPLDAPB	4	Address of DAIR parameter block.

DYNAMIC ALLOCATION PARAMETER BLOCK, FUNCTION CODE X'00' (DSE UPDATE PARAMETER LIST)

Size:	64 bytes.
Created by:	IKJEFD00 -- DAIR08, DAIR24, DAIR2C, DAIR30.
Updated by:	SVC 99 DSE Update routines.
Used by:	SVC 99 DSE Update routines.
Contents:	Addresses and control information for the DSE UPDATE function as invoked by the DAIR2C function, only, to mark DSE blocks not in use. (Other functions enter DSE update processing with the function codes of SVC 99 functions whose execution makes subsequent DSE updating necessary.)

Operation Diagrams
19,25

Displacement Dec.	Hex.	Field Name	Size in bytes	Contents
0	0	CODE	1	X'80'
1	1	OBTDSORG	1	DSORG code returned from OBTAIN macro instruction.
2	2	FCODE	1	Function code X'00'
3	3	OPTION	1	Options. Bit settings, as follows: Bit Meaning when on 0-5 Reserved (0) 6 Mark DSE entries "not in use" except those created for the current task and its originating tasks. 7 Mark a DSE entry "not in use". The address of the DSE entry is placed at offset 5 in this block.
4	4	CONTROL	1	Bit settings, as follows: Bit Meaning when on 0-4 Reserved (0). 5 Data set is permanently allocated. 6-7 Reserved (0).
5	5	TCBADD	3	Address of the TCB for the task for which the data set was allocated, or (if bit 7 in the options field is on) the address of the DSE entry to be marked "not in use".
8	8	PASSWORD	8	The 8-byte password (if used), left justified, padded to the right with blanks; otherwise zeros.

(Continued)

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
16	10	PSWRDTTR	3	The relative track address (TTR) of the password in the password data set.
19	13	DSLNGTH	1	The length of the dsname.
20	14	DSNAME	44	DSNAME for the data set, left justified, padded to the right with blanks.

DYNAMIC ALLOCATION PARAMETER BLOCK, FUNCTION CODE X'01'

Size	Variable.
Created by:	IKJEFD00 -- DAIR08, DAIR1C, DAIR30.
Updated by:	IGC26099.
Used by:	SVC 99 DATASET and DSE routines.
Contents:	Addresses and control information for the DATASET function of dynamic allocation.

Operation Diagrams
20,21

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	PLNGH	2	Size, in bytes, of the parameter block.
2	2	PCODE	1	Function Code X'01'.
3	3	POPTIONS	1	Bit settings as follows: Bit Meaning when set 0 DUMMY data set. No device is to be allocated. I/O operations are to be bypassed. 1 Uncatalog the data set if disposition is delete. 2 TERM=TS data set. Special BSAM/QSAM interface is to be set up so that terminal can be used as I/O device. 3 dsname points to data set name. 4-7 Reserved (0).
4	4	PDDNM1	8	Current ddname for an unallocated, unopened data set.
12	C	--------	1	Reserved (0).
13	D	PDDNM2	8	New ddname for the data set. It may not be the ddname associated with any TIOT entries.
21	15	--------	1	Reserved (0).
22	16	PDISP1	1	Bit settings that indicate the type of data set, as follows: Bit Meaning when on 0-3 Reserved (0). 4 SHR 5 NEW 6 MOD 7 OLD

(Continued)

4

Displacement Dec.	Displacement Hex.	Field Name	Size in Bytes	Contents
22	17	PDISP2	1	Bit settings that indicate the normal disposition of the data set, as follows: <u>Bit</u> <u>Meaning when on</u> 0-3 Reserved (0). 4 KEEP 5 DELETE 6 CATLG 7 UNCATLG
24	18	PDISP3	1	Bit settings that indicate the abnormal disposition of the data set, as follows: <u>Bit</u> <u>Meaning when on</u> 0-3 Reserved (0). 4 KEEP 5 DELETE 6 CATLG 7 UNCATLG
25		PMNM	8	Membername-the name of a member of a partitioned data set (PDS). If less than 8 characters, it must be padded on the right with blanks. If omitted, the entire field must be blank.
33		PPRIME	3	Primary space quantity-indicates how many of the units should be assigned to the data set initially (the units are indicated in the CTB field). Field must be zeros if parameter omitted.
36		PDSNP	4	Address of the dsname Buffer, as follows: <u>Byte</u> <u>Contents</u> 0 The length, in bytes, of the dsname. 1-44 dsname, left justified, padded to the right with blanks.
40	28	PUNM	8	Unitname-specifies information about the unit the data set will use. If omitted this field must be blanks. If it is blank, the system will default to the use of all the direct-access devices. It may be one of the following: 1. address of the specific unit. 2. Type (module number) of unit. 3. Name of a group of units-this name and the group it belongs to are defined during system generation.

(Continued)

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
48	30	PTCTB	1	CTB. Bit settings that control operations performed by dynamic allocation, as follows: <u>Bit</u> <u>Meaning when on</u> 0-1 Specifies the units used for the space parameters, as follows: '01'B-units are average blank length '10'B-units are TRKS '11'B-units are CYLS 2 Reserved (0). 3 RLSE is desired. 4-7 Reserved (0).
49	31	PSEC	3	Secondary space quantity-Indicates how many of the units should be assigned to the data set when the data set exhausts its space. The field must be zero if the parameter is omitted.
52	34	PCLASS	1	Output class-specifies an output class into which the data set should be placed. It may be an alphabetic character from A to Z or a numeric from 0 to 9. Regardless of the output class specified here, the SYSOUT data set will be placed on the queue for the message class. If SYSOUT is not specified, this field must contain blanks.
53	35	PDIREC	3	Directory space quantity-Indicates the size of a directory of a data set with partitioned organization (a BPAM data set). This field must be zero if the parameter is omitted.
56	38	PFORM	4	Form number-indicates that the output should be printed or punched on a specific output form. It is a four digit form number. This field must be blanks if this parameter is omitted.
60	3C	PPGMNM	8	Programname-The member name of a special user program to handle SYSOUT operations. This field must be blanks if this parameter is omitted.
68	44	PBLKSIZ	2	BLKSIZE-Average block length parameter. If specified it must be less than 65,536 bytes. This field must be zero if the parameter is omitted.
70	46	--------	2	Reserved (0).

(Continued)

4

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
72	48	PSERNO	6	Serial Number-The serial number of the volume for this data set. If specified, the request is for a specific volume. Field must be blanks if parameter is omitted.
78	4E	--------	1	Reserved (0).

DYNAMIC ALLOCATION PARAMETER BLOCK, FUNCTION CODE X'02'

Size:	24 bytes.
Created by:	IKJEFD00 -- DAIR18
Used by:	UNALLOC and DSE UPDATE routines.
Contents:	Addresses and control information for the UNALLOC function of SVC 99 dynamic allocation.

Operation Diagrams
20,22

Displacement Dec.	Hex.	Field Name	Size Bytes	Contents
0	0	--------	2	'X0018' the size in bytes, of the parameter block.
2	2	--------	1	X'02' Function code.
3	3	--------	1	Options. Bit settings, as follows: Bit Meaning when on 0 Data set is allocated to more than one user. Dynamic allocation will not remove from the Data Set ENQ Table and will not dequeue the data set. 1 DAIR could not catalog the data set. Dynamic allocation will not uncatalog the data set.
4	4	--------	8	ddname associated with a currently allocated, closed TIOT entry.
12	C	--------	1	Reserved.
13	D	--------	1	Bit settings that indicate the normal disposition for the data set, as follows: Bit Meaning when on 0-3 Reserved (0). 4 KEEP 5 DELETE 6 CATLG 7 UNCATLG
14	E	--------	1	SYSOUT class-A single alphameric character (0-9),(A-Z). (Note: The SUBMIT command processor passes the SUBMIT queue number rather than a SYSOUT class to have its data sets placed in the proper queue.)
15	F	--------	1	Reserved (0)

4

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
16	10	--------	8	An 8 byte field that contains the jobname used to enqueue the SYSOUT data set. If the jobname is omitted, the field must contain blanks; the jobname from the TIOT will be used to enqueue the SYSOUT data set.

DYNAMIC ALLOCATION PARAMETER BLOCK, FUNCTION CODE X'03'

Size:	Variable.
Created by:	IKJEFD00 -- DAIROC.
Used by:	CONCAT and DSE UPDATE routines.
Contents:	Addresses and control information for the CONCAT function of SVC 99 dynamic allocation.

Operation Diagrams
20,24

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	--------	2	Length=4+9n, where n is the number of ddnames to be concatenated.
2	2	--------	1	X'03' Function code.
3	3	--------	1	Reserved (0).
4	4	--------	8	ddname for the first allocated, unopened TIOT entry to be concatenated.
12	C	--------	1	Reserved (0).
13	D	--------	8	ddname for the second allocated, unopened TIOT entry to be concatenated.
.	.			.
.	.			.
.	.			.
.	.			.
13+9n			1	Reserved (0)

Note: At least two ddnames must be specified; a maximum of 255 ddnames may be specified.

DYNAMIC ALLOCATION PARAMETER BLOCK, FUNCTION CODE X'04'

Size:	13 bytes.
Created by:	IKJEFD00 -- DAIR08, DAIR10, DIAR1C, DAIR30.
Used by:	DECONCAT and DSE UPDATE routines.
Contents:	Addresses and control information for the DECONCAT function of SVC 99 dynamic allocation.

Operation Diagrams
20,25

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	--------	2	X'000D'. The size, in bytes, of the parameter block.
2	2	--------	1	Function Code X'04'.
3	3	--------	1	Reserved(0).
4	4	--------	8	ddname for a group of data sets that have been concatenated.
12	C	--------	1	Reserved(0).

DYNAMIC ALLOCATION PARAMETER BLOCK, FUNCTION CODE X'05'

Size:	2 bytes.
Created by:	IKJDAIR STAE routine (IKJEFD00).
Used by:	IGC00099.
Contents:	The function code of X'05', which indicates to the allocation control routine that an ABEND, for which cleanup of the dynamic device request may be necessary for this task, has occurred.

4

DYNAMIC ALLOCATION PARAMETER BLOCK, FUNCTION CODE X'06'

Size:	40 bytes.
Created by:	IKJEFD00 -- DAIR08, DAIR1C, DAIR24, DAIR30.
Used by:	CONVERT and DSE UPDATE routines.
Contents:	Address and control information for the CONVERT function of SVC 99 dynamic allocation.

Operation Diagrams
20,23

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	--------	0	X'0028'. The size, in bytes, of the parameter block.
2	2	--------	1	Function code X'06'.
3	3	--------	1	Options. Bit settings, as follows: Bit Meaning when on 0 Reserved (0). 1 Read JFCB for DSE UPDATE; The DSE DSORG field is zero. 2 Change data set disposition. 3 Change data set status. 4 Change member name. 5 Zero the DCB-oriented fields in the JFCB. Change blocksize. 6 Change status from SHR to exclusive. (Possible only if the current ENQ environment will allow the use attribute to be changed from shared to exclusive.) 7 Change DDNAME1 to DDNAME2.
4	4	--------	8	Current ddname. Ddname associated with the currently unopened data set.
12	C	--------	1	Reserved (0).
13	D	--------	8	New ddname(DDNAME2).
21	15	--------	1	Reserved.
22	16	--------	1	Status. Byte settings, as follows: Byte Meaning when on X'00' omitted X'01' OLD X'02' MOD X'03' NEW X'08' SHR

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
23	17	--------	1	Bit settings that indicate the normal disposition of the data set, as follows: <u>Bit</u> <u>Meaning when on</u> 0-3 Reserved(0) 4 KEEP 5 DELETE 6 CATLG 7 UNCATLG
24	19	--------	1	Bit settings that indicate the abnormal disposition of the data set, as follows: <u>Bit</u> <u>Meaning when on</u> 0-3 Reserved(0). 4 KEEP 5 DELETE 6 CATLG 7 UNCATLG
25	19	--------	8	Membername. The name of a member of a partitioned data set (PDS). If less than 8 characters it must be padded on the right with blanks. If omitted, the entire field must be blank.
33	21	--------	3	Average record length. If bit 5 in the options field is set, this field will be used to fill in the average record length and blocksize fields in the JFCB.
36	24	--------	4	Address of the dsname buffer.

4

DYNAMIC ALLOCATION PARAMETER BLOCK, FUNCTION CODE X'07'

Size:	16 bytes.
Created by:	IKJDAIR.
Used by:	IGC30099 (Dynamic allocation, SVC99).
Contents:	Address and control information for the ATTRIB function of SVC 99 dynamic allocation.

Operation Diagrams
20,27

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	SIZE	2	Size of parameter block.
2	2	CODE	1	Function code-X'07'.
3	3	OPTION UNCHAIN CHAIN	1	Function to perform: Bit 0=1 Unchain ATRCB 1=1 Chain attr. list
4	4	ATRPTR	4	Addr of DAIRACB for chain function or ATRCB for unchain.
8	8	ATRNME	8	Attr-list name per chain function.

DYNAMIC ALLOCATION WORK TABLE (DAWT)

Size:	552 bytes.
Located in:	Subpool 252.
Created by:	IGC00099.
Used by:	All functions.
Updated by:	All functions.
Contents:	Formatted information, as shown below, plus the first, second, and variable work areas (DAWA1, DAWA2, and DAWTVARY, respectively). Note that the field format in DAWTVARY is different for each functional grouping of routines; these various configurations of DAWTVARY appear on the following pages.

Operation Diagram
21-26

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	XCTLAD	4	Address of entry point name for XCTL.
4	4	XCTLDCB	4	DCB address for XCTL.
8	8	XCTLEP	8	Entry point name for XCTL.
16	10	PPARMP	4	Address of SVC parameter list.
20	14	LNGH	2	Size of DAWT.
22	16	--	1	Reserved.
23	17	TIOTNO	1	TIOT number.
24	18	TIOTP	4	TIOT entry address.
28	1C	RTCODE	4	Return code.
32	20	DAWTQMPA	36	QMPA.
68	44	DAWTEIOB	84	ECB/IOB for use by the transient queue manager.
136	88	DAWA1	176	First work area.
312	138	DAWA2	176	Second work area.
488	1E8	DAWAVARY	48	Variable work area.

DYNAMIC ALLOCATION WORK TABLE, VARIABLE AREA (DAWTVARY) AS USED IN THE DATASET FUNCTION

Size:	52 bytes.
Located in:	DAWT -- (Subpool 252).
Created by:	IGC00099.
Used and Updated by:	IGC07099, IGC08099, IGC09099, IGC10099, IGC11099, IGC12099, IGC13099, IGC14099, IGC15099.
Contents:	Work area for the DATASET function.

Operation Diagram
21

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	DAWA1P	4	↑dynamic allocation work area #1.
4	4	SIOTPTR	4	↑SIOT buffer.
8	8	DAWA2P	4	↑dynamic allocation work area #2.
12	C	FTTR	4	TTR of the DSENQ record.
16	10	CURUPLW	4	↑CURRENT UCBLIST.
20	14	DEVTYP	8	Device type, subfielded as follows:
20	14	DMTP	4	↑bit pattern.
24	18	-----	2	Not used.
26	1A	UCBP	2	↑UCB.
28	1C	ENQP	4	↑ENQ dsname parameter list.
32	20	TCBPTR	4	↑Caller's TCB.
36	24	ENQLIST	4	↑ENQ parameter list.
40	28	VFLAGS	1	Flag bits, as follows:
				Bit / Meaning when on
		VDUMMY		0 Dummy data set.
		VDOMIT		1 Omitted dsname.
				or
		VDDR		1 DDR is in system.
		VNEW		2 Disposition is NEW.
		VDELDMT		3 DMT is to be deleted.
		VSPFCUCB		4 Specific UCB.
		VSPFCVOL		5 Specific volume.
		VBPCONST		6 Pattern construction area is built.
		V2321		7 2321 pass.

(Continued)

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
41	29	MSGLV	1	Message level.
42	2A	-----	1	Not used.
43	2B	NAMESIZ	1	Size of dsname.
44	2C	LOADLIST	4	↑Load parameter list.
48	30	CANDSIZE	4	Size of candidate list.

DYNAMIC ALLOCATION WORK TABLE, VARIABLE AREA (DAWTVARY) AS USED IN THE FREEING FUNCTION

Size:	48 bytes.
Located in:	DAWT -- (Subpool 252).
Created by:	IGC00099.
Used by:	IGC01099, IGC02099, IGC03099, IGC04099, IGC05099, IGC06099, IGC21099.
Updated by:	IGC01099, IGC02099, IGC03099, IGC04099, IGC05099, IGC06099, IGC21099.
Contents:	Work area for the UNALLOC function.

Operation Diagram
22

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	TEXTP	4	Address of the TIOT extension.
4	4	JCTP	4	Address of the JCT.
8	8	STTRP	4	Address of the SIOTTTR record.
12	C	SCTP	4	Address of the SCT.
16	10	SIOTP	4	Address of the SIOT.
20	14	JFCBP	4	Address of the JFCB.
24	18	WTOBUFP	4	Address of the WTO buffer.
28	1C	PARMP	4	Address of the macro parameters.
32	20	WMACP	4	Address of the macro volume list.
36	24	--	1	Reserved for the future.
37	25	--	3	Protect key.
40	28	---	8	Reserved for future use.

DYNAMIC ALLOCATION WORK TABLE, VARIABLE AREA (DAWTVARY) AS USED IN THE ATTRIBUTE CONVERSION FUNCTION

Size:	48 bytes.
Located in:	DAWT -- (Subpool 252).
Created by:	IGC00099.
Used by:	IGC16099, IGC17099.
Updated by:	IGC16099, IGC17099.
Contents:	Work area for the CONVERT function.

Operation Diagrams
23

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	DAWA1P	4	Address of DAWT work area 1 (DAWA1).
4	4	DTTR	4	TTR of the DAWA1 record.
8	8	DAWA2P	4	Address of DAWT work area 2 (DAWA2).
12	C	DTTR2	4	TTR of the DAWA2 record.
16	10	TIOT2P	4	Address of the second ddname TIOT entry.
20	14	TTRSIOT1	4	TTR of the first ddname SIOT.
24	18	TTRSIOT1	4	TTR of the second ddname SIOT.
NOTE: The following optional use of this word of storage takes effect if Bit 6 of VFLAGS is on to indicate the need for a second SIOTTR record.				
24	18	TTRSTTR2	3	TTR of SIOTTTR record.
27	1B	NUMTTR2	1	Number of the TTR for ddname2 SIOT.
28	1C	--	8	Initiator enqueue parameter list.
36	24	SYSDSN	8	Enqueue major name.
44	2C	VFLAGS	1	Bit settings, as follows: <u>Bit</u> <u>Meaning when on</u> 0 Reserved. 1 SIOT must be read. 2 Reserved. 3 JFCB must be read. 4 JFCB read only for DSE UPDATE. 5 Reserved. 6 SIOTTTR must be read. 7 Reserved.

4

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
45	2D	--	1	Reserved.
46	2E	TIOT2NO	1	Relative number of the second ddname TIOT entry.
47	2F	--	1	Reserved.

DYNAMIC ALLOCATION WORK TABLE, VARIABLE AREA (DAWTVARY) AS USED IN THE CONCATENATION FUNCTION

Size:	48 bytes.
Located in:	DAWT.
Created by:	IGC00099.
Used by:	IGC18099, IGC19099, IGC20099.
Updated by:	IGC18099, IGC19099.
Contents:	Work area for the CANCAT function.

Operation Diagrams
24

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	TEXTP	4	Address of the TIOT extension.
4	4	N	4	Number of TIOT entries.
8	8	STTRP	4	Address of subject SIOTTTR record.
12	C	TLNGH	4	Length of TIOT.
16	10	NTIOT	4	Address of new TIOT.
20	14	PCAT	4	Address of concatenate table (CATTAB).
24	18	PP3	4	Temporary pointer.
28	1C	J	4	Index variable.
32	20	Q	4	Queue manager parameters.
36	24	PPTR	4	Temporary pointer.
40	28	OPENNUMB	4	Number of open data sets.
44	2C	--	4	Reserved.

DYNAMIC ALLOCATION WORK TABLE, VARIABLE AREA (DAWTVARY) AS USED IN THE UPDATING FUNCTION

Size:	48 bytes.
Located in:	DAWT -- (Subpool 252).
Created by:	IGC25099.
Used by:	IGC25099, IGC26099, IGC27099, IGC29099.
Updated by:	IGC25099, IGC26099, IGC27099, IGC29099.
Contents:	Work area for the UPDATE function.

Operation Diagrams
26

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	DD1ADD	4	Address of the first ddname. OR Address of the TCB if Bit 3 of VFLAGS is on.
0	0	OLDDSE	4	Address of DSE to change.
4	4	DD2ADD	4	Address of the second ddname.
4	4	NSWDSE	4	Address of DSE to replace OLDDSE.
8	8	DAW1P	4	Address of DAWT work area 2 (DAWA2).
12	C	DSNADD	4	Address of the DSLNGTH dsname buffer.
12	C	ADDINIT	4	Address of first DSE.
16	10	ADDNAME1	4	Address of the DSE block containing first ddname or dsname.
16	10	LASTDSE	4	Address of last DSE.
20	14	ADDNAME2	4	Address of the DSE block containing the second ddname.
24	18	AFDSE	4	Address of the first DSE block.
28	1C	ANDSE	4	Address of the new DSE block.
32	20	ADSNDSE	4	Address of the next DSE on the chain.
36	24	------	8	Reserved for future use.

4

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
44	2C	VFLAGS	1	Bit settings, as follows:
				<u>Bit</u> <u>Meaning when on</u>
				0 First ddname found.
				1 Second ddname found.
				2 dsname found.
				3 Search on TCB address.
				4 Reserved for future use.
				5 Only one DSE in chain.
				6 Chain new DSE in front.
				7 Chain new DSE at end.
45	20	CONCOUNT	1	Number of DDs to be concatenated.
46	2E	--	2	Reserved.

ENQUEUE/DEQUEUE PARAMETER LIST FOR ALLOCATION/TERMINATION RESOURCE

Size:	40 bytes.
Located in:	Subpool 252.
Created by:	IGC08099, IGC10099.
Used by:	Normally -- IGC11099 (for enqueue) IGC15099 (for dequeue). Abnormally (in the case of any error exit) for dequeue by IGC11099, IGC12099, IGC15099.
Freed by:	The module that uses it for dequeue, in all cases.
Contents:	Contains the following fields for enqueuing or dequeuing from the allocation/termination resource (replaces the LIST form of the macro instruction). Note that the contents shown below repeat after the second ENDIND field. Note also that the routines enqueue using separate lists, but they dequeue using both lists at once.

Operation Diagrams
21

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	ENDIND	1	X'00' for dequeuing, using both lists at once; when the enqueue is accomplished, the using routine places X'00' in this field. X'FF' for enqueuing.
1	1	MINORLNG	1	X'02'.
2	2	OPTIONS	1	Bit settings as appropriate. (X'40' - Indicates an exclusive request for a system resource; do not wait until the resource is available.)
3	3	RTCODE	1	X'00'.
4	4	--	4	Address of major name.
8	8	--	4	Address of minor name 1.
12	C	ENDIND	1	X'FF' for both enqueue and dequeue.
(These fields repeat for concurrent use by the macro instruction.)				

4

ENQUEUE WORK AREA (EWA) FOR DATA SET NAME ENQUEUE

Size:	28 bytes.
Located in:	Subpool 252.
Created by:	IGC10099.
Used by:	Normally IGC10099 when enqueuing a dsname not previously associated with the job. IGC15099 when enqueuing a data set name to change the use attribute from share to exclusive (RET=CHNGE). Abnormally in the case of an error exit to DEQ the data set name in IGC10099, IGC11099, IGC12099, and IGC15099.
Updated by:	IGC10099
Contents:	Parameter list for the ENQ macro instruction for the data set name, and the parameters for the WAIT/POST mechanism with the initiator.

Operation Diagrams
21

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	INITWRD1	4	Flags (first byte of the field only) with the following meanings when the bit is on: Bit 0 -- Indicates that ENQ is to be done. Bit 1 -- Indicates that DEQ is to be done. Bits 2-7 -- Reserved. Bytes 1-3 -- Address of the ENQ/DEQ parameter list.
4	4	INITWRD2	4	The ECB on which the module waits for the initiator to complete the requested function.
8	8	ENQPARMS	20	Parameter list for enqueuing or dequeuing the data set name.

PATTERN CONSTRUCTION AREA (PCA)

Size:	Variable
Located in:	Subpool 252
Created by:	IGC12099
Used by:	IGC13099
Freed by:	IGC13099 normally, or by IGC12099 in case of an error.
Contents:	Bit pattern for the main UCBs and 2321 subcells eligible to satisfy this request for dynamic allocation.

Operation Diagrams
21

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	None	4	Reserved.
4	4	UCBCNT	2	Number of UCBs represented in this table.
6	6	PCCNT	2	Total number of eligible devices.
8	8	DCCNT	2	Number of datacells represented in this table.
10	A	PCASIZE	2	Size of this pattern construction area.
12	C	MPA	Var.	Bit pattern representing the eligible devices.

4

SYSOUT WORK AREA

Size:	40 bytes.
Located in:	Subpool 252.
Created by:	IGC08099.
Used by:	IGC08099, IGC09099.
Freed by:	IGC09099.
Contents:	Space for the system generated data set name for the SYSOUT data set.

Operation Diagrams
21

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	SWA	40	System generated data set name for the SYSOUT data set.

UNALLOCATE WORK AREA

Size:	584 bytes.
Located in:	Subpool 252.
Created by:	IGC01099.
Used by:	IGC01099, IGC02099, IGC03099, IGC04099, IGC05099, IGC06099.
Updated by:	IGC01099, IGC02099, IGC03099, IGC04099, IGC05099, IGC06099.
Contents:	Contains control blocks, lists and parameters, and a WTO buffer.

Operation Diagrams
22

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	SIOT	176	Step Input/Output table.
176	B0	JFCB	176	Job file control block.
352	160	--	124	WTO buffer.
476	1DC	--	60	Macro volume list.
536	218	--	16	Macro parameter list.
552	228	--	32	Queue manager external parameters.

4

Section 6: Diagnostic Aids

This section contains the following information:

- Messages: DAIR/Dynamic Allocation (Figure 34) -- a summary of messages related to failures in DAIR and SVC 99.
- Register Usage (Figure 35) -- a summary of the use of general registers 0-15 for DAIR.
- Register Usage (Figure 36) -- a summary of the use of general registers 0-15 for SVC 99.
- Return Codes from the Dynamic Allocation Interface Routine (Figure 37) -- a summary of DAIR return codes and their meanings.
- Return Codes from SVC 99 Dynamic Allocation Routines (Figure 38) -- a summary of dynamic allocation return codes and their meanings. DAIR places this return code in the DAnnDARC field of the DAPBnn, where nn is the DAIR entry code.

Other useful diagnostic information is contained in the DAIR work area (DAIRWA) and DAIR parameter blocks. These data areas are described in Section 5.

Message ID	Message	Issued by
IKJ56220I	'1' NOT ALLOCATED, TOO MAY DATA SETS + USE THE FREE COMMAND TO FREE UNUSED DATA SETS	IKJEFF18
IKJ56223I	COMMAND SYSTEM ERROR + DAIR ERROR CODE	
IKJ56224I	INVALID SYSOUT CLASS	
IKJ56225I	DATA SET dsname ALREADY IN USE, TRY LATER + DATA SET IS ALLOCATED TO ANOTHER JOB OR USER	
IKJ56226I	INVALID DATA SET NAME dsname EXCEEDS 44 CHARACTERS	
IKJ56227I	DATA SET dsname ALLOCATED FOR SHARED USE ONLY + DATA SET IS ALLOCATED TO ANOTHER JOB OR USER	
IKJ56227I	'1' NOT ALLOCATED, REQUESTED VOLUME UNAVAILABLE + VOLUME volume name NOT ON SYSTEM OR NOT ELIGIBLE FOR YOUR USE	
IKJ56228I	DATA SET dsname NOT IN CATALOG	
IKJ56229I	DATA SET dsname NOT ALLOCATED, CATALOG ERROR + DATA SET NAME ALREADY IN CATALOG OR WILL CREATE AN INVALID INDEX STRUCTURE	
IKJ56230I	DATA SET dsname NOT UNALLOCATED, MEMBER OF CONCATENATION	
IKJ56231	(1) NOT ALLOCATED, SYSTEM OR INSTALLATION ERROR - CATALOG ERROR CODE 14 or CATALOG I/O ERROR or DYNAMIC ALLOCATION ERROR CODE code	
IKJ56232I	DATA SET dsname NOT ALLOCATED, DATA SET NOT ON VOLUME + CATALOG OR VOLUME INFORMATION INCORRECT	
IKJ56233I	DUPLICATE ATTRIBUTE LIST NAME attribute list name	
IKJ56234I	ATTRIBUTE LIST NAME name NOT FOUND	

Figure 34. Messages: DAIR/Dynamic Allocation (Part 1 of 2)

Message ID	Message	Issued by
IKJ56235I	MEMBER NAME SPECIFIED BUT dsname NOT A PARTITIONED DATA SET	
IKJ56236I	FILE {STEPLIB / JOBLIB} INVALID, FILENAME RESTRICTED	
IKJ56237I	DATA SET dsname NOT ON A DIRECT ACCESS DEVICE NOT SUPPORTED	
IKJ56238I	'1' NOT FREED + SUBALLOCATED DATA SET or GENERATION DATA GROUP or PASSED DATA SET	
IKJ56239I	FILE filename NOT ALLOCATED, filename CURRENTLY ALLOCATED AS A DUMMY + FREE filename AND RE-ENTER COMMAND	
IKJ56241I	(1) NOT ALLOCATED + NO UNIT AVAILABLE or INVALID UNIT IN USER ATTRIBUTE DATA SET	IKJEFF18
IKJ56243I	DATA SET dsname RESIDES ON MULTIPLE VOLUMES, NOT SUPPORTED	
IKJ56244I	DATA SET dsname NOT ALLOCATED, DIRECTORY LARGER THAN PRIMARY QUANTITY	
IKJ56245I	DATA SET dsname NOT ALLOCATED, NOT ENOUGH SPACE ON VOLUMES + USE DELETE COMMAND TO DELETE UNUNSED DATA SETS	
IKJ56246I	DATA SET dsname NOT ALLOCATED, FILE IN USE	
IKJ56247I	{FILE ddname / DATA SET dsname} NOT FREED, IS NOT ALLOCATED	
IKJ56248I	DATA SET dsname NOT ALLOCATED, REQUESTED AS NEW BUT CURRENTLY ALLOCATED	
IKJ56249I	DATA SET dsname NOT ALLOCATED, CURRENTLY ALLOCATED WITH DISPOSITION OF DELETE	

Figure 34. Messages: DAIR/Dynamic Allocation (Part 2 of 2)

Note: The messages in the table above do not emanate automatically as a result of failures in DAIR or SVC 99 dynamic allocation. They are available, however, to any user who wishes interpretive information regarding the error return codes from SVC 99 to DAIR, specifically, or regarding DAIR/dynamic allocation failures in general. To get the messages, any user of DAIR (for example, a command processor, or a command processor user) invokes IKJEFF18, the DAIR error code analyser.

	IKJEFD00	
Register	Name	Contents
0	R0	Work register.
1	R1	Work register.
2	R2	Base register.
3	--	Work register.
4	--	Work register.
5	--	↑ DAIR parameter list.
6	--	Base register.
7	--	Work register.
8	Q	↑GETMAIN area.
9	--	Work register.
10	--	Work register.
11	--	Base register.
12	--	Work register.
13	R13	↑ Auto complier -obtained storage.
14	R14	Work register.
15	R15	Return code.

Figure 35. Register Usage: Dynamic Allocation Interface Routine (DAIR)

	IGC00099		IGC01099	
Register	Name	Contents	Name	Contents
0	--	Work Register	--	Work Register
1	--	↑SVC Parameter List, or Work Area	--	Work Register
2	--	Work Register	--	Work Register
3	--	Count of DDNAMEs to be Checked for JOBLIB or STEPLIB, or Correct Record of SIOTTTR Record	--	↑SIOT
4	--	↑TCB	--	↑TCB
5	--	↑SVRB	--	↑Enqueue Parameters
6	--	↑DAWT	--	↑DAWT
7	--	↑SVC Parameter List	--	↑TIOT
8	--	Work Register	--	↑WTO Buffer
9	--	Work Register	--	↑UCB
10	--	↑TIOT Entry When Searching for DNAME Entry	--	↑Subcell
11	--	Base Register	--	Base Register
12	--	↑Work Area	--	↑Work Area
13	--	Count of DDs to be Concatenated	--	↑Save Area
14	--	Return Register for Subroutines	--	Work Register
15	--	Return Code	--	Return Code

	IGC02099		IGC03099	
Register	Name	Contents	Name	Contents
0	--	Work Register	--	Work Register
1	--	Work Register	--	Work Register
2	--	Work Register	--	↑DAWT
3	--	↑Macro Parameters	--	↑TIOT
4	--	↑TCB	--	↑TCB
5	--	↑UCB ENQ Parameters	--	↑For ENQ Macro
6	--	Work Register	--	Work Register
7	--	↑JFCB, or TSOCVT	--	Temporary SIOT Pointer
8	--	↑SIOT, Contains TJID	--	Length of DSNAME
9	--	Work Register	--	↑DSENQ Name
10	--	↑TJB	--	Work Register
11	--	Base Register	--	Base Register
12	--	↑Work Area	--	↑Work Area
13	--	↑WTO Buffer	--	↑ENQ Parameter List
14	--	Return Register for Subroutines	--	Work Register
15	--	Return Code	--	Return Code

Figure 36. Register Usage: SVC 99 Routines (Part 1 of 8)

	IGC04099		IGC05099	
Register	Name	Contents	Name	Contents
0	--	Work Register	--	REGMAIN Length Parameter Register
1	--	Work Register	--	Entry Parameter Register
2	--	Work Register	--	Work Register
3	--	↑Parameters	--	↑DSB
4	--	↑TCB	--	↑TCB
5	--	Work Register	--	Work Register
6	--	↑TIOT Extension	--	Work Register
7	--	Work Register	--	JCT/SCT Core Length
8	--	Work Register	--	Work Register
9	--	Work Register	--	↑DAWT
10	--	↑WTO Buffer	--	Work Register
11	--	Base Register	--	Base Register
12	--	↑Work Area	--	↑Work Area
13	--	Work Register	--	↑Into DSB
14	--	Return Register for Subroutines	--	Return Register for Subroutines
15	--	Return Code	--	Return Code Register

	IGC06099		IGC07099	
Register	Name	Contents	Name	Contents
0	--	Work Register	--	Work Register
1	--	Work Register	--	Work Register
2	--	Work Register	--	↑DAWA2; ↑DSNAME Buffer
3	--	Work Register	--	Work Register
4	--	↑TCB	--	Work Register
5	--	UCB ENQ Pointer	--	↑DAWT
6	--	Work Register	--	↑SVC Parameters
7	--	Work Register	--	↑TIOT Entry
8	--	Work Register	--	Work Register
9	--	Work Register	--	Work Register
10	--	Work Register	--	Work Register
11	--	Base Register	--	Base Register
12	--	↑Work Area	--	↑Work Area
13	--	Work Register	--	Work Register
14	--	Return Register for Subroutines	--	Return Register for Subroutines
15	--	Return Code	--	Return Code

Figure 36. Register Usage: SVC 99 Routines (Part 2 of 8)

	IGC08099		IGC09099	
Register	Name	Contents	Name	Contents
0	--	Work Register	--	Work Register
1	--	Work Register	--	Work Register
2	--	↑SVC Parameters	--	↑DSNAME Buffer
3	--	Work Register	--	Work Register
4	--	Work Register	--	Work Register
5	--	↑DAWT	--	↑DAWT
6	--	Work Register	--	↑SVC Parameters
7	--	↑WTP ENQ Parameters	--	↑JFCB
8	--	Work Register	--	Work Register
9	--	↑WTP Block	--	↑SIOT
10	--	↑SIOT	--	Work Register
11	--	Base Register	--	Base Register
12	--	↑Work Area	--	↑Work Area
13	--	Work Register	--	Work Register
14	--	Return Register for Subroutines	--	Work Register
15	--	Return Codes	--	Return Code

	IGC10099		IGC11099	
Register	Name	Contents	Name	Contents
0	--	Work Register	--	Work Register
1	--	Work Register	--	Work Register
2	--	Work Register	--	↑UCB
3	--	↑Current DSENQ Entry	--	Entry # in Lookup Table
4	--	↑DSENQ Table	--	Work Register
5	--	↑DAWT	--	↑DAWT
6	--	Work Register	--	↑Device Name Table
7	--	Work Register	--	Work Register
8	--	↑DSENQ Parameters	--	↑User Parameter List
9	--	↑End of DSENQ Table	--	Work Register
10	--	Work Register	--	Work Register
11	--	Base Register	--	Base Register
12	--	↑Work Area	--	↑Work Area
13	--	Work Register	--	Work Register
14	--	Return Register for Subroutines	--	Return Register for Subroutines
15	--	Return Code	--	Return Code

Figure 36. Register Usage: SVC 99 Routines (Part 3 of 8)

	IGC12099		IGC13099	
Register	Name	Contents	Name	Contents
0	--	Work Register	--	Work Register
1	--	Work Register	--	Work Register
2	--	Work Register	--	Entry Number in IOS Lookup Table
3	--	Number of 2321 Devices in the System	--	Work Register
4	--	Work Register	--	↑Main UCB
5	--	↑DAWT	--	Count Used in Shifting Bit Pattern
6	--	↑UCB; Shift Count for Creating PCA	--	Entries in UCBLIST
7	--	↑DMT; Relative Word in the PCA	--	↑DAWT
8	--	Entry Number in the IOS Lookup Table	--	Work Register
9	--	↑DSENQ Entry; Number of Devices in System	--	Contains One Word of Pattern
10	--	Indicates 2321 in the System	--	Eligible Bits in Pattern
11	--	Base Register	--	Base Register
12	--	↑Work Area	--	↑Work Area
13	--	DMT Entry Length; PCA	--	↑UCBLIST
14	--	Return Register for Subroutines	--	Return Register for Subroutines
15	--	Return Code	--	Work Register

	IGC14099		IGC15099	
Register	Name	Contents	Name	Contents
0	--	Work Register	--	Work Register
1	--	Work Register	--	Work Register
2	--	Work Register	--	↑DSNAME ENQ Parameters
3	--	Work Register	--	↑TIOT Entry
4	--	Work Register	--	Work Register
5	--	↑UCB	--	↑UCB
6	--	Work Register	--	↑DAWT
7	--	↑Channel Load Table	--	↑User Parameter List
8	--	↑DAWT	--	Work Register
9	--	↑Candidate List	--	Work Register
10	--	Work Register	--	Work Register
11	--	Base Register	--	Base Register
12	--	↑Work Area	--	↑Work Area
13	--	↑UCBLIST	--	Work Register
14	--	Return Register for Subroutines	--	Return Register for Subroutines
15	--	Return Code	--	Return Code

Figure 36. Register Usage: SVC 99 Routines (Part 4 of 8)

	IGC16099		IGC17099	
Register	Name	Contents	Name	Contents
0	--	Work Register	--	Work Register
1	--	Work Register	--	Work Register
2	--	Work Register	--	Work Register
3	--	Work Register	--	↑DAWT
4	--	↑TCB	--	Work Register
5	--	↑DDNAME2 TIOT Entry	--	Work Register
6	--	Work Register	--	Work Register
7	--	↑DAWT	--	Work Register
8	--	↑SVC Parameter List	--	Work Register
9	--	↑SIOT in DAWA2	--	Work Register
10	--	Work Register	--	Work Register
11	--	Base Register	--	Base Register
12	--	↑Work Area	--	↑Work Area
13	--	↑Save Area	--	Work Register
14	--	Return Register for Subroutines	--	Return Register for Subroutines
15	--	Return Code	--	Return Code

	IGC18099		IGC19099	
Register	Name	Contents	Name	Contents
0	--	Work Register	--	Work Register
1	--	↑For Moving TIOT	--	Work Register
2	--	Work Register	--	Work Register
3	--	Work Register	--	Work Register
4	--	Work Register	--	↑TCB
5	--	Work Register	--	Work Register
6	--	Work Register	--	Work Register
7	--	Register to Search TIOT	--	Work Register
8	--	↑SVC Parameter List	--	Work Register
9	--	Work Register	--	Work Register
10	--	Work Register	--	Work Register
11	--	Base Register	--	Base Register
12	--	↑Work Area	--	↑Work Area
13	--	↑Save Area	--	Work Register
14	--	Work Register	--	Work Register
15	--	Return Code	--	Return Code

Figure 36. Register Usage: SVC 99 Routines (Part 5 of 8)

	IGC20099		IGC21099	
Register	Name	Contents	Name	Contents
0	--	Work Register	--	Work Register
1	--	Work Register	--	Work Register
2	--	Work Register	--	Work Register
3	--	Loop Control	--	Work Register
4	--	↑TCB	--	↑TCB
5	--	Work Register	--	↑UCB ENQ Parameters
6	--	Work Register	--	Work Register
7	--	Loop Control	--	↑DAWT
8	--	Work Register	--	↑TSCVT
9	--	Work Register	--	Work Register
10	--	Work Register	--	Work Register
11	--	Base Register	--	Base Register
12	--	↑Work Area	--	↑Work Area
13	--	Work Register	--	↑To Search TJBs
14	--	Work Register	--	Return Register for Subroutine
15	--	Return Code	--	Return Code

	IGC23099		IGC25099	
Register	Name	Contents	Name	Contents
0	--	Work Register	--	Storage amount for GETMAIN
1	--	Work Register	--	↑To Storage Obtained
2	--	↑DAWT	--	Work Register
3	--	Records to be Read from SIOTTTR Table	--	↑DAWT
4	--	Work Register	--	Work Register
5	--	Number of DDNAMEs	--	↑First DSE
6	--	Work Register	--	Work Register
7	--	↑TIOT Entry	--	Work Register
8	--	Work Register	--	Work Register
9	--	↑TIOT Entry When Searching for Duplicate DDNAMEs	--	Work Register
10	--	Work Register	--	Work Register
11	--	Base Register	--	Base Register
12	--	↑Work Area	--	↑Work Area
13	--	↑Save area	--	Work Register
14	--	Work Register	--	Work Register
15	--	Return Code	--	Return Code

Figure 36. Register Usage: SVC 99 Routines (Part 6 of 8)

	IGC26099		IGC27099	
Register	Name	Contents	Name	Contents
0	--	Work Register	--	Work Register
1	--	Work Register	--	Work Register
2	--	↑DSE Update List	--	Work Register
3	--	Work Register	--	↑DAWT
4	--	Work Register	--	↑First DSE
5	--	Work Register	--	↑TIOT
6	--	Work Register	--	Work Register
7	--	Work Register	--	Work Register
8	--	↑SVC Parameter List	--	Work Register
9	--	Work Register	--	Work Register
10	--	↑TIOT Entry	--	Gets TCB Address
11	--	Base Register	--	Base Register
12	--	↑Work Area	--	↑Work Area
13	--	↑New DSE; Saves Return Code	--	↑TCB Being Used
14	--	Return Register for Subroutines	--	Return Register for Subroutines
15	--	Return Code	--	Return Code

	IGC28099		IGC29099	
Register	Name	Contents	Name	Contents
0	--	Storage length: GETMAIN & FREEMAIN	--	Storage Length: GETMAIN & FREEMAIN
1	--	↑DAWT Upon Entry	--	Storage Address: GETMAIN & FREEMAIN
2	--	↑For DD Lookup Table Entries	--	↑DSE
3	--	↑For Device Entries in TCT I/O Table	--	↑DAWT
4	--	Loop Control	--	Work Register
5	--	↑SMF Record	--	DSE Forward Pointer
6	--	Work Register	--	Work Register
7	--	↑TIOT	--	Work Register
8	--	Work Register	--	Work Register
9	--	Work Register	--	Work Register
10	--	Work Register	--	Work Register
11	--	Work Register	--	Base Register
12	--	Number of UCBs per DD	--	↑Work Area
13	--	Loop Control	--	Saves Return Code
14	--	Work Register	--	Return Register for Subroutines
15	--	Return Code	--	Return Code

Figure 36. Register Usage: SVC 99 Routines (Part 7 of 8)

	IGC30099		IGC31099	
Register	Name	Contents	Name	Contents
0	--	Work Register	--	Work Register
1	--	Work Register	--	↑ First TCB in the Job Step
2	--	Work Register	--	Link Register
3	--	↑ DAWT	--	↑ CVT at Exit, Used for TCB Search
4	--	↑ TCB	--	↑ Current TCB
5	--	↑ SVC Parameter List	--	Work Register
6	--	Work Register	--	↑ TIOT
7	--	Work Register	--	Used in Task Selection
8	--	Work Register	--	Save Register
9	--	Work Register	--	Base Register
10	--	Work Register	--	↑ Alternate TCB; also Used for XCTL
11	--	Base Register	--	↑ DCB
12	--	Work Register	--	↑ DEB; Work Register
13	--	Work Register	--	↑ Save Area
14	--	Work Register	--	Work Register
15	--	Return Codes	--	DD Offset from Beginning of TIOT

Figure 36. Register Usage: SVC 99 Routines (Part 8 of 8)

Routine	Return Code Hexadecimal	Meaning
IKJEFD00	00	Successful.
	04	Invalid parameter list.
	08	Error in catalog management routine. The catalog management return code is stored in the CTRC field.
	0C	Error in dynamic allocation. The dynamic allocation return code is stored in the DARC field.
	10	No entries available for use in the TIOT.
	14	DDNAME requested is currently unavailable.
	18	DSNAME requested is a member of a concatenated group.
	1C	DDNAME or DSNAME specified is not currently allocated.
	20	Error in catalog information routine.
	30	System error -- the completion code is xxxx, an ABEND code that DAIR receives from the STAE macro instruction.

Figure 37. Return Codes: Dynamic Allocation Interface Routine (DAIR)

Return Code Hexadecimal	Meaning	Routine *
0000	Dynamic Allocation completed successfully.	All
0004	Dynamic Allocation could not delete a table that was loaded using a LOAD macro instruction. The data set is still allocated.	IGC11099 IGC12099 IGC13099
0008	The temporary data set was freed and deleted. The disposition specified by the calling routine is invalid for a temporary data set.	UNALLOC
002w	The data set was successfully freed, but the disposition (catalog or uncatalog) was unsuccessful. The hexadecimal digit 'w' is a code indicating the reason for the failure. w Explanation 1 A control volume was required and a utility program must be used to catalog the data set. 2 The data set to be cataloged had previously been cataloged or the data set to be uncataloged could not be located, or no change was made to the volume serial list of a data set with a disposition of CATLG. 3 A specified index did not exist. 4 The data set could not be cataloged because space was not available on the specified volume. 5 Too many volumes were specified for tne data set; because of this, not enough main storage was available to perform the specified cataloging. 6 The data set to be cataloged in a generation index is improperly named. 7 The data set to be cataloged was not opened and no density information was provided. (For dual density tape requests only). 9 An uncorrectable input/output error occurred in reading or writing the catalog.	IGC03099
003x	The data set was successfully freed, but the requested disposition (delete) was unsuccessful. The hexadecimal digit 'x' is a code indicating the reason for failure. x Explanation 1 The expiration date had not occurred. 4 No device was available for mounting during deletion. 5 Too many volumes were specified for deletion. 6 Either no volumes were mounted or the mounted volumes could not be demounted to permit the remaining volumes to be mounted. 8 The SCRATCH routine could not delete the data set from the volume. 9 A job was cancelled and was deleted from any one of the following queues: Input Queues Background Reader Queue Hold Queue Automatic SYSIN Batching (ASB) Queue Output Queues	IGC03099
0104	Dynamic Allocation encountered an I/O error while attempting to read from SYS1.SYSJOBQE.	All

*Note: The symbolic name for the first in a series of routines. For a complete list of SVC 99 Dynamic Allocation routines, refer to Figure 29--Program Hierarchy: SVC 99 Routines

Figure 38. Return Codes: SVC 99 Routines (Part 1 of 5)

Return Code Hexadecimal	Meaning	Routine *
0108	Dynamic Allocation encountered an I/O error while attempting to write to SYS1.SYSJOBQE.	IGC03099 IGC04099 IGC05099 IGC08099 IGC11099 IGC15099 IGC16099 IGC17099 IGC20099
010C	Dynamic Allocation encountered an I/O error while enqueueing on SYS1.SYSJOBQE.	IGC05099
0204	Reserved.	
0208	No space is available on SYS1.SYSJOBQE.	IGC04099 IGC08099 IGC09099 IGC10099
020C	The calling routine made a request for the exclusive use of a shared data set. The request can not be honored.	IGC15099 IGC16099
0210	The data set requested is not available. This data set is allocated to another job and its usage attributes conflict with this request.	IGC10099
0214	A direct access device is not available. To be available it must satisfy the following requirements: o It must be online. o It must be ready. o It must not be pending offline. o It must not be pending an unload. o It must be shareable. o A MOUNT message must not be currently outstanding. o The volume attributes must have been defined.	IGC13099
0218	The required volume was not mounted on an available device. (See Dynamic Allocation return code 214 for the requirements for an available device.)	IGC13099
021C	Incorrect unitname supplied.	IGC11099 IGC12099
0220 through 0264	Reserved.	
0268	Concatenation was requested, but the DCBTIOT offset cannot be found in this job's DEB/DCB chain.	IGC18099
0304	The ddname was not specified by the calling routine.	IGC00099 IGC16099 IGC18099
0308	The ddname specified by the calling routine was not found.	IGC00099 IGC018099
030C	An invalid function code was specified by the calling routine.	IGC00099
0310	The "exchange" option was specified by the calling program and the TIOT entry for the second (new) ddname could not be found.	IGC16099
0314	Restoring ddnames, as per this request, would have resulted in duplicate ddnames -- duplicate ddnames are not permitted.	IGC23099

*Note: The symbolic name for the first in a series of routines. For a complete list of SVC 99 Dynamic Allocation routines, refer to Figure 29 -- Program Hierarchy: SVC 99 Routines

Figure 38. Return Codes: SVC 99 Routines (Part 2 of 5)

Return Code Hexadecimal	Meaning	Routine *
0318	Invalid characters are present in the ddname provided by the caller.	IGC07099 IGC16099
031C	Invalid characters are present in the membername provided by the caller.	IGC07099 IGC16099
0320	Invalid characters are present in the dsname provided by the caller.	IGC07099
0324	Invalid characters are present in the SYSOUT program name provided by the caller.	IGC07099
0328	Invalid characters are present in the SYSOUT form number provided by the caller.	IGC07099
032C	An invalid SYSOUT class was specified by the caller.	IGC04099 IGC07099
0330	A membername was specified but the data set is not a partitioned data set.**	IGC16099
0334	The supplied data set name exceeded 44 characters in length.	IGC07099
0338	The data set disposition specified by the caller is invalid.	IGC01099 IGC09099
033C	More than one mutually exclusive keyword (DSNAME, DUMMY, TERM, or SYSOUT) was specified.	IGC07099
0340	The dsname was not specified and the disposition was not "new". (If the disposition is "new" the dsname may be omitted.)	IGC07099
0344	Dynamic Allocation was specified in a non-TSO environment.	IGC00099
0348-034C	Reserved.	
0350	Jobname field contains zeros. This field may be blank, but may not contain zeros.	IGC04099
0354	Reserved.	--
0358	DELETE cannot be specified if the data set is shared.	IGC01099
035C-0360	Reserved.	--
0364	JOBLIB DDNAME or STEPLIB DDNAME can not be specified. These data sets have been opened and thus cannot be allocated.	IGC00099
0404	The device to be unallocated is not a direct access device. (Only direct access devices are supported for dynamic allocation.)	IGC01099
0408	The new DDNAME is a duplicate of a DDNAME in the TIOT. The calling routine requested allocation of a file name (DDNAME) already used for the job.	IGC07099 IGC16099
040C	The specified ddname is associated with a DYNAM entry. DYNAM entries may not be concatenated.	IGC18099

*Note: The symbolic name for the first in a series of routines. For a complete list of SVC 99 Dynamic Allocation routines, refer to Figure 29 -- Program Hierarchy: SVC 99 Routines

**Note: The SVC 99 routines do not issue these macro instructions. Instead, DAIR receives these codes in the DAxxDARC field of the DAIR parameter block for the DAIRxx subroutine that issued the macro; xx is the number of the issuing subroutine and its parameter block.

Figure 38. Return Codes: SVC 99 Routines (Part 3 of 5)

Return Code Hexadecimal	Meaning	Routine *
0410	The specified ddname is allocated to a data set. The ddname must be associated with a DYNAM entry.	IGC07099
0414	The specified ddname is already allocated to a terminal entry (TERM=TS).	IGC07099
0418	The referenced data set is a member of a concatenated data group. If the data set was dynamically concatenated it must be deconcatenated before this request can be honored. If concatenated at LOGON, the data set may not be freed until LOGOFF.	IGC01099 IGC07099
041C	The referenced data set is a multi-volume data set. Multi-volume data sets (data sets on more than one volume) are not supported by Dynamic Allocation.	IGC01099 IGC09099
0420	The specified ddname is associated with an open data set. (A data set must be closed to be used by Dynamic Allocation.)	IGC01099 IGC07099 IGC16099 IGC18099 IGC23099
0424	Reserved.	
0428	The specified ddname is part of a previously allocated space. Dynamic Allocation cannot free it.	IGC01099
042C	The ddname to be freed is associated with a generation data group. Generation data groups are not supported in Dynamic Allocation.	IGC01099
0430	The specified ddname is associated with a passed data set. Passed data sets cannot be freed or converted.	IGC01099 IGC17099
0504	A serious error of undetermined cause has occurred involving system data.	IGC25099 IGC27099 IGC29099
x7zz	A return code of this form consists of an identifier (x) representing the system macro instruction returning the code, and the code itself (zz) returned by the macro instruction. If "x" equals 1, the LOCATE macro instruction returned the code.** If "x" equals 4, the DADSM macro instruction returned the code. If "x" equals 6, the OBTAIN macro instruction returned the code.** "zz" is the low order byte from register 15 as returned by the macro instruction. The return codes for the LOCATE and the OBTAIN macro instructions are described in IBM System/360 Operating System: System Programmer's Guide, GC28-6550. The return codes for the DADSM macro instruction are as follows: Code — Meaning 00 — The operation completed successfully. 04 — Duplicate name DSCB. 08 — No available DSCB's in the VTOC. 0C — A permanent I/O error occurred in reading or writing a DSCB.	IGC09099 IGC14099 IGC26099

*Note: The symbolic name for the first in a series of routines. For a complete list of SVC 99 Dynamic Allocation routines, refer to Figure 29 -- Program Hierarchy: SVC 99 Routines

**Note: The SVC 99 routines do not issue these macro instructions. Instead, DAIR receives these codes in the DAxxDARC field of the DAIR parameter block for the DAIRxx subroutine that issued the macro; xx is the number of the issuing subroutine and its parameter block.

Figure 38. Return Codes: SVC 99 Routines (Part 4 of 5)

Return Code Hexadecimal	Meaning	Routine *
	Code — Meaning (continued)	
	10 — The absolute track requested is not available.	
	14 — The quantity of space requested is not available.	
	18 — The record length specified is greater than the track length.	
	30 — The number of tracks requested for a split cylinder data set is greater than the number of tracks per cylinder.	
	34 — The disk pack is a DOS volume and the request is not absolute track.	
	38 — The primary quantity of space requested is less than the directory quantity requested.	

*Note: The symbolic name for the first in a series of routines. For a complete list of SVC 99 Dynamic Allocation routines, refer to Figure 29 -- Program Hierarchy: SVC 99 Routines

Figure 38. Return Codes: SVC 99 Routines (Part 5 of 5)

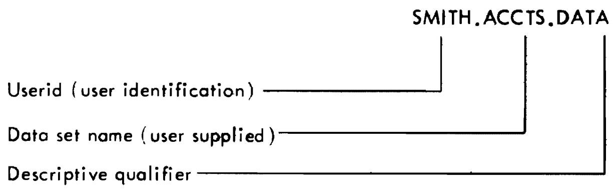
Part 5: Default Service Routine

5

Section 1: Introduction

The default service routine constructs a fully qualified data set name, when provided a partially qualified data set name by the calling routine. A fully qualified data set name has three fields: a userid, a data set name, and a descriptive qualifier.

For example:



For a more detailed description of data set naming conventions, refer to OS/VS2 TSO Command Language Reference, GC28-0646.

In general, default gets control if the terminal user refers to a data set without giving a fully qualified name. The calling routine provides default with the address of the DFPL (default parameter list), which contains the address of the DFPB (default parameter block). The DFPB contains an address that contains the data set name, as provided by the terminal user.

Default prefixes the userid to the data set name, checks the data set name against the system catalog, and if necessary either inserts the proper qualifier or prompts the user to choose a qualifier.

As supplied with TSO, the default service routine resides in SYS1.LINKLIB or SYS1.CMDLIB and executes in the user's foreground region with the protection key assigned to that region. An installation may choose to make default resident in the TSO link pack area (TSLPA) in the region assigned to the Time Sharing Control Task (TSCT). The default service routine requires about 4,000 bytes of main storage.

5

Section 2: Method of Operation

Method of Operation Diagram 29 shows how the default service routine constructs a fully qualified data set name.

The default service routine works this way:

- A TSO command processor calls default quality the data set name. The calling program passes the address of the default parameter list (DFPL) in register 1.
- The DFPL points to tne default parameter block (DFPB), which contains the entry and control codes. The entry codes tell default to use the qualifier provided by the calling routine, the terminal user, or the system catalog. Additional functions that default can performed are specified by control codes. (For example: An entry code of X'04' causes the system catalog to be searched for a data set name qualifier, and a control code of X'20' causes the data set name to be prefixed with the userid.)
- Default uses the catalog information routine (IKJEHCIR) to search the system catalog and find one or more data set qualifiers. If the routine finds more than one qualifier, default prompts the terminal user to choose one of them.
- Default prefixes the userid and adds the data set qualifier, as required, and returns control to the calling program.

ENTRY TO DEFAULT

Default receives control by either a CALL or LINK macro instruction at entry point IKJDFLT in load module IKJEHDEF. At entry, register 1 points to the default parameter list (DFPL).

The default parameter list contains:

- The address of the user profile table (UPT).
- The address of the environment control table (ECT).
- The address of the event control block (ECB).
- The address of the default parameter block (DFPB).

The default parameter block contains an entry code and a control code. These codes are set by the calling program to specify the functions required. Figure 39 describes the entry codes and Figure 40 describes the control codes.

Entry Code	Function Requested	Functions Performed by Default
X'00'	Use the qualifier provided by the caller.	Uses qualifier from DFPB, as provided by the caller.
X'04'	Find a qualifier. If there is more than one, prompt the terminal user to choose one.	• Builds a list of possible qualifiers. • Prompts the terminal user to choose one. • Checks his response against the list.
X'08'	Find a descriptive qualifier, but don't prompt the terminal user.	• Builds a list of possible qualifiers. • Returns control to caller with a control code indicating more than one qualifier was found; thus prompting is necessary.
X'0C'	Use qualifier from DFPB or find one from system catalog, or use a new one submitted by the terminal user.	Does one of the following: • If a qualifier is provided in DFPB, uses it. • If no qualifier is provided: - Builds a list of possible qualifiers. - Sends list to terminal. - Prompts terminal user to choose one from the list or submit a new one.

Figure 39. Entry Codes: Default Service Routine

Control Code Flags	Functions Performed by Default
Bits 0-1	Not used (0).
Bit 2	Prefixes the given data set name with userid.
Bits 3-4	Not used (0).
Bit 5	Returns a copy of any added qualifier to the caller.
Bit 6	Uses the qualifier provided by the caller.
Bit 7	Issues a message telling the terminal user that an old data set is about to be reused.

Figure 40. Control Codes: Default Service Routine

5

PREFIXING USERID TO DSNAME

If bit 2 of the control code byte is on, default prefixes the userid, which was specified at LOGON, to the partially qualified data set name located in the data set buffer supplied by the caller. The address of this buffer is located in the DFPBDSN field of the default parameter block. The format of the data set name buffer is as follows:

Byte	Contents
0-1	The length of the data set name, in bytes.
2-45	The data set name, left justified and padded to the right with blanks.

SEARCHING THE SYSTEM CATALOG

Default invokes the catalog information rouine (IKJEHCIR) to search the system catalog for the required qualifiers. Default must supply the userid and the data set name as a search argument.

The catalog information routine does the following:

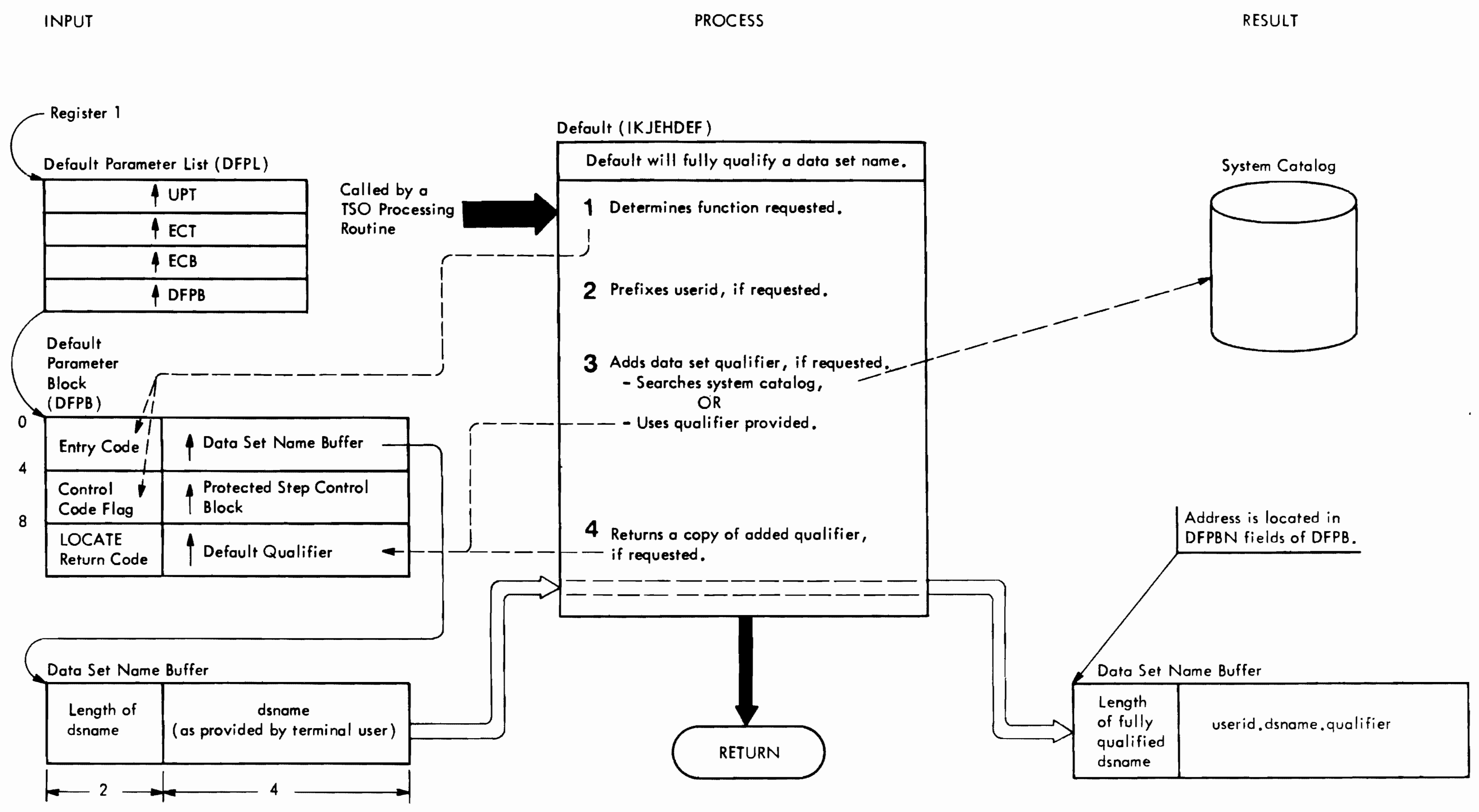
- Issues the LOCATE macro instruction to search the system catalog for the required qualifier.
- Returns a list of qualifiers to default.

EXIT FROM DEFAULT

Default returns to the calling control program by issuing a RETURN macro instruction. All registers, except register 15 which contains the return code, are restored. The return codes appear in Figure 43.

Method of Operation Diagrams

5



INPUT
PROCESS
RESULT
Register 1
Default Parameter List (DFPL)
UPT
ECT
ECB
DFPB
Default Parameter Block (DFPB)
0
4
8
Entry Code
Data Set Name Buffer
Control Code Flag
Protected Step Control Block
LOCATE Return Code
Default Qualifier
Data Set Name Buffer
Length of dsname
dsname (as provided by terminal user)
2
4
Called by a TSO Processing Routine
Default (IKJEHDEF)
Default will fully qualify a data set name.
1 Determines function requested.
2 Prefixes userid, if requested.
3 Adds data set qualifier, if requested.
- Searches system catalog,
OR
- Uses qualifier provided.
4 Returns a copy of added qualifier, if requested.
RETURN
System Catalog
Address is located in DFPBN fields of DFPB.
Data Set Name Buffer
Length of fully qualified dsname
userid.dsname.qualifier

Method of Operation Diagram 29. Default Service Routine (Part 1 of 2)

CROSS REFERENCE TABLE

Key Description	Routine	Label
1 Default performs one or more functions depending on the entry and control codes passed in the DFPB. The entry and control codes are as follows:	Default Service Routine	IKJEHDEF

Entry Code	Function performed by Default
X'00'	Use qualifier provided by terminal user.
X'04'	Search the system catalog for descriptive qualifiers. If more than one exists, build a list of possible qualifiers and prompt the terminal user to choose one.
X'08'	Search system catalog for descriptive qualifiers. Attempt to qualify the data set name but don't interrupt the terminal
X'0C'	Search system catalog for descriptive qualifiers. Accept qualifier provided by terminal user; alert him if it is an old data set.

Control Code Flag	Function performed by Default
Bit 2	Prefix given data set name with userid.
Bit 5	Returns a copy of any added qualifier to caller.
Bit 6	Use the qualifier provided by calling routine.
Bit 7	Issue message to terminal user.

Key Description	Routine	Label
2 Userid is obtained from the PSCBUSER field of the PSCB and is prefixed to the dsname in the data set name buffer. Default searches the system catalog by calling the Catalog Information Routine which in turn issues a LOCATE macro instruction.		ADDUSRID
3 If a data set qualifier is requested, and one is not supplied Default searches the system catalog by calling the Catalog Information Routine. The Catalog Information Routine in turn issues a LOCATE macro instruction. If a qualifier is supplied Default finds the qualifier by referring to location pointed to by the DFPBQUAL field of the DFPB.		CALLCIR
4 If the calling routine requests a copy of the inserted qualifier, Default inserts this copy in the location pointed to by the DFPBQUAL field of the DFPB.		ADDQUAL

Method of Operation Diagram 29. Default Service Routine (Part 2 of 2)

Section 3: Program Organization

This section describes the organization of the default service routine.

The default service routine consists of one load module, IKJEHDEF. As supplied with TSO, the default service routine resides in SYS1.LINKLIB or SYS1.CMDLIB, and requires about 4,000 bytes of main storage.

This table contains information to help you find the appropriate program description or assembly listing.

Label	Common Name	Load Module Name	Assembly module Name	Control Section Name	Description	Diagram
ADDNAME	--	IKJEHDEF	IKJEHDEF	IKJEHDEF	Add qualifier to dsname in buffer.	29
ADDQUAL	--	IKJEHDEF	IKJEHDEF	IKJEHDEF	Attach qualifier to dsname buffer.	29
ADDUSRID	--	IKJEHDEF	IKJEHDEF	IKJEHDEF	Prefix userid to dsname.	29
BLDLIST	--	IKJEHDEF	IKJEHDEF	IKJEHDEF	Build list of qualifiers for terminal user.	29
BADREPLY	--	IKJEHDEF	IKJEHDEF	IKJEHDEF	If improper reply, tell terminal user.	29
CALLCIR	--	IKJEHDEF	IKJEHDEF	IKJEHDEF	Call catalog information routine to find data set qualifiers from system catalog.	29
CHECKRC	--	IKJEHDEF	IKJEHDEF	IKJEHDEF	Check catalog information routine return code.	29
CLEANUP	--	IKJEHDEF	IKJEHDEF	IKJEHDEF	Restore registers, free core.	29
CNTRLTST	--	IKJEHDEF	IKJEHDEF	IKJEHDEF	Check if message is required.	29
COMPARE	--	IKJEHDEF	IKJEHDEF	IKJEHDEF	Check qualifier chosen by terminal user against list.	29
GETQUAL	--	IKJEHDEF	IKJEHDEF	IKJEHDEF	Prompt for qualifier.	29
IKJDFLT	--	IKJEHDEF	IKJEHDEF	IKJEHDEF	Entry point to default.	29
IKJEHDEF	--	IKJEHDEF	IKJEHDEF	IKJEHDEF	Control section.	29
NOTOC	--	IKJEHDEF	IKJEHDEF	IKJEHDEF	Check if userid is to be added.	29
RESPONSE	--	IKJEHDEF	IKJEHDEF	IKJEHDEF	Check response from terminal user.	29
TESTRC	--	IKJEHDEF	IKJEHDEF	IKJEHDEF	Handle PUTLINE return code.	29
TPUT	--	IKJEHDEF	IKJEHDEF	IKJEHDEF	Write a line to the terminal.	29
TPUTGET	--	IKJEHDEF	IKJEHDEF	IKJEHDEF	Prompt terminal user to choose a qualifier.	29

5

Section 5: Data Areas

This section contains the major data areas used by the default service routine. These areas include:

- CIRPARM -- Catalog Information Routine Parameter List
- CSPLARM -- Command Scan Parameter List
- DFPB -- Default Parameter Block
- DFPL -- Default Parameter List
- IOPL -- I/O Service Routine Parameter List

CATALOG INFORMATION ROUTINE PARAMETER LIST (CIRPARM)

Size:	20 Bytes.
Created by:	Default (IKJEHDEF).
Updated by:	N/A.
Used by:	Catalog Information Routine (IKJEHCIR).
Contents:	Parameter List for CIR (IKJEHCIR).

Operation Diagrams
29

Displacement Dec.	Displacement Hex.	Field Name	Size in Bytes	Contents
0	0	CIRPARM	4	CIR option in first byte. Other bytes are unused.
4	4	NAMETTR	4	Pointer to data set name or ttr (relative address).
8	8	CVOLIDPT	4	Pointer to CVOL ID.
12	C	WKAR1PTR	4	Pointer to 265 byte work area.
16	10	WKAR2PTR	4	Pointer to 18 word save area.

COMMAND SCAN PARAMETER LIST (CSPLARM)

Size:	24 Bytes.
Created by:	Default (IKJEHDEF) using mapping macro IKJCSPL.
Updated by:	N/A.
Used by:	Command Scan Routine (IKJSCAN).
Contents:	Parameter List for Command Scan (IKJSCAN).

Operation Diagrams
29

Displacement Dec.	Displacement Hex.	Field Name	Size in Bytes	Contents
0	0	CSPLUPT	4	↑ UPT.
4	4	CSPLECT	4	↑ ECT.
8	8	CSPLECB	4	↑ Command processor's ECB.
12	C	CSPLFLG	4	↑ a flag word.
16	10	CSPLOA	4	↑ Output area.
20	14	CSPLCBUF	4	↑ Command buffer.

5

DEFAULT PARAMETER BLOCK (DFPB)

Size:	12 Bytes.
Created by:	Calling program, using a mapping macro IKJDFPB.
Updated by:	N/A.
Used by:	IKJEHDEF.
Contents:	Parameter block for default (IKJEHDEF).

Operation Diagrams
29

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	DEPBCODE	1	Entry Code, (X'00',X'04',X'08', or X'0C).
1	1	DFPBDSN	3	↑ data set name buffer.
4	4	DFPBCNTL	1	Control code. Bit Meaning when on 0-1 Reserved (0). 2 USERID is to be prefixed. 3-4 Reserved (0). 5 Return added qualifier in buffer pointed to by DFPBQUAL. 6 Add qualifier pointed to by DFPBQUAL. 7 Issue message.
5	5	DFPBPSCB	3	↑ Protected step control block.
8	8	DFPBLORC	1	LOCATE return code. (Code returned here if LOCATE error.)
9	9	DFPBQUAL	3	↑ default qualifier either to be added by default or that was added by default (see bits 5 and 6 of DFPBCNTL).

DEFAULT PARAMETER LIST (DFPL)

Size	16 Bytes.
Created by:	Calling program, using mapping macro IKJDFPL.
Used by:	IKJEHDEF.
Contents:	Parameter List for Default -- IKJEHDEF.

Operation Diagrams
29

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	DFPLUPT	4	† User profile table (UPT).
4	4	DFPLECT	4	† Environment control table (ECT).
8	8	DFPLECB	4	† calling program's event control Block (ECB).
12	C	DFPLDFPB	4	† Default parameter block (DFPB).

I/O PARAMETER LIST (IOPL)

Size:	16 Bytes.
Created by:	Default (IKJEHDEF) using mapping macro IKJIOPL.
Updated by:	N/A.
Used by:	I/O service routines (IKJPUTL and IKJPTGT)
Contents:	Parameter list for I/O service routines.

Operation Diagrams
29

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	IOPLUPT	4	† UPT.
4	4	IOPLECT	4	† ECT.
8	8	IOPLECB	4	† Command Processor's ECB.
12	C	IOPLIOPB	4	† I/O service routine parameter block.

Section 6: Diagnostic Aids

This section includes the following information:

- Default Register usage chart, Figure 41.
- Default Service Routine messages, Figure 42.
- Default Service Routine Return Codes, Figure 43.

Register	Name	Use
0	R0	Work register
1	R1	Parameter register and work register
2	R2	Work register
3	R3	Work register
4	R4	Work register
5	R5	Work register
6	R6	Contains address of defulat parameter block
7	R7	Not used
8	R8	Work register
9	R9	Work register
10	R10	Work register
11	R11	Work register
12	R12	Base register for default
13	R13	Save area register and DSECT base register
14	R14	Link register
15	R15	Branch and return code register

Figure 41. Register Usage: Default Service Routine

I.D.	Message Text
IKJ58600I	QUALIFIERS FOR DATA SET dsname ARE
IKJ58600I	xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
IKJ58600I	xxxxxxxx
IKJ58601A	ENTER QUALIFIER-
*IKJ58601A	DATA SET NAME WAS NOT FULLY QUALIFIED. ENTER DESIRED QUALIFIER FROM ABOVE LIST-
IKJ58602I	INVALID QUALIFIER xxxxxxxx
IKJ58603A	REENTER-
*IKJ58603A	ONLY QUALIFIERS LISTED ARE VALID QUALIFIERS FOR THIS DATA SET NAME. REENTER DESIRED QUALIFIER FROM ABOVE LIST-
*IKJ58603A	QUALIFIERS CONTAIN FROM 1 TO 8 ALPHANUMERIC CHARACTER. REENTER-
IKJ58604A	ENTER OLD OR NEW QUALIFIER-
*IKJ58604A	DATA SET NAME WAS NOT FULLY QUALIFIED. ENTER NEW QUALIFIER OR ONE FROM ABOVE LIST-
IKJ58605I	DATA SET NAME dsname NOT COMPLETE
IKJ58606A	ENTER QUALIFIER FOR dsname-
*IKJ58606A	DATA SET NAME WAS NOT FULLY QUALIFIED. ENTER NEW QUALIFIER-
IKJ58607I	DATA SET dsname IS ABOUT TO BE REUSED
IKJ58608A	ENTER CARRIER RETURN TO CONTINUE OR ATTENTION TO RESPECIFY COMMAND-
IKJ58609I	DATA SET NAME dsname CANNOT BE RESOLVED, SYSTEM ERROR+
*IKJ58609I	PUTLINE ERROR CODE xxxx
*IKJ58609I	PUTGET ERROR CODE xxxx
*IKJ58609I	LOCATE ERROR CODE xxxx
IKJ58610I	DATA SET NAME dsname NOT FULLY QUALIFIED

<u>Note</u>: Lower case letters represent inserted information.

Figure 42. Messages: Default Service Routine

5

Return Code (Dec.)	Meaning of Return Code	Possible Return Code by Entry Code			
		X'00'	X'04'	X'08'	X'0C'
0	Successful operation.	X	X	X	X
4	Unable to obtain qualifier from terminal user. (PUTLINE or PUTGET error).		X		X
8	With qualifiers added, data set length is greater than 44 bytes.	X	X	X	X
12	Permanent I/O error in the system catalog, catalog data set not available, or syntax error in data set name. (The LOCATE return code was X'04', X'14', or X'18'.)	X	X	X	X
16	Data set exits at some level of index other than the lowest index level specified. (The LOCATE return code was X'10').	X	X	X	X
20	One of the data set names was not found. (LOCATE Return Code of X'08'.)	X	X	X	X
24	Attention interruption occurred.	X	X	X	X
28	Invalid parameter: • Invalid entry code, • Data set length not halfword aligned, • Data set length greater than 44 bytes, or • Data set length of 0, except with entry code of X'00'.	X	X	X	X
32	Prompting is necessary to qualify data set name.			X	
36	No qualifiers found. (LOCATE return code X'0C'.)	X	X	X	X

Figure 43. Return Codes: Default Service Routine

Part 6: Catalog Information Routine

6

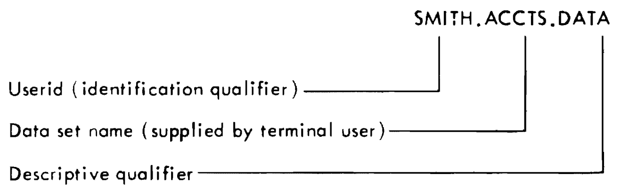
Section 1: Introduction

This description of the catalog information routine assumes that the reader has a knowledge of the information contained in the OS/VS Catalog Management Logic, SY35-0003.

The catalog information routine (IKJEHCIR) retrieves information from the system catalog. This information may include data set name, index name, control volume address, or volume ID.

A fully qualified data set name has three fields: a userid, a data set name, and a descriptive qualifier. For example:

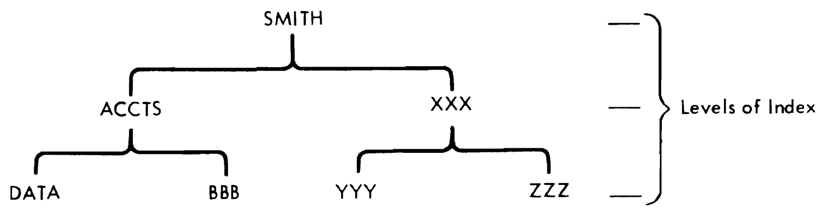
Insert B


SMITH.ACCTS.DATA
Userid (identification qualifier)
Data set name (supplied by terminal user)
Descriptive qualifier

For a more detailed description of data set naming conventions, refer to OS/VS2 TSO Command Language Reference, GC28-0646.

An index name is the name as found in one of the fields of the fully qualified data set name. The system catalog is logically divided into levels of indexes as follows:

Insert C



A control volume address is the location of any direct access volume which contains a portion of the system catalog.

The volume ID is the volume serial number (VOLSER) of an area within auxiliary storage, independently accessed and identified.

The routine that calls IKJEHCIR must supply the userid and the data set name, or it must supply the address of the information requested. The catalog information routine issues the LOCATE macro instruction to search the catalog and return an index block. An index block is a portion of the system catalog containing one or pointers to other index blocks or to data sets.

The catalog information routine then reads the index block, compresses and reformats the information that it contains, and returns the requested information to the caller. If additional information is available at this level of index, the routine informs the calling routine and gives it the address of the next index block. The calling routine may again call the catalog information routine to retrieve an additional portion of the index by specifying this address.

The catalog information routine resides in SYS1.LINKLIB or in SYS1.CMDLIB and executes in the user's foreground region with the protection key assigned to that region. An installation may choose to make the catalog information routine resident in the TSO link pack area (TSLPA) in the region assigned to the Time Sharing Control Task (TSCT). The catalog information routine requires about 800 bytes of main storage.

Section 2: Method of Operation

Method of Operation Diagram 30 shows how the catalog information routine (IKJEHCIR) obtains information from the system catalog.

The catalog information routine works this way:

- When the default routine, or any other TSO problem program, needs information from the system catalog, it calls the catalog Information Routine and passes, in register 1, the address of the catalog information routine parameter block (CIRPARM).

- The first byte of CIRPARM contains an option code, which defines the service requested (for example, X'01' returns the lowest level qualifier associated with the data set name).

- The catalog information routine sets up a parameter block and invokes the LOCATE macro instruction. The LOCATE macro issues an SVC 26 to search the system catalog. SVC 26 uses userid and data set name, or an address to search the system catalog. LOCATE returns the requested information, which may be data set names, volume address, or volume ID, in the work space provided by the catalog information routine.

- On return from the LOCATE routine (IGG0CLC1), the catalog information routine checks the validity of the returned information against the request, reformats the returned information, and returns to the caller.

ENTRY TO CATALOG INFORMATION ROUTINE

The catalog information routine receives control by a CALL or LINK macro instruction at entry point IKJEHCIR. At entry, register 1 points to the catalog information routine parameter list (CIRPARM). CIRPARM contains:

- An option code requesting a particular service, see Figure 44 catalog information routine option codes, for options and resulting functions.

- An address of the search argument. This search argument may be either:
  - A userid and a data set name, which are names of catalog index levels, or
  - A ttr, which is an address relative to the beginning of the system catalog.
- An address of volume identification of a control volume -- the volume containing a portion of the system catalog referred to by the relative address in the search argument.

- Address of work area; this area is supplied by the calling program (on a double word boundary).

- Address of save area; this area is supplied by the calling program.

The catalog information routine returns to the calling program using a RETURN macro instruction. All registers except 15 are restored. At exit, register 15 contains a return code. (See Figure 46, Catalog Information Routine Return Codes.)

Option Code	Option Requested	Response to Caller

X'01'	Data Set Name	Returns all the lowest level qualifiers contained within one index block.
		
		`07 dsname`
		`1  8`
		
		This nine-byte entry represents a lowest level data set name.
		
		`02 gdgname ttr data`
		`1  8       3   4`
		
		This 16-byte list entry represents a generation data group (gdg). A generation data group is the entire collection of chronologically related data sets which can be referred to by the same data set name. For further discussion on generation data groups, refer to IBM System/360 Operating System Data Management Services, GC26-3746. The data field contains four bytes describing the characteristics of the group. This data field contains the following:
		
		Byte 1-2 Not used by Catalog Information Routine
		
		Byte 3-4 Count of generations currently in the index.

X'02'	Index Name	**Returns one lower level qualifier.**
		
		`00 dsname ttr`
		`1  8      3`
		
		This 12-byte list entry contains a data set name qualifier. The ttr points to the beginning of the index containing this data set name.

X'04'	Volume List	**Returns a listing of volumes associated with the search argument.**
		
		`ct code volser1 seq }{ code volsern`
		`1  4    6       2       4    6`
		
		ct -- Number of volume serial numbers described.
		
		code -- Four-byte device code. (Device code designations are shown in IBM System/360 System Programmer's Guide, GC28-6550.)
		
		volser -- Volume serial number.
		
		seq -- Sequence number, for tape devices.

Figure 44. Option Codes: Catalog Information Routine (Part 1 of 2)

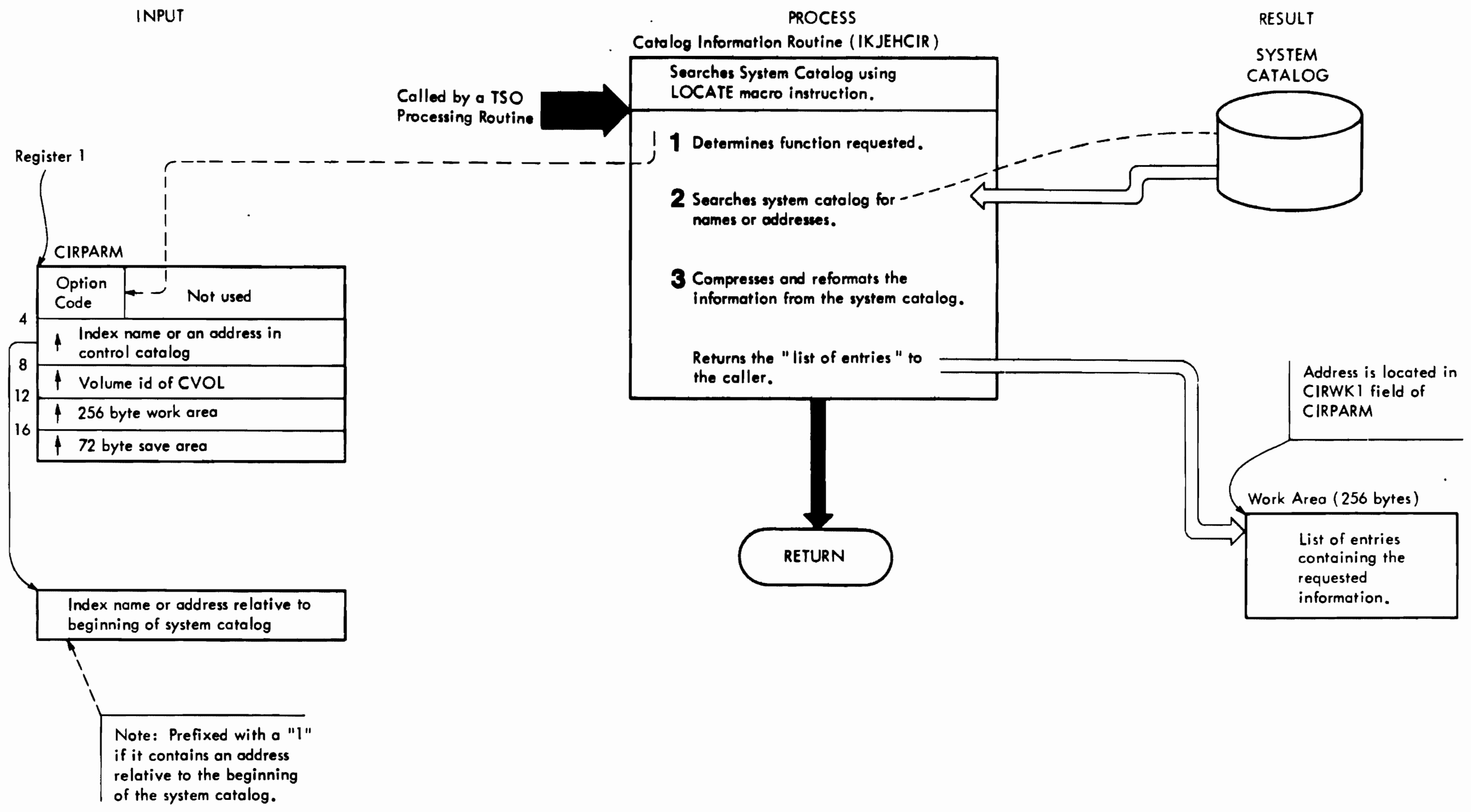
Option Code	Option Requested	Response to Caller
X'05'	Control volume address and data set name.	Returns all the lowest level qualifiers contained within one index block and the address of the volume containing the associated block. 08 | dsname | ct | code | volser1 | seq | code | volsern 1 | 8 | 1 | 4 | 6 | 2 | 4 | 6 ct -- Number of volume serial numbers described. code -- Four-byte device code. volser -- Volume serial number. seq -- Sequence number, for tape devices.
X'08'	Volume ID	Return Volume Identification. 04 | aliasname | ttr | truename 1 | 8 | 3 | 8 This 20-byte list entry represents an alias found in the system catalog. Aliases exist only in the volume index, which is the highest level index. The TTR points to the first block of the index. 05 | dsname | code | volser 1 | 8 | 4 | 6 This 19-byte list entry represents a catalog volume pointer found in the volume index. The code identifies the device type on which the control volume might be mounted. (This field will be zero, if the catalog was constructed prior to Release 17). The volser identifies the control volume (CVOL).
X'FF'		Termination list entry. FF | ttr | volser 1 | 3 | 6 This ten-byte list entry terminates every list regardless of options specified. If this list contains the end of an index, the ttr is zero; otherwise the ttr is the relative address of the next index block. The volser contains the volume identification of the volume containing the catalog being used.

Figure 44. Option Codes: Catalog Information Routine (Part 2 of 2)

6

Method of Operation Diagrams

6


INPUT
PROCESS
Catalog Information Routine (IKJEHCIR)
RESULT
SYSTEM CATALOG
Called by a TSO Processing Routine
Searches System Catalog using LOCATE macro instruction.
1 Determines function requested.
2 Searches system catalog for names or addresses.
3 Compresses and reformats the information from the system catalog.
Returns the "list of entries" to the caller.
RETURN
Register 1
CIRPARM
Option Code
Not used
4
Index name or an address in control catalog
8
Volume id of CVOL
12
256 byte work area
16
72 byte save area
Index name or address relative to beginning of system catalog
Note: Prefixed with a "1" if it contains an address relative to the beginning of the system catalog.
Address is located in CIRWK1 field of CIRPARM
Work Area (256 bytes)
List of entries containing the requested information.

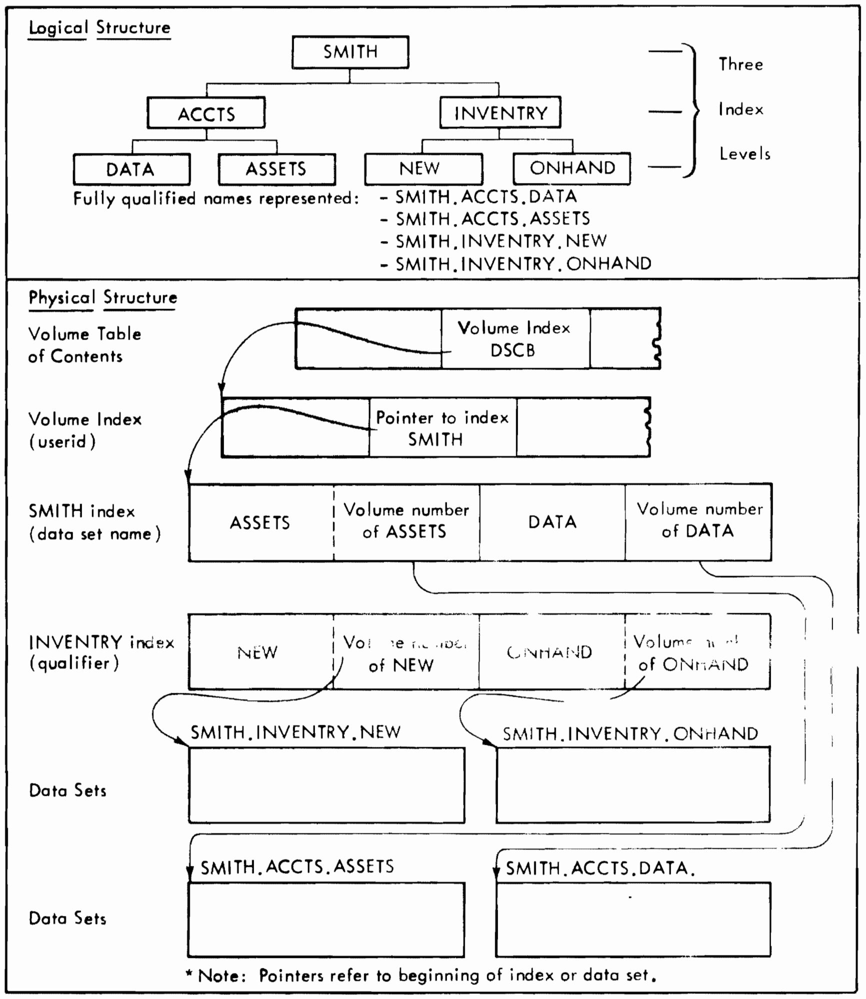
Method of Operation Diagram 29. Catalog Information Routine

Method of Operation Diagram 30. Catalog Information Routine (Part 1 of 2)

CROSS REFERENCE TABLE

Key Description	Routine	Label
1 The Option Codes are as follows: Option Byte — Response to Caller X'01' — Returns all the lowest level qualifiers contained within one index block. X'02' — Returns a lowest level index name. X'04' — Returns a listing of volumes. X'08' — Returns volume information.	Catalog Information Routine	IKJEHCIR
2 The Catalog Information Routine issues the LOCATE macro instruction, which in turn issues a SVC 26.		LOCATE
3 The index block from the system catalog, which is returned by LOCATE, is reformated and returned to the caller.		CODE 00

STRUCTURE OF SYSTEM CATALOG *



* Note: Pointers refer to beginning of index or data set.

Method of Operation Diagram 30. Catalog Information Routine (Part 2 of 2)

Section 3: Program Organization

This section describes the organization of the catalog information routine.

The catalog information routine consists of one load module, IKJEHCIR. As supplied with TSO, the catalog information routine resides in SYS1.LINKLIB or in SYS1.CMDLIB, and requires about 800 bytes of main storage.

Section 4: Directory

This table contains information to help you find the appropriate program description or assembly listing. It correlates information from three sources:

- The source code.
- The executable load modules.
- This manual.

Label	Common Name	Load Module Name	Assembly Module Name	Control Section Name	Description	Diagram
CHKTYPE	--	IKJEHCIR	IKJEHCIR	IKJEHCIR	Examine entry in catalog block.	30
CODE00	--	IKJEHCIR	IKJEHCIR	IKJEHCIR	Process LOCATE return code of zero.	30
DSENTRY	--	IKJEHCIR	IKJEHCIR	IKJEHCIR	Process data set entry.	30
IKJEHCIR	--	IKJEHCIR	IKJEHCIR	IKJEHCIR	Control section name and program entry point.	30
INDEXPTR	--	IKJEHCIR	IKJEHCIR	IKJEHCIR	Process link or index entry.	30
LOCATE	--	IKJEHCIR	IKJEHCIR	IKJEHCIR	Issue LOCATE macro instruction.	30
VOLPTR	--	IKJEHCIR	IKJEHCIR	IKJEHCIR	Process volume control block.	30

Section 5: Data Areas

This section describes the catalog information routine parameter list (CIRPARM) and the parameter list for LOCATE (CAM2). The following information is included:

- Size in bytes.
- The routines that created it.
- Displacements, size and contents.

CATALOG INFORMATION ROUTINE PARAMETER LIST (CIRPARM)

Size:	20 Bytes.
Constructed by:	The calling program.
Updated by:	None.
Used by:	IKJEHCIR.
Contents:	Addresses and control information for IKJEHCIR.

Operation Diagrams
30

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	CIROPT	1	Options used. See Figure 41.
1	1	--	3	Not used.
4	4	CIRSRCH	4	↑ data set name or relative address. (This field is prefixed with a "1", if it contains a relative address. It is prefixed with a "0" if it contains the address of a data set name.)
8	8	CIRCVOL	4	↑ volume ID of CVOL. (If not given, SYSRES is assumed.)
12	C	CIRWA	4	↑ 265 byte work area, aligned on a double word boundary.
16	10	CIRSAVE	4	↑ 72 byte save area.

CAMLST BLOCK (CAM2)

Size:	16 Bytes.
Constructed by:	IKJEHCIR.
Updated by:	IKJEHCIR.
Used by:	SVC 26.
Contents:	Addresses and control information for SVC 26 (Parameter list for input to SVC 26, which is issued by LOCATE macro instruction).

Operation Diagrams
30

Displacement Dec.	Hex.	Field Name	Size in Bytes	Contents
0	0	--	1	Option flag 1. X'C2' Search by Block X'C4' Search by Name
1	1	--	1	Option flag 2*.
2	2	--	1	Option flag 3*.
3	3	--	1	Not used by IKJEHCIR (0).
4	4	--	4	Pointer to data set name search argument.
8	8	--	4	Pointer to the control volume serial number.
12	C	--	4	Pointer to work area, aligned on a double word boundary.

*Set to all zeros by IKJEHCIR to indicate a request to locate an entry in the catalog.

6

Section 6: Diagnostic Aids

This section contains:

- Catalog Information Routine Register usage chart (Figure 45).
- Catalog Information Routine return codes (Figure 46).
- LOCATE macro instruction return codes (Figure 47).

Register	Name	Use
0	R0	Not used.
1	R1	Points to current entry in work area.
2	R2	Points to current entry in block.
3	R3	Work register.
4	R4	Contains parameter list address.
5	R5	Contains option bits.
6	R6	Work register.
7	R7	Points to LOCATE work area.
8	R8	Not used.
9	R9	Not used.
10	R10	Not used.
11	R11	Main base register.
12	R12	Not used.
13	R13	Save area address.
14	R14	Return register.
15	R15	Return code.

Figure 45. Register Usage: Catalog Information Routine

Return Code	Meaning
0	Successful completion of the request.
4	The LOCATE macro instruction has failed. The LOCATE return code will be stored in the first word of the user's parameter list. The catalog information Routine interprets and handles LOCATE return codes 00 and 12 -- LOCATE return codes are shown in Figure 45.
8	Volumes alone were requested (entry code X'04'), but neither a dsname or a ttr to a volume control block was given, and an index block was found instead. The index block that was found is in the work area.
12	Volumes were returned by LOCATE, indicating either a dsname (fully qualified) or a ttr was passed in the parameter list, but options other than volumes were requested. The list of the volumes returned by LOCATE is in the work area.

Figure 46. Return Codes: Catalog Information Routine

Return Code	Meaning
0	Successful completion of the request.
4	Either the required control volume was not mounted or the specified volume does not contain a catalog data set (SYSCATL).
8	The data set name qualifier was not found.
12	Success -- but more names are available.
16	A data set resides at a higher level of index than was requested. For example: data set A.B.C exists but A.B.C.D was requested.
20	A syntax error exists in the data set name.
24	A permanent I/O error was found when processing the catalog.
28	Relative track address supplied is out of the SYSCTLG data set extents.
32	Invalid work area pointer.

Figure 47. Return Codes: LOCATE (IGG0CLC1)

6

Glossary

The following definitions are for the use of these terms in this publication. If you do not find the term you are looking for, refer to the Index or to the IBM Data Processing Glossary, GC20-1699.

abnormal end of task (ABEND): Termination of a task prior to normal completion because of an error condition.

address-constant: A number, or a symbol representing a number, used in calculating storage addresses.

alias: An alternate name for a particular member of a partitioned data set.

allocate: To assign a resource for use in performing a specific task.

allocation of data sets: The process of defining a data set and defining auxiliary storage space. See also dynamic allocation.

alphameric characters: The characters A through Z, digits 0 through 9, and #, $, and @.

ATTACH: A macro instruction that causes the control program to create a new task and indicates the entry point in the program to be given control when the new task becomes active.

attention exit routine: A routine that receives control when an attention interruption is received by the system.

attention interruption: An interruption of instruction execution caused by a remote terminal user hitting the attention key. See also simulated attention.

attention key: A function key on remote terminals that causes an interruption of execution by the CPU.

Attention Scheduler: A part of the Region Control Task that gets control when the terminal user causes an attention interruption. The Attention Scheduler passes control to the appropriate attention exit routine.

attributes: See user attributes.

auxiliary storage: Data storage other than main storage (for example, storage on tape or direct access devices).

background: In TSO, the environment in which jobs submitted through the SUBMIT command or SYSIN are executed. One job step at a time is assigned to a region of main storage, and remains in storage to completion. Contrast with foreground.

background job: In TSO, a job entered through the SUBMIT command or SYSIN. Contrast with foreground job.

background reader: A system task started by the operator to process foreground-initiated background jobs. Output is identical to the normal reader/interpreter output.

break: See receive interruption.

broadcast data set: A system data set containing messages and notices from the system operator, adminstrators, and terminal users.

buffer: See main storage buffer, command buffer.

byte: The representation of a character; eight binary digits (bits) operated upon as a unit.

catalog:
1. A collection of data set indexes that are used by the control program to locate a volume containing a specific data set.
2. To include the volume identification of a data set in the catalog.

cataloged data set: A data set whose name and location are stored in the system catalog.

Catalog Information Routine: A routine that retrieves information from the system catalog for any TSO command processor.

cataloged procedure: A set of job control statements that has been placed in a data set named SYS1.PROCLIB and that can be retrieved by naming it in a job control language (JCL) execute (EXEC) statement.

character: A letter, digit, or other symbol that is used as part of the organization, control, or representation of data. For example, A,B,C,0,1,2, ,+,*,etc.

character-deletion character: A character within a line of terminal input specifying that it and the immediately preceding character are to be removed from the line by a scanning and editing routine.

character string: Any sequence of characters.

command: Under TSO, a request from a remote terminal for the execution of a particular program, called a command processor. The command processor is in a commmand library under the command name. Any subsequent commands processed directly by that command processor are called subcommands.

command buffer: An area of main storage that is assumed to contain a TSO command submitted by the terminal user.

command language: The set of commands, subcommands, and operands recognized by TSO.

command library: A partitioned data set consisting of command processor programs. A user command library can be concatenated to the system command library.

command name: the first term in a command, usually followed by operands.

command procedure: A data set or a member of a partitioned data set containing TSO commands, to be performed sequentially by the EXEC command.

command processor: A problem program executed as the result of entering a command at the terminal. Any problem program can be defined as a command processor by assigning a command name to the program and including the program in a command library.

Command Scan: A TSO service routine that searches the Command Buffer for question mark, command name, or null line. If syntax checking is requested, Command Scan checks the command name to be sure that it starts with an alphabetic character and contains no more than 8 alphanumeric characters.

control block: A storage area that contains a particular type of information used by the operating system to control the use of system resources.

control program: All the routines in the operating system that contribute to the management of resources, programs, and data and implement the data.

control section (CSECT): The smallest separately relocatable unit of a program; that group of coding specified by the programmer to be an entity, all elements of which are to be loaded into contiguous main storage addresses for execution.

control volume: A volume that contains one or more indexes of the catalog.

CP: See "command processor."

DAIR: See Dynamic Allocation Interface Routine.

data definition name (ddname): A name appearing in the data control block assigned to a program; the name is specified in the name field of a data definition (DD) statement.

data management: A general term that collectively describes those functions of the control program that provide access to data sets, enforce data set conventions, and regulate the use of input/output devices.

data set allocation, dynamic: See dynamic allocation.

data set catalog: See catalog.

Data Set Extension (DSE): A control block that contains information about a terminal user's data sets, including the relationship between DDNAMEs and DSNAMEs.

data set organization: The arrangement by data management of information in a data set. For example, sequential orgainzation or partitioned organization.

data set name: The term or phrase used to identify a data set. See also qualified name.

DDNAME: See data definition name.

default: See default value.

Default: A TSO service routine that constructs a fully qualified data set name when provided a partially qualified data set name by the calling routine.

Default Parameter Block (DFPB): An area of main storage used to contain codes and addresses required when calling the Default service routine.

default value: the choice among exclusive alternatives made by the system when no explicit choice is specified by the user.

delimiter: A character used to group and/or separate fields in a line of input.

device type: The general name for a device, specified at system generation. For example, 2311 or 2400.

DSE: See "Data Set Extension."

DSNAME: See "data set name."

dynamic allocation: The process of defining a data set and allocating auxiliary storage space for it during job step execution rather than before job step execution.

Dynamic Allocation Interface Routine (DAIR): A TSO service routine that performs various data management functions.

ECT: See "Environment Control Block."

Environment Control Block (ECT): A control block that contains information about the user's environment in the foreground region.

foreground: In TSO, the environment in which programs are swapped in and out of main storage to allow CPU time to be shared among terminal users. All command processor programs execute in the foreground. Contrast with background.

foreground job: In TSO, any job executing in a foreground region, such as a command processor or a terminal user's program. Contrast with background job.

GETLINE: A TSO service routine used by command processors to obtain input.

group name: The name for a particular collection of devices, specified at the time the system is generated. For example, SYSDA or TAPE.

HELP command: A TSO command that provides the terminal user with reference information on command and subcommand syntax, function, and usage.

IKJDAIR: An alias load module name for the Dynamic Allocation Interface Routine.

IKJEHCIR: The load module name for the Catalog Information Routine.

IKJDEFLT: An alias load module name for the Default service routine.

IKJEHDEF: The load module name for the Default service routine.

IKJPARS: The load module name for the Parse service routine.

IKJPUTL: The entry name for the PUTLINE service routine.

IKJPTGT: The load module name for the STACK, GETLINE, PUTLINE and PUTGET service routines. The entry name for the PUTGET service routine.

IKJSCAN: The load module name for the Command Scan service routine.

informational message: Output on a terminal that tells the terminal user the status of the system and of his terminal session.

index name: In TSO, one of the fields of a qualified data set name.

Input Stack: A push-down list of sources of input for GETLINE and PUTGET. Possible sources are the terminal or an in-storage list.

in-storage list: A chain of input lines in main storage, such as commands in an EXEC procedure, that are used in place of terminal input.

interruption: A transfer of CPU control to the control program of the Operating System. The transfer is initiated automatically by the computing system or by a problem state program through the execution of a supervisor call (SVC) instruction. The transfer of control occurs in such a way that control can later be restored to the interrupted program, or, in systems that perform more than one task at a time, to a different program.

I/O Service Routine List (IOSRL): A control block which contains the address of the first element (bottom element) and the most recently added element (top element) of the input stack. The GETLINE and PUTGET service routines can refer to the IOSRL, but only the STACK service routine can update it.

IOSRL: See I/O Service Routine List.

job control statement: Any one of the control statements in the input job stream that identifies a job or defines its requirements.

job definition: A series of job control statements that define a job.

job management: A major function of the operating system involving the reading and interpreting of job definitions, the scheduling of jobs, the initiation and termination of jobs and job steps, and the recording of job output data.

keyword parameter: A command operand that consists of a specific character string such as FORTLIB or PRINT. See also positional parameter.

line deletion character: A terminal character that specifies that it and all preceding characters are to be deleted from a line of terminal input.

line: A line of one or more characters typed at a terminal. See also logical line, physical line.

load: To place a program into main storage so that it can be executed.

load module: The output of the linkage editor; a program in a form suitable for loading into main storage for execution.

logical line: One or more lines typed at a terminal and treated as a unit. A logical line may consist of one or more physical lines where the symbol "-" indicates continuation. See also physical line.

logical record: A record that is defined in terms of the information it contains rather than by its physical qualities.

LOGON/LOGOFF Scheduler: The TSO control program routine that initiates and terminates a terminal session.

main storage buffer: An area of main storage that is temporarily reserved for use in performing an input/output operation.

mode message: A message that requests the terminal user to enter a line of input.

multi-level message: A chain of informational messages. The first message is the most general; the last message, the most detailed.

multi-line data: A chain of data lines. PUTLINE sends one line after another to the terminal until end-of-chain is reached.

national characters: The characters #, $, and @.

OLD: See Output Line Descriptor.

operand: In the TSO command language, information entered with a command name to define the data on which a command processor operates and to control the execution of the command processor. Some operands are positional, identified by their sequence in the command input line. Other operands are identified by keywords.

output buffer: An area of main storage used to store a data block before it is transferred to an output device.

output class: Any one of up to 36 different output data classes, defined at an installation, to which output data can be assigned.

output device: A machine (such as a printer, terminal, or tape drive) that will accept the output from the system.

Output Line Descriptor (OLD): An area of main storage used to describe information to be sent to the terminal by the PUTLINE and PUTGET service routines.

output writer: The part of the job scheduler that controls the writing of job output data.

Parameter Control Entry (PCE): An entry in the Parameter Control List (PCL). In general, each PCE describes an acceptable TSO command parameter or marks the beginning or end of a field. Each PCE is created by a Parse macro instruction as shown in Table 6.

Parameter Control List (PCL): A data area that contains control information for the Parse service routine. Each element in the list is called a Parameter Control Entry (PCE).

Parameter Descriptor Entry (PDE): An entry in the Parameter Descriptor List (PDL). In general, each PDE describes a TSO command parameter entered by the terminal user or supplied by default.

Parameter Descriptor List (PDL): A data area that describes the TSO command parameters entered by a terminal user or supplied by default. Created by the Parse service routine. Each element in the list is called a Parameter Descriptor Entry (PDE).

PCE: See Parameter Control Entry.

PCL: See Parameter Control List.

PDE: See Parameter Descriptor Entry.

PDL: See Parameter Descriptor List.

Parse: A TSO service routine that searches the Command Buffer for TSO command parameters, checks them for correct syntax, and optionally presents them to a user-supplied validity check exit routine.

partitioned data set: A data set that is stored in direct access storage and can be cataloged like any other data set. It is divided into independent partitions called members, each of which normally contains a program or part of a program.

password: A one- to eight-character symbol assigned to a user that he can be required to supply at LOGON. The password is confidential, as opposed to the user identification. Users can also assign passwords to data sets.

physical line: A line typed at a terminal. See also logical line.

physical record: A record that is defined in terms of physical qualities rather than by the information it contains. (See record.)

positional parameter: A command operand which must appear in a certain order, in relation to other operands. Contrast with keyword parameter.

POST/WAIT: A POST macro instruction followed by a WAIT macro instruction. The purpose is to cause a task switch from the task issuing the POST/WAIT to the task whose ECB is posted. The task switch does not occur until after the WAIT is issued.

procedure list: An in-storage list that contains TSO commands.

problem program: A program which executes in the problem state, is restricted from executing privileged instructions, and executes from main storage with a nonzero protection key.

procedure: See cataloged procedure.

profile: See user profile.

program status word: A doubleword in main storage that controls the order in which instructions are executed.

prompting: A system function that helps a terminal user by requesting him to supply operands necessary to continue processing.

prompting message: A message that requests the terminal to enter another line of input, either a TSO command parameter or data.

protection key: An indicator associated with a task which appears in the program status word whenever the task is in control, and which must match the storage keys of all storage blocks the task is to use.

PSCB: See Protected Step Control Block.

PSW (program status word): A doubleword in main storage that controls the order in which instructions are executed.

PUTGET: A TSO service routine that sends a message to the terminal and obtains a line of input from the current source of input.

PUTLINE: A TSO service routine that sends output to the terminal. PUTLINE selectively puts out messages according to whether or not a user has suppressed prompting or is executing a command procedure.

qualified name: A data set name that is composed of two or more names separated by periods. (For example, MOORE.SALES.JUNE.)

qualifier: In TSO, the lowest level identifier of a qualified name.

RCT: See Region Control Task.

reader/interpreter: A job scheduler function that services an input job stream.

receive interruption: The interruption of a transmission to a terminal by a higher priority transmission from the terminal. Synonymous with break.

record: One or more data fields that represent an organized body of related data, such as all of the basic accounting information concerning a single sales transaction. (See also logical record and physical record.)

reenterable: The attribute or characteristic of a load module allows the same copy of the module in main storage to be used by several tasks concurrently.

region: An area of main storage allocated to a job step and assigned a unique storage protection key. Time sharing jobs share regions. Each job occupies a region briefly, then is swapped out to auxiliary storage and another job is swapped into the vacated main storage area for execution. The jobs are swapped in and out until they are completed.

Region Control Task (RCT): The TSO control program routine handling quiesce/restore and LOGON/LOGOFF. There is one RCT for each active foreground region.

return code: A number placed in a designated register at the completion of a program.

self-defining delimiter: Any character appearing in the first position of certain character strings in the TSO command language. A repetition of the character within the string is interpreted as a delimiter.

separator: A delimiter used to separate fields in an input line to the system.

simulated attention: A function that allows terminals without attention keys to interrupt processing. The terminal is queried (for a specified character string meaning ""attention'') after a specified number of seconds of uninterrupted execution or after a specified number of lines of consecutive output.

STACK: A TSO service routine that manipulates the Input Stack.

STAE (Specify Task Asynchronous Exit): A macro instruction specifying a routine to receive control in the event of the issuing task's abnormal termination (ABEND).

STAI (Subtask ABEND Intercept): A keyword of the ATTACH macro instruction specifying a routine to receive control after the abnormal termination of a subtask.

STATUS: A system macro instruction and its associated SVC routine that makes one or more tasks dispatchable or non-dispatchable.

STATUS START: The form of the STATUS macro instruction that makes one or more tasks dispatchable.

STATUS STOP: The form of the STATUS macro that makes a task non-dispatchable.

storage list: In TSO, an in-storage list that contains data. Contrast with procedure list.

subcommand: In TSO, an explicit request for a particular operation to be performed within the scope of a command processor.

SYS1.CMDLIB: The system command library. A partitioned data set that contains, among other things, the TSO command processors. A user data set may be concatenated to SYS1.CMDLIB.

SYS1.LINKLIB: The system linkage library. A partitioned data set that contains often-used routines. The contents of the linkage library are placed in main storage during initial program loading (IPL).

SYS1.PROCLIB: A system data set containing cataloged procedures.

task: A unit of work for the central processing unit defined by the control program.

TCAM: See Telecommunications Access Method.

Task Control Block (TCB): A system control block that contains task-related information.

TCB: See Task Control Block.

Telecommunications Access Method (TCAM): A generalized terminal I/O support package, providing application program independence of terminal characteristics.

terminal: A device resembling a typewriter that is used to communicate with the system.

terminal job: A foreground job, a session from LOGON to LOGOFF. Also used to refer to the main storage region assigned to a user and associated system control blocks.

Terminal Job Identification (TJID): A two-byte identification assigned to each terminal job.

Terminal Monitor Program (TMP): A program that accepts and interprets commands from the terminal, and causes the appropriate command processors to be scheduled and executed.

terminal user: See user.

TGET: An I/O macro instruction used by TSO problem programs to obtain a line of input from the terminal. Used by the GETLINE and PUTGET service routines.

time sharing: The concurrent sharing of the hardware and information resources of a data processing installation among one or more users who may be located at remote terminals.

Time Sharing Control Task (TSC): A TSO system task that handles system initialization, allocation of time-shared regions, the swapping of user programs into and out of main storage, and general control if the time-sharing operation.

TJID: See Terminal Job Identification.

TMP: See Terminal Monitor Program.

TPUT: An I/O macro instruction used by TSO problem programs to send a line of output to the terminal. Used by the PUTLINE and PUTGET service routines.

TSC: See Time Sharing Control Task.

ttr: A pointer in a partitioned data set directory to the first block of a member on a direct access device. The "tt" represents the relative track from the beginning of the data set. The "r" represents the relative block number on that track.

unit address: The symbolic location of an input/output device.

UPT: See User Profile Table.

user: In TSO, anyone with an entry in the User Attribute Data Set; anyone eligible t log on.

user attributes: A set of parameters in the User Attribute Data Set (UADS). The parameters describe the user to the system; for example, whether he is authorized to use the ACCOUNT command, and what size main storage region he is to be assigned.

User Attribute Data Set (UADS): A partitioned data set with a member for each authorized system user. Each member contains the appropriate passwords, user identifications, account numbers, logon procedure names, and user characteristics defining the user profile.

USERID: See user identification.

user identification (USERID): A one- to seven-character symbol identifying each system user.

user profile: The set of characteristics that describe the user to the system. See also User Profile Table.

User Profile Table: A table of user attributes kept for each active user, built by the Logon/Logoff Scheduler from information in the LOGON command, the UADS, and the user logon procedure.

VTOC: See Volume Table of Contents.

volume: A area of a recording medium that is serviced by a single read/write mechanism whose operation is entirely independent of any other read/write mechanism.

Volume Table of Contents (VTOC): A table of information in a direct access volume that defines the sets of data and unassigned space in the volume and indicates where they are located.

Index

The <u>OS/VS Master Index of Logic</u>, GY28-0603 consolidates the indexes from all the program logic manuals. It may therefore provide reference to additional information about any of the topics listed below through other publications.

SY28-0650-0

OS/VS2 TSO TMP and Service Routines Logic   Printed in U.S.A.   SY28-0650-0

IBM®

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

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

OS/VS2 TSO TMP and Service Routines Logic

SY28-0650-0

READER'S COMMENT FORM

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

Possible topics for comment are:

Clarity Accuracy Completeness Organization Index Figures Examples Legibility

Cut or Fold Along Line

What is your occupation? ____

Number of latest Technical Newsletter (if any) concerning this publication: ______________

Please indicate in the space below if you wish a reply.

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

SY28-0650-0

Your comments, please . . .

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

Cut or Fold Along Line

Fold Fold

First Class
Permit 81
Poughkeepsie
New York

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

Postage will be paid by:

International Business Machines Corporation
Department D58, Building 706-2
PO Box 390
Poughkeepsie, New York 12602

Fold Fold

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

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

OS/VS2 TSO TMP and Service Routines Logic

SY28-0650-0

READER'S COMMENT FORM

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

Possible topics for comment are:

Clarity Accuracy Completeness Organization Index Figures Examples Legibility

Cut or Fold Along Line

What is your occupation? ____

Number of latest Technical Newsletter (if any) concerning this publication: ______________

Please indicate in the space below if you wish a reply.

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

SY28-0650-0

Your comments, please . . .

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

Cut or Fold Along Line

Fold Fold

First Class
Permit 81
Poughkeepsie
New York

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

Postage will be paid by:

International Business Machines Corporation
Department D58, Building 706-2
PO Box 390
Poughkeepsie, New York 12602

Fold Fold

IBM

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

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