SY28-0652-4
File No. S370-39

Systems

OS/VS2 TSO Command Processor Logic Volume IV

(includes LOGON Scheduling)

VS2 Release 3.8

IBM

Fifth Edition (December, 1984)

This is a major revision of and obsoletes SY28-0652-3. See the Summary of Amendments following the Contents for a summary of the changes made to this manual. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

This edition with Supplement SD23-0299 applies to Version 2 of MVS/System Product 5740-XC6 or 5665-291 and Data Facility Product 5665-284 until otherwise indicated in new editions or Technical Newsletters. Changes are periodically made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370 Bibliography*, GC20-0001, for the editions that are applicable and current.

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

Publications are not stocked at the address given below. Requests for 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 has been provided at the back of this publication. If the form has been removed, address comments to IBM Corporation, Information Development, Department D58, Building 921, P.O. Box 390, Poughkeepsie, New York 12602. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Preface

This publication describes the programs that handle the following TSO commands:

ALLOCATE	LISTALC	RENAME
ATTRIB	LISTBC	RUN
CALL	LISTDS	SEND
CANCEL/STATUS	OPERATOR	SUBMIT
EXEC	OUTPUT	TERMINAL
FREE	PROFILE	TIME
HELP	PROTECT	WHEN/END
LINK/LOADGO		

In addition, LOGON Scheduling information is included.

DELETE and LISTCAT information is in *Access Method Services Logic.*

This publication describes program internal logic and organization. It is designed to help the programmer follow the internal operation of a program and determine the location of a program malfunction. The book provides pointers for specific functions; the programmer can use these pointers to access program listing information without having to scan the listings for the data he wants.

The commands are described through the use of Method of Operation Diagrams, a Directory, and a Data Area Usage chart.

The Directory contains a module cross reference for all of the commands described in this book. It cross references load module, object module, entry point, alias, and command name.

The Data Area Usage chart is organized by the data area acronym. The macro name and common name are also listed. Under each data name is a list of modules, by command processor, that alter or create the data area.

Data area descriptions can be found in *Data Areas.* The diagnostic aid description of the TSO terminal messages can be found in *Message Library: TSO Terminal Messages.*

Note that the titles for the TSO library for MVS/XA begin with the "MVS/ Extended Architecture" system prefix.

Associated Publications

MVS/Extended Architecture TSO Terminal Monitor Program and Service Routines Logic
(OS/VS2 TSO Terminal Monitor Program and Service Routines Logic, SY28-0650-3, as amended by Supplement LD23-0262)
MVS/Extended Architecture TSO Command Processor Logic Volume I: ACCOUNT
(OS/VS2 TSO Command Processor Logic Volume I: ACCOUNT, SY28-0651-2, as amended by Supplement LD23-0270)
MVS/Extended Architecture TSO Command Processor Logic Volume II: EDIT
(OS/VS2 TSO Command Processor Logic Volume II: EDIT, SY33-8548-3, as amended by Supplement LD23-0271)
Data Areas
(For *MVS/SP - JES2 Version 2*, LYB8-1191)
(For *MVS/SP - JES3 Version 2*, LYB8-1195)
Macro Usage Table
(For *MVS/SP - JES2 Version 2*, LYB8-1193)
(For *MVS/SP - JES3 Version 2*, LYB8-1197)
Symbol Usage Table
(For *MVS/SP - JES2 Version 2*, LYB8-1192)
(For *MVS/SP - JES3 Version 2*, LYB8-1196)
TSO Terminal Messages, GC28-1310
MVS/Extended Architecture Access Method Services Logic, LY26-3889

Referenced Products

1. All references to MVS System Product indicate either OS/VS2 MVS/System Product-JES3 (5740-XYN) or OS/VS2 MVS/System Product-JES2 (5740-XYS)

Contents

Illustrations

Figures

Diagrams

Diagrams (continued)

Summary of Amendments for SY28-0652-4 as Updated March 1, 1985 by Supplement SD23-0299

This supplement was reissued because of the major revision to base SY28-0652. The parallel TMP structure does not apply to the MVS/XA environment without TSO/E.

Summary of Amendments for SY28-0652-2 as Updated January 14, 1983 by Supplement LD23-0273-0

This supplement supports MVS System Product Version 2. A note has been added to LINK and LOADGO commands to remind readers of the AMODE and RMODE operands available under MVS/Extended Architecture.

Summary of Amendments for SY28-0652-3 as Updated by SN28-4928 OS/VS2 Release 3.8

This technical newsletter incorporates information for LOGON Scheduling.

Introduction

This section contains information on processing that is common to all the TSO command processors.

Terminal Monitor Program

The Terminal Monitor Program (TMP) handles the interfaces between a terminal user and a command processor. The TMP runs under the control program as a subtask of (is ATTACHed by) the TSO LOGON/LOGOFF Scheduler (via the Job Scheduling Subroutine).

Before the TMP in turn attaches its own subtasks (i.e., command processors), it:

- Constructs and initializes the data areas it requires.
- Loads the TIME command processor.
- Sets up ESTAE and ESTAI exits.
- Sets up attention interruption exits.
- Initializes the input stack with a terminal element.
- Issues the EXTRACT macro instruction to obtain pointers to both the STOP/MODIFY ECB and to the Protected Step Control Block (PSCB) that is built by the LOGON/LOGOFF scheduler.
- Informs the terminal it is 'ready' for a command.

When a command processor completes its processing, control is returned to the TMP. For more information on the TMP, please refer to *TSO Terminal Monitor Program and Service Routines Logic.*

Service Routines

There are a number of service routines used selectively by the different command processor packages. These service routines, which are also used by the TMP (unless otherwise noted), include:

- GETLINE, which obtains a line of input from an area defined as its source of input. Normally, this area contains input from the terminal.
- PUTLINE, which sends a line of output to the terminal.
- PUTGET, which sends a line of output to the terminal and waits for a line of input as a response.
- STACK, which establishes the source of input as a terminal; or (if not from a terminal) which places lines of input into areas from which GETLINE or PUTGET can obtain data.
- Command Scan, which checks the syntax of designated data to see if it is syntactically valid.
- Parse (IKJPARS) (not used by the TMP), which checks the syntax of parameters of TSO commands. In certain cases, Parse is directed to take exits to validity checking routines (provided by the processors). The validity checking routines are designed to dynamically assist the parse operation in providing valid input to the command or subcommand processor.
- Dynamic Allocation Interface Routine (IKJDAIR) (not used by the TMP), which provides information to the control program dynamic allocation routines. In turn, these routines allocate, free, and concatenate data sets that relate to a TSO session.

These service routines are documented in full in *TSO Terminal Monitor Program and Service Routines Logic.*

Attention Interruptions

When an attention interrupt has been entered at a terminal, an attention interrupt exit routine will receive control. If a command processor is interrupted, control will pass to the command processor's attention exit routine, if one exists. If not, then control will pass to the TMP's attention exit routine.

TMP Attention Exit Routine

The TMP issues the STAX macro during initialization to place an entry in a control program queue called the Task Attention Interrupt Exit queue. When the attention key is struck during subsequent processing, the control program attention interruption handling routines check the queue, put out the mode message, and pass control to the Attention Exit routine at the address provided through the STAX macro (after obtaining input from the terminal).

The Attention Exit routine issues a PUTGET macro instruction to obtain the input following the attention. Action is taken according to the type of input found, as follows:

New command found
: all previous entries are deleted from the input stack. Control then returns to the TMP where the old command processor is detached and the new one attached.

Null line
: control returns immediately to the task that was operating when the attention key was struck. No ECB is posted. No stack entries are deleted.

?
: a PUTLINE exit is taken to put out second-level messages, if any. (If none, a NO INFORMATION AVAILABLE message is issued.) Then, the TMP Attention Exit routine looks for a new command or a null line as input. Then processing is performed as for the applicable input type above.

Time command
: the TIME command processor receives control. Upon completion, TIME returns control to the TMP Attention Exit routine, which then looks for either a new command, or a null line, as input.

ABEND Processing

When the TMP issues the ATTACH macro to activate a command processor as a subtask, the ESTAI operand is included as part of the ATTACH macro. The ESTAI operand specifies the address of the TMP's ESTAE Exit routine. The main purpose of the ESTAI Exit routine -- in the command processor environment -- is to intercept an ABEND and thereby retain processing control. When a command processor experiences an ABEND, the TMP's ESTAI Exit routine gets control to ensure the following:

- The user is notified that his command processor has experienced an ABEND.
- The READY message is issued.

Action is taken according to the type of input found (as a response to the READY message), as follows:

New command found (except TIME or TEST)
: the command processor that has experienced the ABEND is deleted via the DETACH macro, (thereby restricting the ABEND), and the new command processor is activated as a subtask.

Null line
: control is returned to the point of interruption to allow the ABEND to process (a dump will occur if a SYSABEND or a SYSUDUMP has been specified on a DD Card).

?
: the second level message containing the ABEND code is issued. The ESTAI Exit routine then looks for either a new command, or a null line, as input.

TIME command
the TIME command processor receives control. Upon completion, TIME returns control to ESTAI, which then looks for either a new command or a null line as input.

Error Termination Procedure

When a command processor terminates with an error condition, the input stack is flushed (via the STACK service routine) and the terminal input queue is cleared (with the TCLEARQ macro instruction).

Message Handling

Most TSO command processors have message CSECTs. The address of a particular message is provided (by the command processor) to the PUTLINE service routine – which writes the message to the terminal.

A message can be either single or multi-level. Either type may require that PUTLINE insert variables (such as names, userids, etc.) to complete the message.

LOGON Scheduling

When a terminal user logs on or logs off TSO, the LOGON scheduling routines are invoked to handle the request. The routines interface with started task control (STC) and the TMP.

Diagram 1. Hierarchy of M.O. Diagrams (Part 1 of 3)

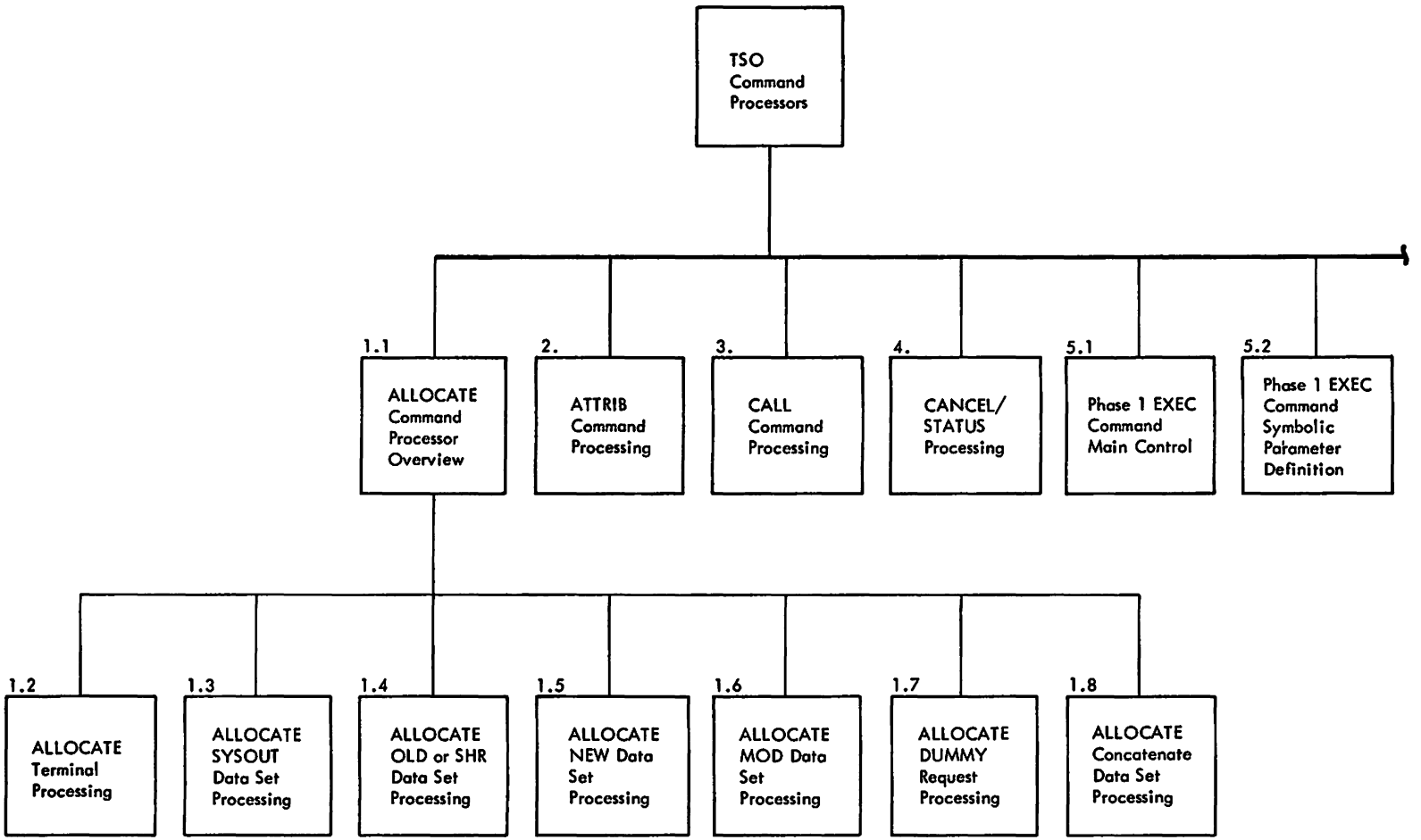


Diagram 1. Hierarchy of M.O. Diagrams (Part 2 of 3)

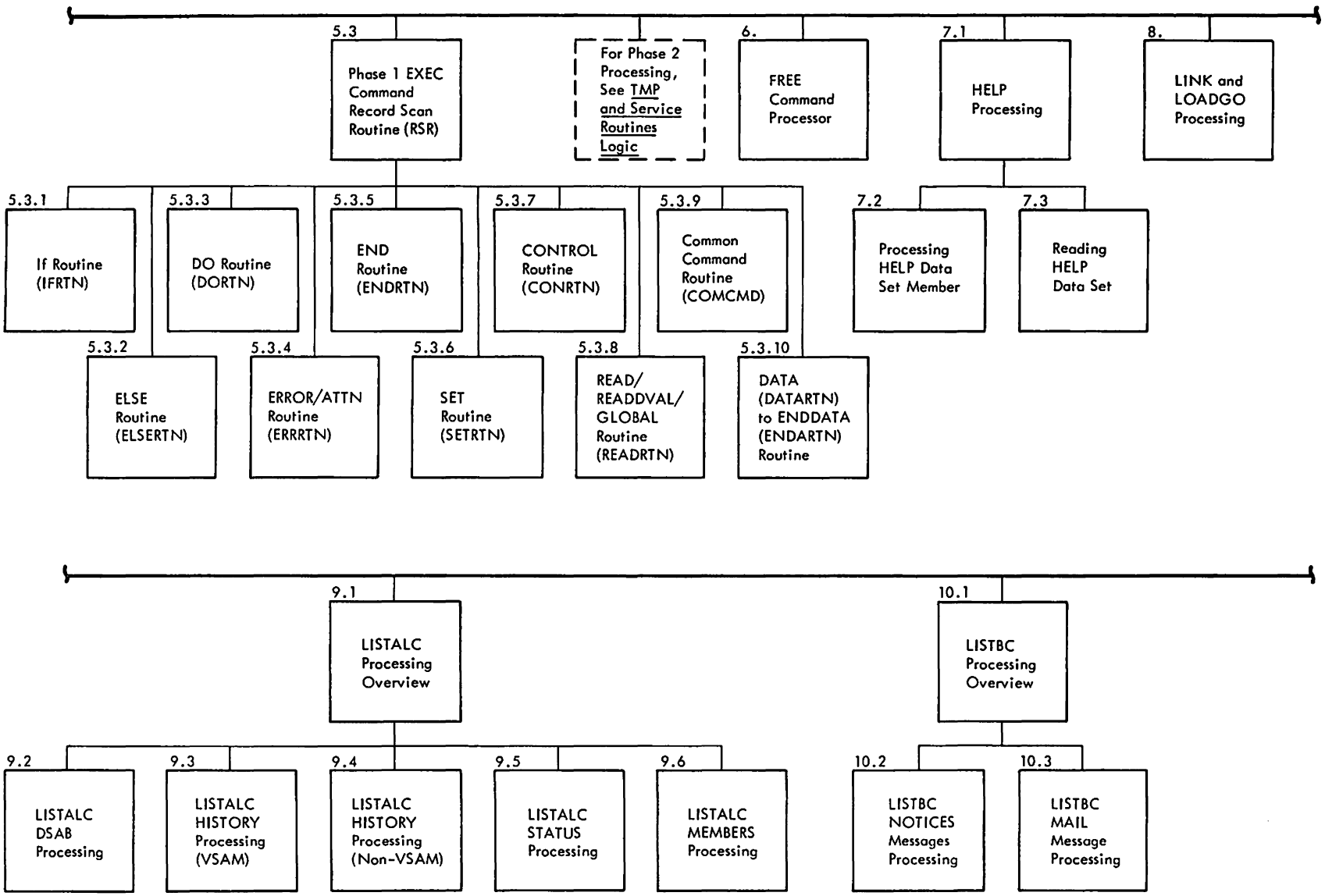


Diagram 1. Hierarchy of M.O. Diagrams (Part 3 of 3)

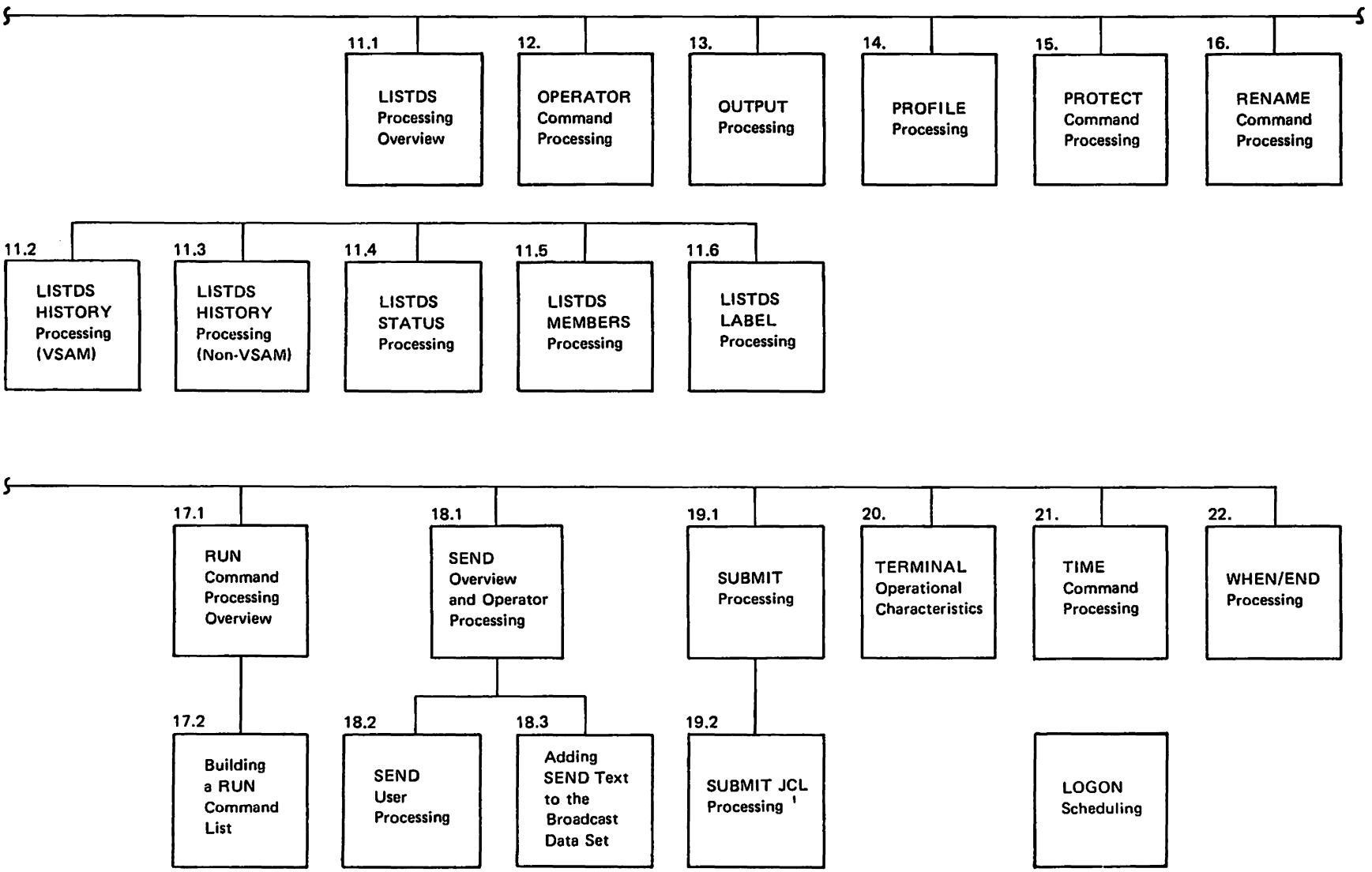


Diagram 1.1. ALLOCATE Command Processor Overview (Part 1 of 2)

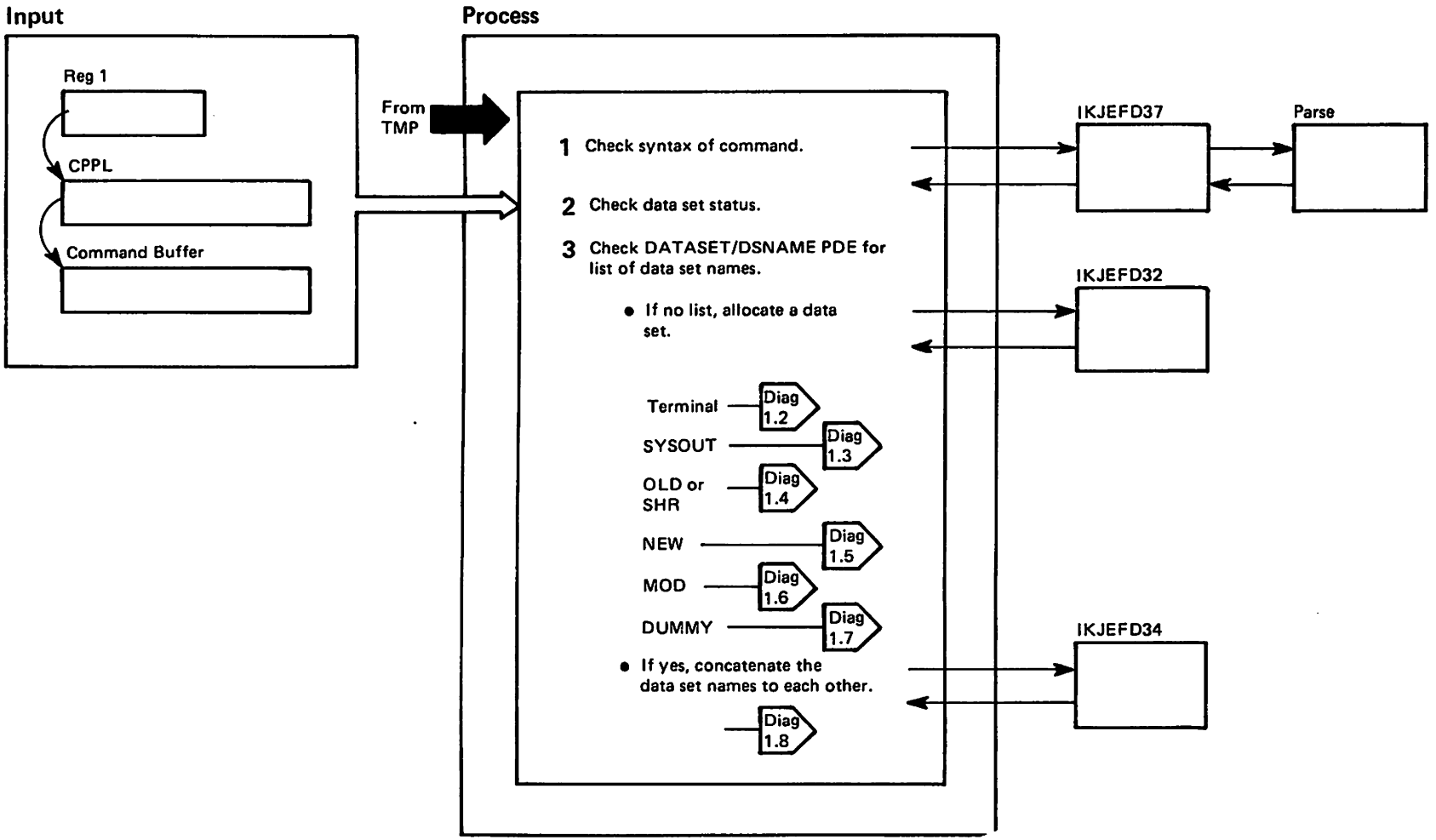


Diagram 1.1. ALLOCATE Command Processing Overview (Part 2 of 2)

1 Check the ALLOCATE command for correct syntax. Invoke module IKJEFD37 to build a Parameter Control List (PCL). The PCL is passed to Parse which syntax checks the command and returns a Parameter Description List (PDL) with an entry for each parameter.

2 Check for data set status parameters (OLD, SHR, MOD, NEW, or SYSOUT). If no status parameters were specified, set defaults or prompt user for status.

3 Check the DATASET/DSNAME PDE for a list of data set names.

- If there is no list of data set names, IKJEFD32 allocates a data set. (Types of data sets: Terminal, SYSOUT, OLD, SHR, NEW, MOD, DUMMY.)
- If there is a list of data set names, concatenate the data set names to each other.

Object Module: IKJEFD30

Diagram 1.2. ALLOCATE Terminal Processing (Part 1 of 2)

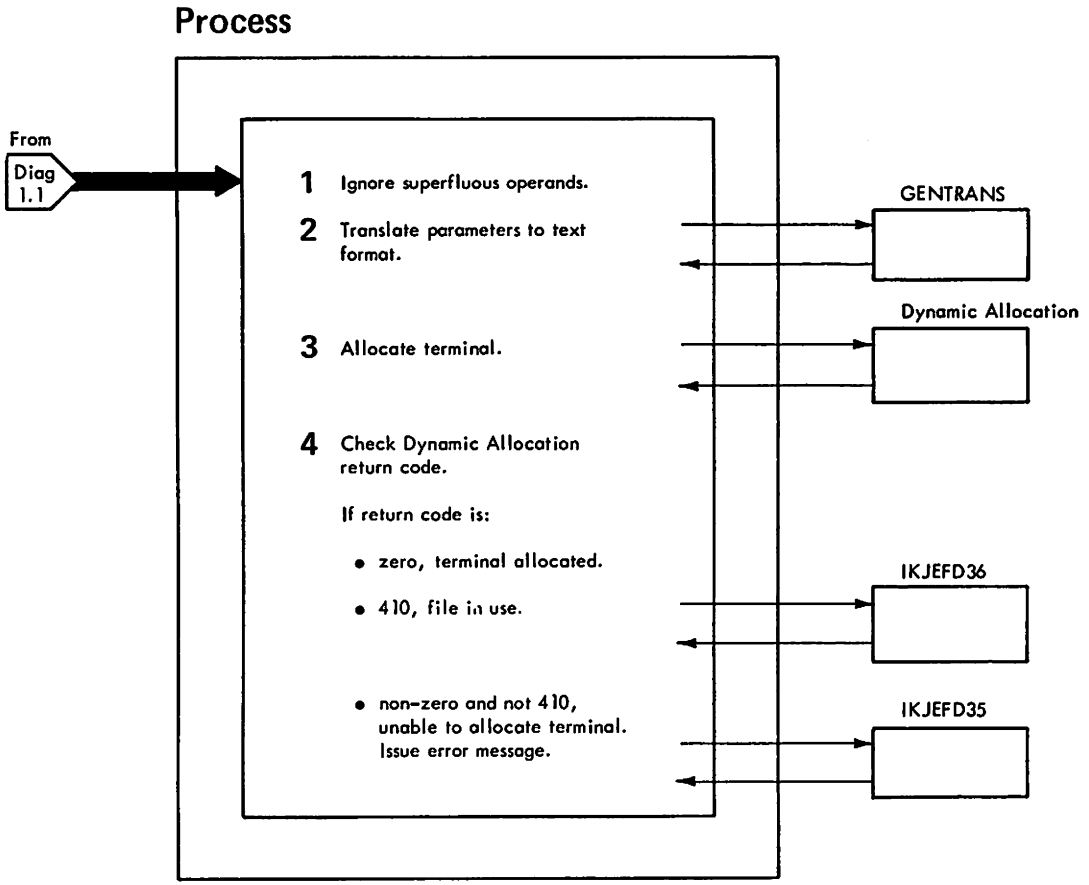


Diagram 1.2. ALLOCATE Terminal Processing (Part 2 of 2)

1 Ignore all operands except DATASET/DSNAME, USING, FILE/DDNAME, and BLOCK/BLKSIZE. (The BLOCK/BLKSIZE parameter is processed by object module IKJEFD33.)

2 Set up the text unit specifying terminal allocation. Use GENTRANS to translate the parameters to dynamic allocation text format. The pointer to the text units is returned from GENTRANS in the IKJZB831 parameter list.

3 Use Dynamic Allocation to allocate the terminal. Register 1 points to a pointer to the dynamic allocation request block.

4 Check the dynamic allocation return code.

- If the return code is zero, the terminal has been allocated.
- If the return code is 410, the file is in use. IKJEFD36 prompts the user to enter 'FREE', to free and reallocate the file, or 'END' to terminate the command.
- If the return code is non-zero and not 410, allocation of a terminal has failed. IKJEFD35 uses DAIR failure message routine IKJEFF18 to analyze the return code and send the appropriate error message to the user.

Control is returned to IKJEFD30.

Object Module: IKJEFD32

Diagram 1.3. ALLOCATE SYSOUT Data Set Processing (Part 1 of 2)

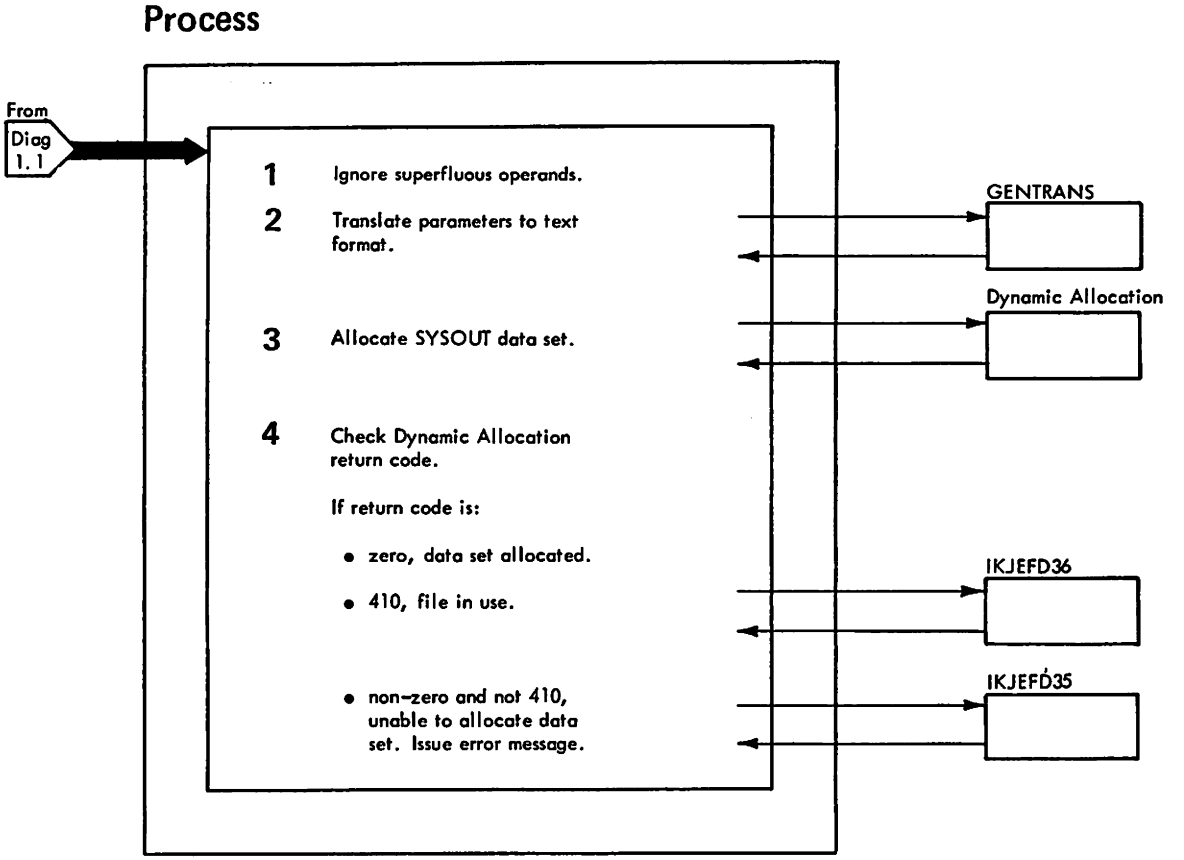


Diagram 1.3. ALLOCATE SYSOUT Data Set Processing (Part 2 of 2)

1 Ignore the following operands: DATASET/DSNAME, MSVGP, VOLUME, PRIVATE, LABEL, POSITION, MAXVOL, DIR, VSEQ, and disposition. (The BLOCK/BLKSIZE parameter is processed by object module IKJEFD33.)

2 Set up the text unit specifying SYSOUT data set allocation if necessary. Use GENTRANS to translate the parameters to dynamic allocation text format. The pointer to the text units is returned from GENTRANS in the IKJZB831 parameter list.

3 Use Dynamic Allocation to allocate the SYSOUT data set. Register 1 points to a pointer to the dynamic allocation request block.

4 Check the dynamic allocation return code.

- If the return code is zero, the data set has been allocated.
- If the return code is 410, the file is in use. IKJEFD36 prompts the user to enter 'FREE', to free and reallocate the file, or 'END' to terminate the command.
- If the return code is non-zero and not 410, allocation of the SYSOUT data set has failed. IKJEFD35 uses DAIR failure message routine IKJEFF18 to analyze the return code and send the appropriate error message to the user.

Control is returned to IKJEFD30.

Object Module: IKJEFD32

Diagram 1.4. ALLOCATE OLD or SHR Data Set Processing (Part 1 of 2)

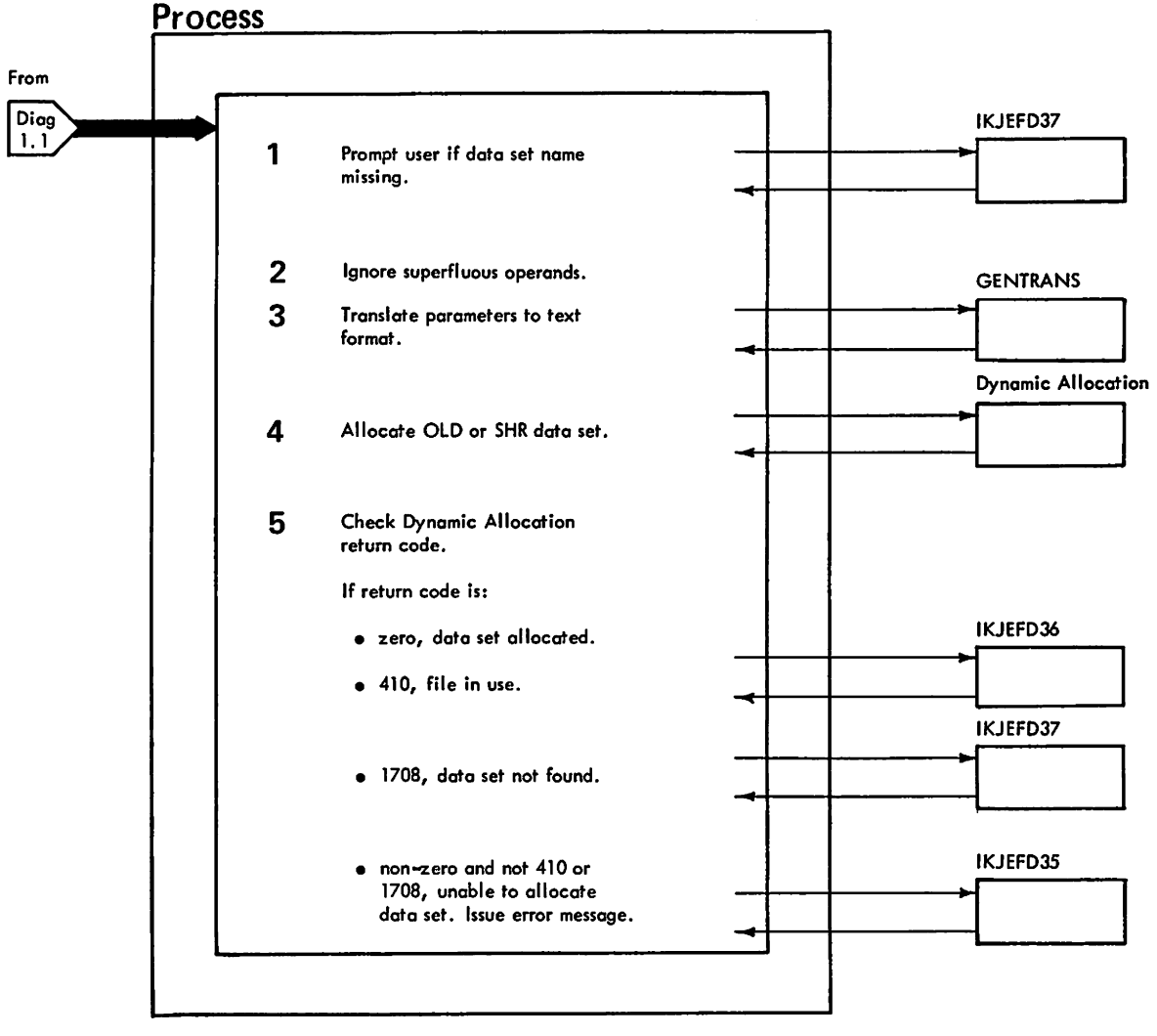


Diagram 1.4. ALLOCATE OLD and SHR Data Set Processing (Part 2 of 2)

1 Check for data set name on the DATASET/DSNAME keyword. If the name is missing use IKJEFD37 to prompt the user to supply one.

2 Ignore the following operands: SPACE, DIR, HOLD/NOHOLD, DEST, RELEASE, ROUND, and BLOCK/BLKSIZE/AVBLOCK/TRACKS/CYLINDERS.

3 Use GENTRANS to translate the parameters to dynamic allocation text format. The pointer to the text units is returned from GENTRANS in the IKJZB831 parameter list.

4 Use Dynamic Allocation to allocate the OLD or SHR data set. Register 1 points to a pointer to the dynamic allocation request block.

5 Check the dynamic allocation return code.

- If the return code is zero, the data set has been allocated.
- If the return code is 410, the file is in use. IKJEFD36 prompts the user to enter 'FREE', to free and reallocate the file, or 'END' to terminate the command.
- If the return code is 1708, the data set was not found. IKJEFD37 prompts the user for a new data set name. Repeat step 3, 4, and 5.
- If return code is non-zero and not 410 or 1708, allocation of the OLD or SHR data set has failed. IKJEFD35 uses the DAIR failure message routine IKJEFD18 to analyze the return code and send the appropriate error message to the user.

Control is returned to IKJEFD30.

Object Module: IKJEFD32

Diagram 1.5. ALLOCATE NEW Data Set Processing (Part 1 of 2)

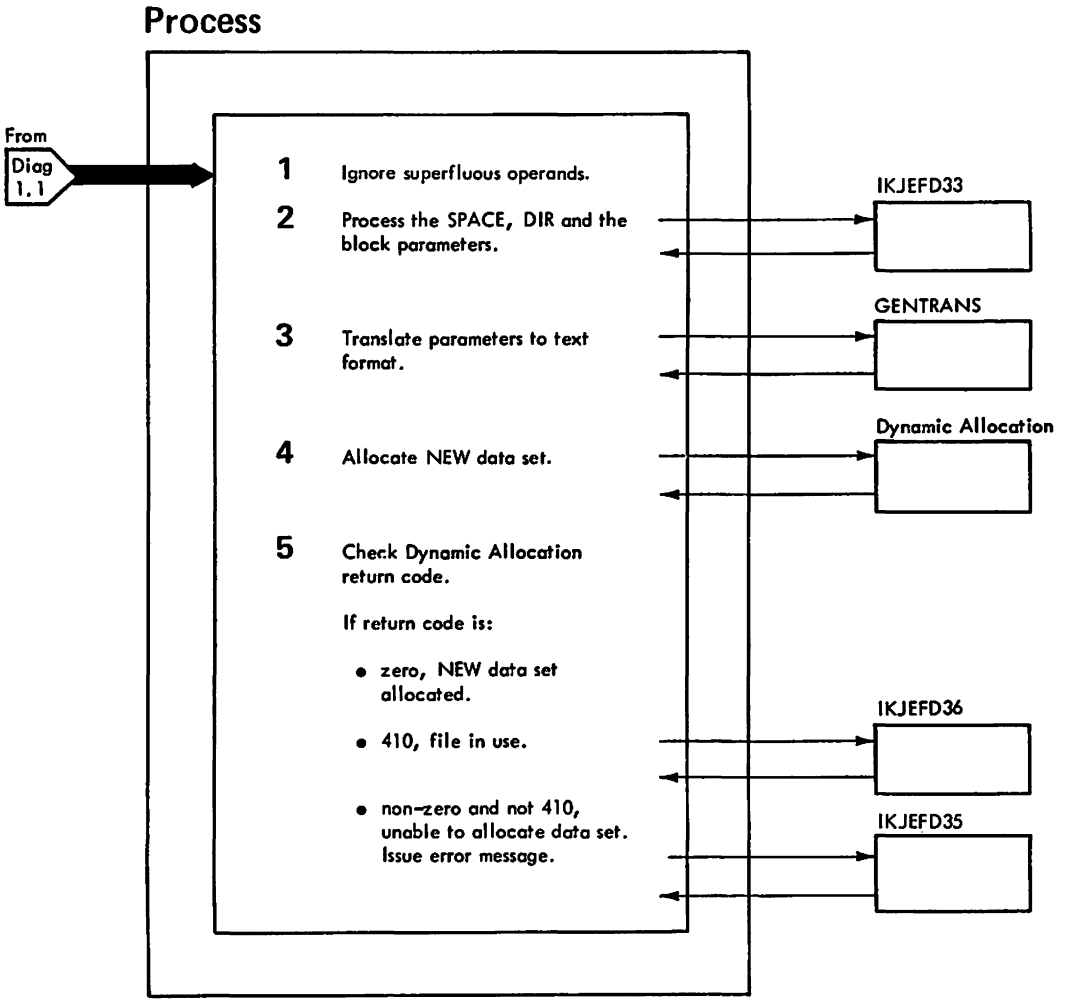


Diagram 1.5. ALLOCATE NEW Data Sets Processing (Part 2 of 2)

1 Ignore the following operands: DEST and HOLD/NOHOLD.

2 Use IKJEFD33 to process the SPACE, DIR, and block parameters.

3 Use GENTRANS to translate the parameters to dynamic allocation text format. The pointer to the text units is returned from GENTRANS in the IKJZB831 parameter list.

4 Use Dynamic Allocation to allocate the NEW data set. Register 1 points to a pointer to the dynamic allocation request block.

5 Check the dynamic allocation return code.

- If the return code is zero, the data set has been allocated.
- If the return code is 410, the file is in use. IKJEFD36 prompts the user to enter 'FREE', to free and reallocate the file, or 'END' to terminate the command.
- If the return code is non-zero and not 410, allocation of the NEW data set has failed. IKJEFD35 uses the DAIR failure message routine IKJEFF18 to analyze the return code and send the appropriate error message to the user.

Control is returned to IKJEFD30.

Object Module: IKJEFD32

Diagram 1.6. ALLOCATE MOD Data Set Processing (Part 1 of 2)

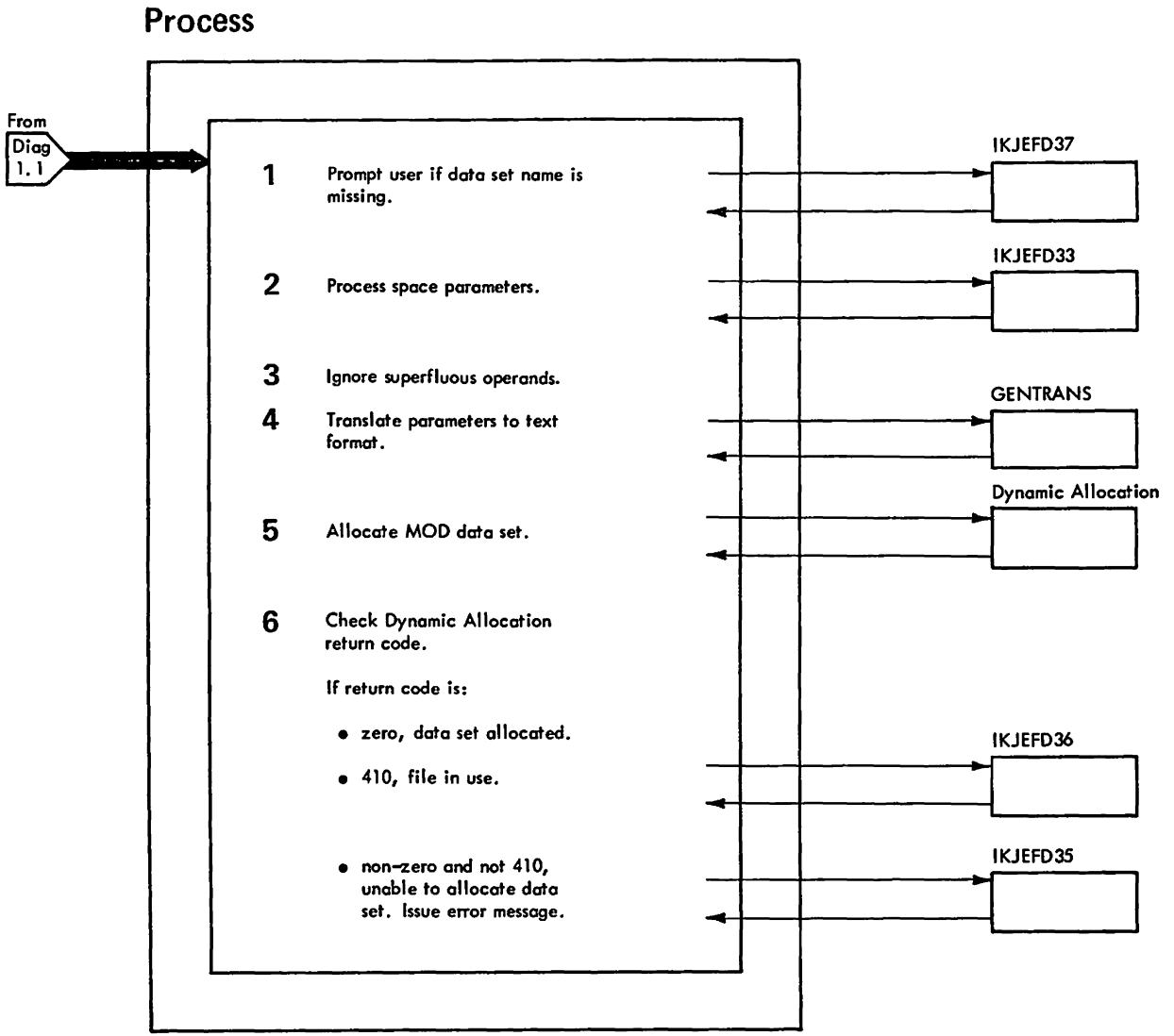


Diagram 1.6. ALLOCATE MOD Data Set Processing (Part 2 of 2)

1 Check for data set name on the DATASET/DSNAME keyword. If the name is missing use IKJEFD37 to prompt the user to supply one.

2 Use IKJEFD33 to process space parameters.

3 Ignore the following operands: DEST and HOLD/NOHOLD.

4 Use GENTRANS to translate the parameters to dynamic allocation text format. The pointer to the text units is returned from GENTRANS in the IKJZB831 parameter list.

5 Use Dynamic Allocation to allocate the MOD data set. Register 1 points to a pointer to the dynamic allocation request block.

6 Check the Dynamic Allocation request code.

- If the return code is zero, the data set has been allocated.
- If the return code is 410, the file is in use. IKJEFD36 prompts the user to enter 'FREE', to reallocate the file, or 'END' to terminate the command.
- If the return code is non-zero and not 410, allocation of the MOD data set failed. IKJEFD35 uses the DAIR failure message routine IKJEFF18 to analyze the return code and send the appropriate error message to the user.

Control is returned to IKJEFD32.

Object Module: IKJEFD32

Diagram 1.7. ALLOCATE DUMMY Request Processing (Part 1 of 2)

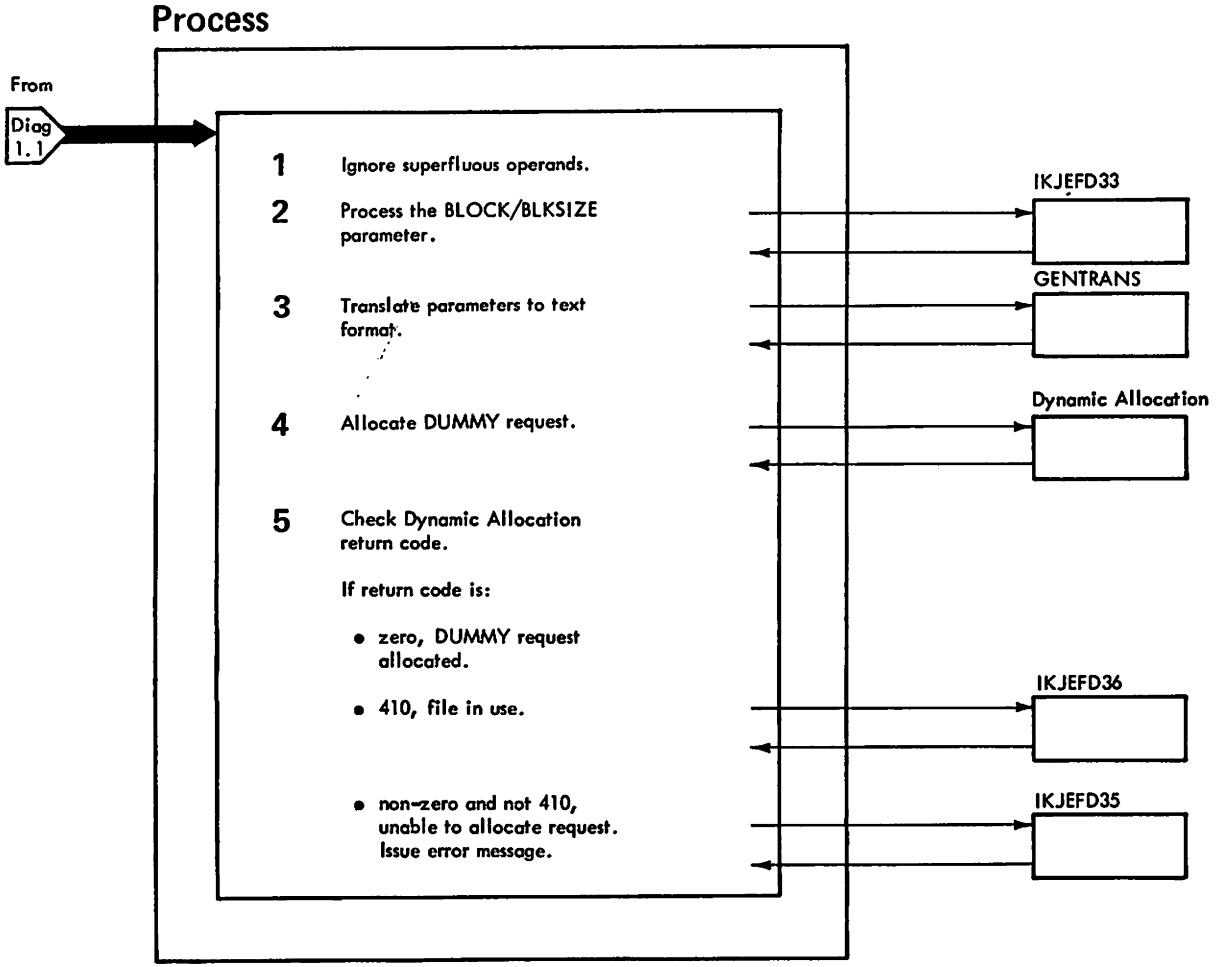


Diagram 1.7. ALLOCATE DUMMY Request Processing (Part 2 of 2)

1 Ignore all operands except DUMMY, FILE/DDNAME, BLOCK/BLKSIZE, AVBLOCK, TRACKS, CYLINDERS, and USING.

2 Use IKJEFD33 to process the BLOCK/BLKSIZE parameter.

3 Use GENTRANS to translate the parameters to dynamic allocation text format. The pointer to the text units is returned from GENTRANS in the IKJZB831 parameter list.

4 Use Dynamic Allocation to allocate the DUMMY request. Register 1 points to a pointer to the dynamic allocation request block.

5 Check the dynamic allocation return code.

- If the return code is zero, the DUMMY request has been allocated.
- If the return code is 410, the file is in use. IKJEFD36 prompts the user to enter 'FREE', to free and reallocate the file, or 'END' to terminate the command.
- If the return code is non-zero and not 410, allocation of the DUMMY request has failed. IKJEFD35 uses the DAIR failure message routine IKJEFF18 to analyze the return code and send the appropriate error message to the user.

Control is returned to IKJEFD30.

Object Module: IKJEFD32

Diagram 1.8. ALLOCATE Concatenate Data Set Processing (Part 1 of 2)

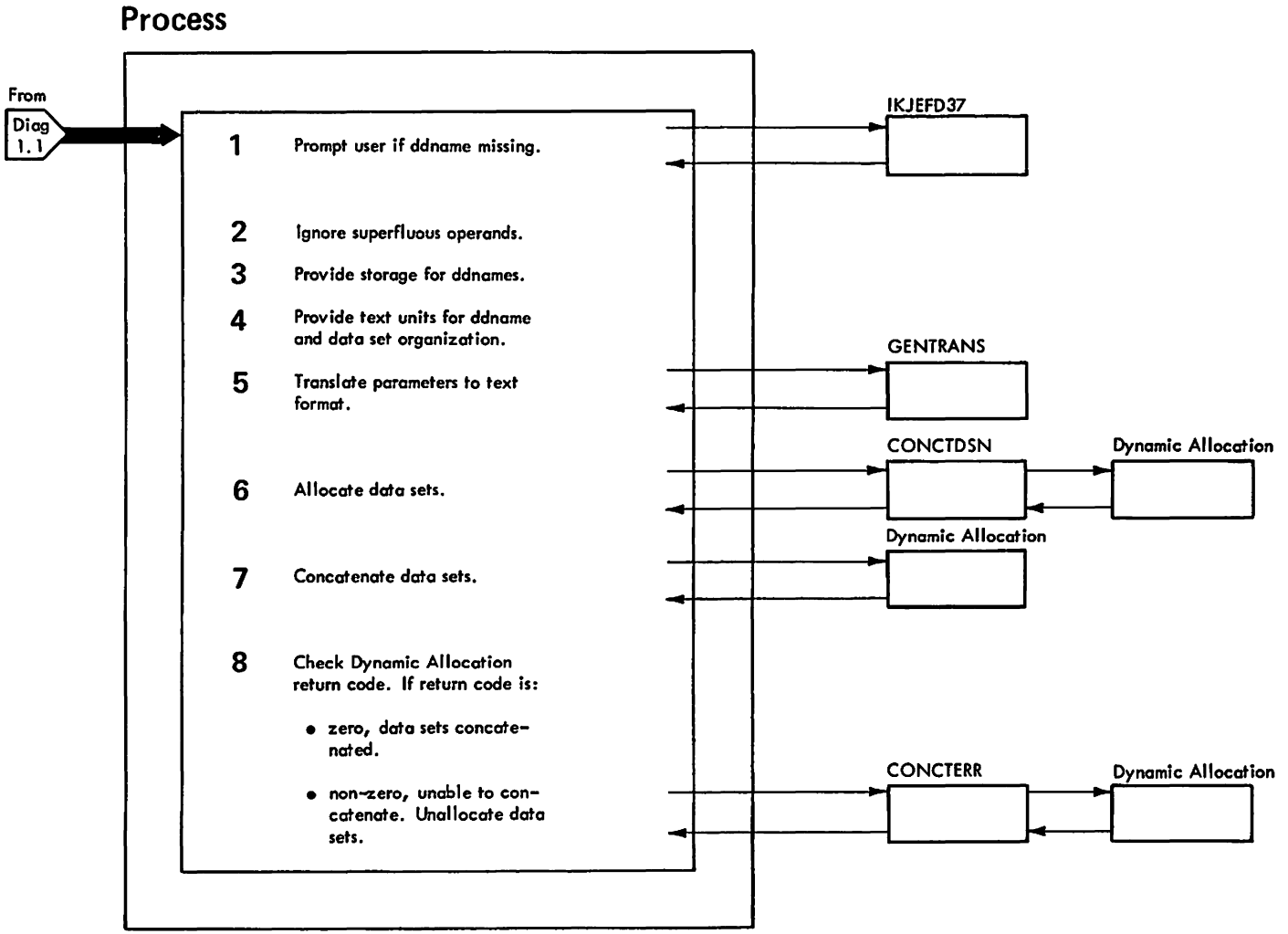


Diagram 1.8. ALLOCATE Concatenate Data Set Names Processing (Part 2 of 2)

1 Check for ddname on the FILE/DDNAME keyword. If the name is missing, use IKJEFD37 to prompt the user to supply one.

2 Ignore all operands except DATASET/DSNAME, FILE/DDNAME, and STATUS.

3 Provide the required amount of storage needed to save the ddnames.

4 Provide text units for the ddname and the data set organization to be returned by Dynamic Allocation.

5 Use GENTRANS to translate the parameters to dynamic allocation text format. The pointer to the text units is returned from GENTRANS in the IKJZB831 parameter list.

6 Use CONCTDSN to request Dynamic Allocation to allocate all of the data sets in the list.

7 Use Dynamic Allocation to concatenate the data set names.

8 Check the Dynamic Allocation return code.

- If the return code is zero, the data set names have been concatenated.
- If the return code is non-zero, concatenation of the data set names has failed. CONCTERR requests Dynamic Allocation to unallocate the data sets.

Control is returned to IKJEFD30.

Object Module: IKJEFD34

Diagram 2. ATTRIB Command Processing (Part 1 of 2)

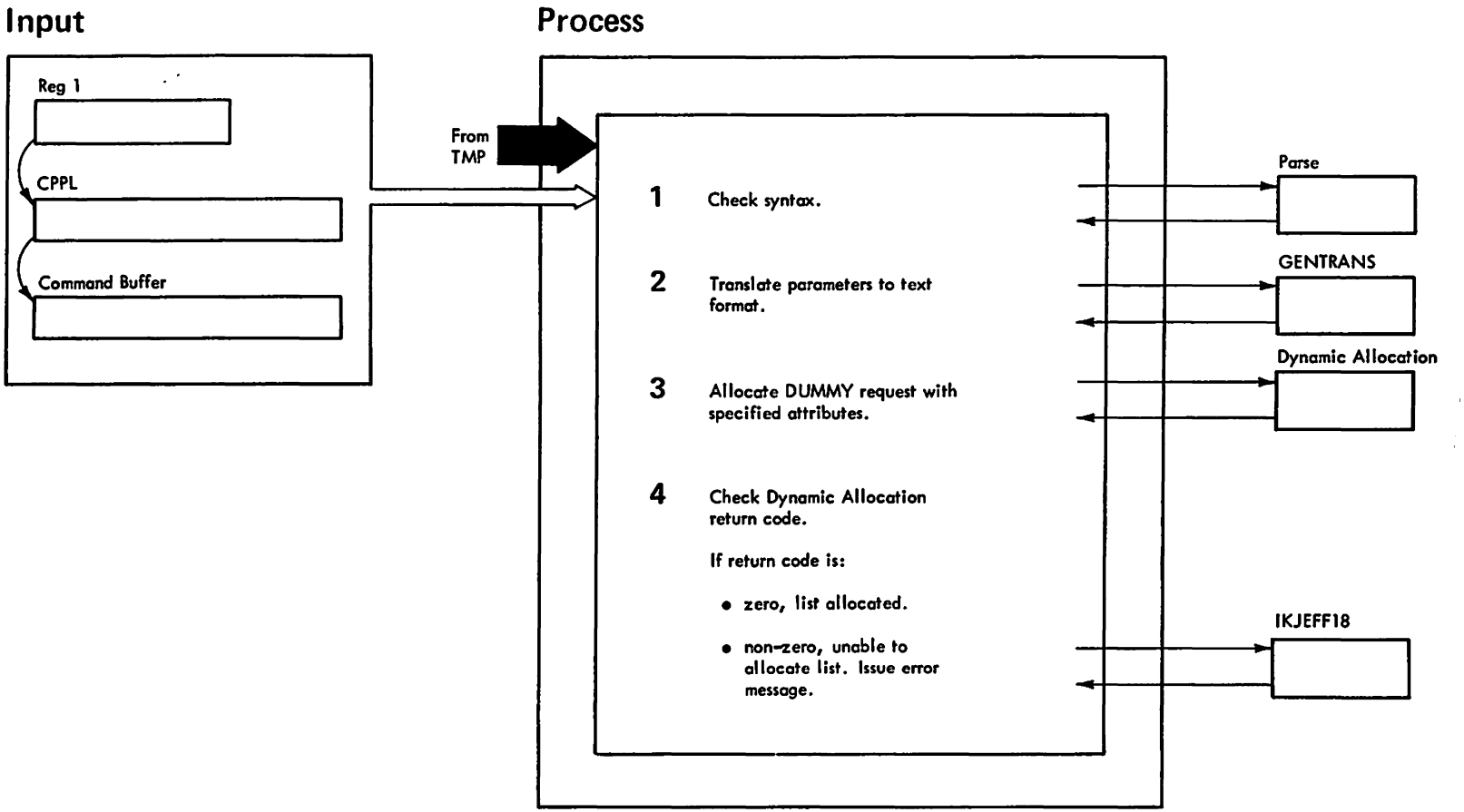


Diagram 2. ATTRIB Command Processing (Part 2 of 2)

1 Use Parse to scan and syntax check the command. For those operands Parse can not fully check (like LRECL), validity check exits in ATTRIB are used. The validity check exit for the name uses Dynamic Allocation to check the validity of the attr-list-name. If Parse failed, (non-zero return code) an error message will be issued and control is returned to the TMP.

2 Use GENTRANS (IKJCB831) to translate the parameters from Parse output to text format. For those parameters which GENTRANS can not translate (DEN, DSORG, LRECL, EXPDT, and BUFOFF), ATTRIB builds the text units. If GENTRANS failed (non-zero return code), an error message is issued, Parse output is freed, and control is returned to the TMP.

3 Use Dynamic Allocation to allocate a DUMMY request with the specified attributes.

4 Check the Dynamic Allocation return code.

- If the return code is zero, the request has been allocated and control is returned to the TMP.
- If the return code is non-zero, allocation has failed. Use the DAIRFAIL message routine IKJEFF18 to analyze the return code and send the appropriate error message to the user. Control is returned to the TMP.

Object Module: IKJEFATT

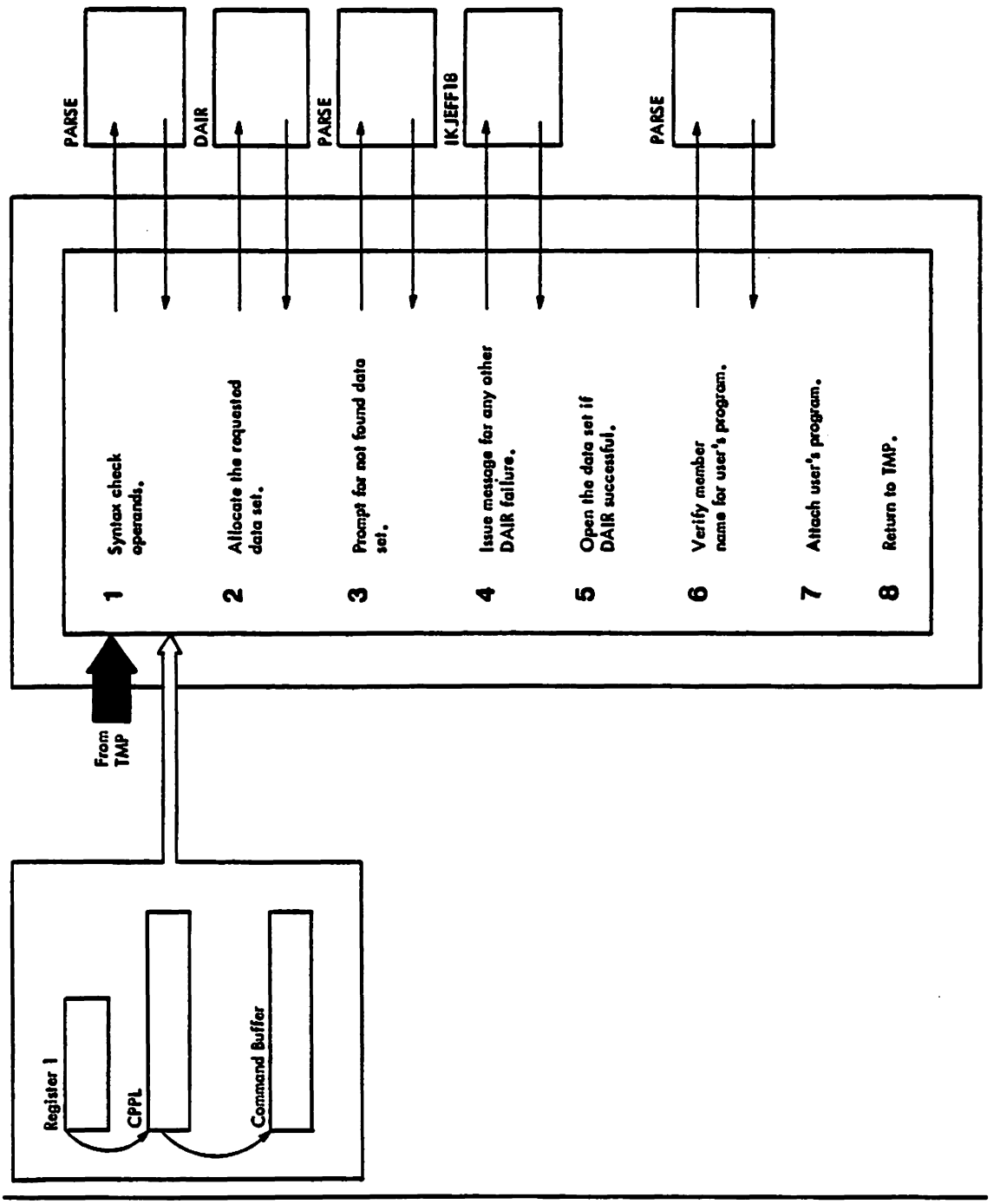


Diagram 3. CALL Command Processing (Part 1 of 2)

Diagram 3. CALL Command Processing (Part 2 of 2)

Extended Description

1 IKJEFT08 uses Parse to scan and syntax check the command operands. Register 1 points to the Parse parameter list (PPL). Upon return from Parse, IKJEFT08 checks to see if a member name has been supplied. If not, it re-uses Parse to prompt the user to supply it. The operand information is placed in buffers.

2 IKJEFT08 examines the data set name supplied as an operand to the CALL processor, to see if it is fully qualified. If it is not, it appends the word '.LOAD' to the right of it. It then places the program member name, the password, if any, and a pointer to the fully qualified data set name in the DAIR parameter block for a data set (DAPB08). IKJEFT08 uses DAIR (Dyanamic Allocation Interface Routine). The X'0008' operation code in the first field of the DAPB08 requests DAIR routine to allocate the data set to the user, if this has not previously been done. The remaining fields are either blanked out or set to zero.

For further information about DAIR, see *OS/VS2 Terminal Monitor Program and Service Routines Logic.*

3 If the DAIR return code indicates a not-found data set, IKJEFT08 notifies the user with a message. It then re-uses Parse to prompt the user to resupply the data set name. IKJEFT08 then re-uses DAIR after the user supplies a name.

4 If any other non-zero return code is issued by DAIR, allocation failed. The DAIR Failure Message routine (IKJEFF18) analyzes DAIR's input and output and sends an appropriate message to the user.

5 When DAIR is successful, IKJEFT08 moves the ddname supplied by DAIR to the DCB for the requested data set, and opens the data set.

6 IKJEFT08 issues a BLDL macro against the opened data set to verify the member name of the program to be attached. If the return code is a 4, (member name not found,) it uses Parse once more to prompt the user to resupply the member name. The BLDL macro is again issued to verify the newly supplied member name.

7 Once the BLDL macro has successfully verified the program member name, an ATTACH macro is issued to attach and pass control to it. If a parameter list was supplied, it will be truncated if more than 100 characters.

8 Upon completion of the attached program, IKJEFD00 releases all work and parameter areas and returns control to the TMP.

Object Module: IKJEFT08

Page of SY28-0652-4
As Updated March 1, 1985
By Suppl. SD23-0299 for 5665-291, 5740-XC6

This page intentionally left blank.

This page intentionally left blank.

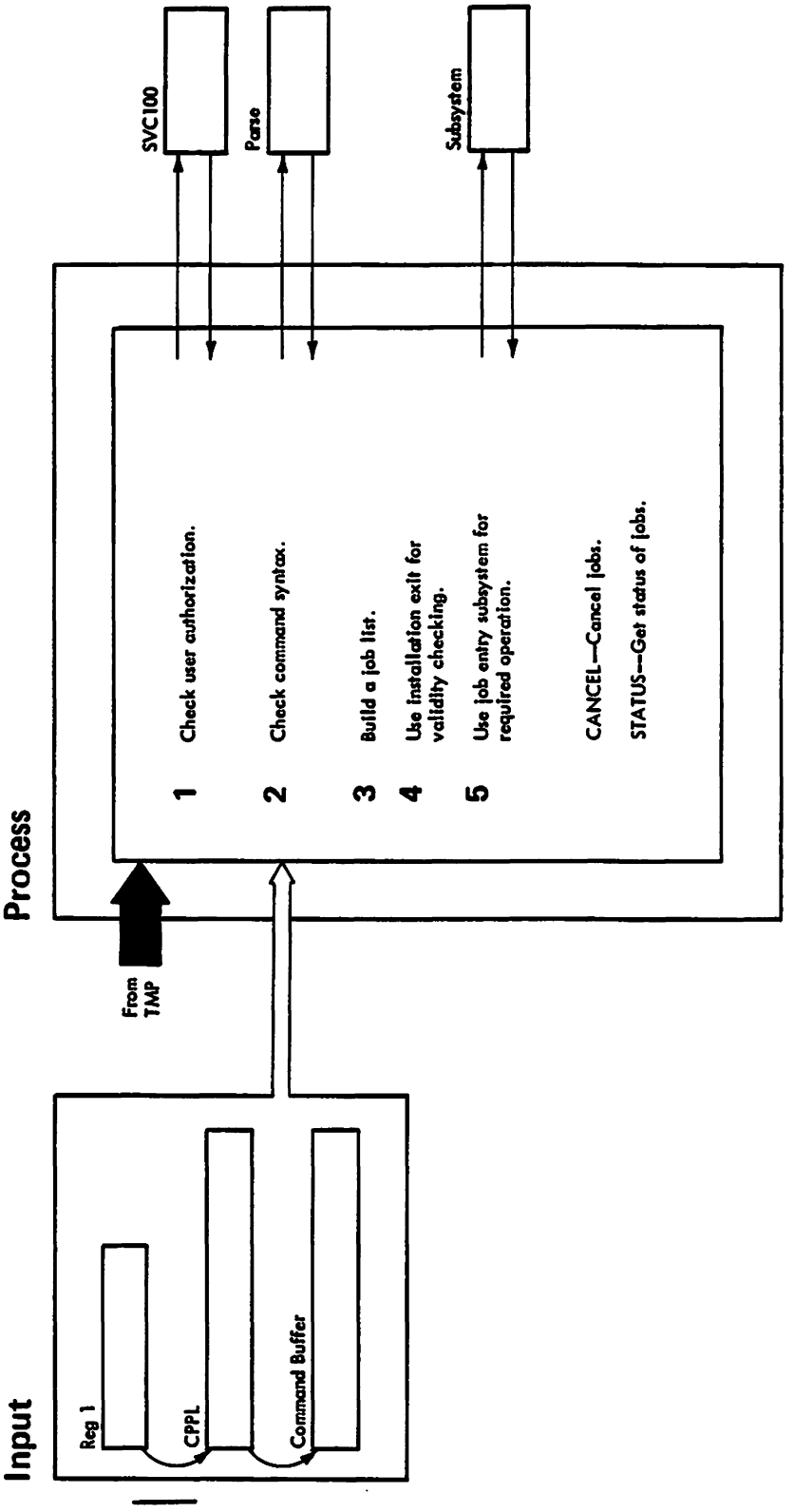


Diagram 4. CANCEL/STATUS Processing (Part 1 of 2)

Diagram 4. CANCEL/STATUS Processing (Part 2 of 2)

1 Use SVC100 to check the user's authorization to enter the command. Information is passed to SVC100 in the FIBPARMS parameter list. If the user is not authorized to enter the CANCEL or STATUS commands, an error message is issued and control is returned to the TMP.
Object Modules: IKJEFF58 (CANCEL), IKJEFF56 (CANCEL), IKJEFF56 (STATUS).

2 Use Parse to check the command syntax.

For CANCEL, use the Parse validity check exit (IKJEFF49) to check the jobname or jobname (jobid) and the STATUS interface to the job entry subsystem to validity check the command.

For STATUS, no exit is used. If no parameters were specified on the STATUS command, this step is skipped.
Object Modules: IKJEFF50 (STATUS), IKJEFF57 (CANCEL)

3 Build a list of jobs from the names indicated on the STATUS or CANCEL command.
Object Modules: IKJEFF50, (STATUS), IKJEFF57, (CANCEL)

4 Use installation exit IKJEFF53 to validity check the list of jobs.

For STATUS commands with no parameters, this step is skipped.

For CANCEL commands, the IBM supplied exit will reject jobnames that are not userid plus one or more characters.
Object Module: IKJEFF51

5 Use the job entry subsystem to perform the required operation. Information is passed to the job entry subsystem via the SSOB parameter list.

For CANCEL, use IKJEFF54 to request the job entry subsystem to cancel the requested jobs.

For STATUS, use the job entry subsystem to return the status of the requested jobs. For a STATUS command with no operands, the subsystem uses the userid plus one character to search the system queues for a job name.
Object Module: IKJEFF52

Page of SY28-0652-4
As Updated March 1, 1985
By Suppl. SD23-0299 for 5665-291, 5740-XC6

EXEC Command Processing Operation

EXEC command processing proceeds in separate operations known as Phase 1 and Phase 2.

Phase 1 receives control from the TMP. The operation includes reading the CLIST records from the input data set, building the in-storage command procedure and the Symbolic Name and Symbolic Value tables, then placing the command procedure just built on the input Stack. Phase 1 is described in this publication.

Phase 2, the statement executor, receives control from the GETLINE function of the I/O Service Routines as each record is removed from the command procedure (provided that the procedure was built by EXEC Phase 1) with a parameter list containing pointers to the UPT, the ECT, the ECB, and the GTPB. Phase 2 is described in *OS/VS2 Terminal Monitor Program and Service Routines Logic.*

Phase 1 Processing

Phase 1 processing is done either explicitly or implicitly.

The explicit form is defined when the user enters 'EXEC' with a data set name; optionally, a value list (in quotes); and options. EXEC will open the data set, explicitly named with the command, and read the records (PROC statement and commands) into storage.

The implicit form is defined when the user enters only a member name (optionally preceded by a percent sign) of a partitioned command procedure library, optionally followed by a value list. However, the command procedure library must have been previously allocated to the file name 'SYSPROC' prior to the implicit EXEC command. This could have been accomplished either by step allocation at LOGON time or or by using the TSO ALLOCATE command. When the user enters an implicit EXEC command *without* the percent sign, EXEC does not get control immediately; the TMP will go through its normal routine of attaching the implicit EXEC command as a TSO command, using the LINK library data sets. If ATTACH fails (806 ABEND), the EXEC command is invoked implicitly, Optionally, the user can precede the implicit EXEC name with a percent sign which will signal the TMP to bypass the ATTACH or the LINK library data sets and to ATTACH the EXEC command implicitly. The EXEC command will use DAIR to determine if filename 'SYSPROC' exists allocated. If there is no 'SYSPROC' data set allocated, the user gets "COMMAND name-entered NOT FOUND"; otherwise, the EXEC command will OPEN/FIND the member name specified. If the FIND fails, the message "COMMAND name-entered NOT FOUND" will be issued. If the FIND is successful, the records (PROC statement and Commands) in that member will be read into storage. If the EXEC command has been attached by another command as an implicit EXEC command and the member name entered could not be found as described by the two cases above, then the message will read "SUBCOMMAND name-entered NOT FOUND".

A STAE routine is established initially to prevent fragmenting subpool 78 storage in the event that EXEC ABENDs, by freeing any subpool 78 storage that had been explicitly gotten.

The input data set is allocated using DAIR. If the invocation is explicit, EXEC will use the entry code for data set allocation. If the invocation is implicit, EXEC will use the entry code for Information Retrieval to determine if a command procedure library is presently allocated.

The Dataset I/O function of the I/O Service Routines is utilized to read records from the input data set. The Service Routines will be loaded or addressed via a Resident Service Routine Vector table, if available, then GETLINE will be used for the input records.

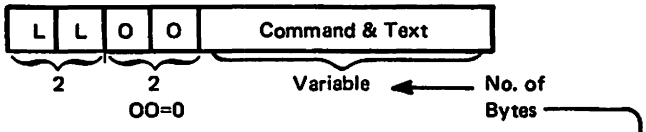
A set of control variables are defined and initialized for the immediate CLIST by placing the control variable names in the Symbolic Name table (SNTAB) and their values in the Symbolic Value table (SVTAB). A flag byte in the SNTAB defines the entry as a variable that can or can not be set by the user.

If the first record is a PROC statement, all the symbolic parameters defined on the PROC statement are syntax checked. The EXEC command saves the names of the parameters in the Symbolic Name table and any default values in the Symbolic Value table located in subpool 78 storage. Next, the EXEC command will build a PCL dynamically based on the PROC statement definitions and then invoke Parse to syntax check any replacement values in the value list. Parse passes the values found in the value list back to EXEC, which updates the Symbolic Value table accordingly (updates positional parameters entries and replaces keyword default values with those specified by the user).

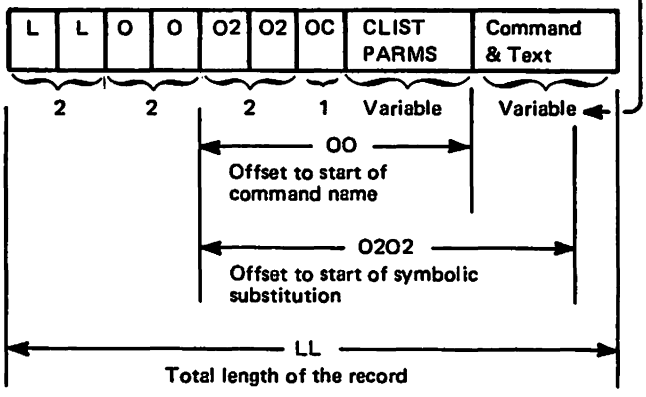
After the Symbol Name and Value tables have been built, the EXEC command will get subpool 78 storage to construct the in-storage command procedure. Each record is then read from the input data set and identified (scanned to determine the command name and whether a label was present), and copied to the in-storage command procedure. All CLIST statements are uniquely defined; any other statement is considered to be a TSO

command. Each in-storage command procedure record will have the following format:

On Input:



In CP Storage:



OC = Operation code defining the command; defined operation codes.

Code	Description
00	TSO Command
01	GOTO Statement
02	IF Statement
03	ELSE Statement
04	WRITE Statement
05	WRITENR Statement
06	EXIT Statement
07	DO Statement
08	END Statement
09	CONTROL Statement
0A	ERROR Statement
0B	TERMIN Statement
0C	READ Statement
0D	SET Statement
0E	RETURN Statement
0F	Internal GOTO Statement
10	READDVAL Statement
11	ATTN Statement
12	OPENFILE Statement
13	CLOSFILE Statement
14	GETFILE Statement
15	PUTFILE Statement
FF	Used internally to IKJCT432 for GLOBAL Statement

CLIST PARMS – some of the CLIST statements will require immediate parsing or certain addresses to be preserved as the in-storage command procedure record is being constructed. This area contains information relevant to those CLIST statements.

COMMAND TEXT – complete copy of the record as it was read from the CLIST data set.

DATA/ENDDATA statements are not placed in the in-storage command procedure. Statements between DATA and ENDDATA statements are considered to be TSO commands and are not syntax checked for labels.

GLOBAL statements are removed by the Phase 1 EXEC command processing and are not placed in the in-storage command procedures.

When all the records in the CLIST data set have been processed as described above, the STACK macro is used to place the command procedure on the input Stack using the EXEC option in the Stack interface. An existing but previously reserved field defined in the LSD will be utilized to address the EXEC data area, which will be used by EXEC Phase 2 to obtain the addresses of the Symbol Name and Value tables. This EXEC data area also will be used by the Stack service routine when removing and freeing the space occupied by this command procedure from the input Stack.

The in-storage command procedure record will be different for each of the CLIST statements; however, all TSO commands will result in the same basic command procedure record.

The following is a description of the processing that will occur, by command, as well as the description of each command CLIST PARMS Area:

- The IF statement processing routine will be reentrant and will invoke the Command Identification process in IKJCT432 to identify the command following the THEN clause; the IF statement will be divided into two logical commands. The IF routine will now generate an internal GOTO with a four-byte address in the CLIST PARMS area of the next in-storage record to be created. Next, the CLIST PARMS area is updated with a four-byte address of the start of the false path. Next, the IF routine will get the next input record from the input CLIST data set and invoke the Command Identification process to identify the record. This is done so that the correct relationship will exist for IF and ELSE statements (if the ELSE statement is used). Then, control is returned to EXEC mainline for processing of the next input CLIST record.

- The ELSE statement processing routine is reentrant and will verify that the input parameter list (IKJCT432) contains a flag indicating the ELSE statement processing was invoked by an IF statement; otherwise, the user has invalid syntax. The Command Identification process (IKJCT432) will be invoked to process the command following the ELSE: the ELSE statement is broken into two logical commands. Then the ELSE routine will update the address in the CLIST PARMS area of the internal GOTO prior to the ELSE in-storage record to the real end of the false path and return to EXEC mainline for processing of the next input record.
- The DO statement processing routine is reentrant and will get a record from the input data set and invoke the Command Identification process in IKJCT432 as many times as necessary until an END or an alternate END is processed. The DO routine will set a flag byte in the CLIST PARMS area to indicate the presence of a WHILE operand. When the corresponding END has been processed and the matching DO statement contained a WHILE operand, an internal GOTO statement is generated with the address of the corresponding DO statement and the false path address in the CLIST PARMS area is updated with address of the next CLIST record to be created. No internal GOTO is generated if the DO statement had no WHILE operand. Control returns to EXEC mainline for processing of the next input record.
- The ERROR and ATTN statement processing is reentrant and will invoke the Command Identification process (IKJCT432) to identify the command specified as the error action unless the OFF operand was present, or if no operands were found. When a command action is specified, the ERROR or ATTN statement will be broken into two logical in-storage Stack records; upon return from Command Identification, a 4-byte address in the CLIST PARMS area is initialized to the address of the next in-storage record to be created. A Flag byte in the CLIST PARMS area is used to indicate whether NO operands were found, OFF was specified, or a command was found. Then control returns to EXEC to process the next input record.

The SET statements use four bytes in the CLIST PARMS area to store the address of the Symbolic Name Element for the symbolic parameter specified on the left of the equal sign.

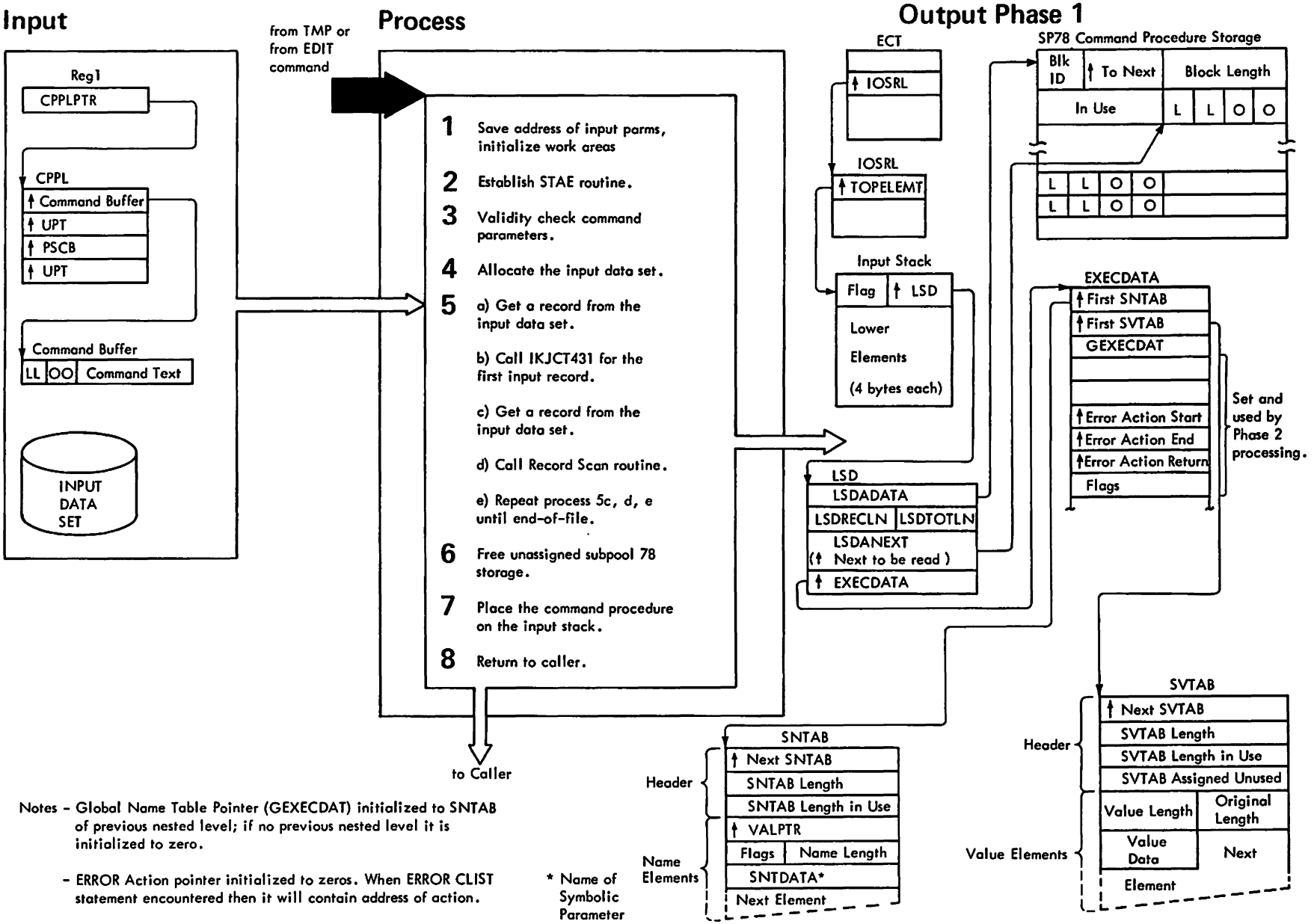
The READ and READDVAL statements use a variable amount of the CLIST PARMS area, depending on the number of symbolic parameters that have been specified. First are two bytes used as a counter, which will contain the number of symbolic parameters that were specified. Following the counter bytes is a four-byte entry for each symbolic parameter specified, which will contain the address of the corresponding element in the Symbolic Name table.

The CONTROL statement uses two bytes of the CLIST PARMS area for a flag area. The flag bytes will contain indicators corresponding to the options specified on the CONTROL statement. When an alternate END was specified, the EXEC Phase 1 Common Data Area (ECDA) fields will be updated to reflect those changes.

The Internal GOTO statement is used by the IF and DO statement processors for controlling the flow around false or conditional paths. It will use four bytes of the CLIST PARMS area to save the target address.

The remainder of the statements have no CLIST PARMS information and pass control to a common in-storage stack build routine to place an op code in the command procedure record and return to EXEC Main Control.

Diagram 5.1. Phase 1 EXEC Command Main Control (IKJCT430) (Part 1 of 2)



Notes - Global Name Table Pointer (GEXECDAT) initialized to SNTAB of previous nested level; if no previous nested level it is initialized to zero.

- ERROR Action pointer initialized to zeros. When ERROR CLIST statement encountered then it will contain address of action.

Diagram 5.1. Phase 1 EXEC Command Main Control (IKJCT430) (Part 2 of 2)

1 Save input CPPLPTR. Initialize the common work area.

2 Establish a STAE routine.

3 Syntax check input parameters.

- For explicit EXEC commands, invoke Parse to obtain dataset name containing the command procedure, determine if the value list was specified, and obtain the CLIST options (LIST or PROMPT). Use IKJEFF19 to analyze any errors.
- For implicit EXEC commands, invoke Command Scan to obtain the member name (Command Procedure Library) and to determine if any operands were specified in the value list.

  Retain Parse output with pointer to the value list.

4 Invoke IKJDAIR to allocate the input data set.

- For explicit EXEC commands, Data Set Allocation will be requested.
- For implicit EXEC commands, Information Retrieval will be requested to determine whether 'SYSPROC' is allocated.

Use IKJEFF18 for any DAIR errors and use STAE routine to clean up and return to TMP with RC=12.

5 Use I/O Service Routines Dataset I/O to read input data set.

- Load IKJGETL (IKJSTCK).
- Stack a dataset I/O element, using I/O Service Routines with the input ddname specified as input with no output ddname, using the prompt option.

  a) Issue GETLINE for an input record.

  b) If this is first statement in the data set, then call Symbolic Parameter Definition (IKJCT431).

  If the return code is 4 proceed to Step 5A. If the return code is > 4 then use STAE Exit to cleanup and return to TMP, otherwise continue at C.

  Return Code 0 — Indicates PROC statement was not present.

  4 — Indicates PROC was present.

  c) Call Record Scan routine IKJCT432. If the return code is greater than 4, then use STAE Exit to clean up and return to the TMP.

  If the return code is 4 from IKJCT432 (EOF) then proceed to Step 6 for end-of-file cleanup.

  d) Check for end of input file. If not end of file return to Step 5 for next record.

6 For end of file on input, clean up unused storage for Name table, leave some space in the Value Table, and place pointer to EXECDATA in LSD. Clean up any SP78 storage not used for CLIST. If the procedure can not be executed due to error then proceed to Step 2 of the STAE routine to clean up.

7 Issue STACK with options specified by user to make command procedure the current input source.

8 Return to caller with RC=0 for normal completion. For error, see below.

Error Processing (STAE)

1. Indicate entry to STAE routine via ABEND using a flag for ABEND entry.
2. Free all subpool 78 storage which had been explicitly gotten — EXECDATA, SNTAB(s), SVTAB(s), LSD, command procedure storage area(s).
3. Issue a TCLEARQ and a STACK macro with the DELETE=ALL option.
4. Return to caller. For ABENDs, return to ABEND with no retry requested. For EXEC error situations, return to the TMP with a preset return code (RC=12); command procedure is not executed.

Diagram 5.2. Phase 1 EXEC Command Symbolic Parameter Definition (IKJCT431) (Part 1 of 3)

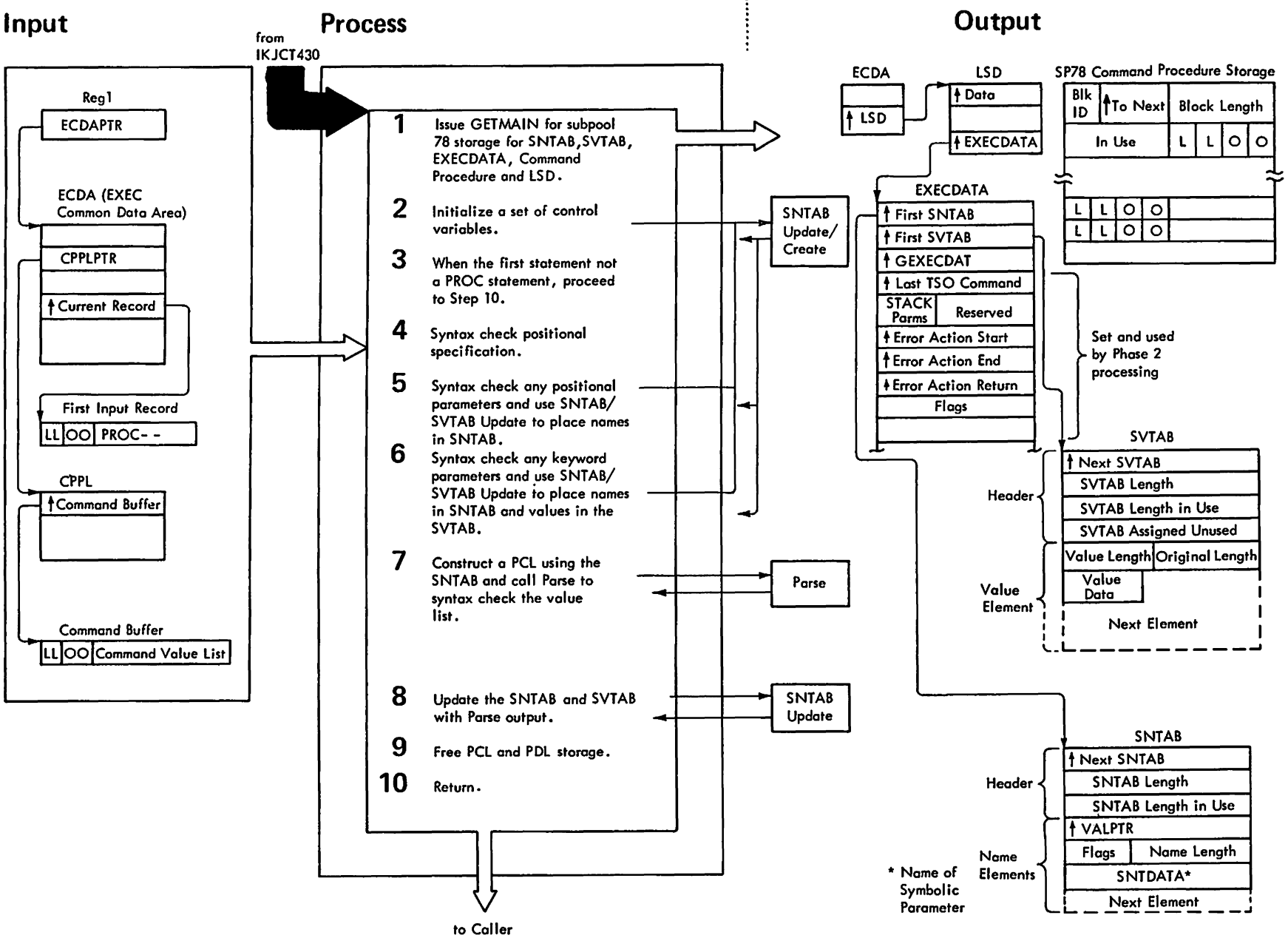


Diagram 5.2. Phase 1 EXEC Command Symbolic Parameter Definition (IKJCT431) (Part 2 of 3)

1 Save input pointer to the Common Data Area (CDA). Get storage for the Symbolic Name/Value tables, EXECDATA, LSD, and the command procedure from subpool 78. Initialize table header information for each block. If storage is unavailable, notify user and return with RC=16.

2 Initialize the control variables for this command procedure.

NAME	*VAL*
LASTCC	0
MAXCC	0
SYSPROC	Logon Proc. from TIOT
SYSTIME	Create entry; indicate evaluate immediately, initialize to a null entry
SYSDATE	Create entry; indicate evaluate immediately, initialize to a null entry
SYSUID	get from PSCE
SYSPREF	Create entry, indicate immediate evaluation, initialize to a null entry
SYSSCAN	initialize to default
SYSDVAL	initialize to null entry
SYSDIM	initialize to 00
SYSNEST	initialize to YES or NO depending on CLIST nesting
SYSICMD	to implicit name or null if explicit EXEC
SYSPCMD	initialize to null entry
SYSSCMD	initialize to null entry

3 Determine if the first statement is a PROC statement. When first statement is not a PROC statement then return with RC=0.

4 Syntax check positional specifications. Positional specifications must be numeric. Find the next non-numeric – a non-numeric is considered to be a comma, blank or tab or start of comment. Any other characters will be syntactically incorrect with user notified and "PROC not executable" set, then return.

5 Syntax check all positional parameters and place name in SNTAB. Accumulate a total number of positional parameters and the total length in bytes.

- Skip separators.
- First character must be alphameric.
- Find next non-alphameric.
- If name less than 252 characters and correct* then call IKJCT431 SNTAB/SVTAB Update routine to place name in SNTAB. If the PROC is not executable, then bypass adding to SNTAB.
- Repeat the above steps of 5 until all positional parameters are processed.

* If parameters syntactically incorrect then notify user, set "PROC not executable"; continue syntax checking any remaining parameters.

6 Syntax check keywords specified on PROC statement.

- Skip separators.
- First character can be alphameric or left paren. If alphameric, it is start of a new keyword so call IKJCT431 SNTAB/SVTAB to place previous keyword in SNTAB. Upon return, continue syntax checking until end of keyword is found (1-31 characters, the first of which is alphabetic and remainder alphameric*). Repeat above steps of 6 until all keywords are processed.

  If it was a left paren then syntax check the value for the current keyword. Value can be a quoted string or a character string*. Repeat above steps of 6 until all keywords are processed.

* If parameters are syntactically incorrect then notify user, set "PROC not executable"; continue syntax checking any remaining parameters.

7 If errors have occurred at this point, return with any accumulated return code. Otherwise, using the SNTAB dynamically construct a PCL representative of the PROC statement parameters and call Parse to parse the value list. If Parse fails then notify user and return. Use IKJEFF1S to analyze Parse errors.

8 Update the SNTAB/SVTAB with the Parse output. Take values from the PDL and place in the SVTAB. If operation runs out of storage, then get new blocks of SNTAB/SVTAB. If necessary, copy SVTAB to new block and free first block. If storage unavai.able, notify user and return (return code 16).

9 Free the PCL and PDL storage.

10 Return RC=0 First record not a PROC statement.
RC=4 First record was a PROC statement.
RC=16 Not enough storage (GETMAIN failure).

Diagram 5.2. Phase 1 EXEC Command Symbolic Parameter Definition (IKJCT431) (Part 3 of 3)

SNTAB Element Update/Create Routine

1. Locate the SNTAB pointer in EXECDATA.
2. Search the SNTAB to see if name already defined.
   - If name already defined and the request was to create a label entry, then this is an error situation. Notify user of duplicate label and set "PROC not executable" and return.
   - If name already exists and the request was to create a Symbolic Parameter element, then notify user of multiply defined parameter and set "PROC not executable" and return.
   - If name already exists and request was to locate, then return address of element RC=0.
   - If name does not exist and request was to locate an element, then create a new SNTAB element from the remaining storage. If storage remaining is insufficient, then get storage for a new block and extend the SNTAB. Set the last bit in the last element of previous SNTAB and create new entry in next SNTAB (if GETMAIN error return RC=16).

   When the request was to create a label entry, initialize the value pointer to the current location in the CP SP (78) storage and label flag. Return with RC=0. When the request was to create an element, set the appropriate type flags, adjust the SNTAB amount in use, and check to see if a value is required. If yes, call the SVTAB Update routine to create a value element. Return with address of element (SNTAB) with return code of SVTAB updates.
3. Return RC = 0 – Request performed address of element returned.
   RC = 16 – Not enough storage.

SVTAB Element Update/Create Routine

1. If request was to update an element, then determine if value will fit in old spot. If enough room then reuse the entry and return RC=0. If not enough room, change request to a create. Add element space to the free space in SVTAB.
2. If entry was to create an element, then determine if enough storage available. If not issue GETMAIN for a new block, copy all values from previous SVTAB. table up to current SNTAB element (if GETMAIN error, return with RC=16).
3. Create the new element: update SNTAB-SNTVLPTR to new entry and adjust the SVTAB amount in use.
4. Return RC = 0 – Request performed; the SNTVLPTR has the address of the SVTAB element.
   RC = 16 – Not enough storage.

Diagram 5.3. Phase 1 EXEC Command Record Scan Routine (IKJCT432) (Part 1 of 2)

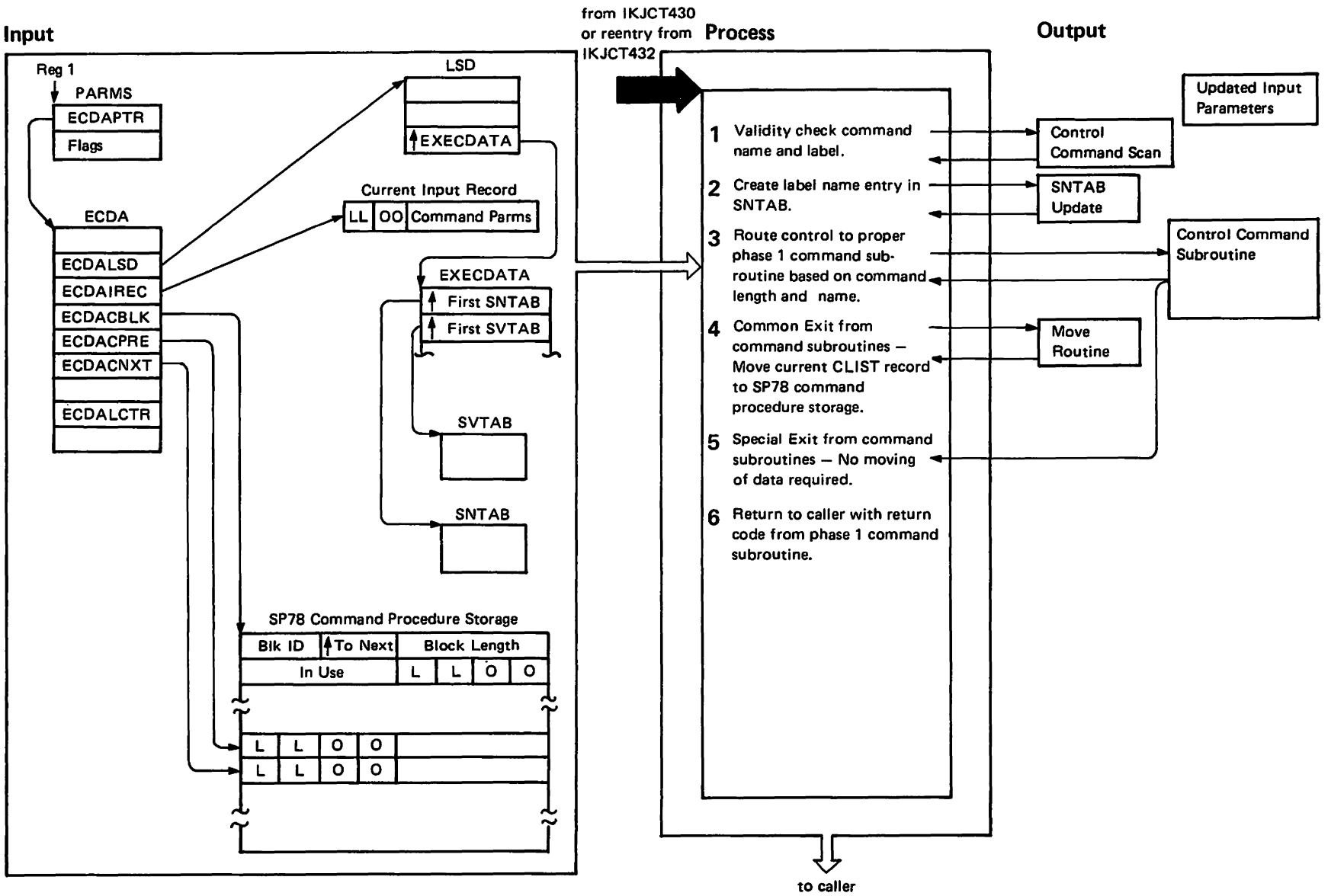


Diagram 5.3. Phase 1 EXEC Command Record Scan Routine (IKJCT432) (Part 2 of 2)

1 Call Control Command Scan.

2 If label was found and was syntactically correct, place name and record address in the Symbolic Name table. Otherwise, if there is what appears to be a label syntax but specified incorrect, then assume the record to be a TSO command. When the return code from the SNTAB Update routine is non-zero, then proceed to Step 4 with following return code –

0 when SNTAB Update RC not 16
16 when SNTAB Update RC=16

a) If not a syntactically correct command name and it contained a valid label, then assume it to be a TSO command. If there was no label, then assume the command is a TSO command in disguise.

If a label appears by itself, this is an error; notify user, set "PROC not executable", RC=0 and return via Step 4.

If no label or command appears, then return with RC=0 via Step 4.

3 Determine command type.

a) If the command was a TSO command in disguise, then call Common Command routine with the TSO op code.

b) Determine if the command is an END or alternate END; if so, call ENDRTN. Otherwise, use the length to index into branch table and go to proper length routine. (COMCMD = Common Command Routine, OP = op code.)

Length=1

1. Call COMCMD OP(00).

Length=2

1. IF Command ? Call IFRTN (IF Routine).
2. DO Command ? Call DORTN (DO Routine).
3. Call COMCMD OP(00).

Length=3

1. SET Command ? Call SETRTN (SET Routine).
2. Alternate END or END ? Check for end specification. YES – Call ENDRTN (END Routine).
3. Call COMCMD OP(00).

Length=4

1. GOTO Command ? Call COMCMD OP(01).
2. ELSE Command ? Call ELSERTN (ELSE Routine).
3. READ Command ? Call READRTN (READ/READDVAL/GLOBAL Routine).
4. EXIT Command ? Call COMCMD OP(06).
5. DATA Command ? Call DATARTN (DATA Routine).
6. ATTN Command ? Set OP(11). Call ERRRTN (ERROR/ATTN Routine).
7. Call COMCMD OP(00).

Length=5

1. ERROR Command ? Set OP(0A), Call ERRRTN (ERROR/ATTN Routine).
2. WRITE Command ? Call COMCMD OP(05).
3. Call COMCMD OP(00).

Length=6

1. TERMIN Command ? Call COMCMD OP(13).
2. GLOBAL Command ? Call READRTN OP(FF) (READ/READDVAL/GLOBAL Routine).
3. RETURN Command ? Call COMCMD OP(0E).
4. Call COMCMD OP(00).

Length=7

1. CONTROL Command ? Call CONRTN (CONTROL Routine).
2. WRITENR Command ? (Call COMCMD OP(04).
3. ENDDATA Command ? Call ENDARTN (ENDDATA Routine).
4. GETFILE Command ? Call COMCMD OP(14).
5. PUTFILE Command ? Call COMCMD OP(14).
6. Call COMCMD OP(00).

Length=8

1. READDVAL Command ? Call READRTN OP(10) (READ/READDVAL/GLOBAL Routine).
2. OPENFILE Command ? Call COMCMD OP(12).
3. CLOSFILE Command ? Call COMCMD OP(13).
4. Call COMCMD OP(00).

4 Common Exit – Call Common Move routine to move CLIST record to subpool 78 using current line pointer.

5 Special Exit – No moving required.

6 Return

RC=0	Record processed successfully.
RC=4	End-of-file occurred.
RC=16	Not enough storage (GETMAIN failure).
RC=20	GETLINE error.

Diagram 5.3.1. IF Routine (IKJCT432) (Part 1 of 2)

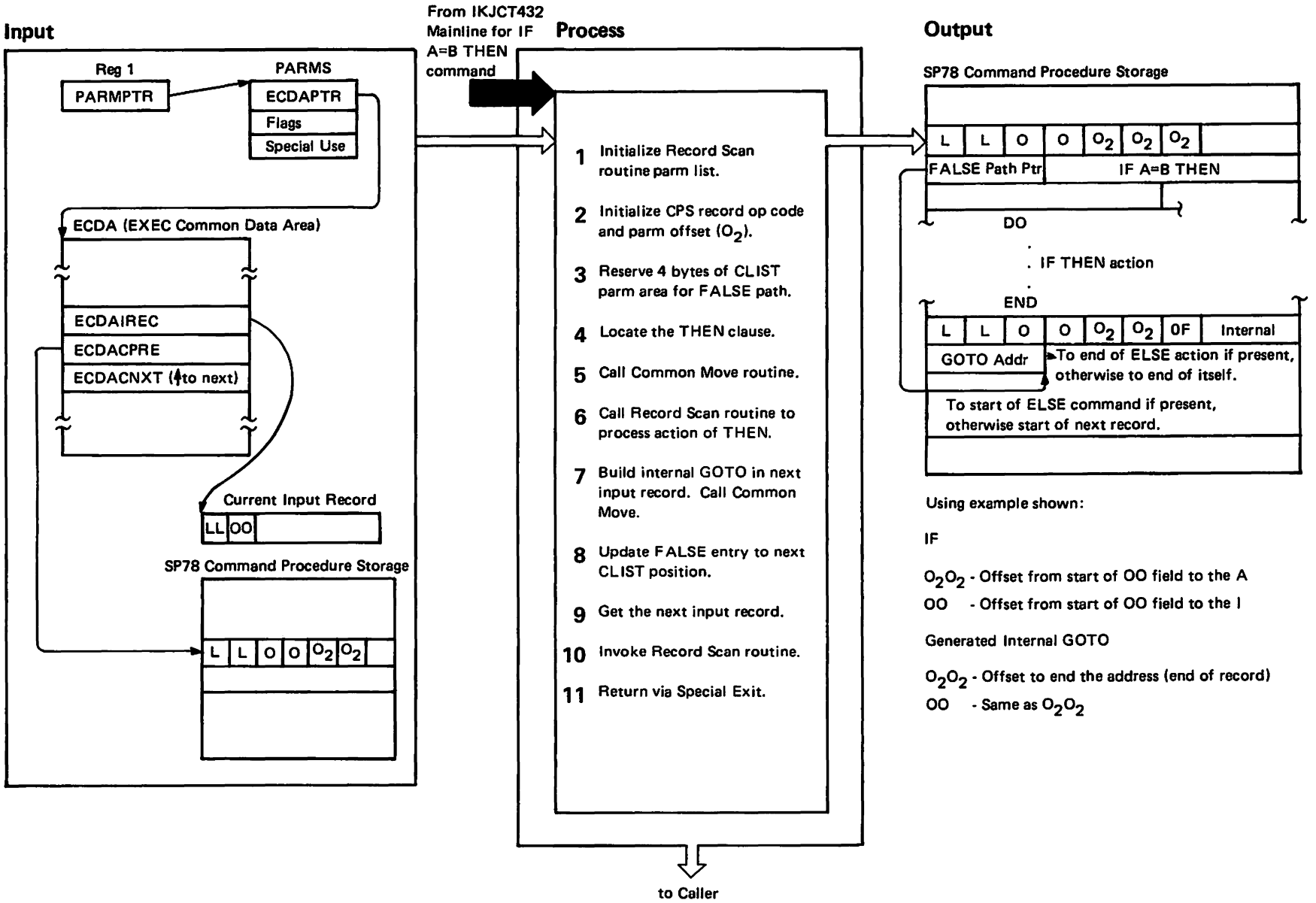


Diagram 5.3.1. IF Routine (IKJCT432) (Part 2 of 2)

1 Save input pointer to the Common Data Area (CDA) pointer and flag area.

2 Initialize the command procedure op code to X'02' and set the parameter offset (02) equal to the offset of the first parameter of the command (offset to A). Update LL of current CP record.

3 Reserve four bytes of CLIST Parm Area for the FALSE path address.

4 Locate the ending position of the THEN keyword by skipping separators and operands until the THEN clause is found.

5 Call the Common Move routine to move the record starting with IF and ending with THEN, to Command Procedure Storage Area. Respecify SP (78) command base after move in case record would not fit in the current CP block. If return code from Move non-zero then return with Move RC via Step 9.

6 Adjust offset in record to next position following N of THEN and call the Record Scan routine to process the command following. (Preserve the original input buffer address and length for correct freeing of buffer later.)

7 Upon return from the Record Scan routine construct an internal GOTO CLIST statement with an address in CLIST parms of the next command procedure record to be created and call the Common Move routine to place in the CP Storage (if the RSR return code was not zero then return with RSR RC via Step 9). Initialize the GOTO address to the next SP (78) command procedure record to be created. If the Move Return code was non-zero then return with Move RC via Step 9.

8 Update the FALSE path address in the IF CP Storage record to point to the next record to be created in CP Storage. (Same address as in internal GOTO.) If an END of DO loop was processed during the object of the 'THEN' processing then return via special EXIT to close the DO loop.

9 Call GETALINE to obtain the next input buffer (allows nesting of IF THEN... ELSE relationships properly).

10 Call Record Scan routine (IKJCT432) to process record obtained in 9 if not end of file.

11 Return via the Special Exit because no move is necessary with GETALINE return code.

Diagram 5.3.2. ELSE Routine (IKJCT432) (Part 1 of 2)

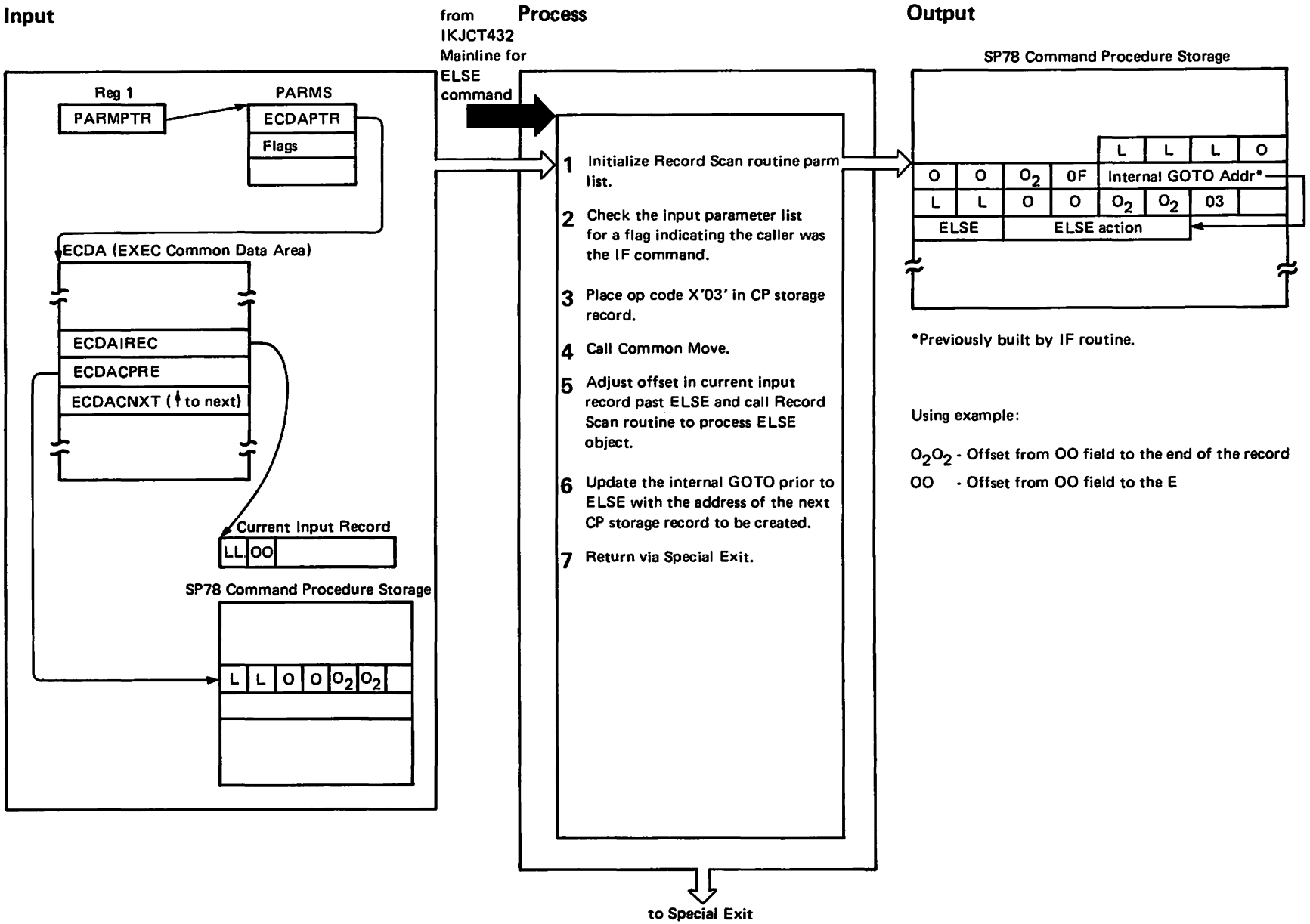


Diagram 5.3.2. ELSE Routine (IKJCT432) (Part 2 of 2)

1 Save input pointer to the CDAPTR and flag area.

2 Make sure previous CP Storage record is an internal GOTO with an address the same as that of the next command procedure record to be created. Save address of the internal GOTO for later update; if this is not true then the user has a syntax error; notify the user and set the PROC not executable switch-continue command syntax check at Step 5.

3 Place the op code X'03' in the current CP storage record, update O2 offset and LL of current CP record. (Preserve the original input buffer address and length for correct freeing of input buffer, later.)

4 Call the Common Move routine to move ELSE to CP storage. If the Move return code is non-zero, then return with Move RC via Step 7.

5 Update the offset in the current input record past the word ELSE and call the Record Scan routine to process the object of the ELSE. If RSR return code non-zero return with RSR RC via Step 7.

6 Update the address in the internal GOTO prior to the ELSE to point to the next CP storage record to be created.

7 Return via Special Exit.

Diagram 5.3.3. DO Routine (IKJCT432) (Part 1 of 2)

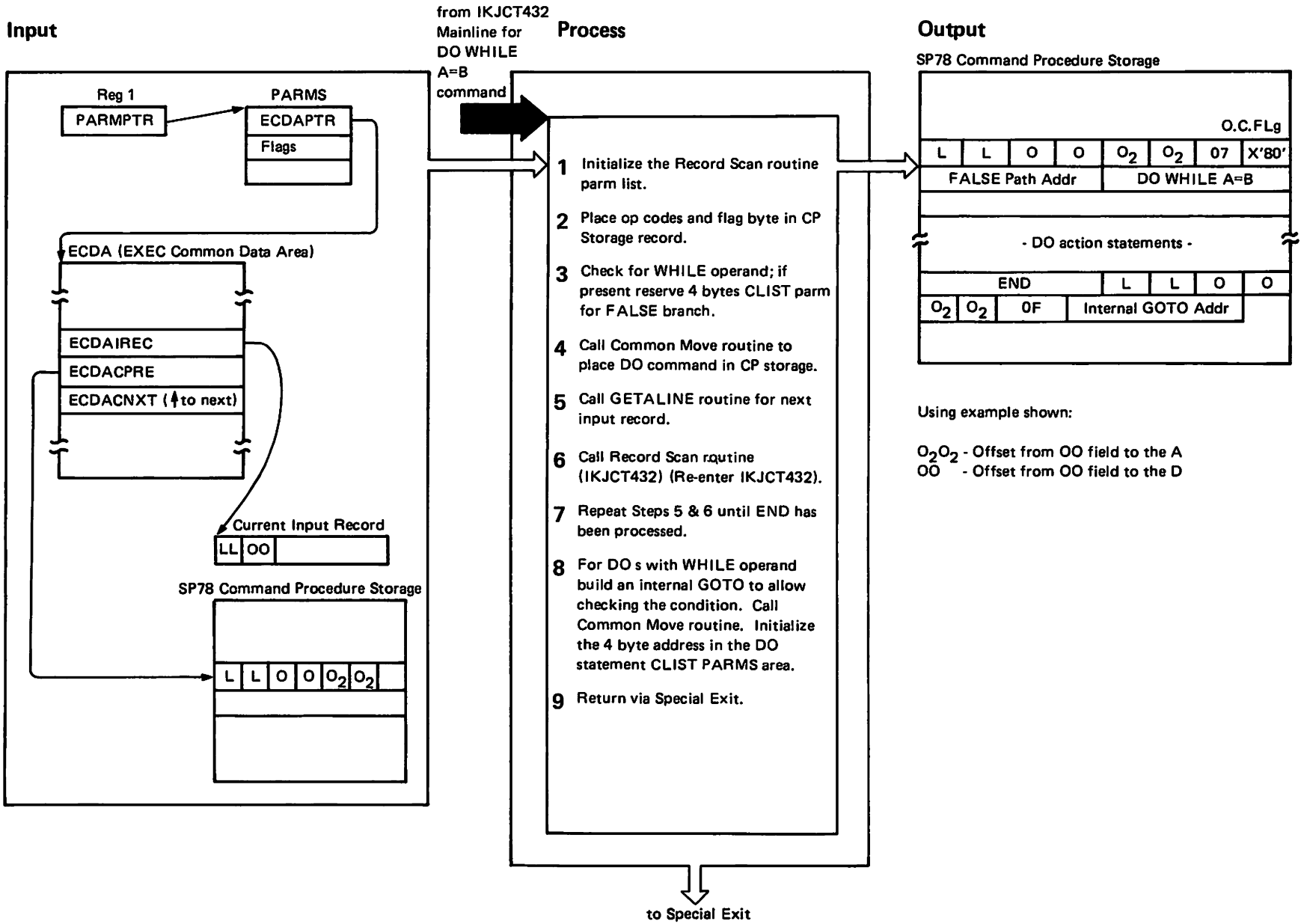


Diagram 5.3.3. DO Routine (IKJCT432) (Part 2 of 2)

1 Save pointer to the CDAPTR and flag area.

2 Place op code 07 and flag byte X'00' in the CLIST parm area. Update DO Count. Update 0202 offset.

3 Determine if the WHILE operand was specified. If so reserve 4 bytes in the CLIST parm area and set flag byte X'80'. This will be the address of FALSE path. Adjust LL of current CP record. If there was no WHILE operand leave flag type X'00'.

4 Call the Common Move routine to place the DO statement in the CP storage area. Respecify SP (78) command base after move in case record would not fit previous block. If return code from Move non-zero then return with Move RC via Step 9.

5 Call the GETALINE routine to get the next record from the input data set. If end of file has occurred then notify user of the open DO group, set PROC not executable. For end of file and other errors return with the GETLINE error via Step 9.

6 Initialize a flag area and CDAPTR parm to be used as input to the Record Scan routine and call Record Scan routine. If the return code from RSR is non-zero notify the user of the open DO group and return with RSR RC via Step 7.

7 Repeat Steps 5 and 6 until an END CLIST statement has been processed (this will be denoted by conditions in the flag area).

8 If the DO statement had a WHILE operand then build an internal GOTO with the address initialized to that of the original DO statement. Call the Common Move routine to place internal GOTO into CP storage. Update the FALSE address in the original DO statement to the address of the next record to be created.

9 Return via Special Exit.

Diagram 5.3.4. ERROR/ATTN Routine (IKJCT432) (Part 1 of 2)

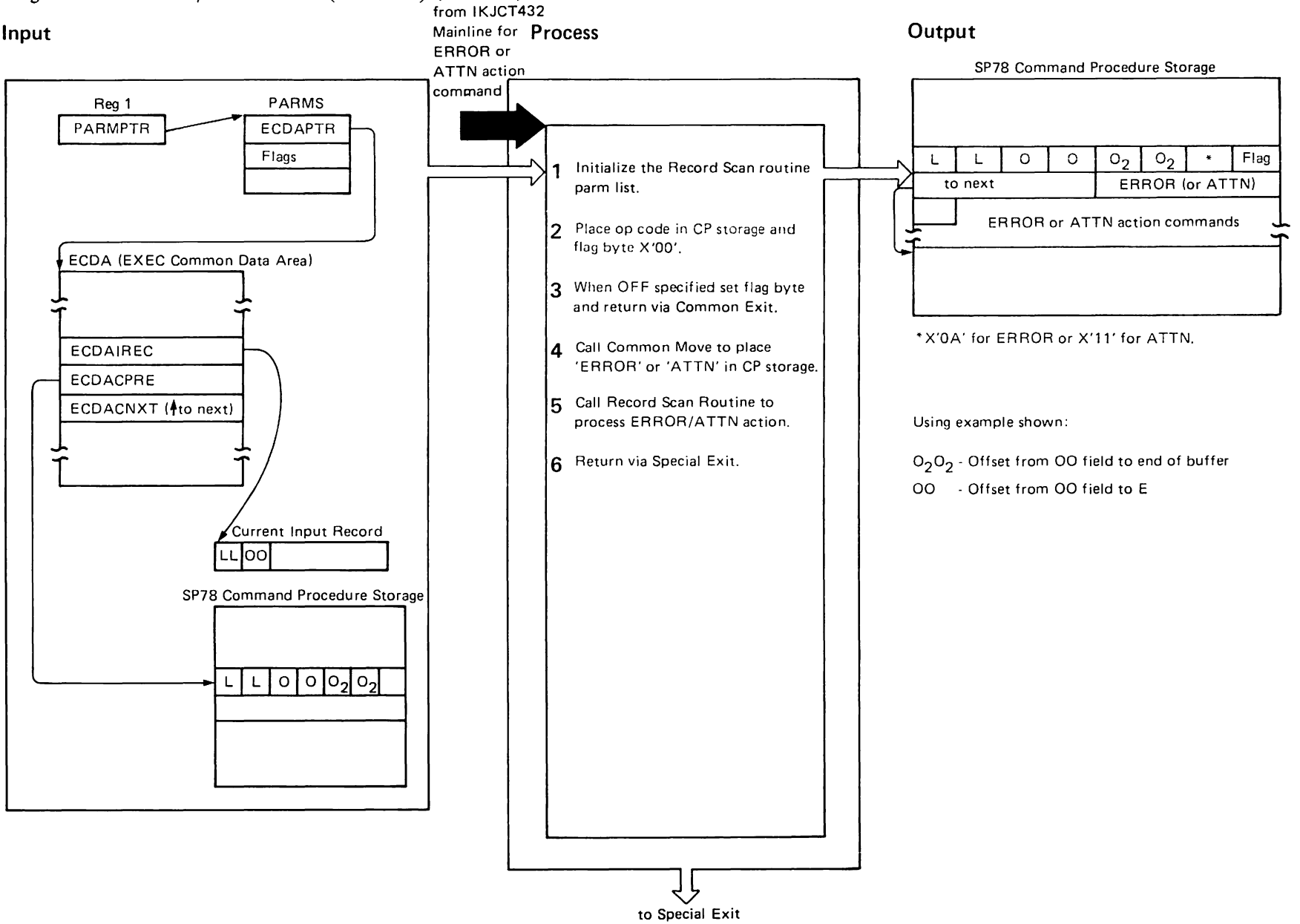


Diagram 5.3.4. ERROR/ATTN Routine (IKJCT432) (Part 2 of 2)

1 Save address of input parms – CDAPTR and flags.

2 Initialize a flag byte to zero; X'00'. Adjust LL and O2O2 of current CP record.

3 Determine if OFF was specified. If OFF was requested then set flag byte to X'80' and return via Common Exit. If no operands specified set flag byte to X'40' and return via Common Exit.

4 Reserve 4 bytes for the End of Action address. Adjust LL and O2O2 of current CP record. CALL the Common Move routine to place 'ERROR' in the CP storage. If Move return code non-zero then return with Move RC via Step 6.

5 Call Record Scan routine to process the action following 'ERROR'. Update the End of Action address in the CLIST Parm Area. If RSR return code non-zero, then return with RSR RC via Step 6.

6 Return via Special Exit.

Diagram 5.3.5. END Routine (IKJCT432) (Part 1 of 2)

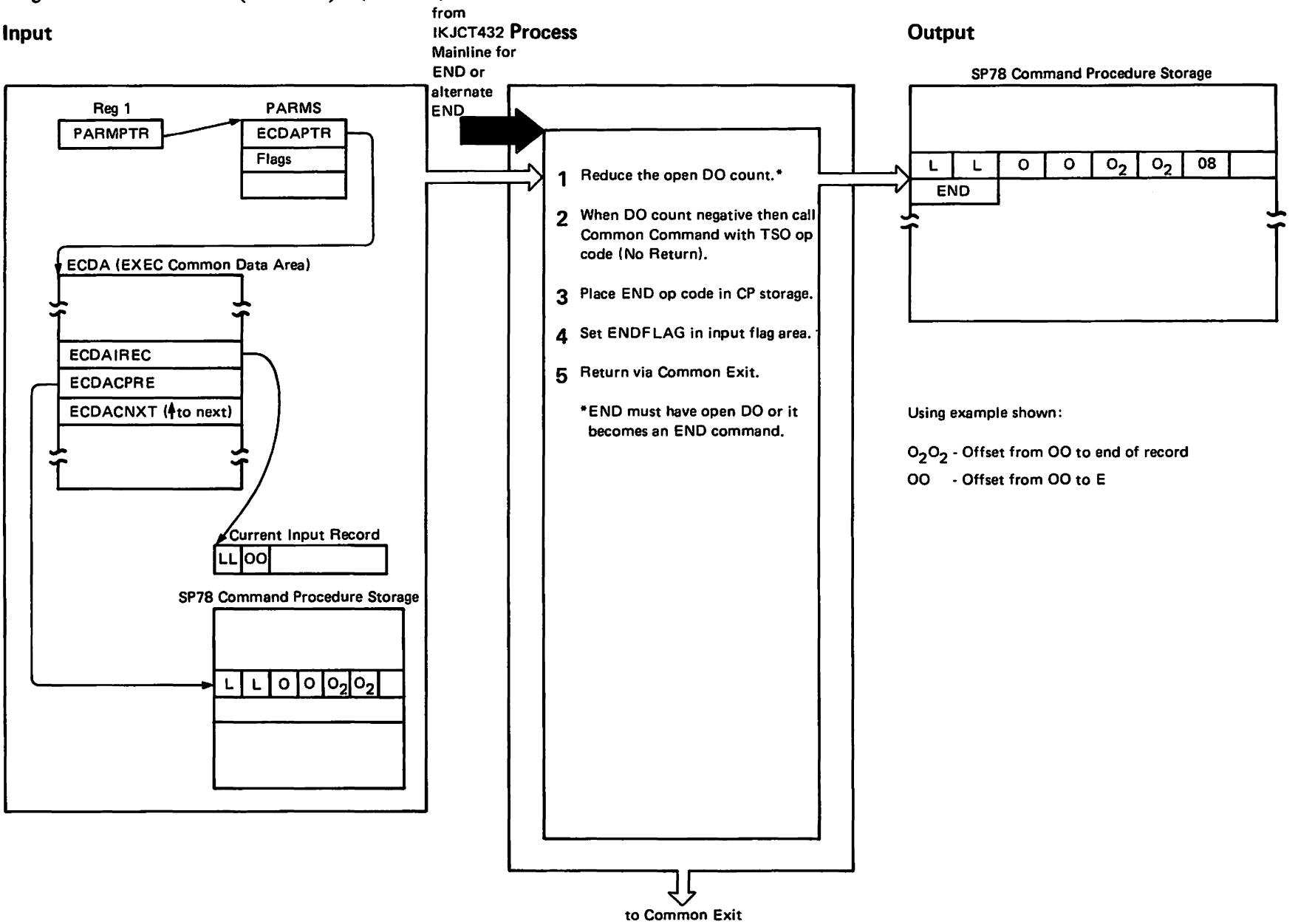


Diagram 5.3.5. END Routine (IKJCT432) (Part 2 of 2)

1 Lower the open DO count by ONE.

2 If the DO count goes negative then there was no matching DO. If this was an alternate end specification (ECDAEND field other than 'END') then it is an error; notify the user and set the PROC not executable switch. Otherwise, zero the DO count then treat as if it was a TSO command by invoking the COMCMD routine with the TSO op code X'00'. COMCMD routine will return via Common Exit.

3 Set END Flag in input parm flag area. Set op code to X'08' and initialize O2O2 and LL.

4 Return via Common Exit.

Diagram 5.3.6. SET Routine (IKJCT432) (Part 1 of 2)

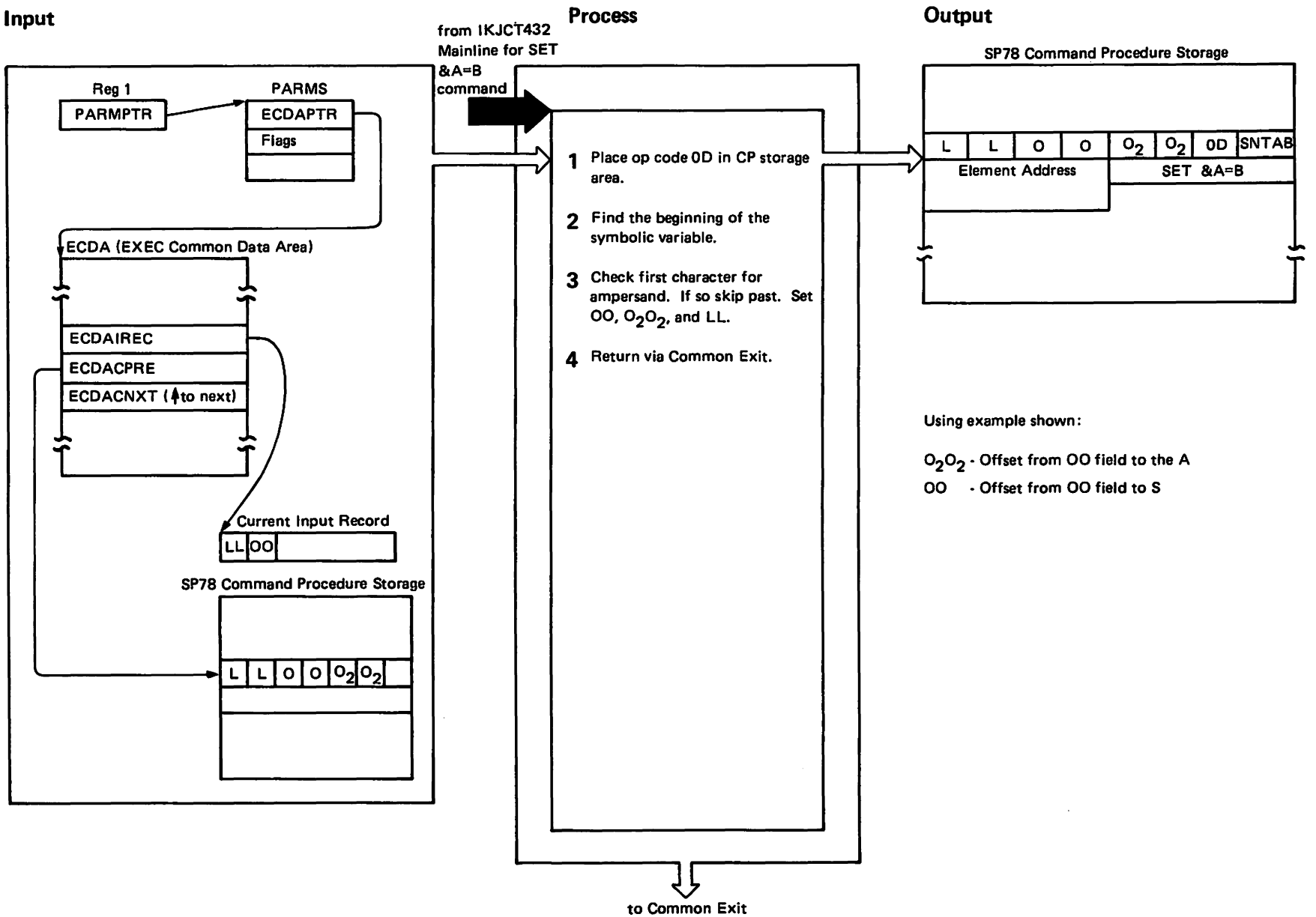


Diagram 5.3.6. SET Routine (IKJCT432) (Part 2 of 2)

1 Place op code '0D' in CP storage area.

2 Locate the start of symbolic variable. If none exist, issue a message and return to Special Exit with return code.

3 If the first character is an ampersand, skip past. Set OO, 0202, and LL.

4 Return via Common Exit.

Diagram 5.3.7. CONTROL Routine (IKJCT432) (Part 1 of 2)

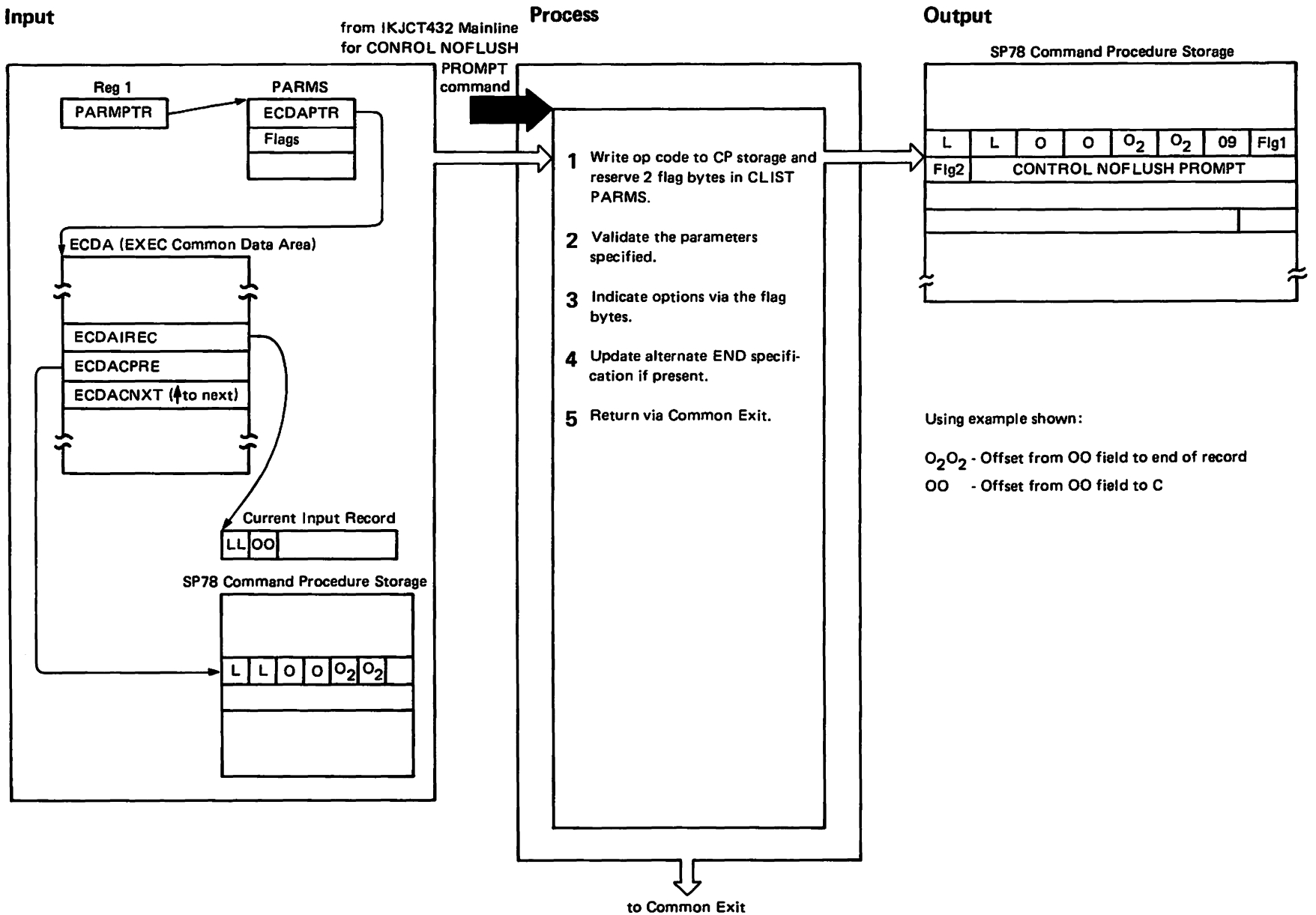


Diagram 5.3.7. CONTROL Routine (IKJCT432) (Part 2 of 2)

1 Place op code X'09' in the CP storage area. Reserve 2 bytes for flag options. Update the 0202 and LL fields.

2 Validate the parameters specified. Syntax checking will not invoke any prompting situations. If a parameter is incorrect, notify the user and set the "PROC not executable" switch and continue syntax check of operands. The lowest unique number of characters are allowed for each keyword.

3 Indicate the options found on the CONTROL statement by setting or resetting bits in flag bytes.

4 If an alternate ending sequence was specified then update the alternate END field and length in the Common Data Area.

5 Return via Common Exit.

Diagram 5.3.8. READ/READDVAL/GLOBAL Routine (IKJCT432) (Part 1 of 2)

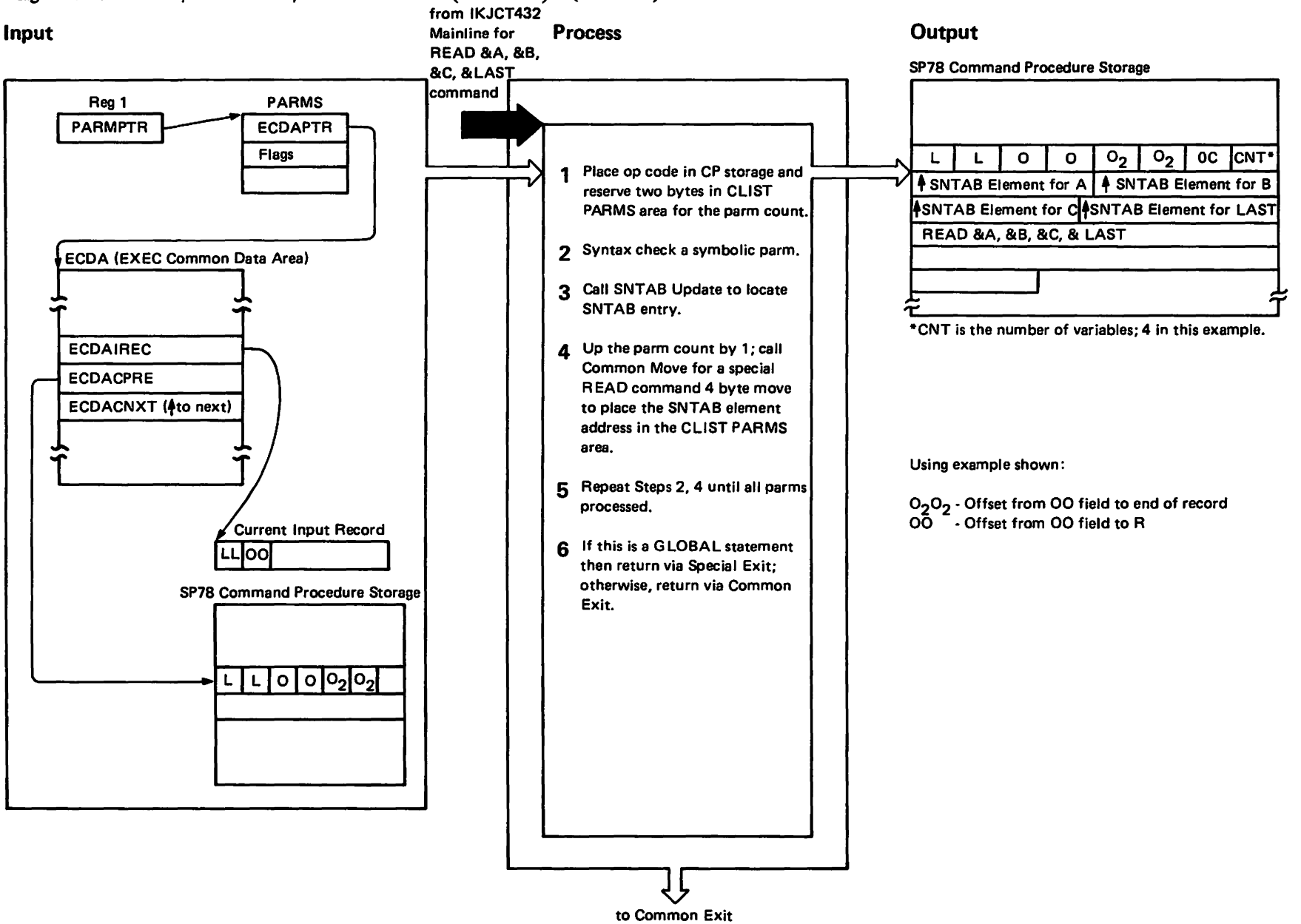


Diagram 5.3.8. READ/READDVAL/GLOBAL Routine (IKJCT432) (Part 2 of 2)

1 If command is GLOBAL proceed to Step 2. Place op code in CP storage record and reserve two bytes for a count field set to X'0000'.

2 Syntax check a Symbolic Parameter Specification. Determine name and length. If syntax incorrect notify user, set PROC not executable and continue syntax check of any more parameters.

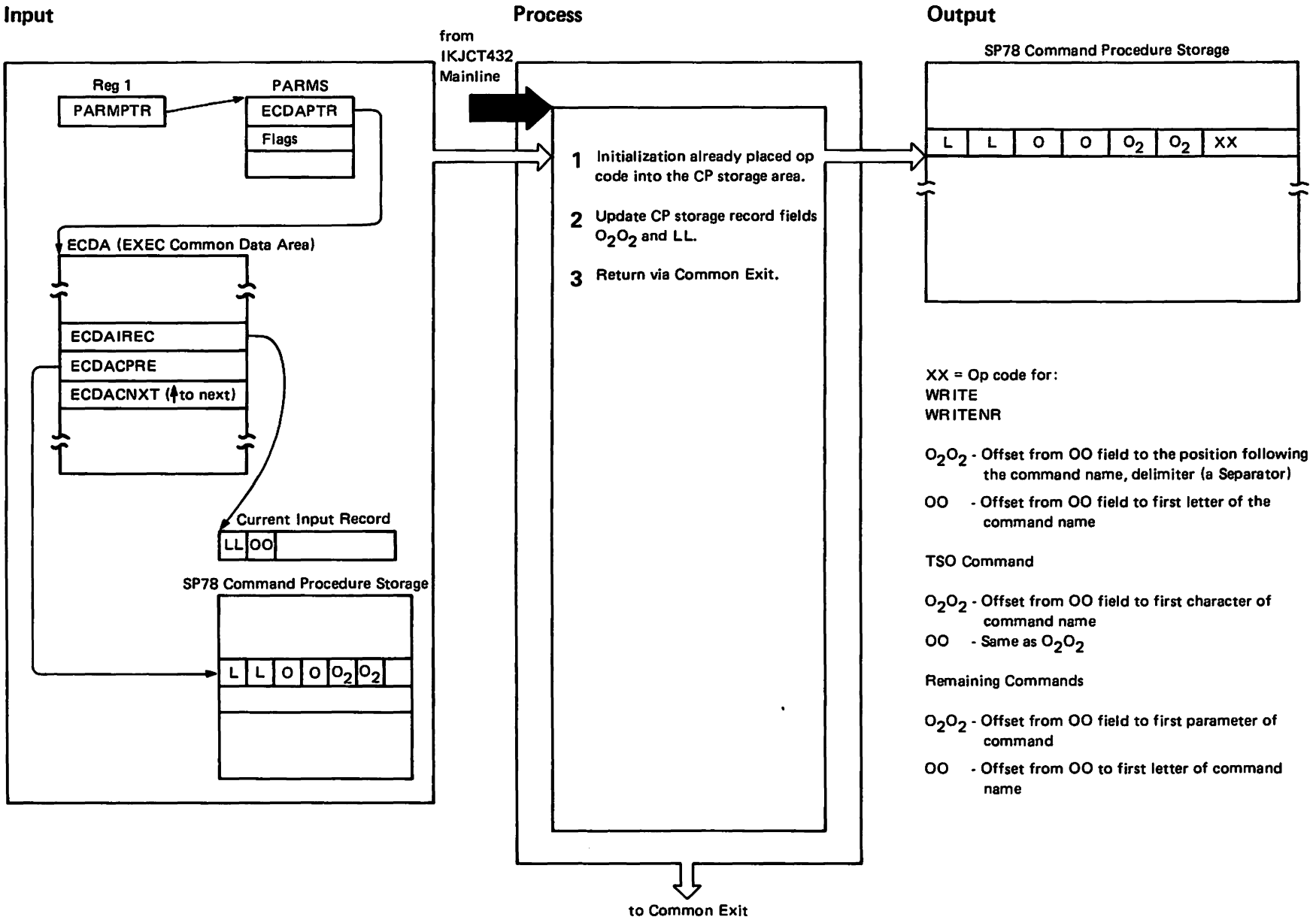
3 Call SNTAB Update routine to locate or create an element in SNTAB. If command is GLOBAL, proceed to Step 2 to process next parameter name; if no further parameters, return via Special Exit. Call Common Move to place address of element in CLIST Parm by indicating call by READ command. If the return code from the SNTAB Update routine or Move routine is non-zero then return with SNTAB Update or Move RC via Step 6.

4 Add one to the parm count. Update O2O2 and LL fields.

5 Repeat Steps 2-4 until all symbolic parameters have been syntax checked or count exceeds 256. If count exceeds 256, notify user that PROC is not executable and return.

6 If the command length was eight, then set the op code to 10. Return via Common Exit.

Diagram 5.3.9. Common Command Routine (IKJCT432) (Part 1 of 2)



XX = Op code for:
WRITE
WRITENR

O_2O_2 - Offset from OO field to the position following the command name, delimiter (a Separator)

OO - Offset from OO field to first letter of the command name

TSO Command

O_2O_2 - Offset from OO field to first character of command name

OO - Same as O_2O_2

Remaining Commands

O_2O_2 - Offset from OO field to first parameter of command

OO - Offset from OO to first letter of command name

Diagram 5.3.9. Common Command Routine (IKJCT432) (Part 2 of 2)

1 Initialization already moved the op code to the current CP record.

2 Update the OO, O2O2 and the LL fields in the CP record as follows:

- TSO commands: O2O2 = offset to first position of command name; OO = same as O2O2.
- WRITE(NR) commands: O2O2 = offset to separator following last character of command name; OO = offset from OO field to the first letter of the command name.
- All other: O2O2 = offset to first operand following the command name; OO = offset from OO to first letter of command name.

3 Return via Common Exit.

Diagram 5.3.10. DATA to ENDDATA Routine (IKJCT432) (Part 1 of 2)

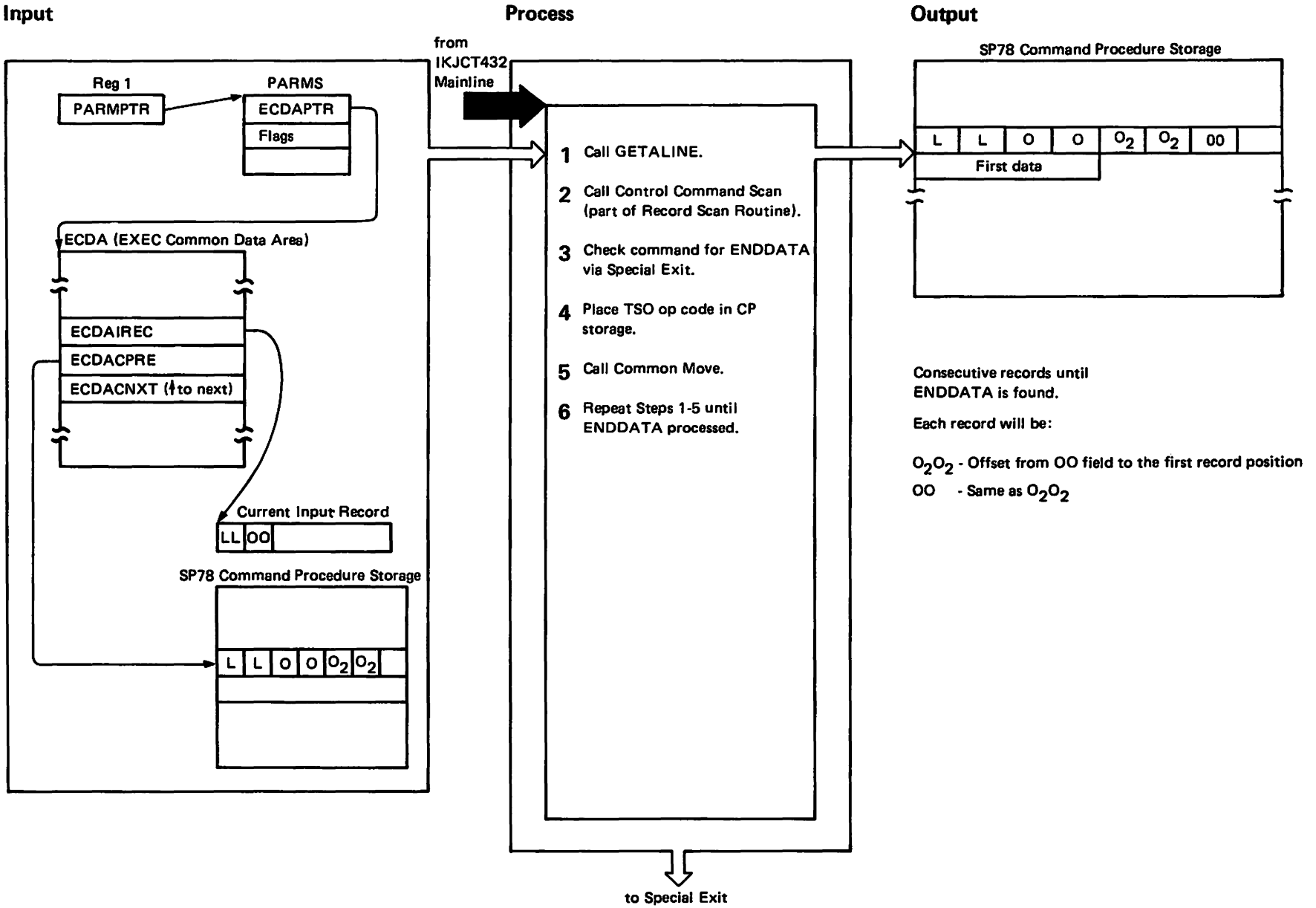


Diagram 5.3.10. DATA to ENDDATA Routine (IKJCT432) (Part 2 of 2)

1 GETALINE

1. Issue a FREEMAIN for the previous input record.
2. Issue a GETLINE for the next input record. If end of data occurs then return with Return Code = 4. Other errors, notify user and return with Return Code = 20.
3. Update the current input record pointer in the Common Data Area (CDA).
4. Return to caller.

2 Control Command Scan

1. Skip separators.
2. Validity check the first character for alphabetic.

   If the first character is an ampersand or a percent sign it will be valid; assume this is a TSO command.

   If the first character is not alphabetic, then assume this is a TSO command.
3. Find the end of parameter by scanning to next non-alphameric.
4. If the non-alphameric was a colon and the routine has not already processed a label, then update the label output area; length of label can not exceed 8 characters. After updating the label output area return to Step 1 to syntax check the command name specification. If routine has already processed a label or the length was greater than 8 characters then assume this is a TSO command.
5. If the non-alphameric is not a colon, this is the end of the command name. Validity check command length and update the command output area. If length not 8 or less then assume this is a TSO command.
6. Return.

3 ENDDATA Routine

The ENDDATA routine should never get control unless there is a missing data statement. ENDDATA routine should notify user of error, set PROC not executable and return with RC=0.

4 Call SNTAB Update routine to put TSO op code in CP storage.

5 CP Common Move Routine

1. Locate the current offset into the current CP storage record by adding LL at base of record. If PROC not executable switch is on then return.
2. Determine if the amount of storage remaining in the command procedure area is large enough to contain the data to be moved. If storage remaining in the command procedure is insufficient then we must get an additional block of Subpool 78 storage for the command procedure.
   - Issue a GETMAIN (SP78) for at least 2K* block of storage for the command procedure area. Initialize the 3 word header in beginning of new Command Procedure Storage Block. If GETMAIN fails then notify user, set PROC not executable and return RC=16.

   *Or for an amount of storage that will contain the CP header area, the current record, and an internal GOTO (max prefix).

   - Move the current prefix area from the old block of storage to the new block of storage (from the LL to a length of LL).
   - Free any excess storage over 8 bytes in the old block.
3. Move the record to the CP storage block. The address of the data to be moved and its length will be in the Common Data Area. Placement of data in the command procedure area will start at the current value of LL into the current command procedure record prefix (ECDACNXT address).
4. Then update the command procedure record LL field to the new length. Update the available storage size in the command procedure block header.
5. If this was a call by the READ Command then return; otherwise, update the previous and next record pointers in the CDA and return.

6 Repeat steps 1 - 5 until ENDDATA is processed.

Diagram 6. FREE Command Processor (Part 1 of 2)

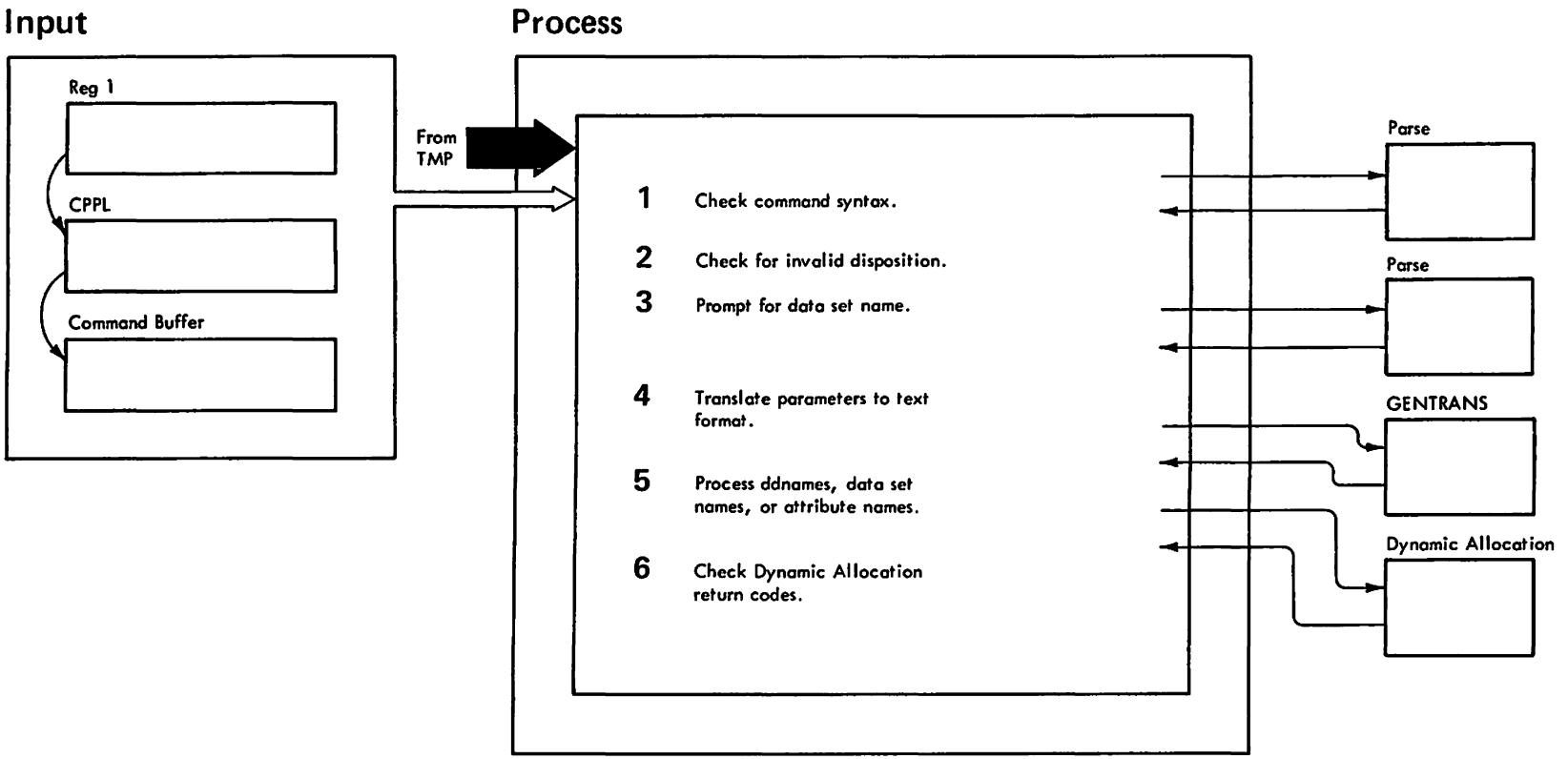


Diagram 6. FREE Command Processor (Part 2 of 2)

1 Use Parse to syntax check the command. Check the Parse return code; if it is non-zero return to the TMP.

2 Check to see if DEST, HOLD, or NOHOLD was specified with a data set disposition of KEEP, DELETE, CATALOG, or UNCATALOG. If yes, issue an error message and return to the TMP.

3 Check to see if a data set name, file name, or attribute list name was entered. If no, pass control to Parse and prompt the user for a data set name. When prompting is complete, overlay the original PDE with the new PDE from prompt.

4 Use GENTRANS to translate the parameters to text format. The pointer to the text unit is returned from GENTRANS in the IKJZB831 parameter list. If the return code is non-zero, return control to the TMP.

5 Use the unallocate function of Dynamic Allocation to unallocate files, data sets, or attribute lists.

6 Check the Dynamic Allocation return code and information reason code.

- If both codes are zero, a file, data set, or attribute list was unallocated.
- If either code was non-zero, unable to unallocate. Use the DAIR failure message routine IKJEFF18 to analyze the return code and send the appropriate error message to the user.

Control is returned to the TMP.

Object Module: IKJEFD20

Diagram 7.1. HELP Processing (Part 1 of 2)

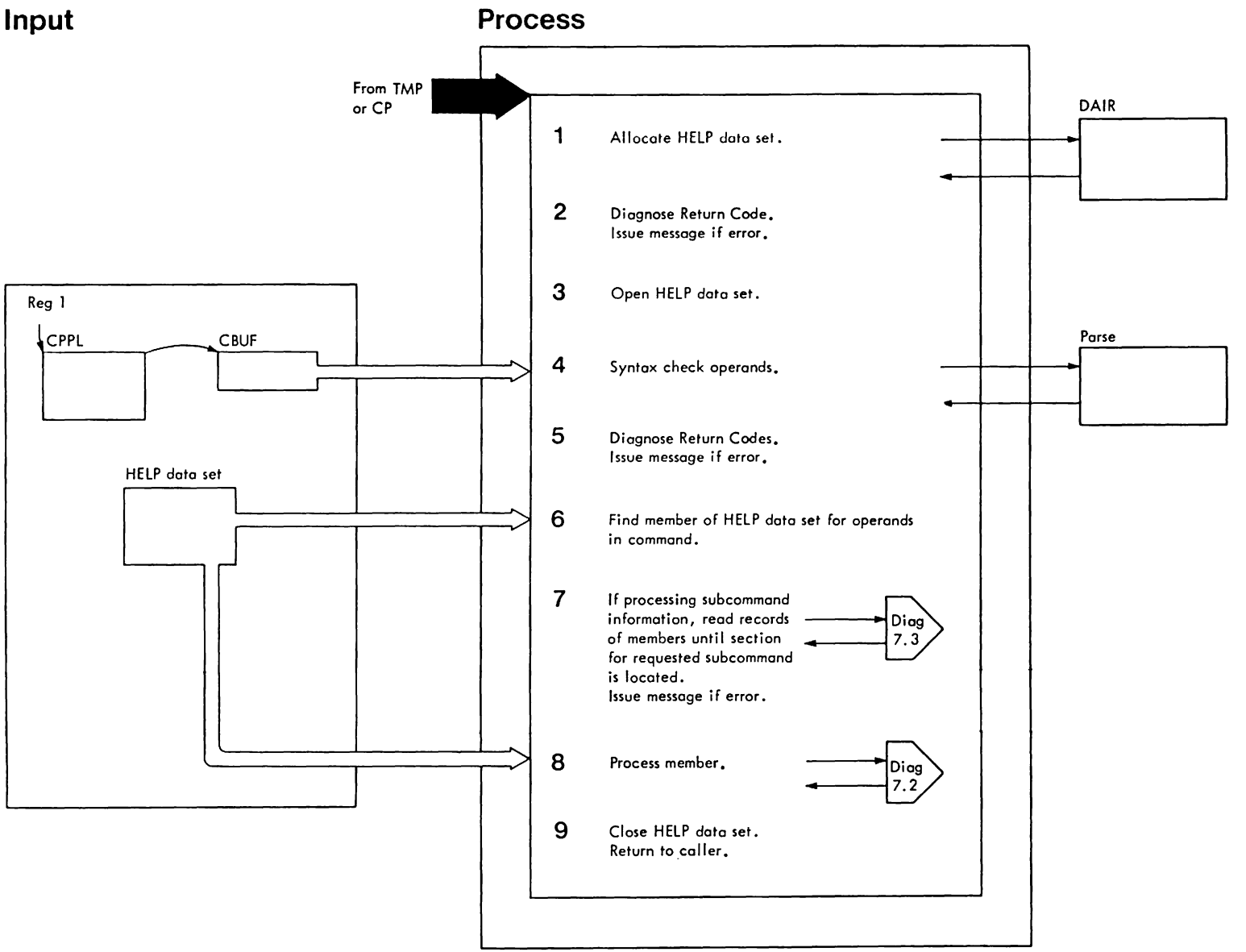


Diagram 7.1. HELP Processing (Part 2 of 2)

Extended Description

1 Using DAIR, (The Dynamic Allocation Interface Routine), allocate the HELP data set.

2 Check return code:
- a) If non-zero, use IKJEFF18 (DAIRFAIL) to diagnose error and send message to user, return to caller.
- b) If zero and DSORG is not PO (partitioned), issue message to user, return to caller.

3 Open HELP data set. If Open fails, issue message and return to caller.

4 Use Parse to check syntax of command.

5 If Parse was unsuccessful, issue messages and return to caller.

6 FIND member of HELP data set.
- a) If not in subcommand mode (command attached by the TMP is HELP or H), and HELP entered with no operands, find member 'COMMANDS'.
- b) If not in subcommand mode and HELP entered with operands, use first operand as member name.
- c) If in subcommand mode and HELP is first operand, find member 'HELP'.
- d) If in subcommand mode and HELP is not first operand, use the name of the name of the command attached by TMP (from ECT) as the member name.

7 If case d) of step 6, read record from member of HELP data set. See Diagram 7.3. Search each record for subcommandname indicator '=' and then for subcommandname requested on command. Keep reading records until end-of-file (error) or subcommand name found.

8 Process the HELP data set member. See Diagram 7.2.

9 Close the HELP data set. Return to caller.

Object Module: IKJEFH01

Diagram 7.2. Processing HELP Data Set Member (Part 1 of 2)

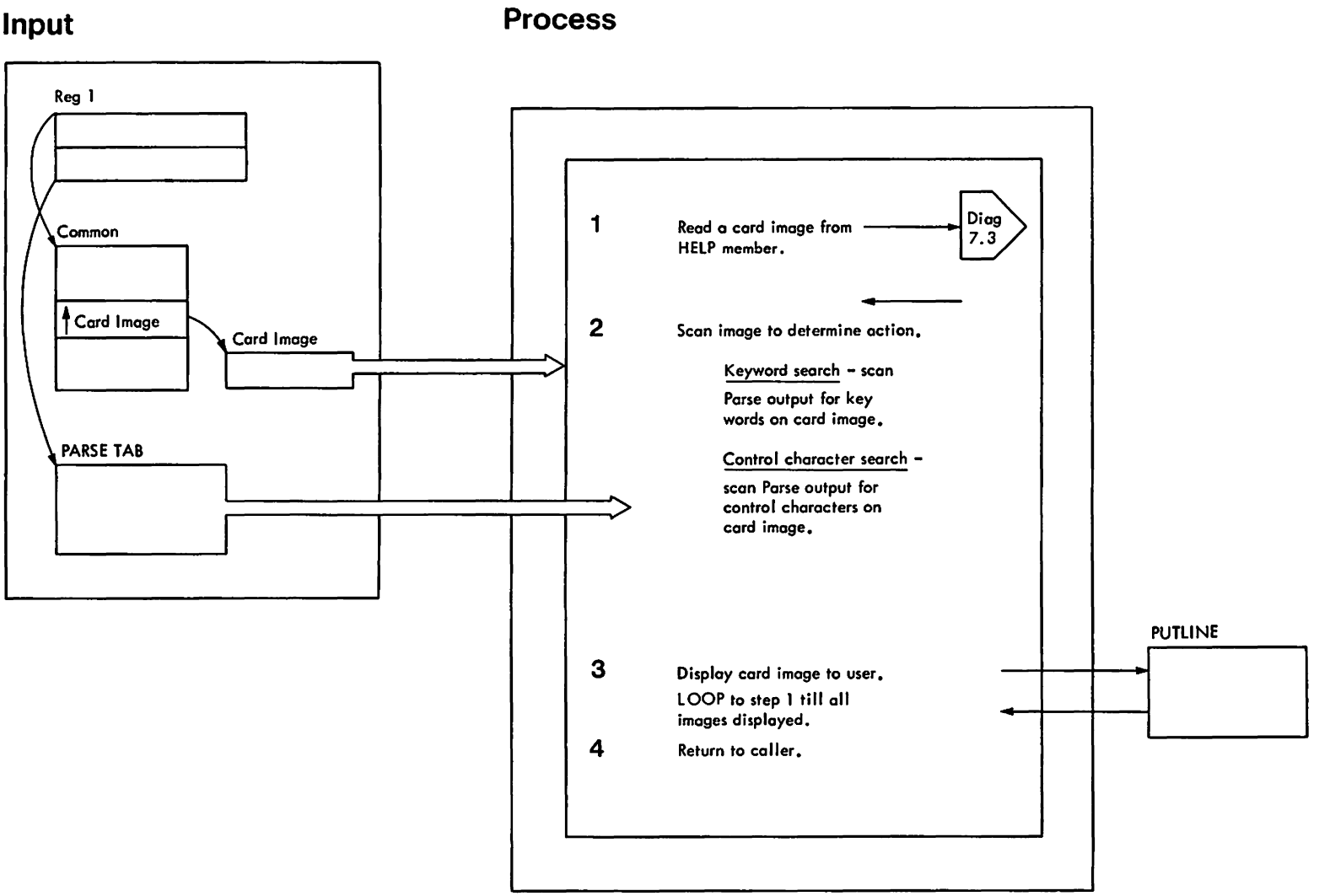


Diagram 7.2. Processing HELP Data Set Member (Part 2 of 2)

Extended Description

1 Obtain a card image from the HELP data set member. See Diagram 7.3.

2 Parse output calls for either keyword or control character information. Scan card to find match for Parse output.

3 If scan finds match, use PUTLINE via IKJEFF02 to display card image to user.

4 If all information has been displayed, return to caller, else process step 1 again.

Object Module: IKJEFH02

Diagram 7.3. Reading HELP Data Set (Part 1 of 2)

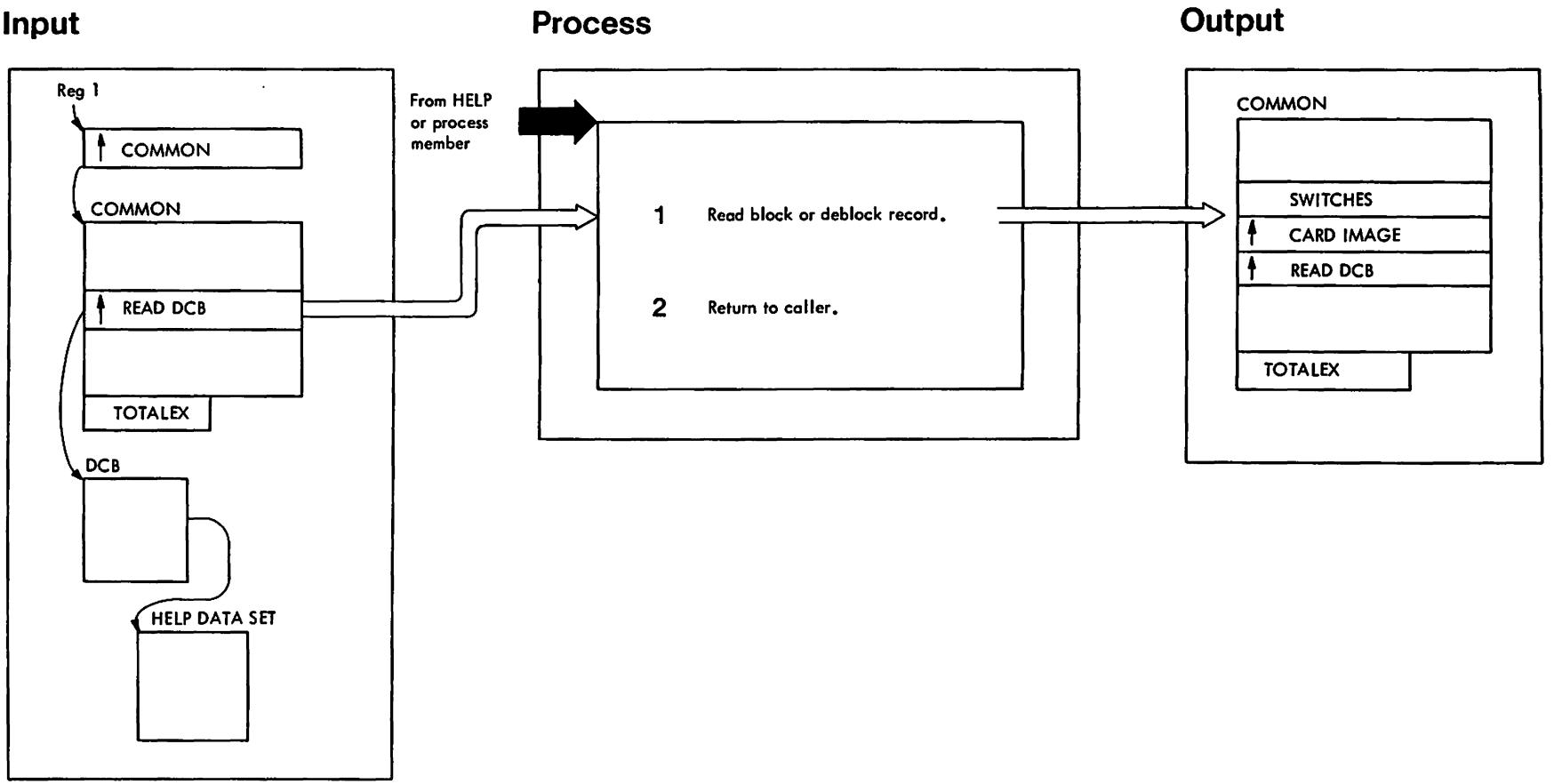


Diagram 7.3. Reading HELP Data Set (Part 2 of 2)

Extended Description

1 The HELP data set is blocked. Read a block and deblock a record or deblock a card image record. In case of I/O error or end of data, set switches.

2 Return to caller.

Object Module: IKJEFH03

Diagram 8. LINK and LOADGO Processing (Part 1 of 2)

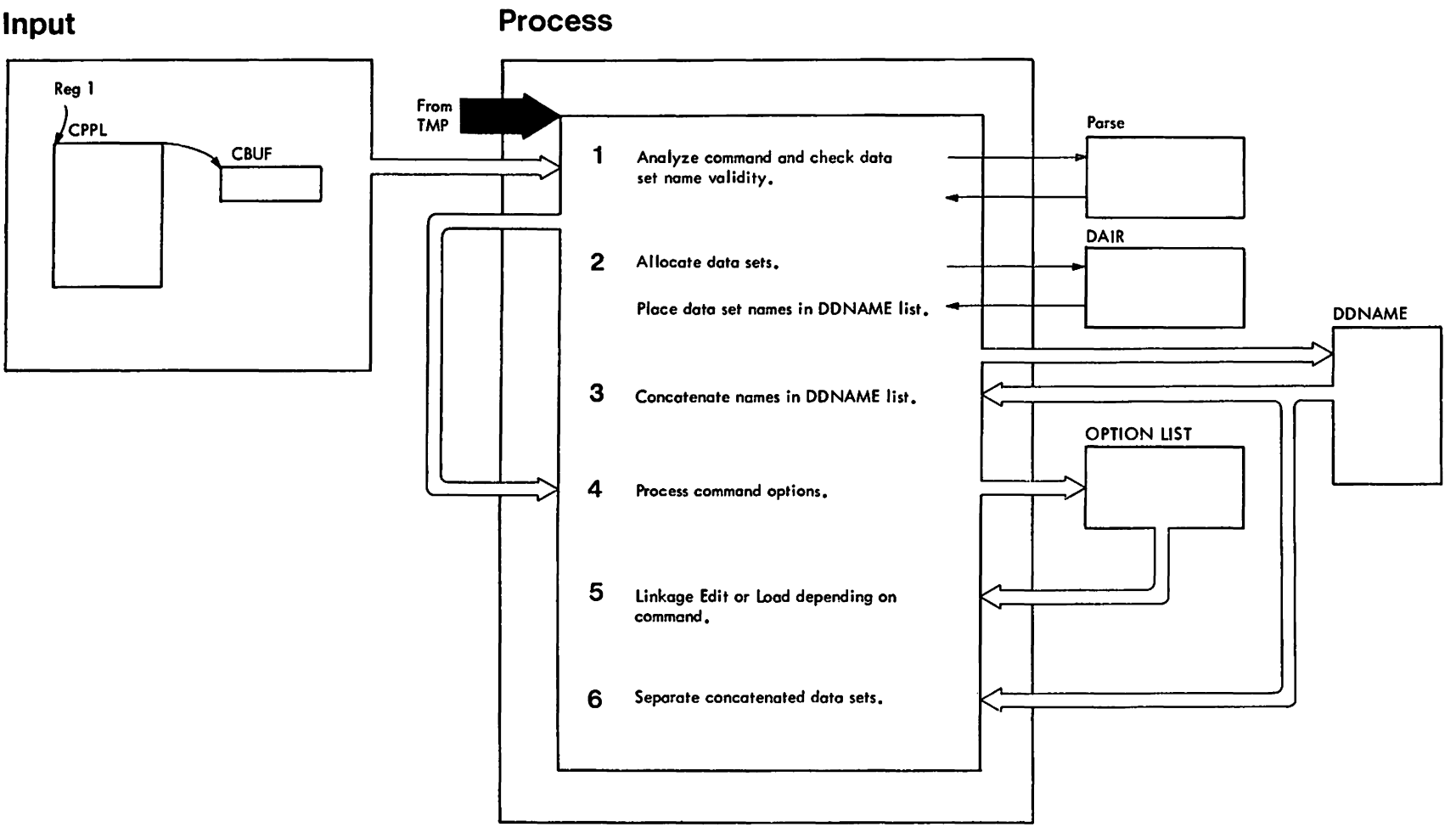


Diagram 8. LINK and LOADGO Processing (Part 2 of 2)

Extended Description

1 Use Parse to analyze syntax of commands. Check data set names for valid qualifiers. Set LKLD to indicate whether command is LINK or LOADGO.

2 Use DAIR to allocate data sets. Place each data set name in DDNAME list.

3 Use DAIR to concatenate ddnames in ddname list. (DDNMS)

4 Using Parse output, process command options. Prompt for missing operands, set defaults. Place results in option list. (OPLEN)

Note that the LINK and LOADGO commands have been updated with AMODE and RMODE options to support the MVS/XA environment.

5 Link to either the Linkage Editor or the Loader depending on LKLD switch passing the option list and ddname list through OUTPARM.

6 On return, separate concatenated data sets and return to the TMP.

Page of SY28-0652-4
As Updated March 1, 1985
By Suppl. SD23-0299 for 5665-291, 5740-XC6

Diagram 9.1. LISTALC Processing Overview (Part 1 of 2)

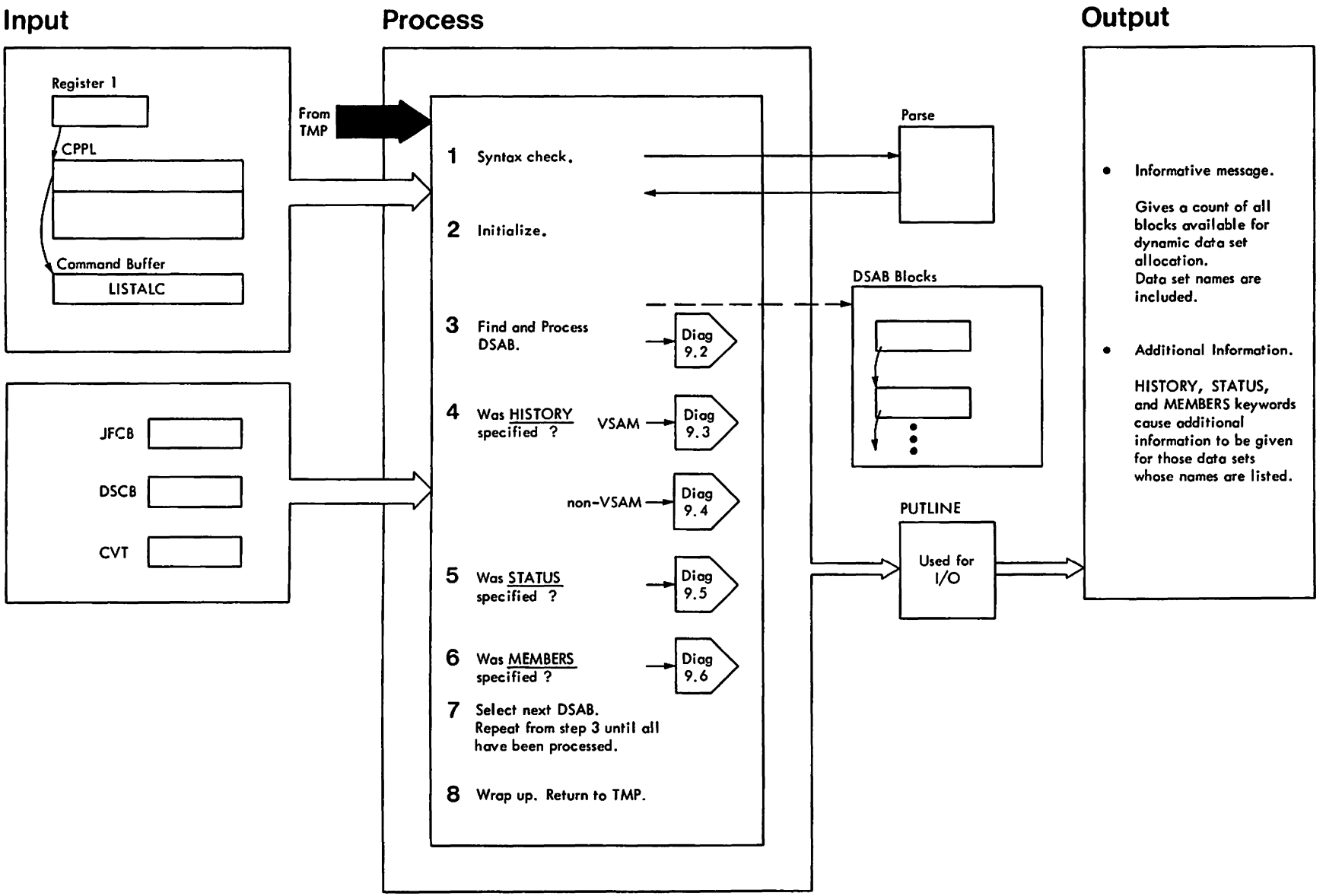


Diagram 9.1. LISTALC Processing Overview (Part 2 of 2)

1 The Parse routine syntax checks the command. Upon return, the Parse return code is checked.
Possible messages: IKJ58304I, IKJ58305I

2 Set option byte to reflect options selected by user. If HISTORY, MEMBERS, or SYSNAMES were specified, get workarea and place address in OBTWA. Store JFCB work area address. Store DCB address.
Possible message: IKJ58303I

3 Obtain a pointer to the Data Set Attribute Block DSAB chain through SVC99. After the DSAB is located, check for HISTORY and STATUS and print applicable headings. Then check the DSAB to see if it is available for allocation. The DSAB is considered available if the data set is not in use and not permanently allocated. This condition is indicated on the output line by an asterisk (*) preceding the data set name.

4 After basic processing of a DSAB, check to see if HISTORY was requested. If yes, process HISTORY information. See Diagram 9.3 (VSAM) or 9.4 (Non-VSAM).

5 If STATUS was specified, process STATUS information, see Diagram 9.5.

6 Write HISTORY and/or STATUS information, if applicable.
Possible messages: IKJ58301I, IKJ58300I

7 After all processing of the DSAB is complete, process the next DSAB. If no DSABs remain to be processed, return control to the TMP.

Object Module: IKJEHAL1

Diagram 9.2. LISTALC DSAB Processing (Part 1 of 2)

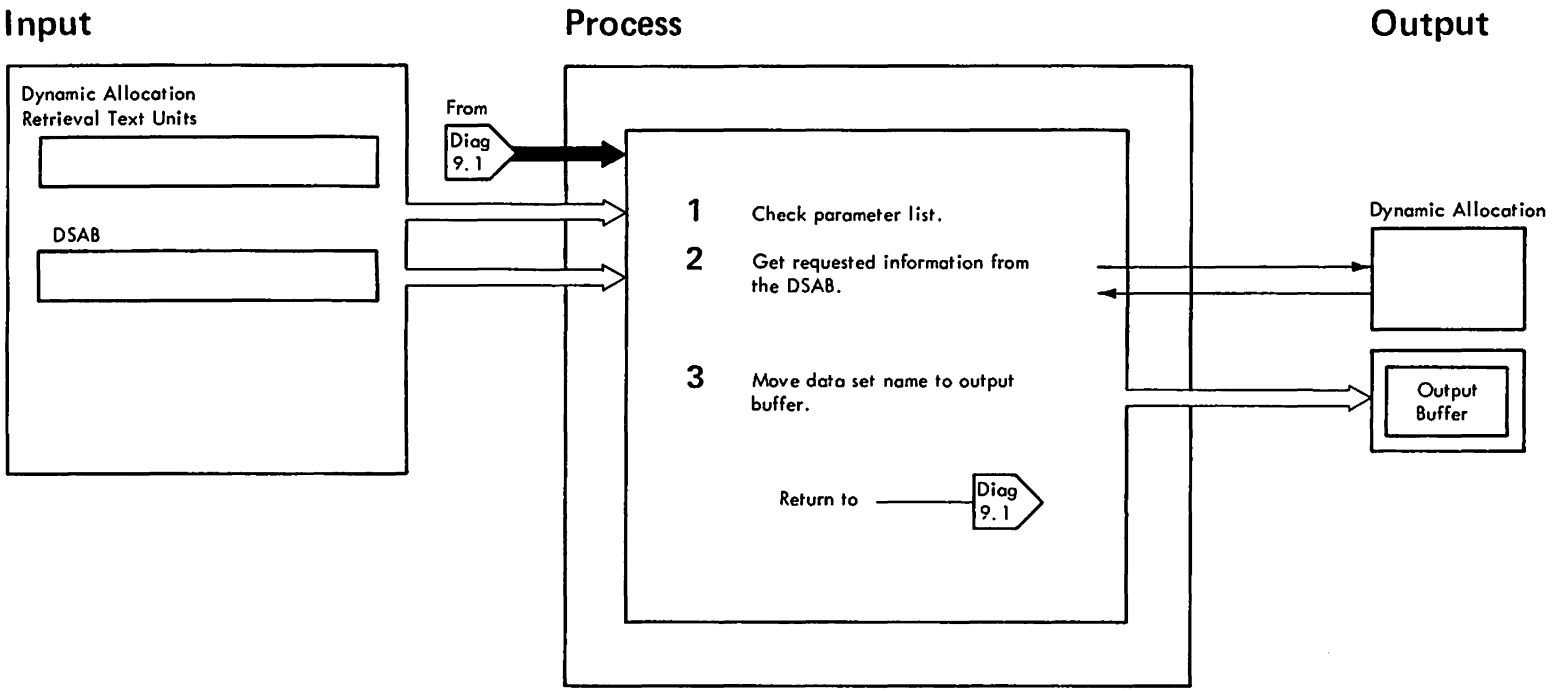


Diagram 9.2. LISTALC DSAB Processing (Part 2 of 2)

1 Check the dynamic allocation parameter list (IEFZB4D0) to see if it has been initialized for use by Dynamic Allocation. If it is initialized continue processing. If not, build dynamic allocation text units describing the data to be returned about each allocated data set.

2 Use Dynamic Allocation to get information requested by the text units from the DSAB.

3 Get data set name from the DSAB and move it to the output buffer. If a data set name is not available, put the appropriate message in the output buffer.

Object Module: IKJEHAL1

Diagram 9.3. LISTALC HISTORY Processing (VSAM) (Part 1 of 2)

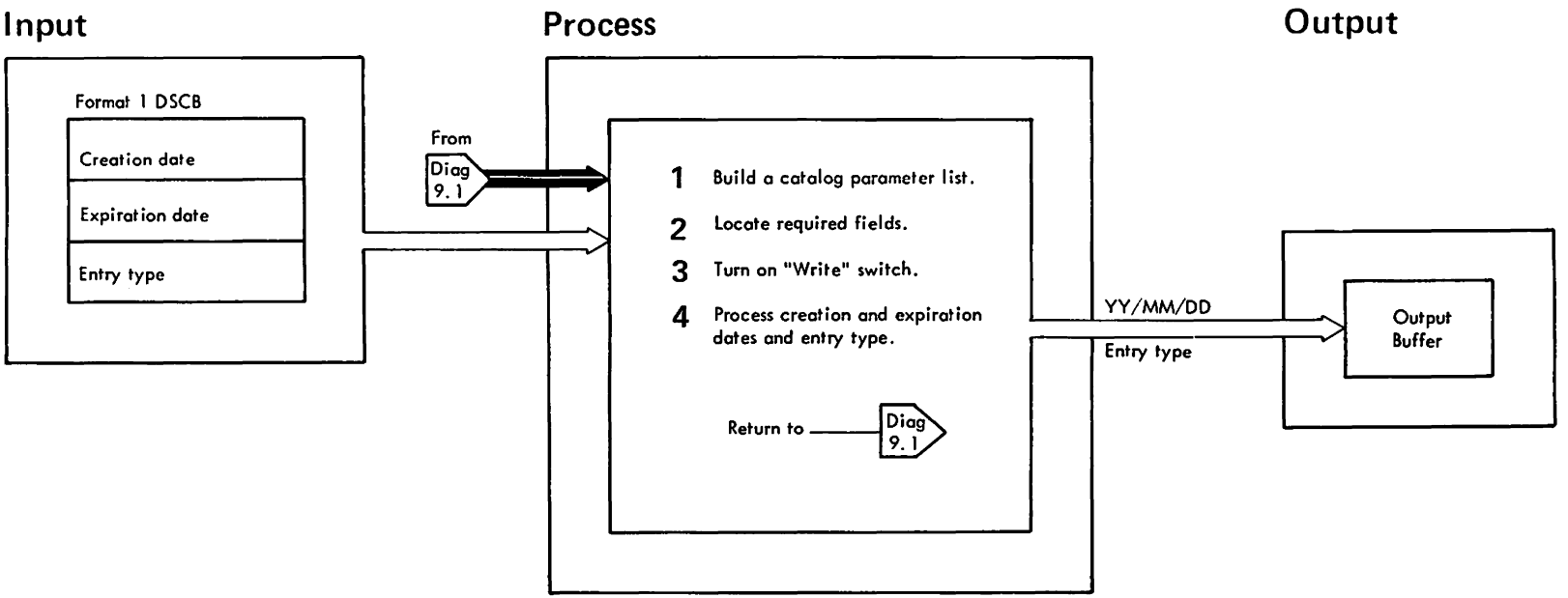


Diagram 9.3. LISTALC HISTORY Processing (VSAM) (Part 2 of 2)

1 Build a catalog parameter list using information and the data set name obtained from the DSAB by SVC99. The parameter list specifies the named data set to be retrieved from the VSAM catalog, the entry type (indicates VSAM or non-VSAM data set) and the expiration and creation dates for the data set are to be returned in the work area.

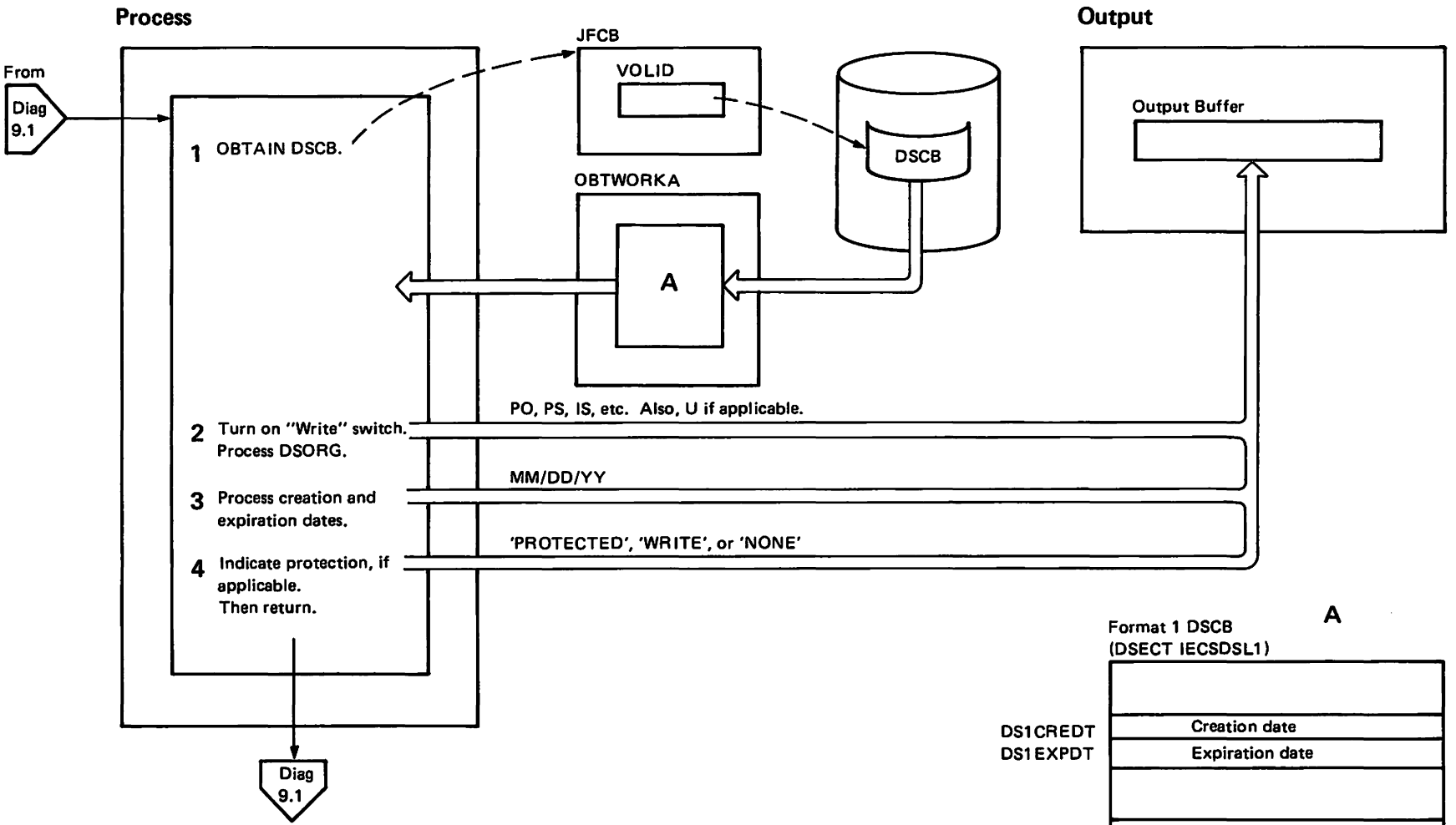
2 Locate the required fields in the CTGPL and the CTGFL to return expiration date, creation date, and entry type.

3 Pass control to IKJEHVHS, the VSAM HISTORY processing routine, which turns on the "Write" switch to indicate that the buffer should be written when all options have been processed.

4 Convert creation and expiration dates into MM/DD/YY format and move them and the entry type to the output buffer.

Object Module: IKJEHAL1
CSECT: IKJEHHST

Diagram 9.4. LISTALC HISTORY Processing (Non-VSAM) (Part 1 of 2)



A

Format 1 DSCB
(DSECT IECSDSL1)

DS1CREDT	Creation date
DS1EXPDT	Expiration date
DS1DSORG	Data Set Organization
DS1DSIND	Protect indicators

Diagram 9.4. LISTALC HISTORY Processing (Non-VSAM) (Part 2 of 2)

1 Check the DSADDNAM field in the DSAB for blanks. If the field contains blanks, it is part of a concatenation. Issue LOCATE to find the volume serial for the OBTAIN; otherwise, issue a RDJFCB to find the volume serial of the volume containing the DSCB.

Issue an OBTAIN macro instruction.

2 Then pass control to IKJEHHST, the HISTORY processing routine, which turns on the 'Write' switch and checks for data set organization.

The organization indicator (PO, PS, etc.) is then placed in the buffer, along with the unmovable indicator (U), if applicable.

3 Creation and expiration date are converted into MM/DD/YY format and placed in the buffer.

4 A check is made for password protection. If none, a check is made for write protection. The applicable indication is placed in the buffer.

Control is then returned to IKJEHAL1, where a check is made for STATUS processing. (If none, the write switch is checked and found 'on', then the buffer is written via PUTLINE; if STATUS processing is applicable, the STATUS information is placed in the buffer prior to checking the write switch.)

Object Module: IKJEHAL1
CSECT: IKJEHHST

Diagram 9.5. LISTALC STATUS Processing (Part 1 of 2)

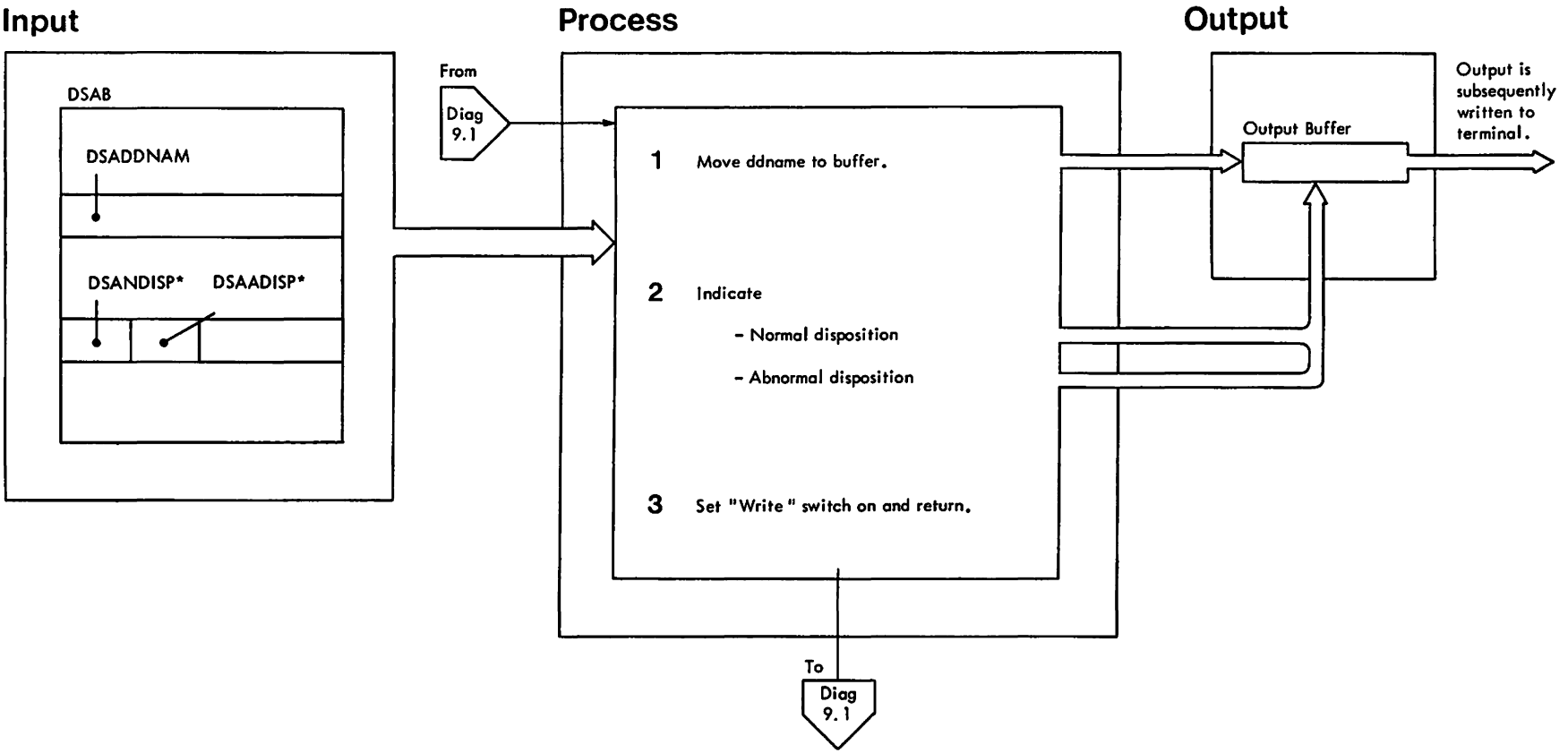


Diagram 9.5. LISTALC STATUS Processing (Part 2 of 2)

1 A check is made to see if HISTORY information is contained in the output buffer area. If so, this will affect the pointer to the proper location in the buffer.

The ddname in field DSADDNAM is moved to the output buffer area. The ddname can be blanks if the data set was concatenated.

2 Then a test is made for normal disposition. The appropriate indication is placed in the output buffer. The output-buffer pointer is then updated to point to the next location.

A test is now made for disposition in the event of an abnormal termination.

The appropriate disposition is moved to the buffer.

3 The 'Write' Switch is turned on and control is returned to IKJEHAL1.

Diagram 9.6. LISTALC MEMBERS Processing (Part 1 of 2)

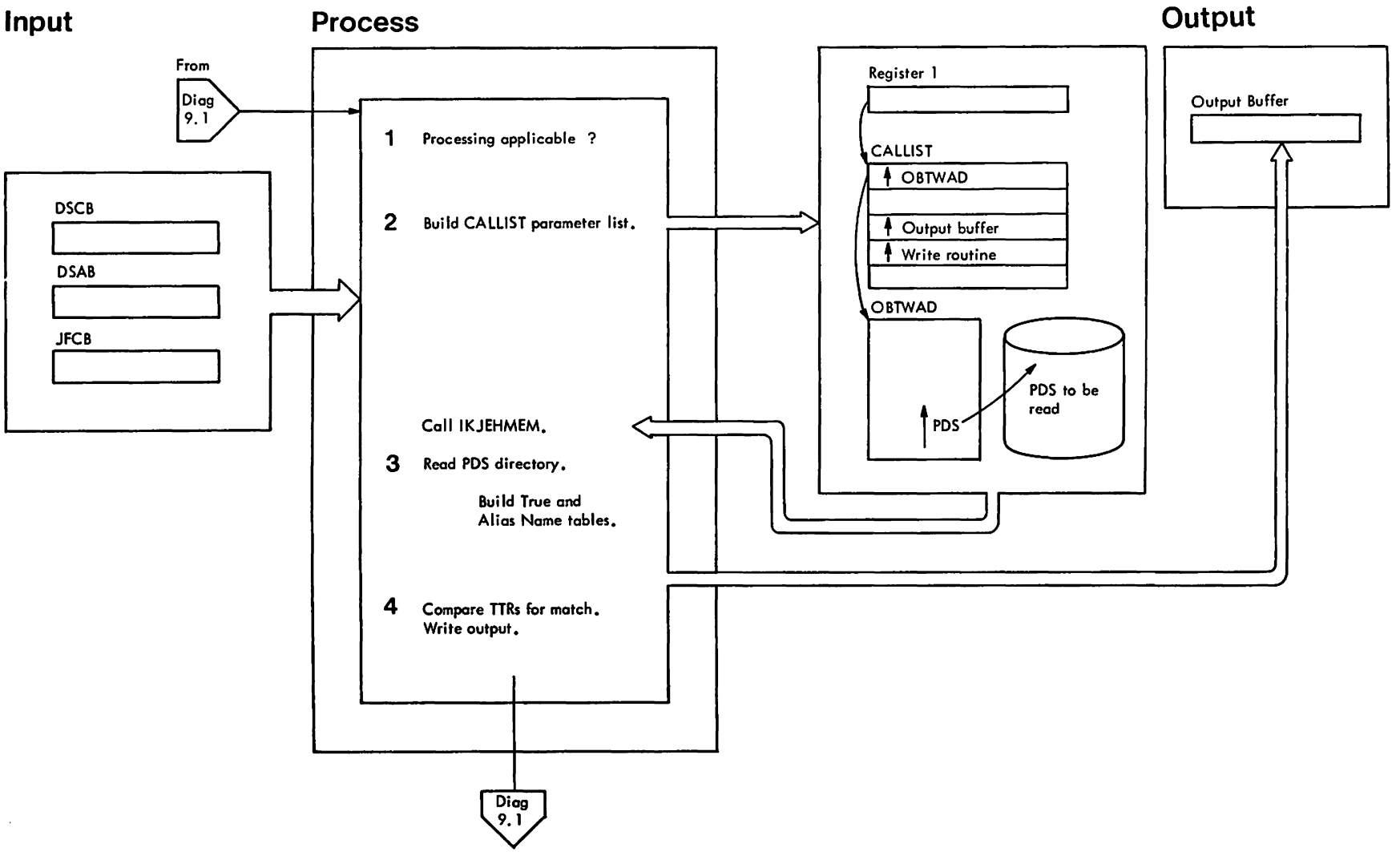


Diagram 9.6. LISTALC MEMBERS Processing (Part 2 of 2)

1 IKJEHMMR makes a number of checks, prior to passing control to IKJEHMEM, to ensure that processing is applicable.

- DSORG in the DSCB is checked to ensure that it is PO.
- This user's userid must be the first qualifier in the data set name.
- The ddname cannot be blank (which would indicate concatenation).
- The dynamic concatenation bit in the DSAB is checked. If on, DSADDNAM is compared to DCBDDNAM. If unequal, MEMBERS processing can continue. (If equal, this is at least the second data set of a concatenation cluster.)

2 A RDJFCB is issued (unless HISTORY was also specified, in which case it is not required), the CALLIST parameter list for IKJEHMEM is constructed, and control is passed to IKJEHMEM.

3 IKJEHMEM initializes the True and Alias Name tables, then reads the PDS directory into them. Name blocks are obtained and chained dynamically, as required.

4 A true name is moved to the output buffer. The true name TTR is compared with all of the alias name TTRs. Applicable aliases are moved to the buffer. (The calling routine's write routine is used to write the buffer.) This action is repeated until all true names have been processed. Alias names that do not match any true name are then grouped by TTR and written. A message is provided to indicate that no true name exists for them.

Control then returns to IKJEHMMR, where the return code is checked and control is passed to IKJEHAL1.
Possible message: IKJ58301I

Object Module: IKJEHMEM
CSECT: IKJEHMMR

Diagram 10.1. LISTBC Processing Overview (Part 1 of 2)

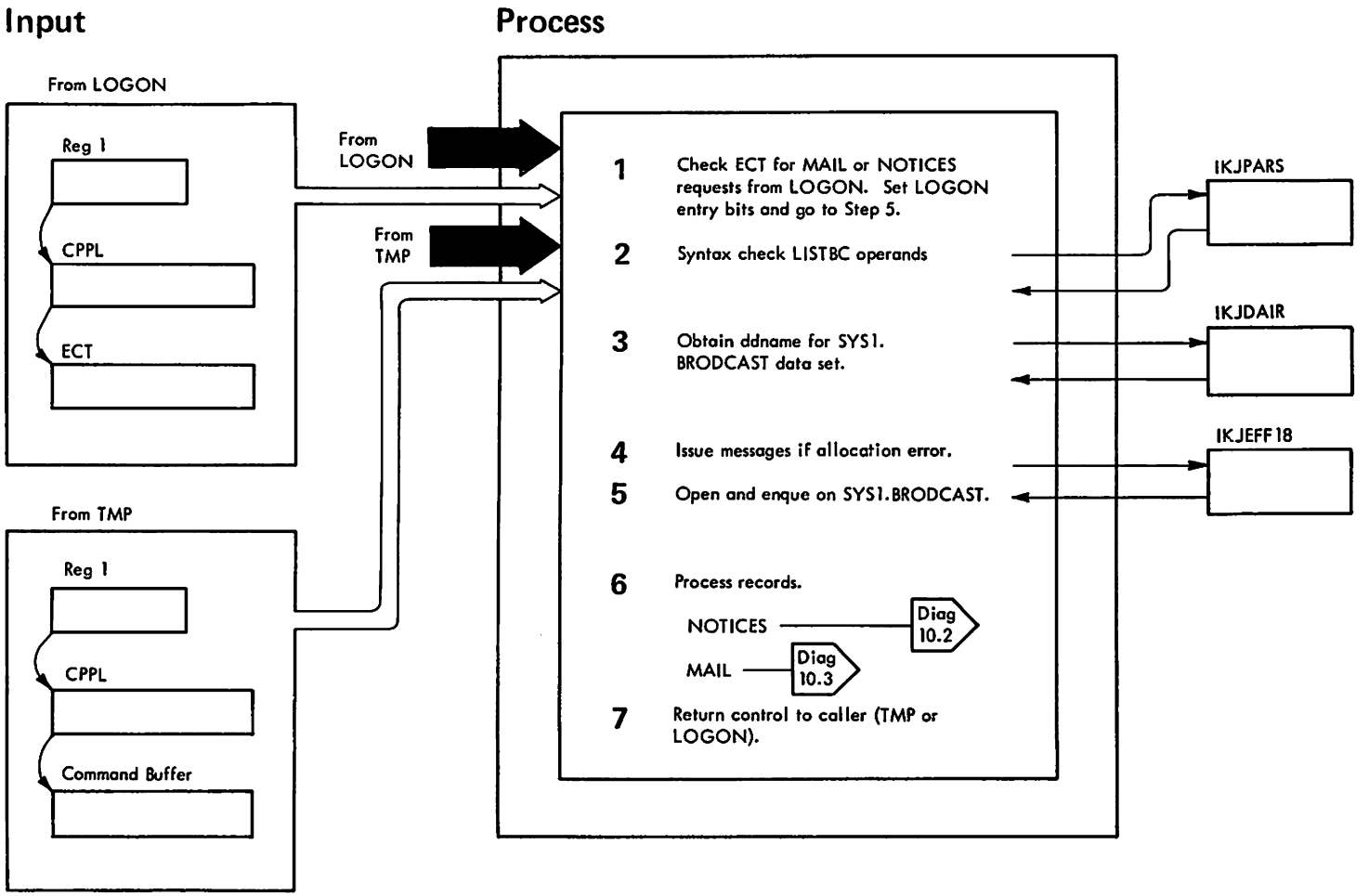


Diagram 10.1. LISTBC Processing Overview (Part 2 of 2)

1 Entry point IKJEES73 is used to bypass the use of Parse.

2 A check is made to determine if explicit operands have been supplied for LISTBC. If they exist, the Parse parameter list (PPL) is created with Register 1 pointing to it. Parse is then used to syntax check the LISTBC operands. If the NOMAIL and NONOTICES bits are both set, control is returned to the TMP. If Parse fails, it returns a non-zero return code. When LISTBC detects this, it issues an error message and returns control to the TMP.

3 If Parse was successful or was not required, LISTBC proceeds to create a DAIR parameter list (DAPL) which is pointed to by register 1. The DAPL contains a pointer to an X'08' type DAIR parameter block (DAPB08). DAIR is then used to obtain a ddname. If DAIR is successful, the ddname it returns is placed in the DCB which describes the SYS1.BRODCAST data set located on a direct access device.

4 If allocation of a ddname is not successful, DAIR returns a non zero return code. In this case, the common DAIR failure message routine (IKJEFF18) is used to issue an error message. Control is returned to the TMP.

5 If there are no errors, LISTBC does an enqueue for record 1 (qname 'SYSIKJBC', rname '000000'X) and opens SYS1.BRODCAST for output.

6 Processing of NOTICES messages and MAIL user messages is done as shown in Diagrams 10.2 and 10.3. The user is informed if there are no messages of the types requested.

7 If entry was from the TMP, dequeueing is done from SYS1.BRODCAST and it is closed. Control is then returned to the TMP.

If entry was from LOGON, control is returned to LOGON.

Object Module: IKJEES70, IKJEES75

Diagram 10.2. LISTBC NOTICES Message Processing (Part 1 of 2)

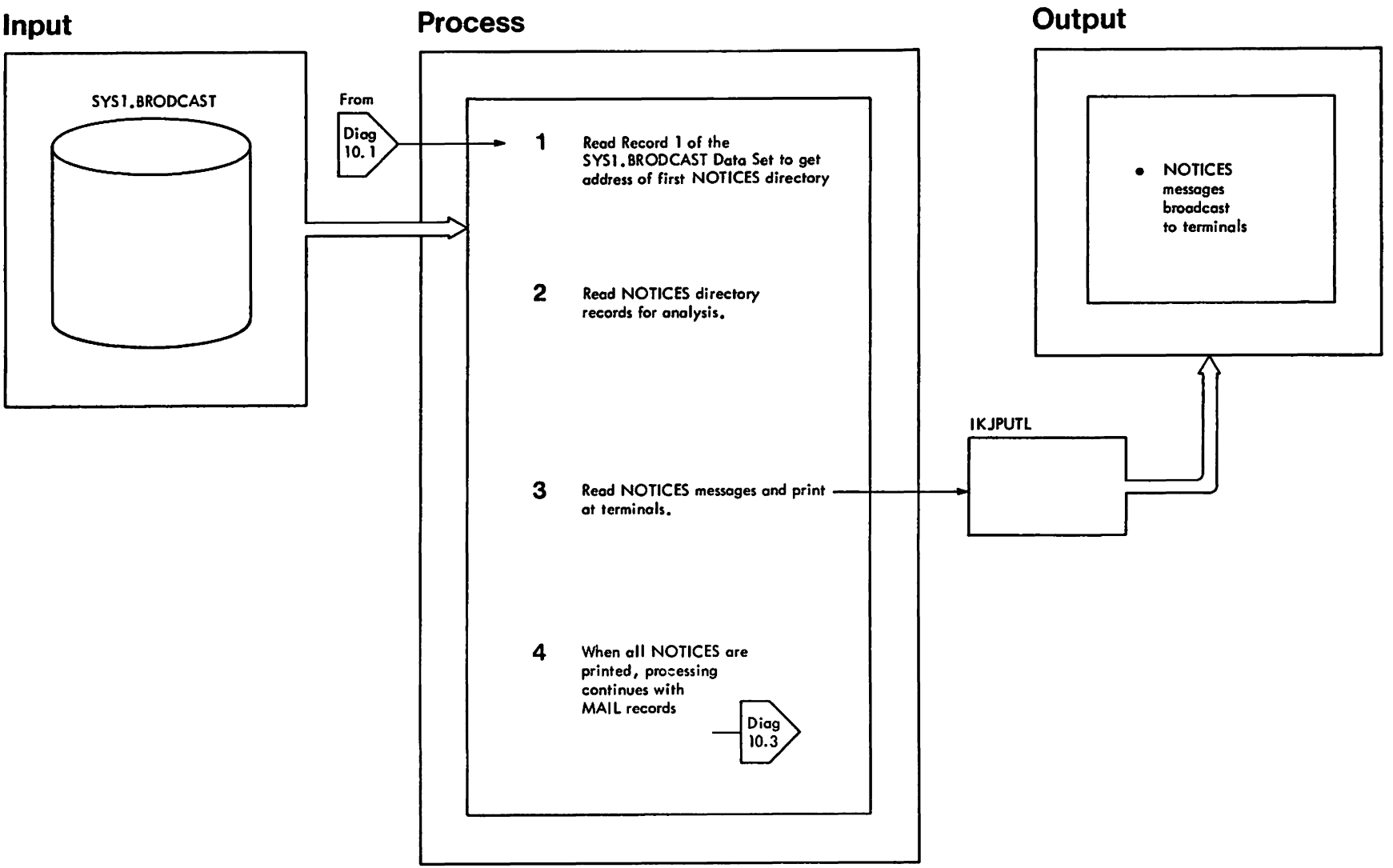


Diagram 10.2. LISTBC NOTICES Message Processing (Part 2 of 2)

1 Read record 1 of the SYS1.BRODCAST data set to check for proper format (level indicator is 2) and get the pointer to the first NOTICES directory record. If there is a format error an error message is issued and control is returned to the caller. If NONOTICES was specified, processing of NOTICE records is bypassed and processing continues with MAIL records.

2 Read the first NOTICES directory record into a buffer and scan the buffer for active NOTICES.

3 Broadcast the message if the high order bit of the message number is set to zero. The message text is read into a buffer which is pointed to by an entry in the IO Parameter List (IOPL). The PUTLINE service routine (IKJPUTL) is used with register 1 pointing to the IOPL. IKJPUTL transmits the active NOTICES message to the terminal.

4 Examine the RBA for the next NOTICES directory record. If the RBA is non-zero, that RBA is used to locate the next NOTICES directory record. The record is read into the buffer and processed as in step 2. If the RBA is zero, no more directory records exist. Control is passed to the MAIL message processing routine.

Object Module: IKJEES70, IKJEES75

Diagram 10.3. LISTBC MAIL Message Processing (Part 1 of 2)

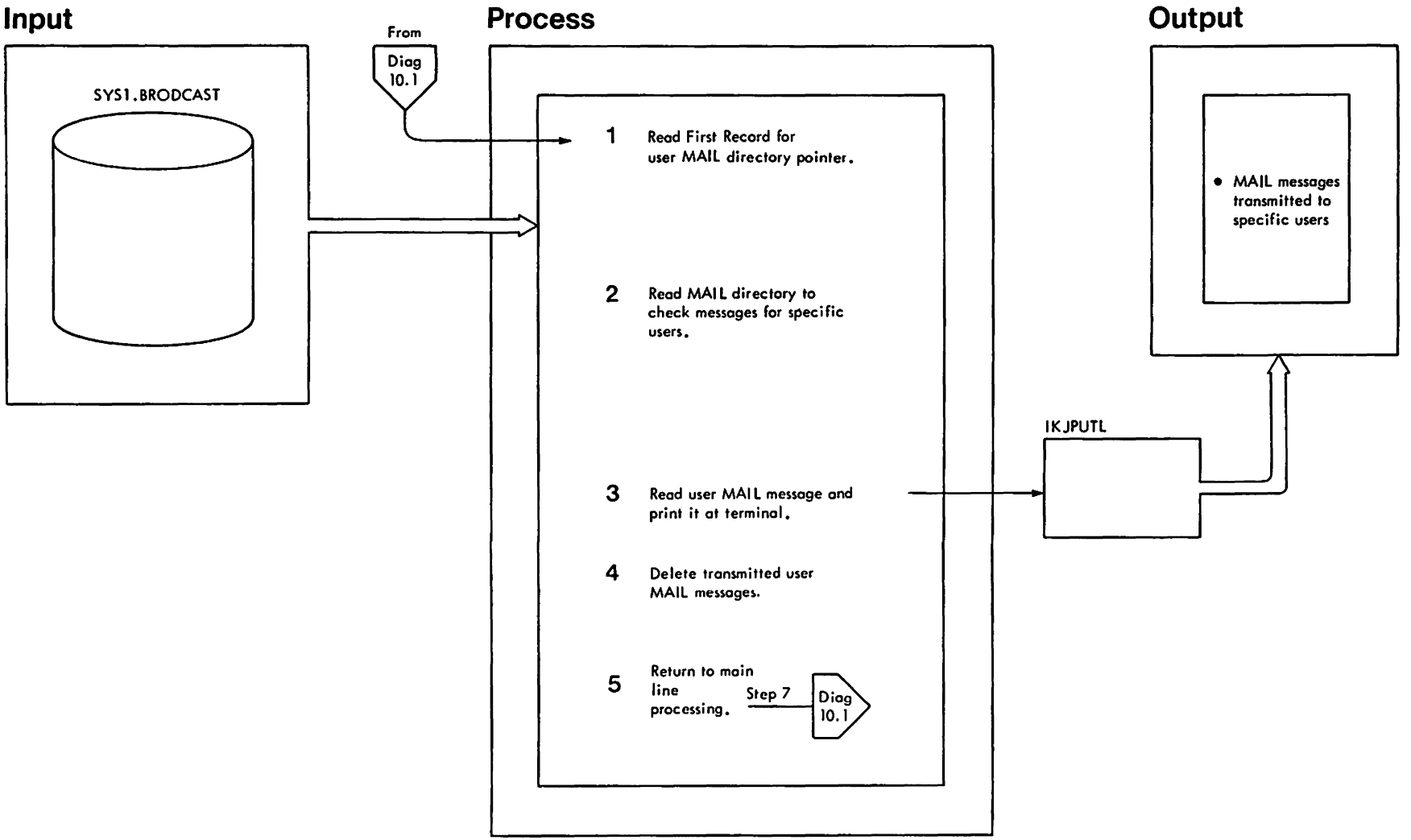


Diagram 10.3. LISTBC MAIL Message Processing (Part 2 of 2)

1 Check if the NOMAIL bit was set on. If it was set, branch to the LISTBC exit procedure described in step 7 of diagram 10.1. Read Record 1 of the SYS1.BRODCAST Data Set to obtain the pointer to the relative block address (RBA) of the initial user MAIL directory record.

2 The first MAIL directory record is read into a buffer where the user identification (userids) can be examined. The userid in the PSCB created by LOGON is successively compared to all userids in the first MAIL directory record. If a match is found in this record, processing continues as described in step 3 below. If a X'7F' is found in scanning the record, it indicates that an end of record condition has been reached. The RBA pointer to the next directory record is examined. If it is not zero, it is used by BDAM to read that next MAIL directory record into a buffer. It is then scanned for the matching userid in the same manner as above. The final MAIL directory record has all zeros as its RBA for the next directory. If this condition is reached, it means all userids in the entire chain of MAIL directory records have been searched without finding a matching userid. In this case, no messages are transmitted to the requesting user.

3 When the matching userid is found in the MAIL directory, its associated RBA pointer to its message chain, is used to read the message text record into a buffer. PUTLINE is invoked with its IOPL containing a pointer to the message text buffer. The user MAIL message is transmitted to the user by PUTLINE. The RBA pointer to additional MAIL messages for that same user is checked. If the RBA pointer is not zero, it is used to read in the next MAIL message, which is again transmitted by PUTLINE. This process continues until the zero RBA message pointer is reached.

4 Each MAIL message record sent to the requesting userid is then deleted. The message record itself is written back with a X'FF' key, which tells BDAM that this is an inactive record which can be used for incoming messages.

5 Upon completion of MAIL message processing, a branch is taken to the LISTBC exit procedure described in step 7 of diagram 10.1.

Object Module: IKJEES70, IKJEES75

Diagram 11.1. LISTDS Processing Overview (Part 1 of 2)

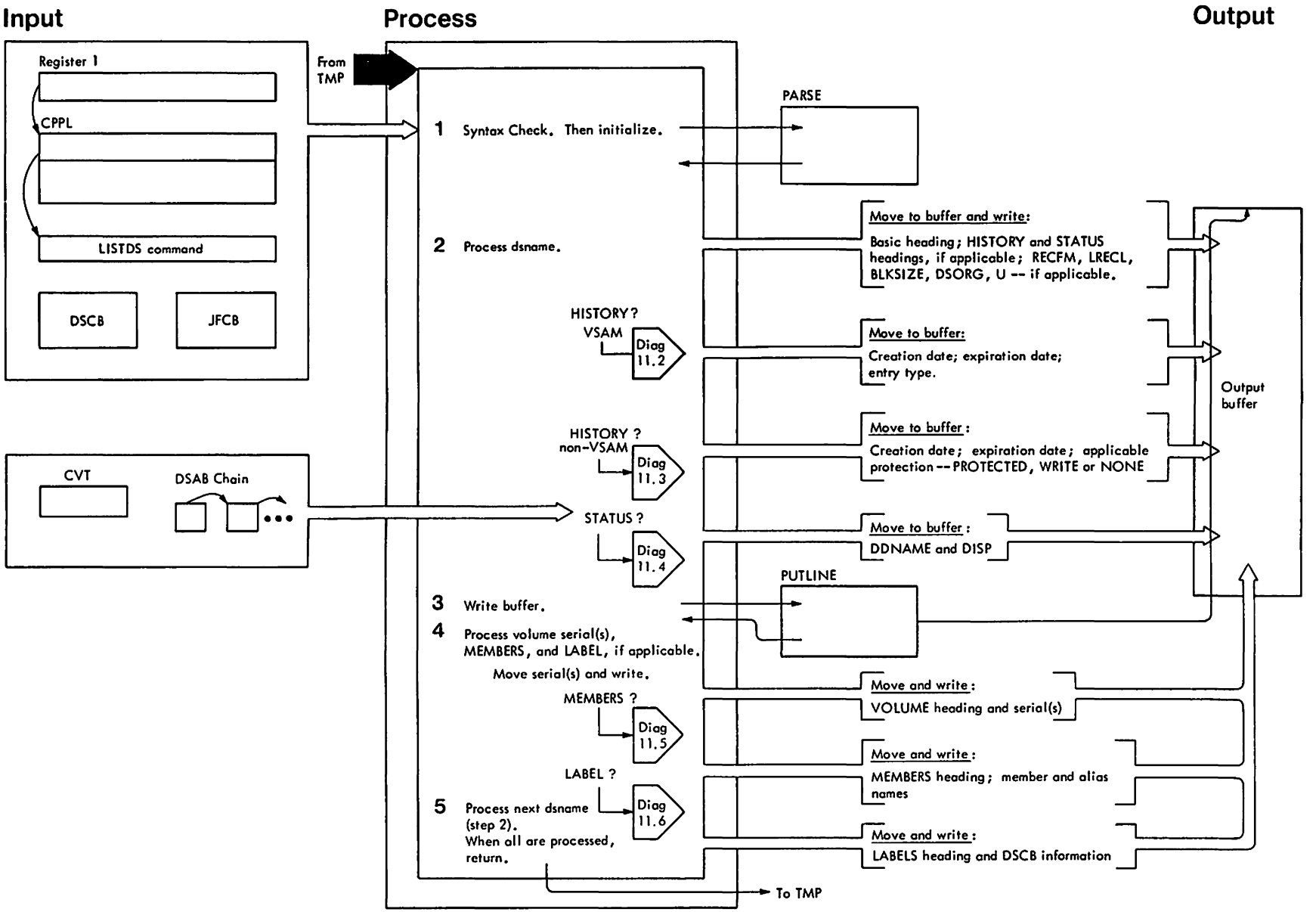


Diagram 11.1. LISTDS Processing Overview (Part 2 of 2)

1 The Parse routine receives control to check the command buffer for incorrect or unspecified parameters upon return from Parse, the return code is checked, and then appropriate option bits are set to reflect the specified options. If MEMBERS is requested, load module IKJEHMEM is loaded into storage. If STATUS is specified, the DSAB is located. The LEVEL keyword or an '*' indicates that the data set name is generic.
Possible messages: IKJ58511I, IKJ58512I

2 The first entry in the dsname list is pointed to. The NXDSNAME subroutine examines the dsname and fully qualifies it, if necessary.
Possible messages: IKJ58503I, IKJ58509I, IKJ58502I, IKJ58513I

Then the fully-qualified name is moved to the output buffer and written.

Non-VSAM Data Sets: The LOCATE macro is used in an attempt to locate the dsname through the catalog. If LOCATE passes back a non-zero return code, DAIR is used (with an X'08' operation code) in an attempt to see if the data set is otherwise accessable. If the data set cannot be located, the user is informed and processing continues with the next dsname.
Possible messages: IKJ58503I, IKJ58506I, IKJ58507I, IKJ58508I, IKJ58510I, IKJ58512I

Depending on which way the data set was found (using LOCATE or DAIR), set up is performed prior to issuing an OBTAIN. If LOCATE found the data set, the LOCATE switch is set. If DAIR found the data set, the DDNAME is moved to a DCB and the RDJFCB macro is issued to get the JFCB.

Then the OBTAIN macro is issued to bring the DSCB into storage. If the return code is not equal zero, processing continues with the next dsname in the list. Otherwise, heading information is moved to the buffer and written.

VSAM Data Sets: The Catalog Information Routine is used to supply a list of names. VSAM LOCATE is used to indicate whether the data set is VSAM or not. LOCATE also supplies the required attributes if the data set is VSAM.
Possible message: IKJ58504I

HISTORY is processed, if applicable. See Diagram 11.2 (VSAM) or 11.3 (non-VSAM)

STATUS is processed, if applicable. See Diagram 11.4

3-4 First the buffer is written, then volume serials are placed, one by one, in the buffer and printed.
Possible message: IKJ58504I

Then a check is made for MEMBERS and LABEL processing.
See Diagrams 11.5 and 11.6 respectively.

5 After MEMBERS and/or LABEL have been processed, additional dsnames, if any, are processed. When all have been processed, control returns to the TMP.

Object Module: IKJEHDS1
CSECT: IKJEHBSC

Diagram 11.2. LISTDS HISTORY Processing (VSAM) (Part 1 of 2)

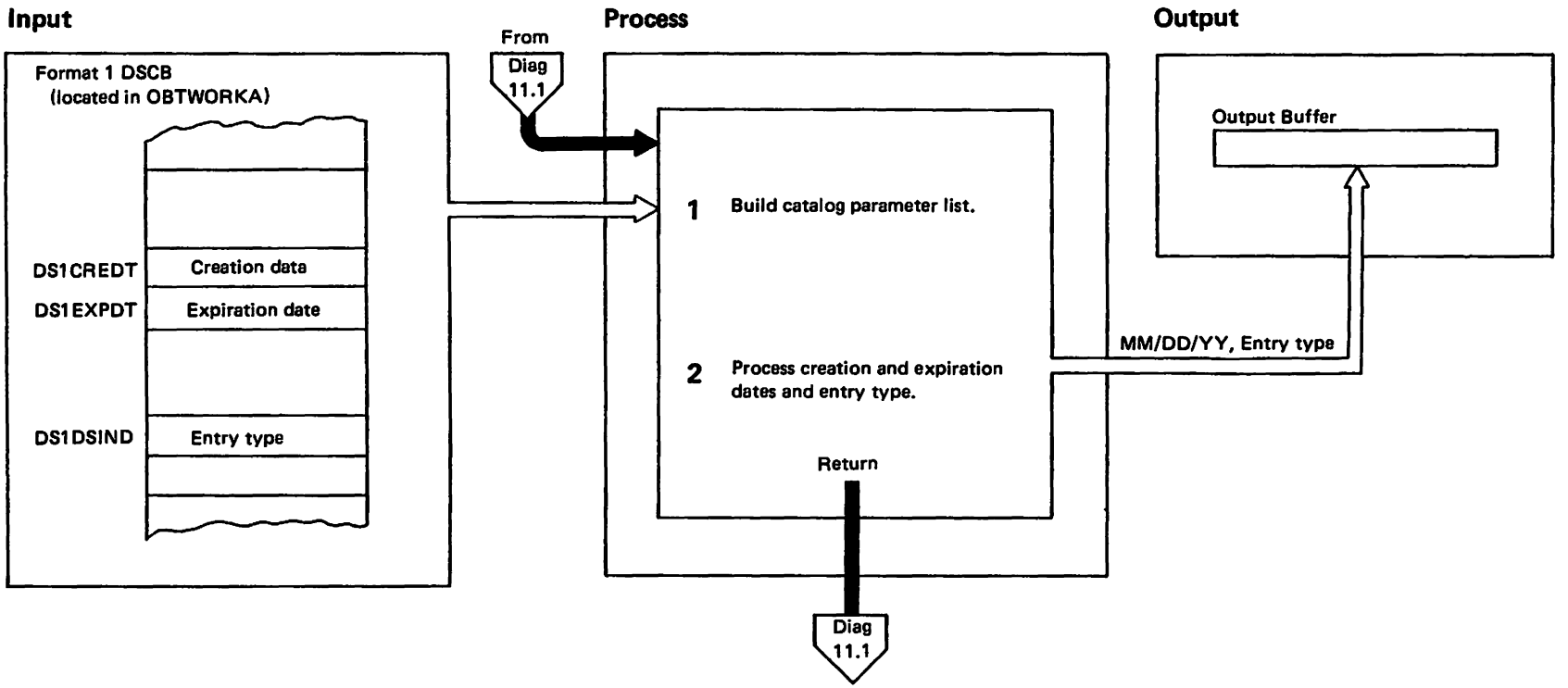


Diagram 11.2. LISTDS HISTORY Processing (VSAM) (Part 2 of 2)

1 Build a catalog parameter list using information and the data set name obtained from the DSAB by SVC99. The parameter list specifies the data set name to be retrieved from the VSAM catalog, the entry type, creation and expiration dates, logical record length, volume serials, and physical blocksize for the data set.

2 Upon entry from the IKJEHDS1 routine, IKJEHHIS gets a save area, then sets up to convert the creation date from YMMDD format to MM/DD/YY.

The creation date is converted and placed in the buffer. If no date was found, a default of 00/00/00 is placed in the buffer. This process is repeated for the expiration date. The entry type code is also moved into the output buffer.

Object Module: IKJEHDS1
CSECT: IKJEHHIS

Diagram 11.3. LISTDS HISTORY Processing (Non-VSAM) (Part 1 of 2)

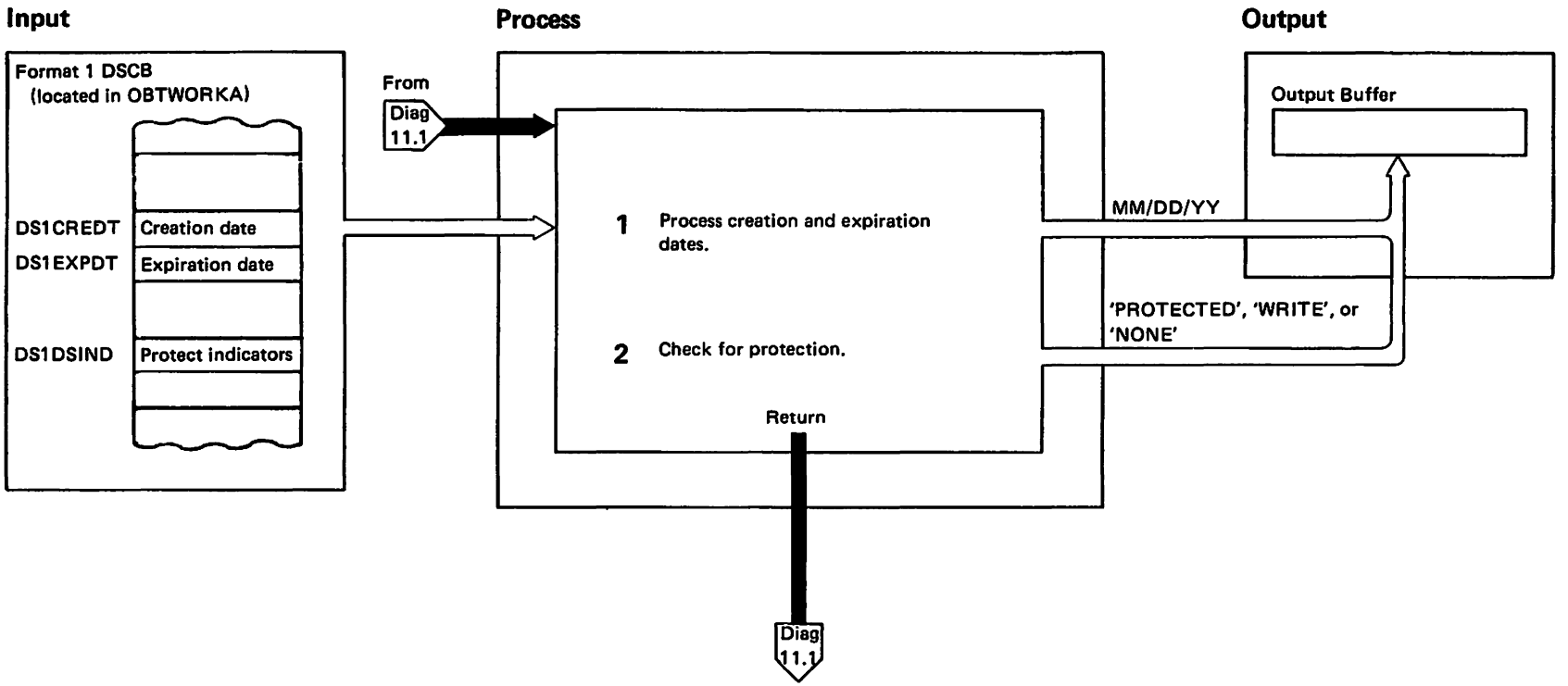


Diagram 11.3. LISTDS HISTORY Processing (Non-VSAM) (Part 2 of 2)

1 Upon entry from the IKJEHDS1 routine, IKJEHHIS gets a save area, then sets up to convert the creation date from YMMDD format to MM/DD/YY.

The creation date is converted and placed in the buffer. If no date was found, a default of 00/00/00 is placed in the buffer. This process is repeated for the expiration date.

2 Then a check is made for password protection. If password protection applies, 'PROTECTED' is placed in the buffer. Otherwise, a check is made for WRITE protection. If WRITE protection applies, 'WRITE' is placed in the buffer. Otherwise 'NONE' is placed in the buffer.

Then the space obtained for the save area is freed and control returns to the IKJEHDS1 routine. See Diagram 11.1.

Object Module: IKJEHDS1
CSECT: IKJEHHIS

Diagram 11.4. LISTDS STATUS Processing (Part 1 of 2)

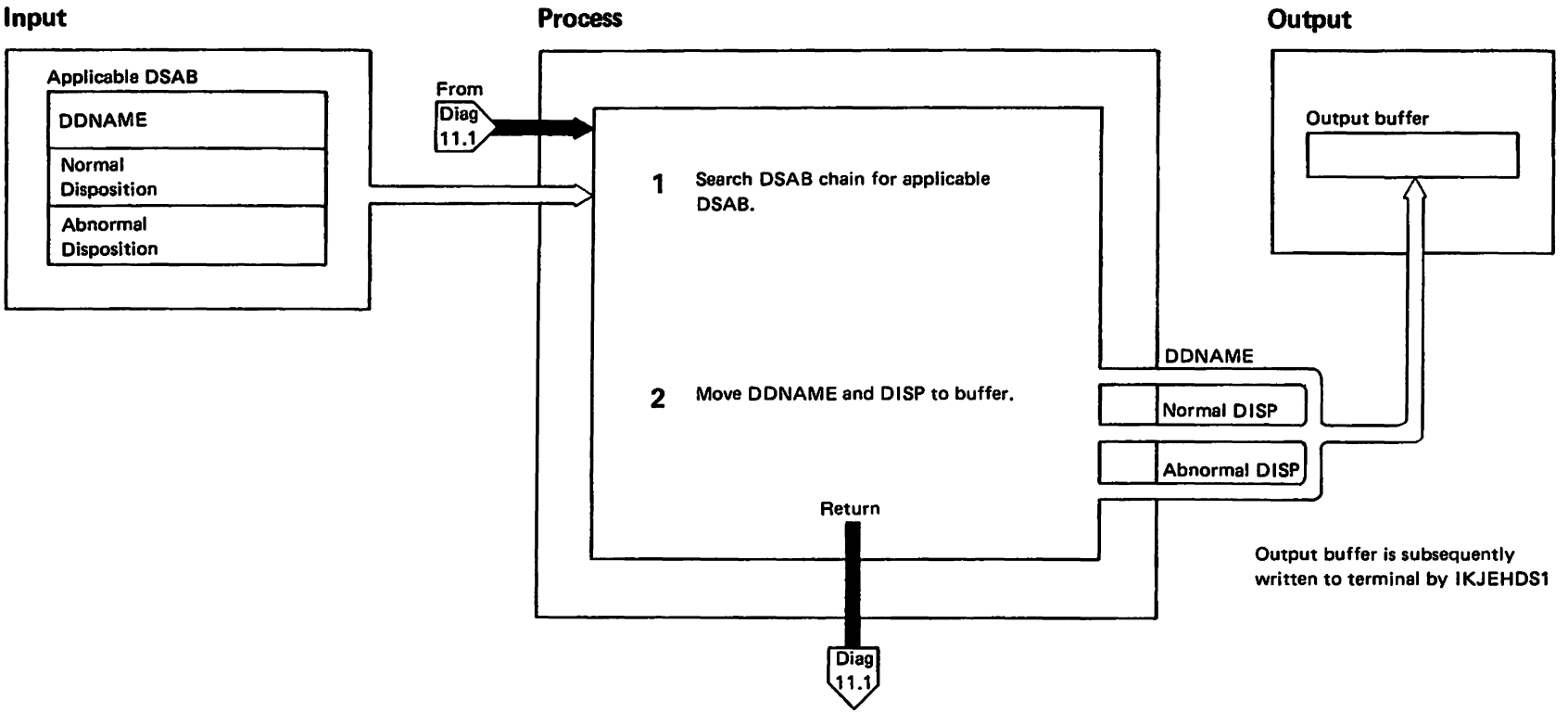


Diagram 11.4. LISTDS STATUS Processing (Part 2 of 2)

1 SVC99 searches the DSAB chain for a data set name that matches the name in the data set list. The search is done to obtain allocation information about the date set name. If no match is found, control is returned to IKJEHDS1.

2 When a dsname match is found, a check is made to see if HISTORY is also specified (in which case, the buffer is filled partially with HISTORY information that has not yet been written). If yes, the offset to the output buffer area is adjusted accordingly.

Then the DDNAME is moved to the output buffer.

The status bits are then tested for normal disposition, and the appropriate word (KEEP, DELETE, CATLG, or UNCATLG) is placed in the buffer.

After the output buffer offset is adjusted, the status bits are tested for disposition in the event of an abnormal termination. The appropriate disposition is placed in the buffer.

Control returns to the IKJEHDS1 routine. See Diagram 11.1.

Object Module: IKJEHDS1
CSECT: IKJEHSTA

Diagram 11.5. LISTDS MEMBERS Processing (Part 1 of 2)

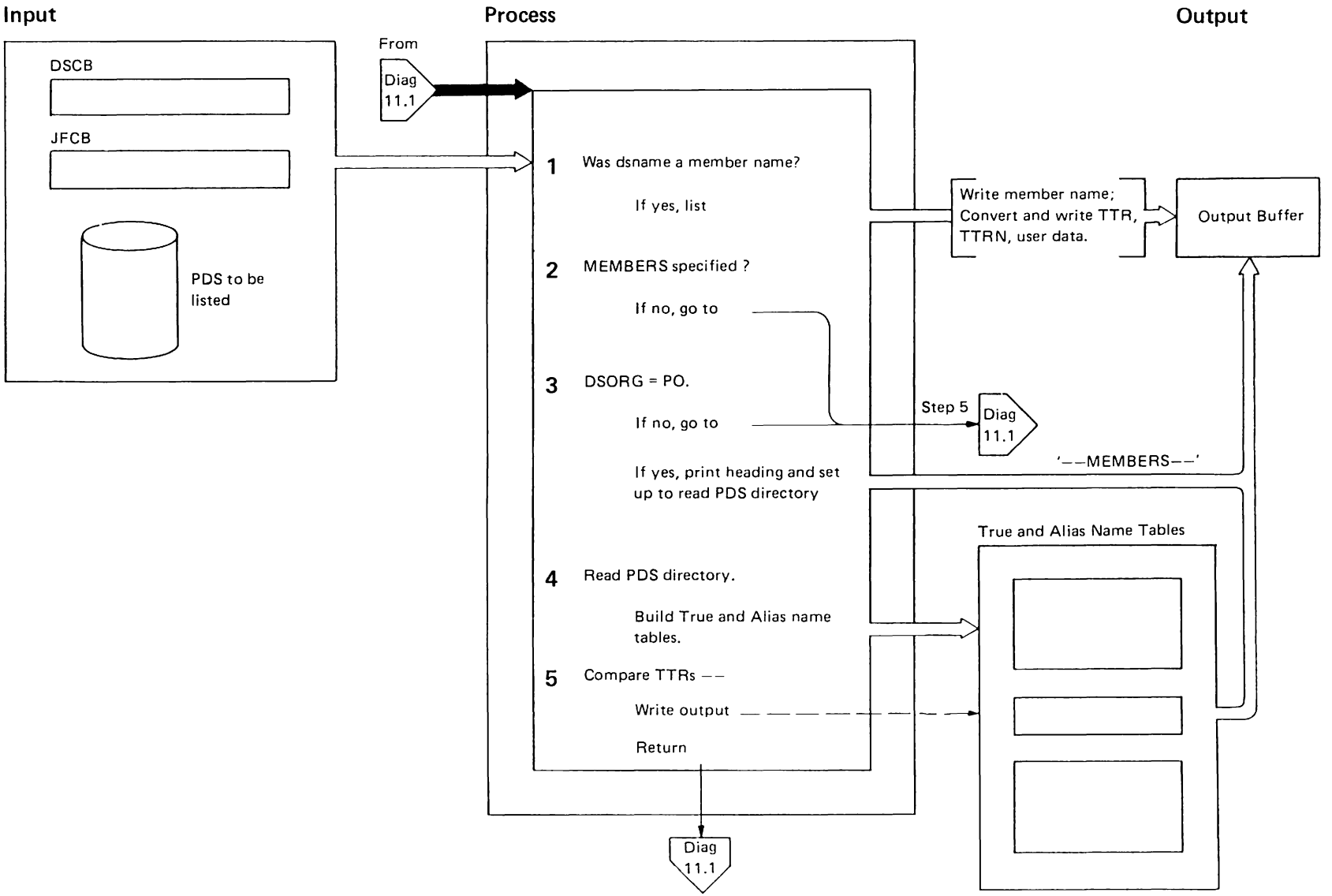


Diagram 11.5. LISTDS MEMBERS Processing (Part 2 of 2)

1 After volume information has been printed, a check is made to see whether the current dsname is a member name. If not a member name, a check for label processing is first made, then a check is made to see if the MEMBERS keyword is specified (step 2).

If the dsname was a member name, routine MNAMROUT is given control to print specific information for the member. If necessary, MNAMROUT issues a RDJFCB and passes control to DAIR to allocate the data set.

2 Then LABEL processing takes place, if applicable (see Diagram 11.6). After this, a check is made to see if MEMBERS was specified. If no, processing continues from step 5 of Diagram 11.1. Otherwise control is passed to the MEMBERS interface routine, MEMROUT.

3 A check is made to ensure that the organization is partitioned. If not, control returns to step 5 of Diagram 11.1. Otherwise the MEMBERS heading is written. Then a check is made to see if the JFCB has already been read. If yes, processing continues from step 4, below. Otherwise, DAIR is used to allocate the data set. Then the DDNAME is placed in DCBDDNAM of OBTDCB and an RDJFCB is issued prior to reading the PDS directory. Then control passes to IKJEHMEM.
Possible messages: IKJ58502I, IKJ58514I

4 IKJEHMEM initializes tables to contain the true and alias names, then reads the PDS directory into the tables. Name blocks are obtained and chained dynamically, as required.

5 Then a true name is moved to the output buffer. The true name TTR is compared to all of the alias name TTRs. Applicable aliases are moved to the buffer. (MEMROUT's write routine, which uses PUTLINE, is used to write the buffer.) This action is repeated until all true names have been processed. Alias names not matching any true name are then grouped by TTR and written. A message is provided to indicate that no true name exists for them. Control returns to MEMROUT.
Possible message: IKJ58501I

MEMROUT checks the return code, then returns control to step 5 of Diagram 11.1 to process the next name in the list.

Object Modules: IKJEHDS1, IKJEHMEM

Diagram 11.6. LISTDS LABEL Processing (Part 1 of 2)

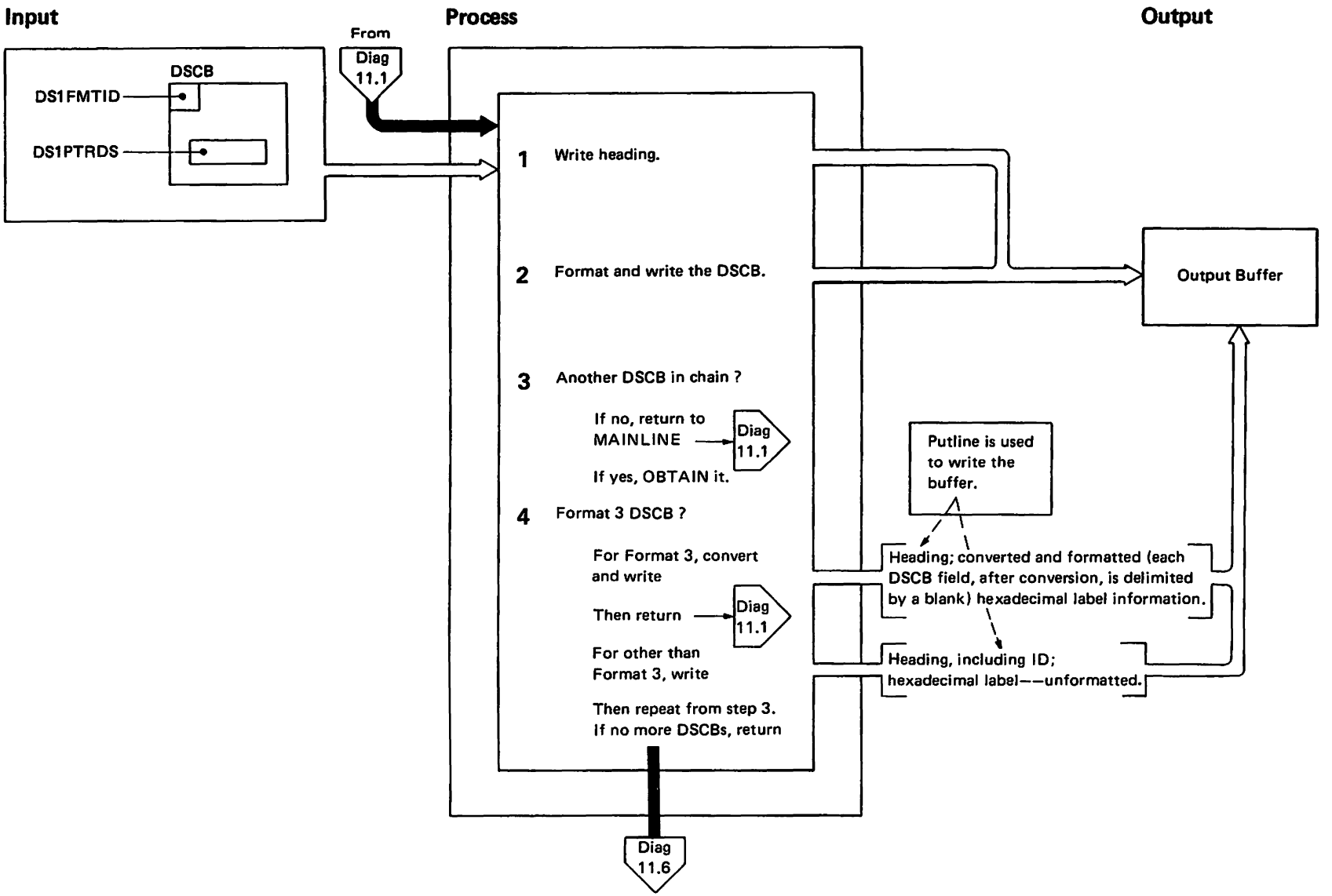


Diagram 11.6. LISTDS LABEL Processing (Part 2 of 2)

1 After getting a save area, IKJEHLBL uses PUTLINE to write the heading for the DSCB.

2 IKJEHLBL refers to DS1FMTID for the address of the DSCB information to be converted. The DSCB information is then converted from binary to hexadecimal and written one line at a time. Formatting consists of separating each field by a blank.

3 DS1PTRDS is then checked to determine if any DSCBs are chained to the format 1 DSCB just processed. If none, control returns to MEMBCHK in the IKJEHDS1 routine.

If another DSCB is found, an OBTAIN is issued for it.
Possible message: IKJ58505I

4 A check for a Format 3 DSCB is made. (A Format 3 DSCB is formatted as in step 2 and 3, above. If a Format 3 DSCB is found, control returns to the IKJEHDS1 routine after the DSCB is processed.

If the DSCB is other than a Format 3 DSCB, no formatting takes place. That is, the information is converted to hexadecimal and dumped 36 bytes at a time. Then a check is made for another DSCB. If none, storage is freed and control returns to MEMBCHK in the IKJEHDS1 routine.

Object Module: IKJEHDS1
CSECT: IKJEHLBL

Diagram 12. OPERATOR Command Processing (Part 1 of 2)

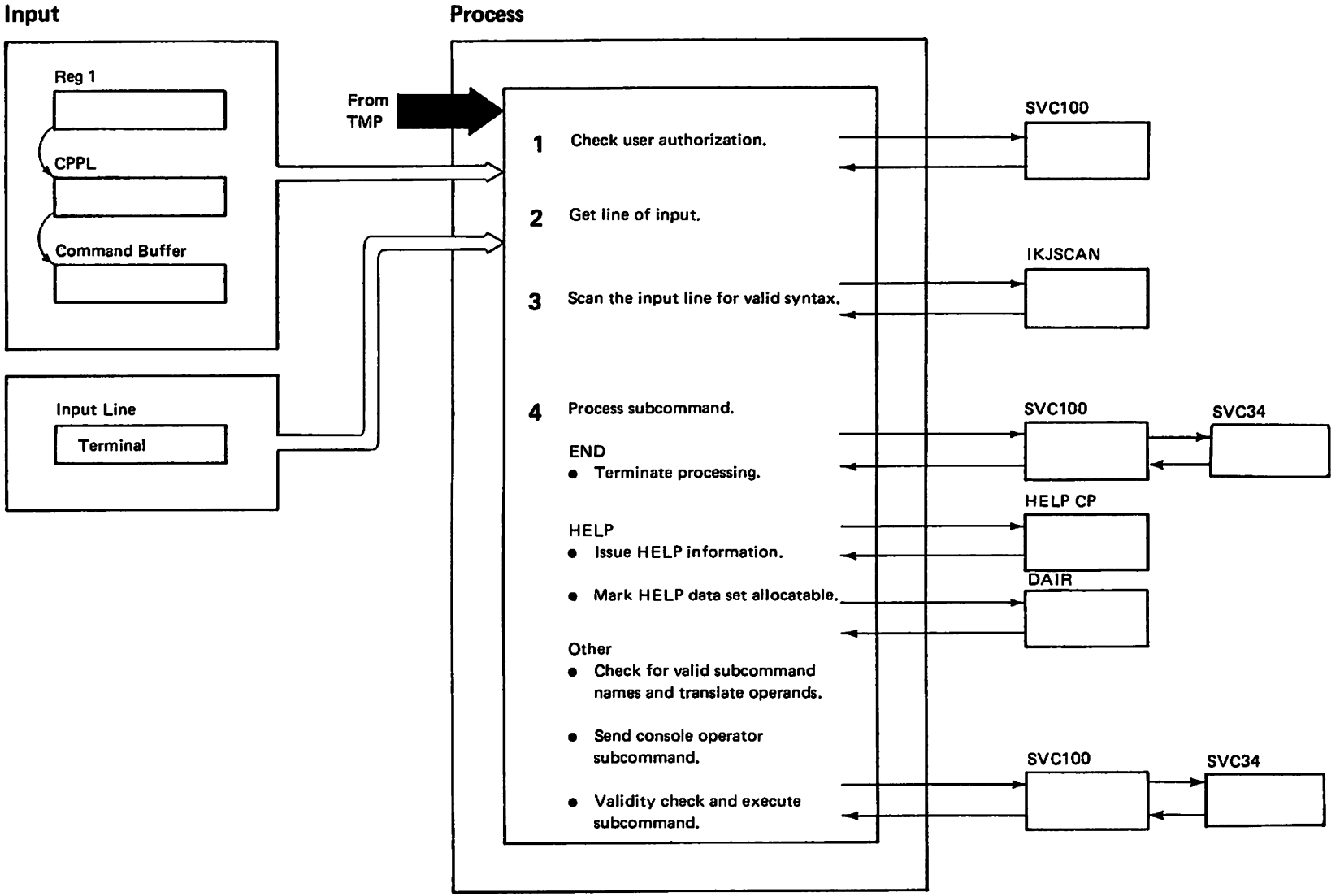


Diagram 12. OPERATOR Command Processing (Part 2 of 2)

1 After the STAE and ATTEN exits are set up, SVC100 checks the user's authority to enter the OPERATOR command. Information is passed to SVC100 in the FIBPARMS parameter list. If the user is not authorized, OPERATOR will issue an error message and return control to the TMP.

2 Use PUTGET to get a line of input from the terminal and to issue the command mode message if required.

3 Scan the input line with IKJSCAN for valid syntax. If the subcommand syntax was invalid an error message is issued and PUTGET gets another line of input.

4 Process OPERATOR subcommands.

END

- This routine is used to terminate processing due to an error or when an END subcommand is issued by the terminal user to terminate OPERATOR command processing. SVC100 is used to stop active monitors and to issue SVC34 to schedule executions of the subcommand. All buffers are freed and service routines are deleted. Control is returned to the TMP.

HELP

- ATTACH the HELP command processor to send the terminal user the HELP information. If the ATTACH failed control is passed to step 4-END.
- When HELP is finished, use DAIR to mark data sets used by HELP as available for allocation. If DAIR fails, control is passed to step 4-END.
- Processing continues with step 2.

Other

- Check the subcommand name against a list of allowable names (DISPLAY, MONITOR, SEND, CANCEL, and STOPMN). If parameters were specified on the DISPLAY, MONITOR, CANCEL, or STOPMN subcommands, translate the operands to upper case for use by SVC100.
- Send the console operator a message with the subcommand that was entered.
- Initialize the FIBPARMS parameter list and use SVC100 to validity check and to issue SVC34 to schedule executions of the subcommand. If the validity check fails, an error message is issued to the terminal user. Processing continues with step 3. If there is an error other than validity, processing continues with step 4-END.

Object Modules: IKJEE100, IKJEE1A0, and IKJEE150

Diagram 13. OUTPUT Processing (Part 1 of 2)

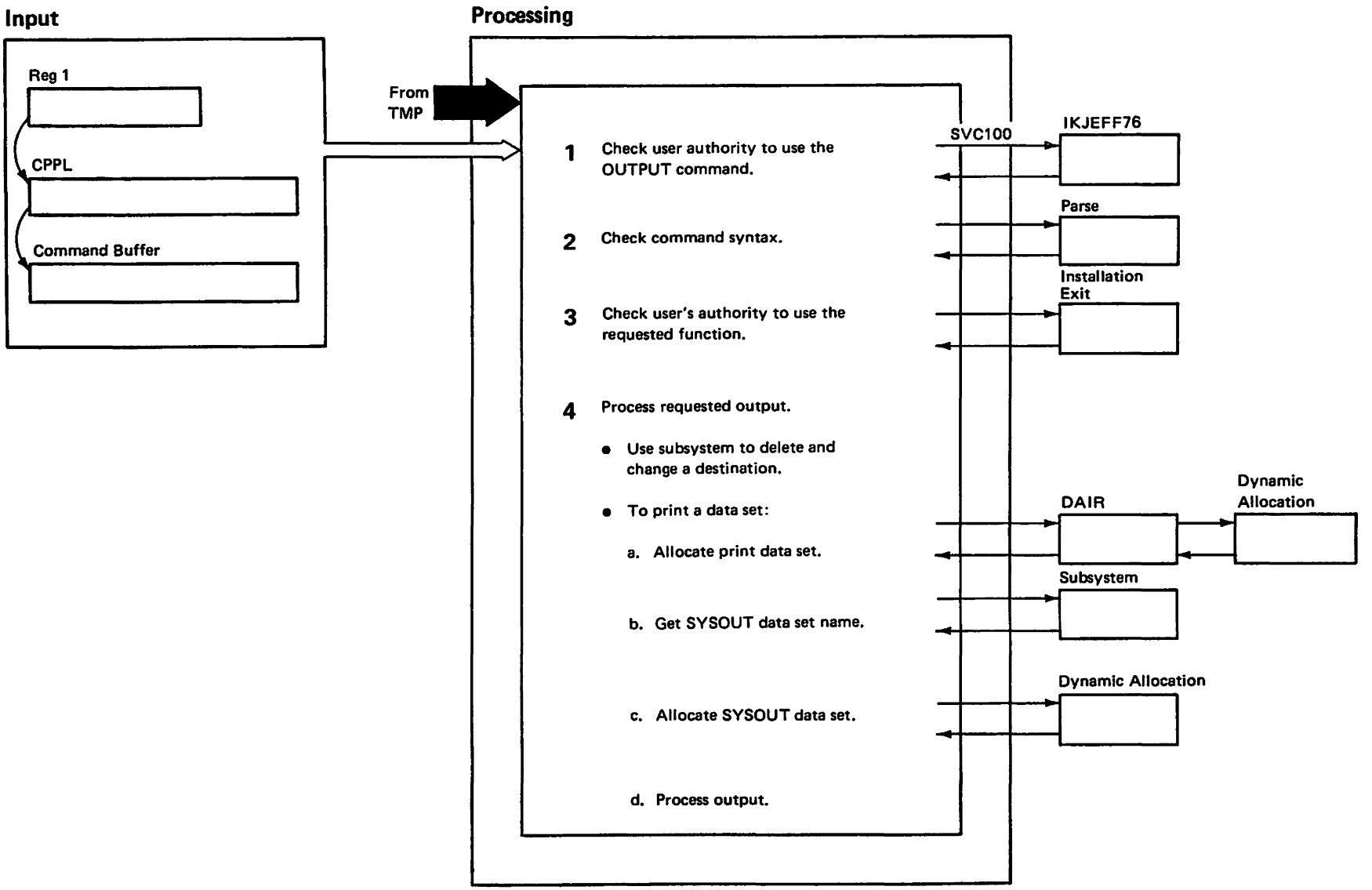


Diagram 13. OUTPUT Processing Summary (Part 2 of 2)

1 Use SVC100 to post the TMP (IKJEFTSC) requesting IKJEFF76 be attached under a parallel task structure. Information is passed to IKJEFF76 in the FIBPARMS parameter list. IKJEFF76 checks the user's authorization to enter the command. If the user is not authorized to enter foreground initiated background commands, the system issues an error message and returns control to the TMP.
Object Module: IKJCT466

2 Use Parse to check the syntax of the command.
Object Module: IKJCT469

3 Use an installation exit to check the userid for authorization to use the requested function on the job specified. If there is no installation exit the IBM supplied exit IKJEFF53 is used.
Object Module: IKJECT469

4 Determine the operation to be performed: print, delete, or change the destination (station or class) of a data set.

- To delete or change the destination of a data set, set up an interface to the subsystem and request the subsystem to perform the requested operation. Return control to the TMP.
  Object Modules: IKJCT469, IKJCT462
- To print a data set:
  a. Use Dynamic Allocation to allocate a PRINT data set via the DAIR interface.
     Object Module: IKJCT469, IKJCT473
  b. Use the job entry subsystem to select all system output data sets for a specific jobname and class.
     Object Module: IKJCT462
  c. Use Dynamic Allocation to allocate a system output data set by data set name.
     Object Module: IKJCT462
  d. Process the system output data set until an end-of-file condition or an attention.

     For an end-of-file condition, check for more data sets. If there are no more, return control to the TMP.

     For an attention, process the requested subcommand and all remaining data sets and return control to the TMP.
     Object Modules: IKJCT462, IKJCT470, IKJCT471, IKJCT463

Diagram 14. PROFILE Processing (Part 1 of 2)

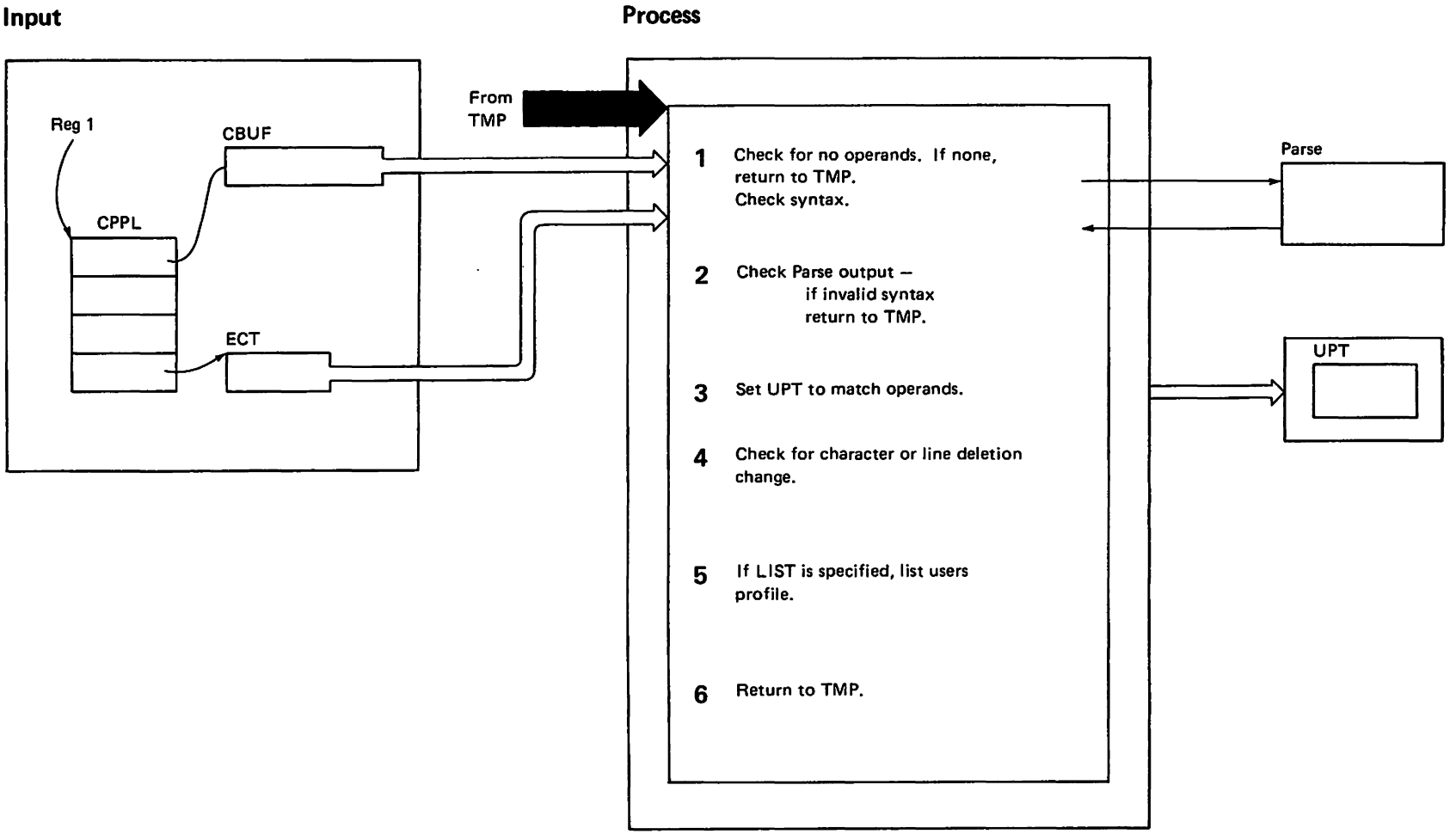


Diagram 14. PROFILE Processing (Part 2 of 2)

Extended Description

1 Check the ECT for the presence of operands in CBUF (The Command Buffer). If there are none, issue a message to the user and return to the TMP.

Invoke Parse to check the syntax of the operands.

2 Check the Parse return code.

non-zero—means an operand was not valid and prompting failed. Issue an error message unless the return code indicates the user was in noprompt mode. Return to the caller in any case.

zero—means Parse was successful.

3 Set the UPT (The User Profile Table) to conform to the user options, checked by Parse.

4 If a new line or character deletion character was among the operands, issue a STCC macro to change the terminal line or character deletion characters. Check the return-code.

- non-zero—means reissue a STCC macro with the former line or character delete characters, and issue an error message.
- zero—means issue SVC100 to update the PSCB with the new line-delete and character-delete change requests.

5 If the operand LIST has been specified, list the users profile.

6 Free storage, set the return code, and return to the TMP.

zero—means successful processing.

non-zero—means unsuccessful processing.

Object Module: IKJEFT82

Diagram 15. PROTECT Command Processing (Part 1 of 2)

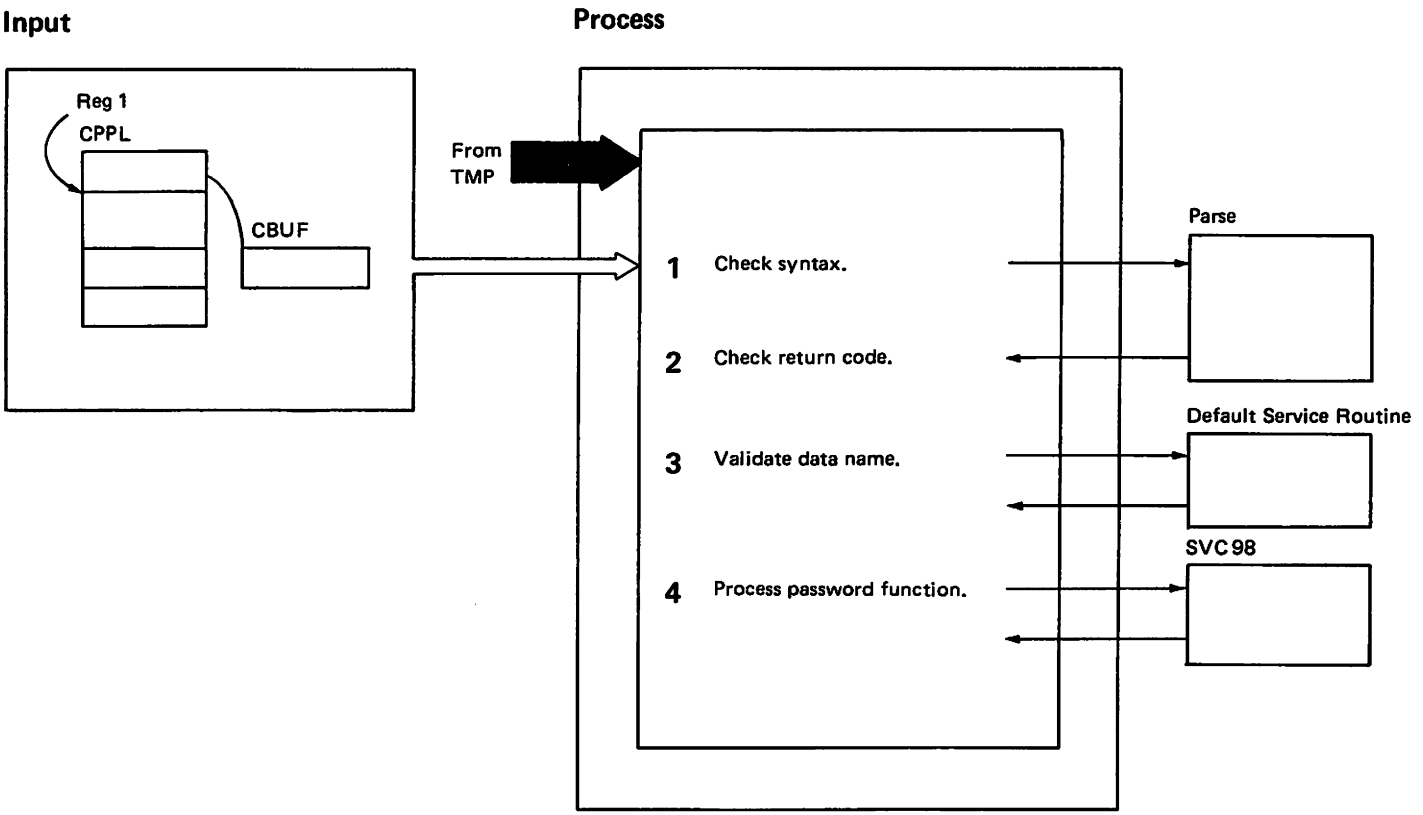


Diagram 15. PROTECT Command Processing (Part 2 of 2)

1 Use Parse to scan and check the command for proper syntax.
Possible messages: IKJ58102I, IKJ58112I

2 After checking the Parse return code, move the control password, if one was specified, to the SVC98 buffer.
Possible message: IKJ58108I

3 If the data set name was not fully qualified, it is fully qualified using IKJEHDEF (The Default Service Routine).
Possible messages: IKJ58103I, IKJ58111I, IKJ58112I

4 Check the function to be performed, and fill in the parameter list (SVCPARMS) for SVC 98 accordingly. The first byte of the parameter list contains a hexadecimal value indicating the function, as follows.

- X'01' ADD an entry to the password data set.
- X'02' REPLACE an entry in the password data set.
- X'03' DELETE an entry from the password data set.
- X'04' LIST protection, security counter, and optional data information of a protected data set. (The last 80 bytes of the password data set entry for this data set password is placed in the 80 byte buffer pointed to by the SVC parameter list.)

Issue SVC 98 return control to the TMP issuing error messages, depending on the return code provided by SVC 98.
Possible messages: IKJ58101I, IKJ58101I, IKJ58104I, IKJ58105I, IKJ58106I, IKJ58107I, IKJ58110I, IKJ58112I

Object Module: IKJEHPRO

Diagram 16. RENAME Command Processing (Part 1 of 2)

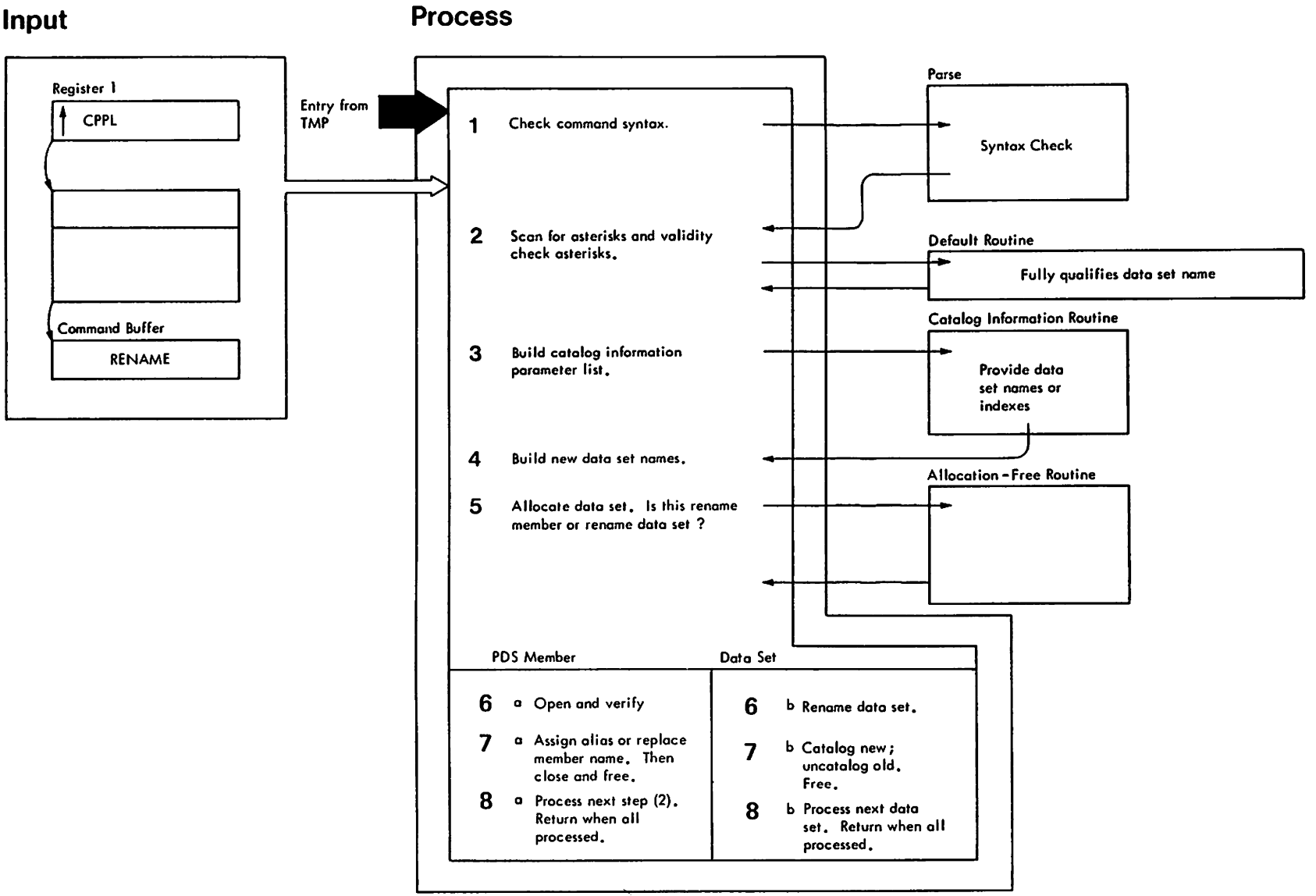


Diagram 16. RENAME Command Processing (Part 2 of 2)

1 The Parse subroutine syntax checks the command.
Possible messages: IKJ58202I, IKJ58223I

Storage is obtained for work areas.

2 A user id is prefixed if necessary. The data set name is scanned for asterisks. If none, prompting is done for any necessary qualification of data set names by the default routine. Then operation continues from step 4. If asterisks are found, they are checked to ensure that they occur in the same relative position within the fully qualified data set names.
Possible messages: IKJ58206I, IKJ58208I, IKJ58209I, IKJ58218I, IKJ58225I, IKJ58227I

3 If asterisks were found, the catalog information routine is used to look up candidates for renaming.
Possible messages: IKJ58201I, IKJ58219I

4 The new data set names are built in preparation for the renaming operation.
Possible messages: IKJ58205I, IKJ58208I

5 Allocation is done to make use of the system enqueueing facility which ensures that the data set is not renamed while some other user is using it. (Also, this enables the OPEN and CLOSE operation for partitioned members.)
Possible messages: IKJ58201I, IKJ58202I, IKJ58211I, IKJ58212I, IKJ58213I, IKJ58214I, IKJ58215I, IKJ58229I, IKJ58225I, IKJ58229I

6a OPEN and BLDL are used to open the PDS.
Possible messages: IKJ58203I, IKJ58204I, IKJ58207I, IKJ58217I

7a STOW is used to assign the alias or new member name.
Possible messages: IKJ58207I, IKJ58217I, IKJ58223I, IKJ58226I

8a The data set is closed and unallocated.
Possible messages: IKJ58201I, IKJ58207I, IKJ58216I, IKJ58222I, IKJ58224I

6b RENAME is used to rename the data set.

7b CATALOG is used to catalog new and uncatalog old.
Possible messages: IKJ58210I, IKJ58227I, IKJ58228I

8b Repeat from step 4, if applicable.
Possible messages: IKJ58210I, IKJ58227I, IKJ58228I

Object Module: IKJEHREN

Diagram 17.1. RUN Command Processing Overview (Part 1 of 2)

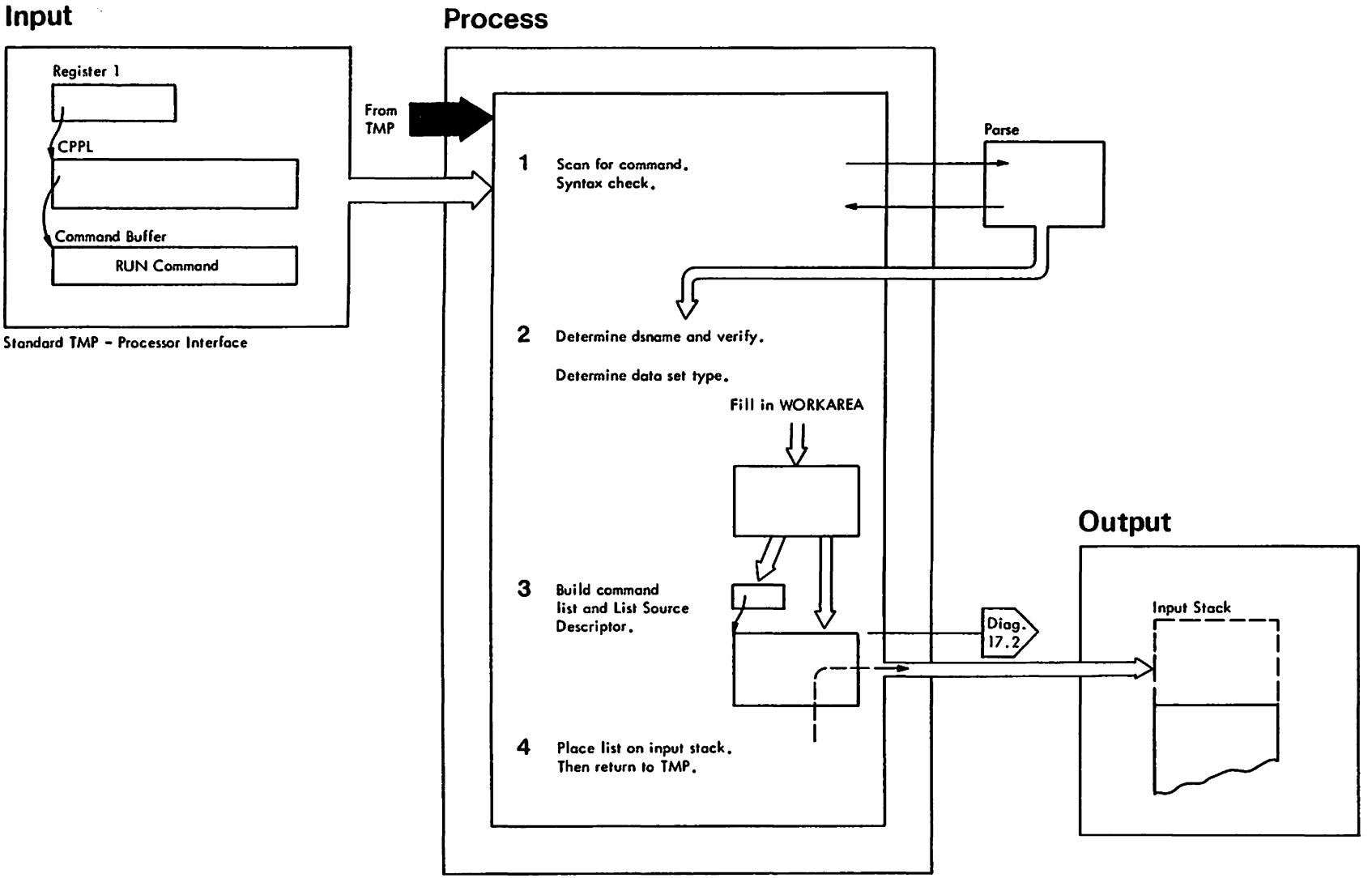


Diagram 17.1. RUN Command Processing Overview (Part 2 of 2)

Extended Description

1 IKJEFR00 uses Parse to scan and syntax check the RUN command. Prompting occurs if required parameters are missing or if syntactically incorrect parameters are present.

2 Upon return from Parse, the return code is checked. If an error was encountered, a message is issued to the user; otherwise, processing continues.

Control passes to a routine that examines the specified data set or member name and places applicable information into a buffer in WORKAREA. If the data set is fully qualified, an indicator is set. If a password is specified, the password and length are placed in WORKAREA.

Then the data set type (ASM, etc) is determined and placed in the data set type buffer of WORKAREA. Parse is again used, if necessary, to prompt for the data set type.

DAIR is then given control to search for a data set having the specified name. First the set of currently allocated data sets is searched; then if necessary, the system catalog. If the data set is found, processing continues; otherwise, the user is prompted for a respecification, and another search is made.

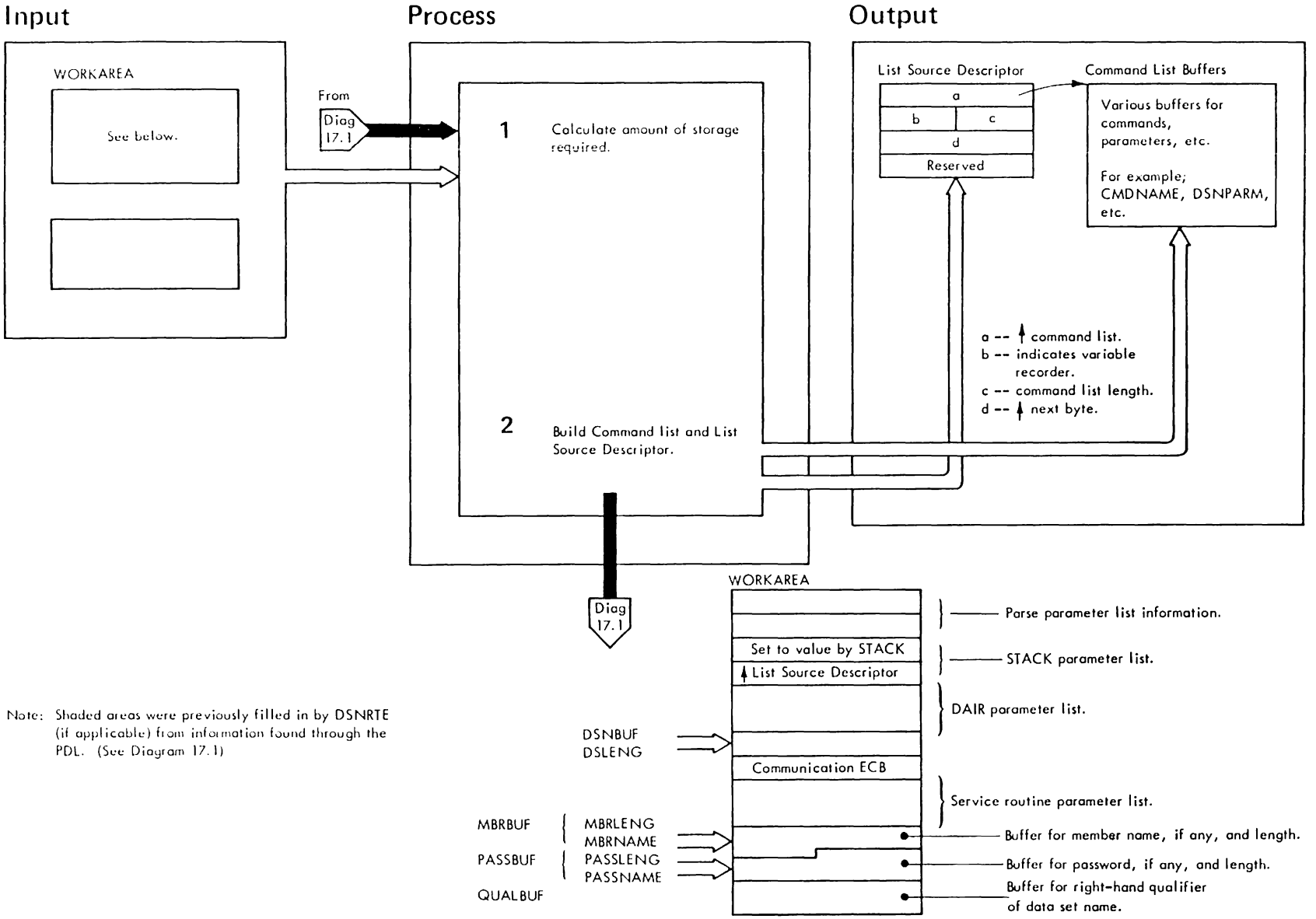
3 When a data set is verified as existing, storage is obtained in shared subpool 78 for an in-storage command list and a table (the List Source Descriptor) describing the list. See Diagram 17.2 for details of this operation.

4 After the in-storage command list and List Source Descriptor are built, the address of the List Source Descriptor is placed in the STACK parameter list and control is passed to Stack. This routine places the command list on the input stack.

Then control returns to the TMP. The TMP will select the next command from the top of the input stack.

Object Module: IKJEFR00

Diagram 17.2. Building a RUN Command List (Part 1 of 2)



Note: Shaded areas were previously filled in by DSNRTE (if applicable) from information found through the PDL. (See Diagram 17.1)

Diagram 17.2. Building a RUN Command List (Part 2 of 2)

Extended Description

1 WORKAREA fields and parse information previously located through the PDL are examined to calculate the amount of storage required for the command list. Included in the calculation are:

- The length of the List Source Descriptor (16 bytes).
- The size of the compiler command.
- The length of the data set name.
- Compiler parameters, if any.
- LOADGO command size (for ASM, FORT, PLI with OPT operand, or COBOL). This size includes control information length; LOADGO length; LOADGO data set name length; the length of the WHEN/END command, which is used to prevent execution of the program in the event the compiler does not complete successfully; parameter information, if any; COBLIB, PLIBASE, and FORTLIB length (for COBOL, PLI with OPT operand, and FORT data sets, LIB operand length, including the length of the data set list contained within parentheses).

2 The parameters are checked for validity with compiler types. Issue a message if they are invalid.

The command list and List Descriptor are built. The List Source Descriptor is filled in as the command list is constructed.

First, the compiler command (type) is built. Control information consists of a two-byte length field followed by two bytes containing 0. The compiler command is moved to the appropriate buffer (CMDNAME).

Then the data set name is moved to the command list buffer (DSNPARM).

If a compiler parameter is specified (for BASIC, IPLI, or GOFORT), it is placed in the buffer, along with the parameter length.

If the compiler is ASM, FORT, PLI with OPT operand, or COBOL, the WHEN command is built and placed in the list. Then the LOADGO command is created. This consists of placing in the buffer the proper data set name, applicable parameters, and for the FORT, PLIBASE, and COBOL data sets, FORTLIB or COBLIB, respectively. The length of the LOADGO command is placed in the control field.

After the command list is complete, it is placed on the input stack (See step 4 of Diagram 17.1)

Object Module: IKJEFR00

Diagram 18.1. SEND Overview and Operator Processing (Part 1 of 2)

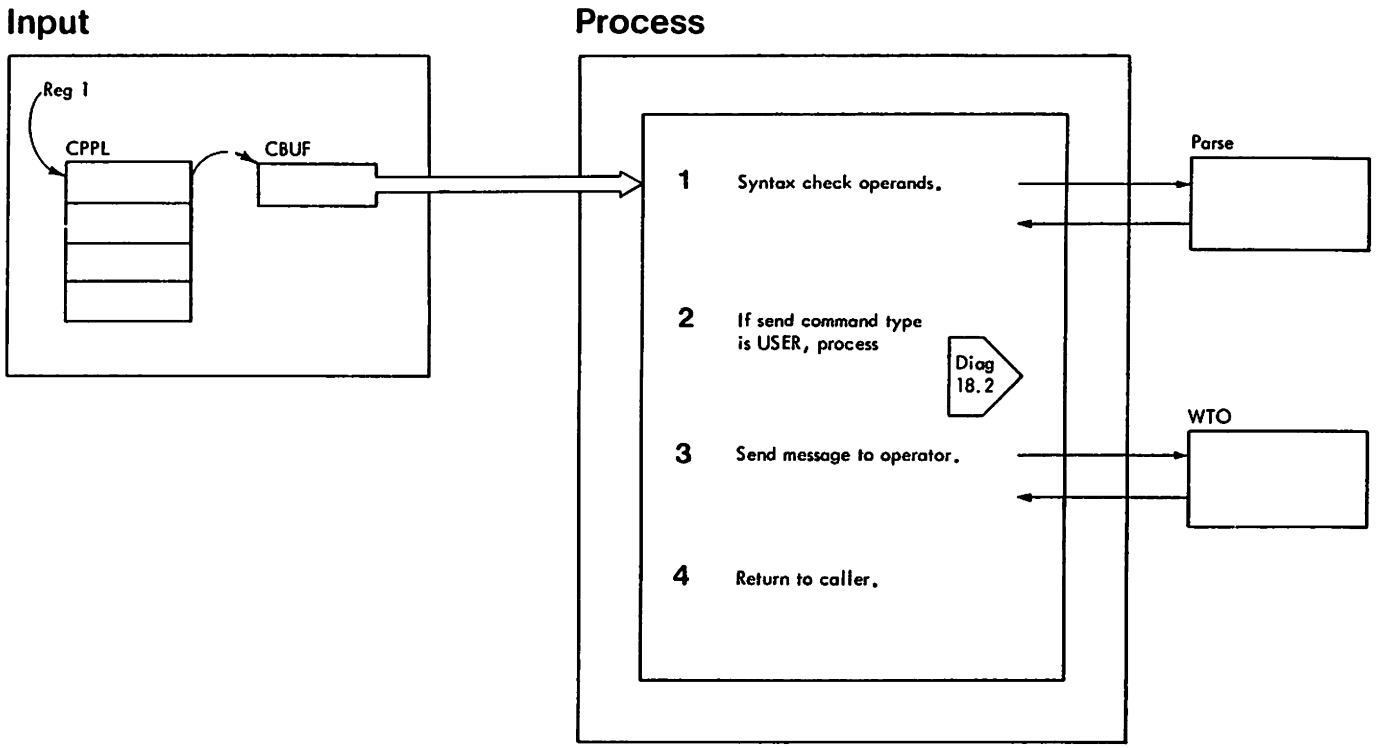


Diagram 18.1. SEND Overview and Operator Processing (Part 2 of 2)

1 Use Parse to syntax check operands.

2 If Parse output of command operands shows the command type is USER, process. See Diagram 18.2.

3 Otherwise command type is Operator or console id. Issue a WTO macro instruction.

4 Return control to the TMP.

Object Module: IKJEES10

Diagram 18.2. SEND User Processing (Part 1 of 2)

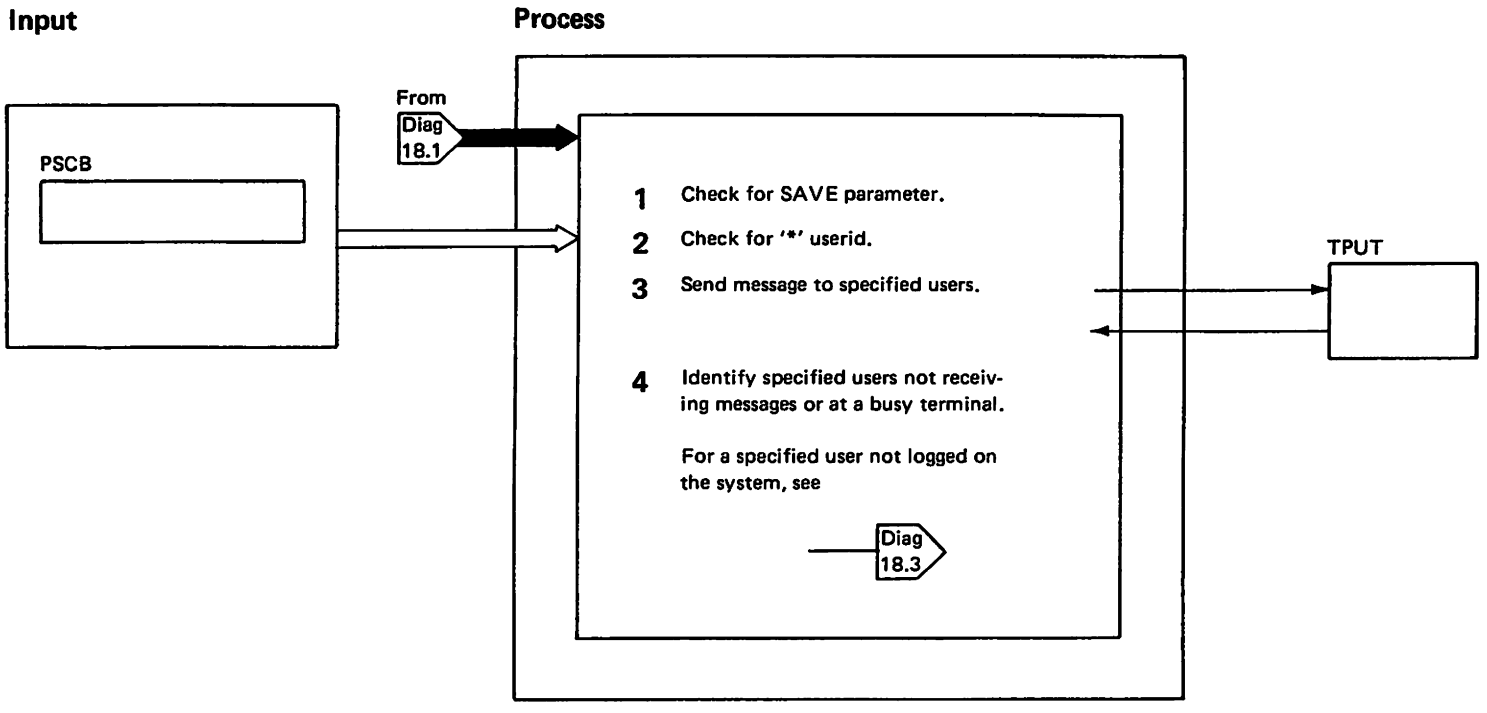


Diagram 18.2. SEND User Processing (Part 2 of 2)

1 Check SEND command for a SAVE parameter. If the SAVE parameter was used, see Diagram 18.3.

2 Check for an '*' used as a userid. If an '*' was used, get the userid from the Protected Step Control Block (PSCB).

3 Use TPUT (SVC93) to send a message to all users specified in the SEND command.

4 Send a warning message to the issuer of the SEND command identifying all specified users not receiving messages or at a busy terminal.

For users specified in the SEND command but not logged on the system, see Diagram 18.3.

Object Module: IKJEES11

Diagram 18.3. Adding SEND Text to the Broadcast Data Set (Part 1 of 2)

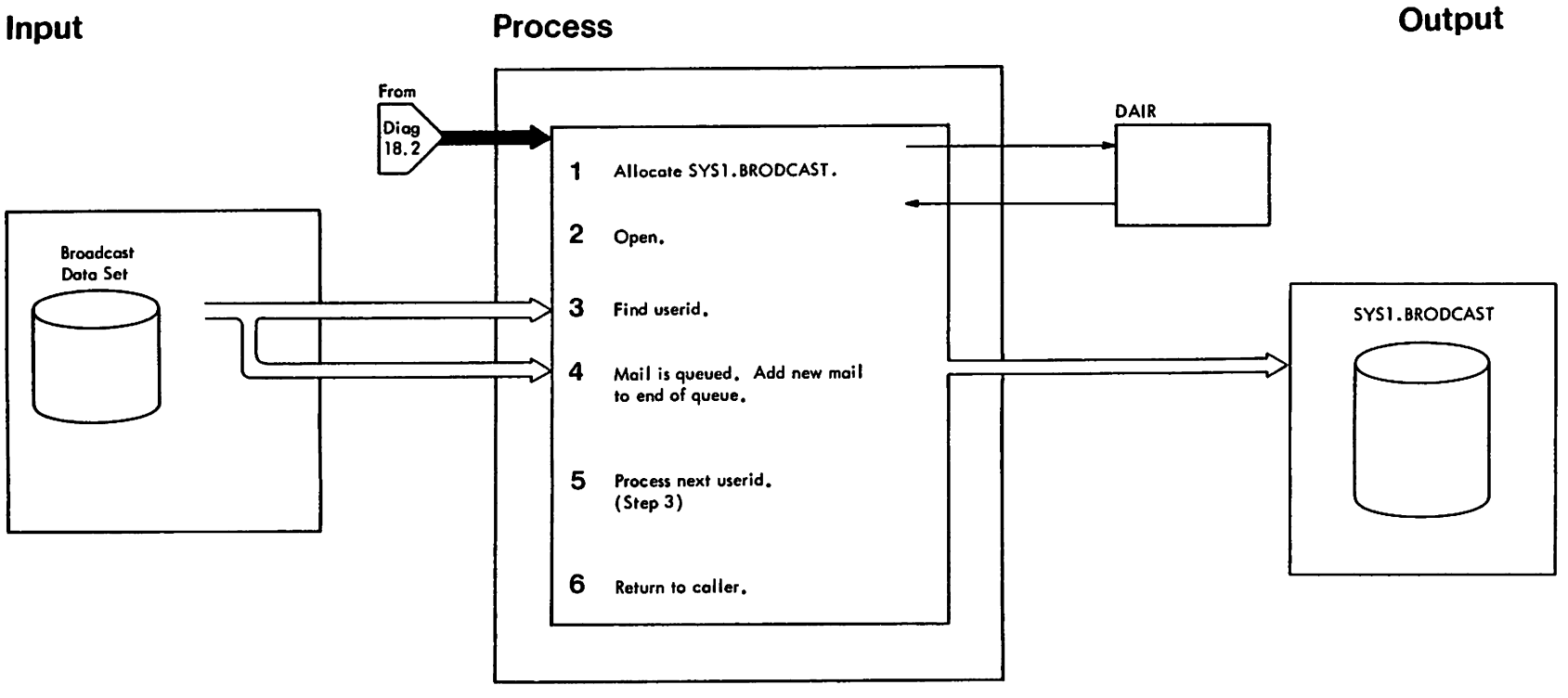


Diagram 18.3. Adding SEND Text to the Broadcast Data Set (Part 2 of 2)

1 Allocate SYS1.BRODCAST Using DAIR.

2 On successful completion, open SYS1.BRODCAST and enqueue on record 1.

3 Search the directory to find an entry for the userid.

4 Messages are queued in the data set. Put the SEND message text on the queue.

5 Continue processing all userids.

6 Close SYS1.BRODCAST and free it using DAIR. Return to the caller.

Diagram 19.1. SUBMIT Processing (Part 1 of 2)

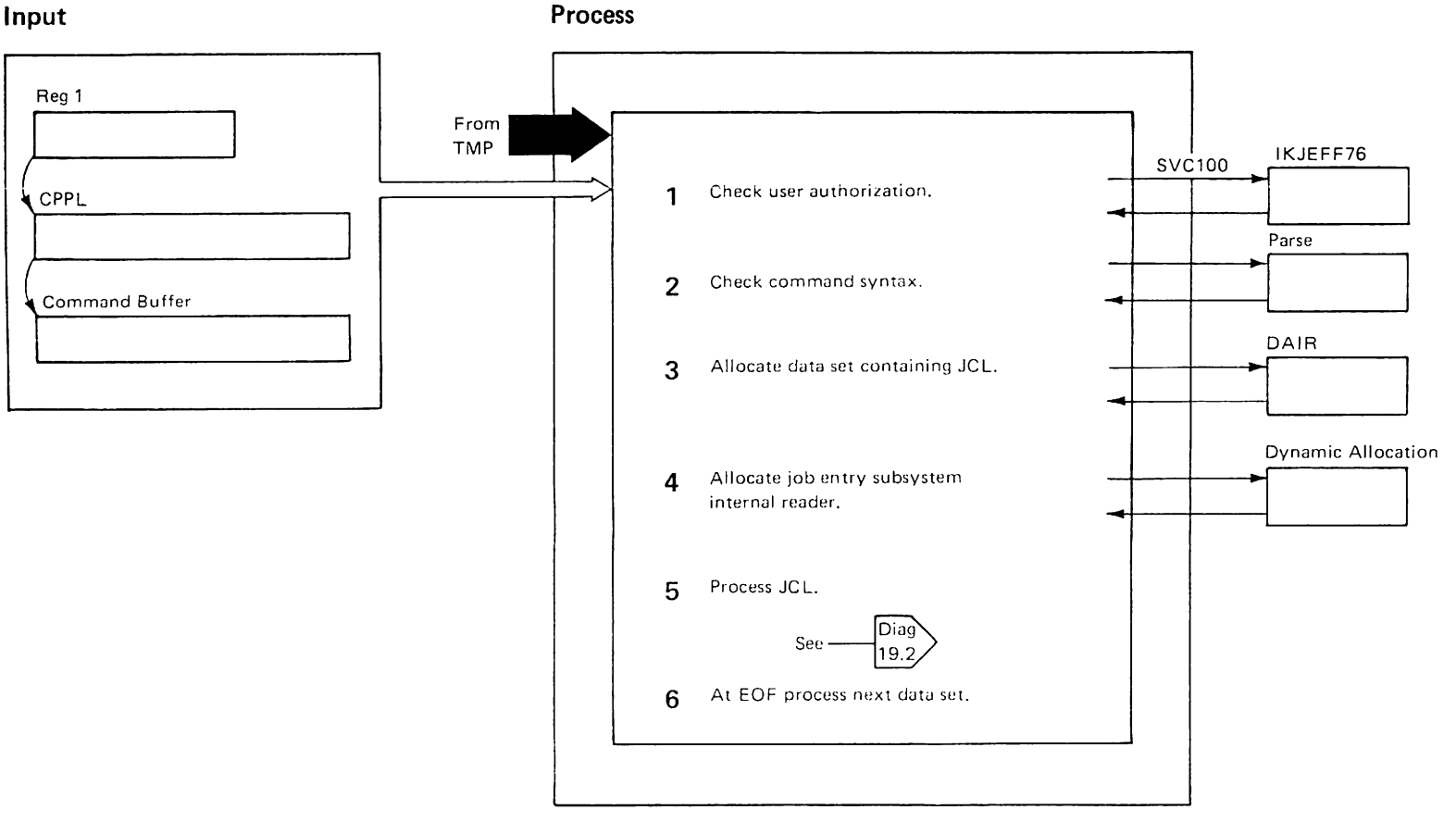


Diagram 19.1. SUBMIT Processing (Part 2 of 2)

1 Use SVC100 to post the TMP (IKJEFTSC) requesting IKJEFF76 be attached under a parallel task structure. Information is passed to IKJEFF76 in the FIBPARMS parameter list. IKJEFF76 checks the user's authorization to enter the command. If the user is not authorized to enter foreground initiated background commands, the system issues an error message and returns control to the TMP.
Object Module: IKJEFF01

2 Use Parse to check the command syntax. Parse validity check exit IKJEFF16 is entered to get the fully qualified data set name from the Default Service Routine.
Object Module: IKJEFF04

3 Allocate, using DAIR, input data sets containing JCL. Also build control and history tables for JCL processing.
Object Module: IKJEFF04

4 Use Dynamic Allocation to allocate a job entry subsystem internal reader and open the internal reader.
Object Module: IKJEFF15

For an attention or ABEND, the internal reader will be closed and the last job submitted will be flushed.
Object Modules: IKJEFF20, IKJEFF15

5 Read JCL statement and process.
Object Module: IKJEFF05. See Diagram 19.2

6 Process the next data set when an end-of-file is encountered. After the last file is processed, control is returned to the TMP.
Object Module: IKJEFF05

Diagram 19.2. SUBMIT JCL Processing (Part 1 of 2)

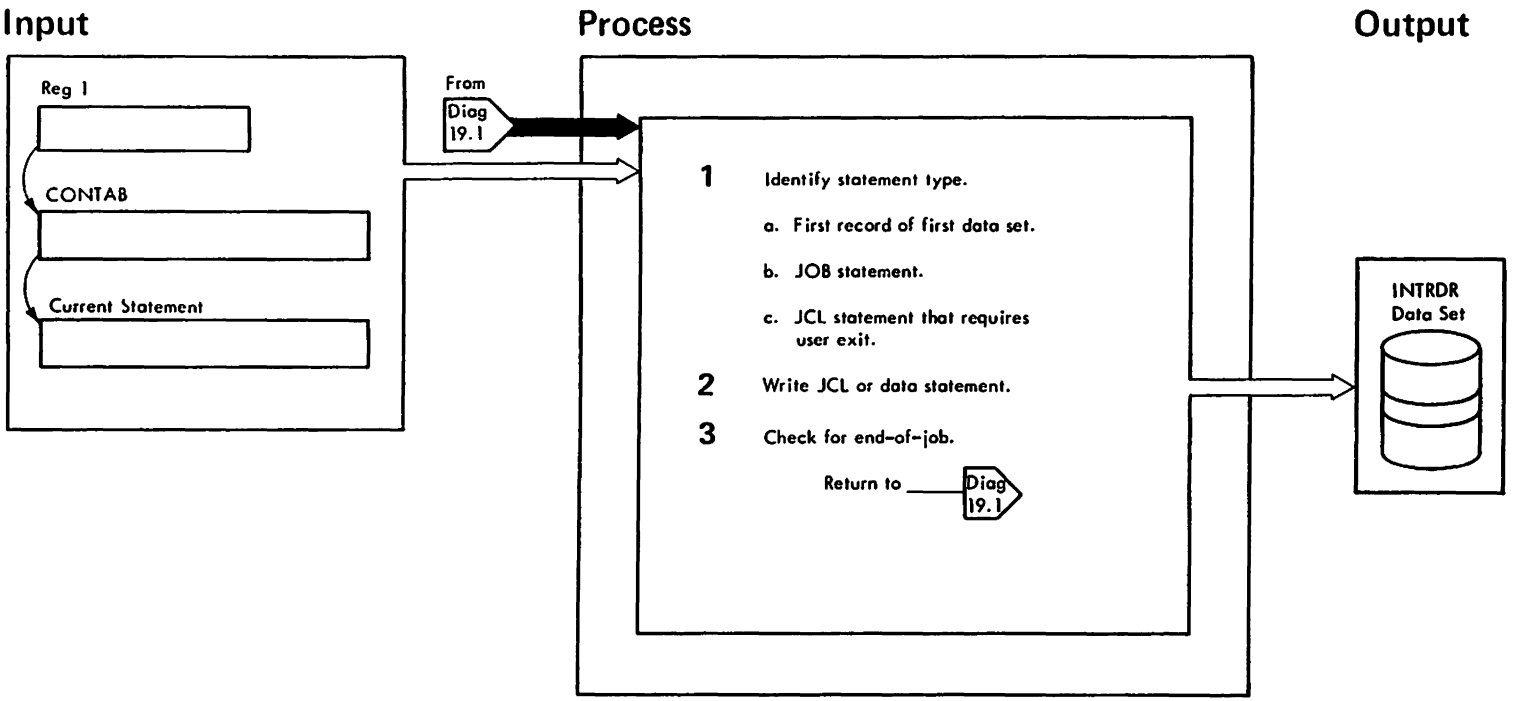


Diagram 19.2. SUBMIT JCL Processing (Part 2 of 2)

1 Identify the statement as data or type of JCL.
Object Module: IKJEFF07

- First record that is not a subsystem control card for first data set and not a JOB statement, create a JOB statement.
  Object Module: IKJEFF08
- JOB statement. Verify that the job name is not equal to the userid. If it is equal, the user is prompted for an identifying character.
  Object Module: IKJEFF13
- JCL statement that requires user exit (IKJEFF10) for installation required processing.
  Object Module: IKJEFF09, IKJEFF10

2 Write JCL or data statement to job entry subsystem internal reader data set.

3 Check for end-of-job. If an end-of-job, get a jobid from the job entry subsystem for the 'job submitted' message. If not end-of-job get the next statement. (See Diagram 19.1.)

Object Module: IKJEFF05

Diagram 20. TERMINAL Operational Characteristics (Part 1 of 2)

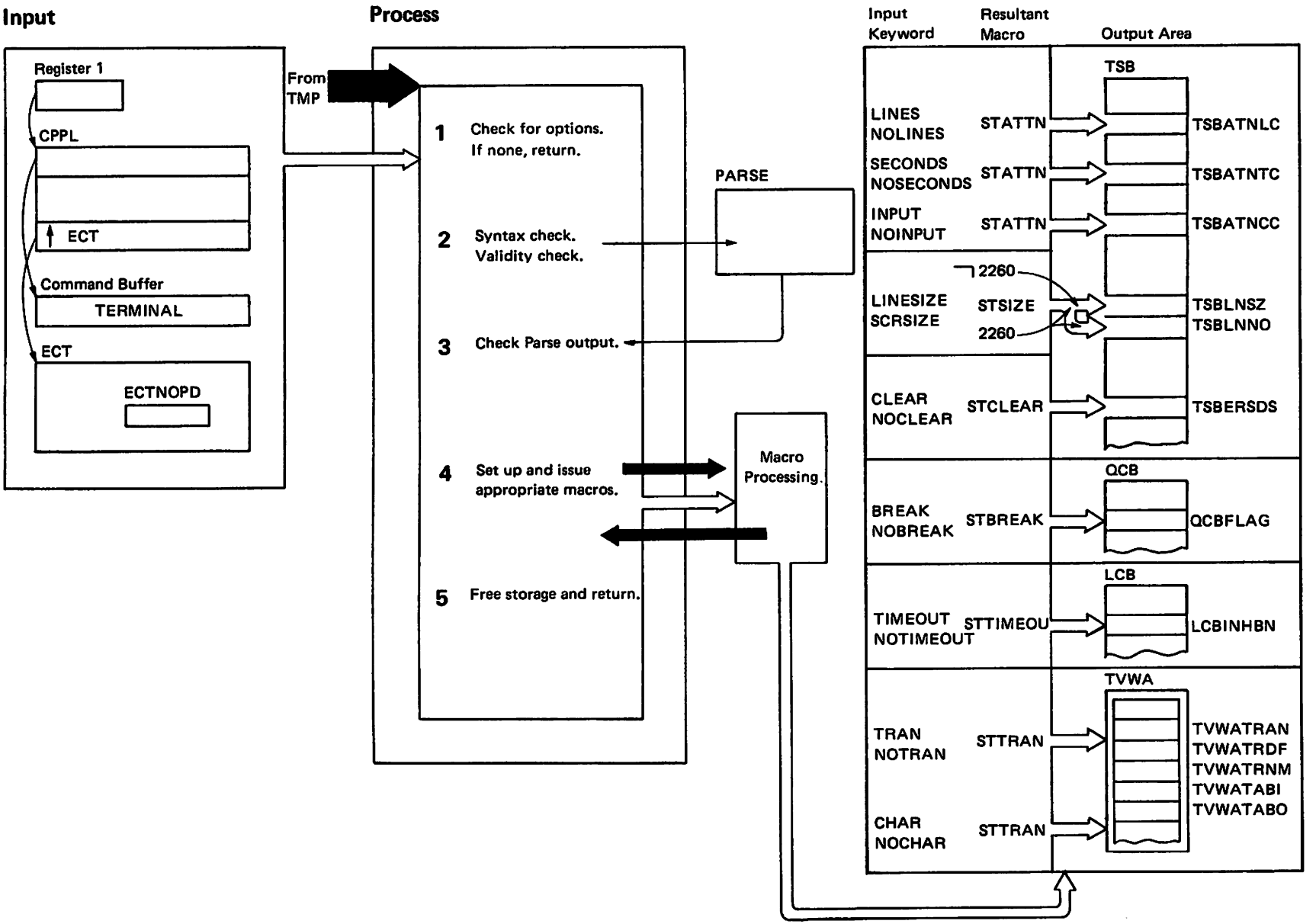


Diagram 20. TERMINAL Operational Characteristics (Part 2 of 2)

Extended Description

1 Check the ECTNOPD bit of the ECT to see if there are any TERMINAL keywords. If none, the command is ignored, a message is printed and control returns to the TMP.

2 If LINE or SECOND keywords are provided, Parse will temporarily pass control back to the appropriate routine for validity checking. When Parse returns control to IKJEFT80 for mainline processing, it provides a return code that informs IKJEFT80 of the results of the validity checks.

3 IKJEFT80 checks the PDL pointer (PDEPTR) to see if the PDL is zeros. If it is, the terminal command is ignored, a message is written and control returns to the TMP.

4 The TERMINAL status macros are set up and issued according to the specified keyword values. The end result is the placing of values in output fields, as shown at the right of this diagram.

5 TERMINAL frees main storage, sets return codes and returns control to the TMP.

zero means successful processing.

non-zero ('12') means unsuccessful processing.

Object Module: IKJEFT80

Diagram 21. TIME Command Processing (Part 1 of 2)

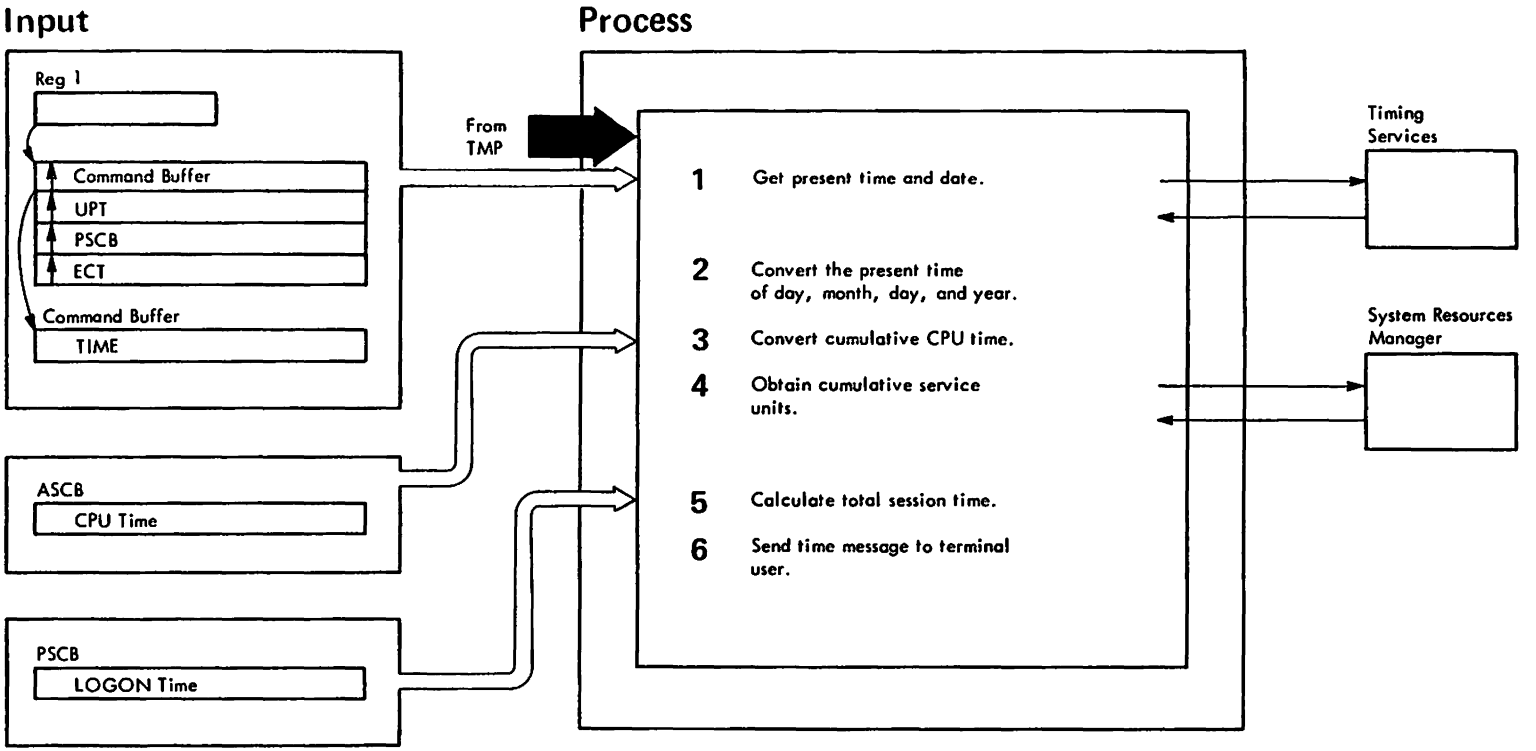


Diagram 21. TIME Command Processing (Part 2 of 2)

1 The TIME macro obtains the present time of day and date. If Timing Services indicates the hardware clock is inoperative, control is passed to TIMERR and an error message is issued. Control is then returned to the TMP.

2 Convert the time and date returned by the TIME macro to printable format and save in a message buffer.

3 Convert the cumulative CPU time to printable format and save in a message buffer. The CPU time is maintained in the ASCB in store-clock units.

4 Use SVC95 (SYSEVENT), issued with code 38, to obtain cumulative service units from the System Resources Manager. Convert the returned data to a printable format and save in a message buffer.

5 Calculate total session time by subtracting the LOGON time of day from the present time. The LOGON time of day is maintained in the PSCB. Convert the session time to printable format and save in a message buffer.

6 Use the PUTLINE service routine to send the TIME message to the terminal. Control is returned to the TMP.

Object Module: IKJEFT25

Diagram 22. WHEN/END Processing (Part 1 of 2)

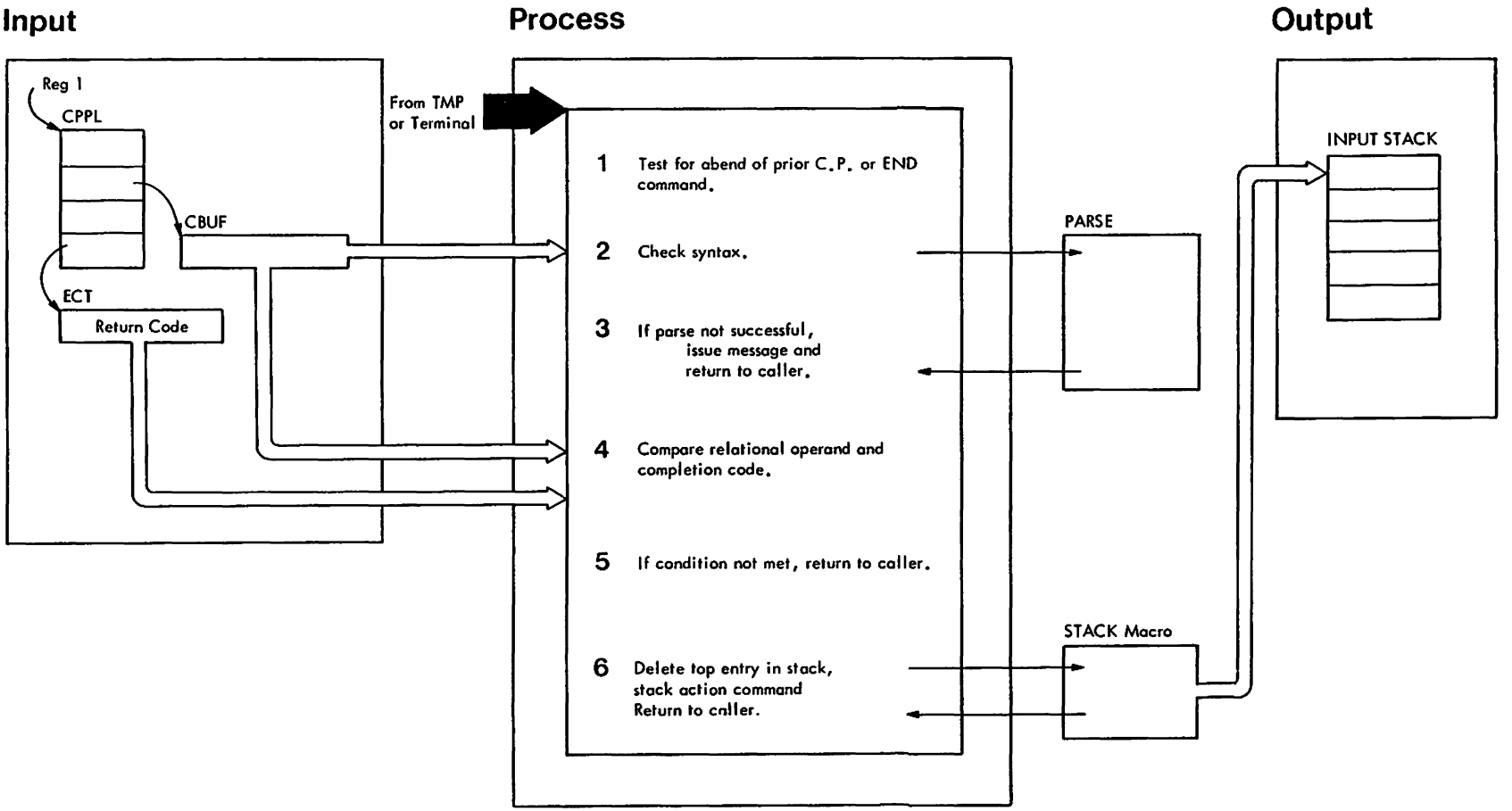


Diagram 22. WHEN/END Processing (Part 2 of 2)

Extended Description

1 Test for previous command processor abend. If abend occurred, issue message and return to caller. If the command is END, delete the top element from the stack and return to the caller.

2 Use Parse to check the syntax of the WHEN command.

3 Check Parse return code, if non-zero Parse was unsuccessful. Issue a message and return to the caller,

4 Compare the WHEN relational operator with the completion code of the previous command processor.

5 If condition is not met, return to the caller.

6 If condition is met, use STACK to delete the top entry in the input stack and add the action command to the stack if it is not the END command. Return to the caller.

Object Module: IKJEFE11

Name	Type	Load	Object	Entry	Alias	CP
AKJLKL01	Object	AKJLKL01		AKJLKL01		LINK/LOADGO
	Entry	AKJLKL01	AKJLKL01			LINK/LOADGO
AKJLKL02	Object	AKJLKL02		AKJLKL02		LINK/LOADGO
	Entry	AKJLKL02	AKJLKL02			LINK/LOADGO
AKJLKMSG	Object	AKJLKL01				LINK/LOADGO
ALLOC	Alias	ALLOCATE	IKJEFD30	IKJEFD30		ALLOCATE
ALLOCATE	Load		IKJEFD30	IKJEFD30	ALLOC	ALLOCATE
			IKJEFD31			ALLOCATE
			IKJEFD32			ALLOCATE
			IKJEFD33			ALLOCATE
			IKJEFD34			ALLOCATE
			IKJEFD35			ALLOCATE
			IKJEFD36			ALLOCATE
			IKJEFD37			ALLOCATE
ATTR	Alias	ATTRIB	IKJEFATT	IKJEFATT		ATTRIB
ATTRIB	Load		IKJEFATT	IKJEFATT	ATTR	ATTRIB
CALL	Alias	IKJEFT02	IKJEFT08	IKJEFT08		CALL
CANCEL	Load		IKJEFF58	IKJEFF58		CANCEL/STATUS
END	Alias	IKJEFE11	IKJEFE11	IKJEFE11		WHEN/END
EX	Alias	EXEC	IKJCT430	IKJCT430		EXEC
EXEC	Load		IKJCT430	IKJCT430	EX	EXEC
			IKJCT431			EXEC
			IKJCT432			EXEC
			IKJCT435			EXEC
FREE	Load		IKJEFD20	IKJEFD20		FREE
H	Alias	HELP	IKJEFH01	IKJEFH01		HELP
HELP	Load		IKJEFH00			HELP
			IKJEFH01	IKJEFH01	H	HELP
			IKJEFH02			HELP
			IKJEFH03			HELP
IEEVSDIO	Object	SEND				SEND
IGC0010{	Load		IKJEFF00	IKJEFF00		See Note
({=x'C0')						
			IKJEFF20			See Note
IKJCT430	Object	EXEC		IKJCT430		EXEC
	Entry	EXEC	IKJCT430			EXEC
IKJCT431	Object	EXEC				EXEC
IKJCT432	Object	EXEC				EXEC
IKJCT435	Object	EXEC				EXEC
IKJCT460	Object	IKJCT469				OUTPUT
IKJCT462	Object	IKJCT469				OUTPUT
IKJCT463	Object	IKJCT469				OUTPUT
IKJCT464	Object	IKJCT469				OUTPUT
IKJCT466	Object	OUTPUT		IKJCT466	OUT	OUTPUT
	Entry	OUTPUT	IKJCT466		OUT	OUTPUT
IKJCT467	Object	IKJCT469			IKJCT467	
	Alias	IKJCT469	IKJCT467			
IKJCT469	Load		IKJCT460			OUTPUT
			IKJCT462			OUTPUT
			IKJCT463			OUTPUT
			IKJCT464			OUTPUT
			IKJCT469	IKJCT469		OUTPUT
			IKJCT470			OUTPUT
			IKJCT471			OUTPUT
			IKJCT472			OUTPUT
			IKJCT473			OUTPUT
	Object	IKJCT469		IKJCT469		OUTPUT
	Entry	IKJCT469	IKJCT469			OUTPUT

Note: CANCEL/OPERATOR/OUTPUT/PROFILE/STATUS/SUBMIT

Name	Type	Load	Object	Entry	Alias	CP
IKJCT470	Object	IKJCT469				OUTPUT
IKJCT471	Object	IKJCT469				OUTPUT
IKJCT472	Object	IKJCT469				OUTPUT
IKJCT473	Object	IKJCT469				OUTPUT
IKJEES10	Object	SEND		IKJEES10	SE	SEND
	Entry	SEND	IKJEES10		SE	SEND
IKJEES11	Object	SEND				SEND
IKJEES20	Object	SEND				SEND
IKJEES70	Object	IKJEES73		IKJEES73		LISTBC
		LISTBC		IKJEES70	LISTB	LISTBC
	Entry	LISTBC	IKJEES70		LISTB	LISTBC
IKJEES73	Load		IKJEES70	IKJEES73		LISTBC
			IKJEES74			LISTBC
			IKJEES75			LISTBC
	Entry	IKJEES73	IKJEES70			LISTBC
IKJEES74	Object	IKJEES73				LISTBC
		LISTBC				LISTBC
IKJEES75	Object	IKJEES73				LISTBC
		LISTBC				LISTBC
IKJEE1A0	Object	OPERATOR				OPERATOR
IKJEE100	Object	OPERATOR		IKJEE101		OPERATOR
				IKJEE100	OPER	OPERATOR
	Entry	OPERATOR	IKJEE100		OPER	OPERATOR
IKJEE101	Entry	OPERATOR	IKJEE100			OPERATOR
IKJEE150	Object	OPERATOR				OPERATOR
IKJEFATT	Object	ATTRIB		IKJEFATT	ATTR	ATTRIB
	Entry	ATTRIB	IKJEFATT		ATTR	ATTRIB
IKJEFD20	Object	FREE		IKJEFD20		FREE
	Entry	FREE	IKJEFD20			FREE
IKJEFD30	Object	ALLOCATE		IKJEFD30	ALLOC	ALLOCATE
	Entry	ALLOCATE	IKJEFD30		ALLOC	ALLOCATE
IKJEFD31	Object	ALLOCATE				ALLOCATE
IKJEFD32	Object	ALLOCATE				ALLOCATE
IKJEFD33	Object	ALLOCATE				ALLOCATE
IKJEFD34	Object	ALLOCATE				ALLOCATE
IKJEFD35	Object	ALLOCATE				ALLOCATE
IKJEFD36	Object	ALLOCATE				ALLOCATE
IKJEFD37	Object	ALLOCATE				ALLOCATE
IKJEFE11	Load		IKJEFE11	IKJEFE11	WHEN	WHEN/END
					END	WHEN/END
			IKJEFE15			WHEN/END
			IKJEFE16			WHEN/END
	Object	IKJEFE11		IKJEFE11	WHEN	WHEN/END
					END	WHEN/END
	Entry	IKJEFE11	IKJEFE11		WHEN	WHEN/END
					END	WHEN/END
IKJEFE15	Object	IKJEFE11				WHEN/END
IKJEFE16	Object	IKJEFE11				WHEN/END
IKJEFFCA	Alias	STATUS	IKJEFF56	IKJEFF56		CANCEL/STATUS
IKJEFF00	Object	IGC0010{ ({=X'C0')		IKJEFF00		See Note
	Entry	IGC0010{ ({=X'C0')	IKJEFF00			See Note
IKJEFF01	Object	SUBMIT		IKJEFF01	SUB	SUBMIT
	Entry	SUBMIT	IKJEFF01		SUB	SUBMIT
IKJEFF02	Load		IKJEFF02	IKJEFF02		SUBMIT
	Object	IKJEFF02		IKJEFF02		SUBMIT
	Entry	IKJEFF02	IKJEFF02			SUBMIT
IKJEFF03	Object	IKJEFF04			IKJEFF03	SUBMIT
	Alias	IKJEFF04	IKJEFF03			SUBMIT
IKJEFF04	Load		IKJEFF02			SUBMIT
			IKJEFF03		IKJEFF03	SUBMIT
			IKJEFF04	IKJEFF04		SUBMIT
			IKJEFF05			SUBMIT
			IKJEFF07			SUBMIT

Note: CANCEL/OPERATOR/OUTPUT/PROFILE/STATUS/SUBMIT

Name	Type	Load	Object	Entry	Alias	CP
			IKJEFF08			SUBMIT
			IKJEFF09			SUBMIT
			IKJEFF13			SUBMIT
			IKJEFF15			SUBMIT
			IKJEFF16			SUBMIT
	Object	IKJEFF04		IKJEFF04		SUBMIT
	Entry	IKJEFF04	IKJEFF04			SUBMIT
IKJEFF05	Object	IKJEFF04				SUBMIT
IKJEFF07	Object	IKJEFF04				SUBMIT
IKJEFF08	Object	IKJEFF04				SUBMIT
IKJEFF09	Object	IKJEFF04				SUBMIT
IKJEFF10	Load		IKJEFF10	IKJEFF10		SUBMIT
	Object	IKJEFF10		IKJEFF10		SUBMIT
	Entry	IKJEFF10	IKJEFF10			SUBMIT
IKJEFF13	Object	IKJEFF04				SUBMIT
IKJEFF15	Object	IKJEFF04				SUBMIT
IKJEFF16	Object	IKJEFF04				SUBMIT
IKJEFF19	Load		IKJEFF19	IKJEFF19		SUBMIT
	Object	IKJEFF19		IKJEFF19		SUBMIT
	Entry	IKJEFF19	IKJEFF19			SUBMIT
IKJEFF20	Object	IGC0010{ ({=x'C0')				See Note
IKJEFF49	Object	IKJEFF57				CANCEL/STATUS
IKJEFF50	Load		IKJEFF02			CANCEL/STATUS
			IKJEFF50	IKJEFF50		CANCEL/STATUS
			IKJEFF55		IKJEFF55	CANCEL/STATUS
	Object	IKJEFF50		IKJEFF50		CANCEL/STATUS
	Entry	IKJEFF50	IKJEFF50			CANCEL/STATUS
IKJEFF51	Load		IKJEFF51	IKJEFF51		CANCEL/STATUS
			IKJEFF52			CANCEL/STATUS
			IKJEFF54			CANCEL/STATUS
	Object	IKJEFF51		IKJEFF51		CANCEL/STATUS
	Entry	IKJEFF51	IKJEFF51			CANCEL/STATUS
IKJEFF52	Object	IKJEFF51				CANCEL/STATUS
IKJEFF53	Load		IKJEFF53	IKJEFF53		CANCEL/STATUS
	Object	IKJEFF53		IKJEFF53		
	Entry	IKJEFF53	IKJEFF53			
IKJEFF54	Object	IKJEFF51				CANCEL/STATUS
IKJEFF55	Object	IKJEFF50			IKJEFF55	CANCEL/STATUS
	Alias	IKJEFF50	IKJEFF55			CANCEL/STATUS
IKJEFF56	Object	STATUS		IKJEFF56	ST IKJEFFCA	CANCEL/STATUS
	Entry	STATUS	IKJEFF56		ST IKJEFFCA	CANCEL/STATUS
IKJEFF57	Load		IKJEFF59			CANCEL/STATUS
			IKJEFF57	IKJEFF57		CANCEL/STATUS
	Object	IKJEFF57		IKJEFF57		CANCEL/STATUS
	Entry	IKJEFF57	IKJEFF57			CANCEL/STATUS
IKJEFF58	Object	CANCEL		IKJEFF58		CANCEL/STATUS
	Entry	CANCEL	IKJEFF58			CANCEL/STATUS
IKJEFF76	Object	IGC0010{ ({=x'C0')		IKJEFF76		See Note
IKJEFF77	Object	IGC0010{ ({=x'C0')		IKJEFF77		See Note
IKJEFT08	Load		IKJEFT08	IKJEFT08	CALL	CALL
	Object	IKJEFT02		IKJEFT08	CALL	CALL
	Entry	IKJEFT02	IKJEFT08		CALL	CALL
IKJEFH00	Object	HELP				HELP
IKJEFH01	Object	HELP		IKJEFH01	H	HELP
	Entry	HELP	IKJEFH01		H	HELP
IKJEFH02	Object	HELP				HELP
IKJEFH03	Object	HELP				HELP
IKJEFR00	Load		IKJEFR00	IKJEFR00	RUN	RUN
					R	RUN

Note: CANCEL/OPERATOR/OUTPUT/PROFILE/STATUS/SUBMIT

Name	Type	Load	Object	Entry	Alias	CP
	Object	IKJEFR00		IKJEFR00	RUN	RUN
					R	RUN
	Entry	IKJEFR00	IKJEFR00		RUN	RUN
IKJEFT25	Load		IKJEFT25	IKJEFT25	TIME	TIME
	Object	IKJEFT25		IKJEFT25	TIME	TIME
	Entry	IKJEFT25	IKJEFT25		TIME	TIME
IKJEFT80	Object	TERMINAL		IKJEFT80	TERM	TERMINAL
	Entry	TERMINAL	IKJEFT80		TERM	TERMINAL
IKJEFT82	Object	PROFILE		IKJEFT82	PROF	PROFILE
	Entry	PROFILE	IKJEFT82		PROF	PROFILE
IKJEHAL1	Load		IKJEHAL1	IKJEHAL1	LISTA	LISTALC
	Object	IKJEHAL1		IKJEHAL1	LISTA	LISTALC
	Entry	IKJEHAL1	IKJEHAL1		LISTA	LISTALC
IKJEHDS1	Load		IKJEHDS1	IKJEHDS1	LISTDS	LISTDS
	Object	IKJEHDS1		IKJEHDS1	LISTDS	LISTDS
	Entry	IKJEHDS1	IKJEHDS1		LISTDS	LISTDS
IKJEHMEM	Load		IKJEHMEM	IKJEHMEM		LISTDS/LISTALC
	Object	IKJEHMEM		IKJEHMEM		LISTDS/LISTALC
	Entry	IKJEHMEM	IKJEHMEM			LISTDS/LISTALC
IKJEHPRO	Load		IKJEHPRO	IKJEHPRO	PROTECT	PROTECT
	Object	IKJEHPRO		IKHEHPRO	PROTECT	PROTECT
	Entry	IKJEHPRO	IKJEHPRO	IKJEHPRO	PROTECT	PROTECT
IKJEHREN	Load		IKJEHREN	IKJEHREN	RENAME	RENAME
	Object	IKJEHREN		IKJEHREN	RENAME	RENAME
	Entry	IKJEHREN	IKJEHREN		RENAME	RENAME
LINK	Load		LINK	LINK		LINK/LOADGO
	Object	LINK		LINK		LINK/LOADGO
	Entry	LINK	LINK			LINK/LOADGO
LISTA	Alias	IKJEHAL1	IKJEHAL1	IKJEHAL1		LISTALC
LISTB	Alias	LISTBC	IKJEES70	IKJEES70		LISTBC
LISTBC	Load		IKJEES70	IKJEES70	LISTB	LISTBC
			IKJEES74			LISTBC
			IKJEES75			LISTBC
LISTDS	Alias	IKJEHDS1	IKJEHDS1	IKJEHDS1		LISTDS
LOAD	Alias	LOADGO	LOADGO	LOADGO		LINK/LOADGO
LOADGO	Load		LOADGO	LOADGO	LOAD	LINK/LOADGO
	Object	LOADGO		LOADGO	LOAD	LINK/LOADGO
	Entry	LOADGO	LOADGO		LOAD	LINK/LOADGO
OPER	Alias	OPERATOR	IKJEE100	IKJEE100		OPERATOR
OPERATOR	Load		IKJEE1A0			OPERATOR
			IKJEE100	IKJEE101		OPERATOR
				IKJEE100	OPER	OPERATOR
			IKJEE150			OPERATOR
OUT	Alias	OUTPUT	IKJCT466	IKJCT466		OUTPUT
OUTPUT	Load		IKJCT466	IKJCT466	OUT	OUTPUT
PROF	Alias	PROFILE	IKJEFT82	IKJEFT82		PROFILE
PROFILE	Load		IKJEFT82	IKJEFT82	PROF	PROFILE
PROTECT	Alias	IKJEHPRO	IKJEHPRO	IKJEHPRO		PROTECT
R	Alias	IKJEFR00	IKJEFR00			RUN
RENAME	Alias	IKJEHREN	IKJEHREN	IKJEHREN		RENAME
RUN	Alias	IKJEFR00	IKJEFR00	IKJEFR00		RUN
SE	Alias	SEND	IKJEES10	IKJEES10		SEND
SEND	Load		IEEVSDIO			SEND
			IKJEES10	IKJEES10	SE	SEND
			IKJEES11			SEND
			IKJEES20			SEND
ST	Alias	STATUS	IKJEFF56	IKJEFF56		CANCEL/STATUS
STATUS	Load		IKJEFF56	IKJEFF56	ST	CANCEL/STATUS
					IKJEFFCA	CANCEL/STATUS
SUB	Alias	SUBMIT	IKJEFF01	IKJEFF01		SUBMIT
SUBMIT	Load		IKJEFF01	IKJEFF01	SUB	SUBMIT
TERM	Alias	TERMINAL	IKJEFT80	IKJEFT80		TERMINAL
TERMINAL	Load		IKJEFT80	IKJEFT80	TERM	TERMINAL
TIME	Alias	IKJEFT25	IKJEFT25	IKJEFT25		TIME
WHEN	Alias	IKJEFE11	IKJEFE11	IKJEFE11		WHEN/END

Acronym	Macro	Common Name	Command Processor	Module/Access
ACB	IFGACB	VSAM Access Method Control Block	OUTPUT SUBMIT	IKJCT462 (create) IKJEFF15 (create)
ALLOCWA	IKJZT430	ALLOCATE Work Area	ALLOCATE	IKJEFD30 (create) IKJEFD32 (alter) IKJEFD33 (alter) IKJEFD34 (alter) IKJEFD35 (alter) IKJEFD36 (alter) IKJEFD37 (alter)
COMPROC	IKJEXEC	Command Procedure Storage Block	EXEC	IKJCT431 (create) IKJCT430 (alter) IKJCT432 (alter)
CONTAB	IKJEFFCT	SUBMIT Internal Control Table	SUBMIT	IKJEFF04 (create) IKJEFF15 (alter)
CPPL	IKJCPPL	Command Processor Parameter List	OUTPUT	IKJCT463 (create)
CSOA	IKJCSOA	Command Scan Output Area	ALLOCATE EXEC OPERATOR	IKJEFD36 (create) IKJCT430 (create) IKJEE100 (create) IKJEE150 (create)
CSPL	IKJCSPL	Command Scan Parameter List	ALLOCATE EXEC OPERATOR OUTPUT	IKJEFD36 (create) IKJCT430 (create) IKJEE100 (create) IKJEE150 (create) IKJCT463 (create)
CTGFL	IEZCTCFL	VSAM Catalog Control Field List	LISTALC LISTDS RENAME	IKJEHAL1 (create) IKJEHDS1 (create) IKJEHCIR (create)
CTGPL	IEZCTGPL	VSAM Catalog Parameter List	LISTALC LISTDS RENAME	IKJEHAL1 (create) IKJEHDS1 (create) IKJEHCIR (create)
DAPB00	IKJDAP00	DAIR Parameter Block 00	EXEC	IKJCT430 (create)
DAPB04	IKJDAP04	DAIR Parameter Block 04	ALLOCATE LINK/LOADGO OUTPUT RUN	IKJEFD32 (create) AKJLKL01 (create) IKJCT473 (create) IKJEFR00 (create)
DAPB08	IKJDAP08	DAIR Parameter Block 08	CALL EXEC LINK/LOADGO LISTDS OUTPUT RENAME SUBMIT	IKJEFG00 (create) IKJCT430 (create) AKJLKL01 (create) IKJEHDS1 (create) IKJCT473 (create) IKJEHREN (create) IKJEFF04 (alter) IKJEFF16 (create)

Acronym	Macro	Common Name	Command Processor	Module/Access
DAPB0C	IKJDAP0C	DAIR Parameter Block 0C	LINK/LOADGO	AKJLKL01 (create)
DAPB10	IKJDAP10	DAIR Parameter Block 10	LINK/LOADGO	AKJLKL01 (create) AKJLKL02 (create)
DAPB18	IKJDAP18	DAIR Parameter Block 18	EXEC LINK/LOADGO OUTPUT PROTECT RENAME	IKJCT430 (create) AKJLKL01 (create) IKJCT473 (create) IKJEHPRO (create) IKJEHREN (create)
DAPB1C	IKJDAP1C	DAIR Parameter Block 1C	LINK/LOADGO OUTPUT	AKJLKL01 (create) IKJCT473 (create)
DAPB24	IKJDAP24	DAIR Parameter Block 24	HELP	IKJEFH01
DAPD28	IKJDAP28	DAIR Parameter Block 28	LINK/LOADGO SUBMIT	AKJLKL01 (create) IKJEFF04 (create)
DAPB2C	IKJDAP2C	DAIR Parameter Block 2C	LISTDS OPERATOR OUTPUT	IKJEHDS1 (create) IKJEE100 (create) IKJCT463 (create)
DAPB34	IKJDAP34	DAIR Parameter Block 34	LINK/LOADGO	AKJLKL01 (create)
DAPL	IKJDAPL	DAIR Parameter List	ALLOCATE CALL EXEC HELP LINK/LOADGO LISTALC LISTDS OPERATOR OUTPUT PROTECT RENAME RUN SUBMIT	IKJEFD32 (create) IKJEFG00 (create) IKJCT430 (create) IKJEFH01 (create) AKJLKL01 (create) AKJLKL02 (create) IKJEHAL1 (create) IKJEHDS1 (create) IKJEE100 (create) IKJCT463 (create) IKJCT473 (create) IKJEHPRO (create) IKJEHREN (create) IKJEFR00 (create) IKJEFF04 (create)
DCB	DCBD	Data Control Block DSECT	HELP LISTBC OUTPUT SEND SUBMIT	IKJEFH01 (create) IKJEES75 (create) IKJCT463 (create) IKJCT469 (create) IKJCT471 (alter) IEEVSDI0 (create) IKJEFF05 (create)
DFPARMS	IKJEFFDF	DAIRFAIL (IKJEFF18) Parameter List	ALLOCATE ATTRIB EXEC OUTPUT SUBMIT	IKJEFD32 (create) IKJEFATT (create) IKJCT430 (create) IKJCT467 (create) IKJEFF04 (create) IKJEFF15 (create)

Acronym	Macro	Common Name	Command Processor	Module/Access
DFPB	IKJDFPB	Default Parameter Block	LISTDS PROTECT RENAME	IKJEHDS1 (create) IKJEHPRO (create) IKJEHREN (create)
DFPL	IKJDFPL	Default Parameter List	LISTDS PROTECT RENAME	IKJEHDS1 (create) IKJEHPRO (create) IKJEHREN (create)
D08ADDED	IKJEFFD8	SUBMIT Extension to DAPB08	SUBMIT	IKJEFF04 (alter) IKJEFF16 (create)
ECB	IHAECB	Event Control Block	CANCEL/STATUS OUTPUT SUBMIT	IKJEFF50 (create) IKJEFF56 (create) IKJEFF57 (create) IKJCT463 (create) IKJCT469 (create) IKJEFF01 (create) IKJEFF04 (create) IKJEFF19 (create)
ECDA	IKJEXEC	Phase 1 Exit Common Data Area	EXEC	IKJCT430 (create) IKJCT431 (alter) IKJCT432 (alter)
ECT	IKJECT	Environment Control Table	CALL LINK/LOADGO OPERATOR OUTPUT	IKJEFG00 (alter) AKJLKL02 (alter) IKJEE150 (alter) IKJCT460 (alter)
EXECDATA	IKJEXEC	EXEC Command Control Data Area	EXEC	IKJCT431 (create)
ESTAEWA		ESTAE Exit Work Area	OUTPUT	IKJCT460 (alter) IKJCT464 (alter) IKJCT469 (create)
EXITL	IKJEFFIE	FIB Installation Exit Parameter List	SUBMIT	IKJEFF09 (create) IKJEFF10 (alter)
FFB2	IKJEFFB2	FIB Module's Parameter List from SVC 100	SUBMIT	IKJEFF04 (create)
FFIB	IKJEFFIB	Parameter List to SVC 100	CANCEL/STATUS OPERATOR OUTPUT PROFILE SUBMIT	IKJEFF56 (create) IKJEE100 (create) IKJCT466 (create) IKJEFT82 (create) IKJEFF01 (create)
GFPARMS	IKJEFFGF	GNRLFAIL and VSAMFAIL (IKJEFF19) Parameter List	EXEC OUTPUT SUBMIT	IKJCT430 (create) IKJCT431 (create) IKJCT432 (create) IKJCT467 (create) IKJEFF05 (create) IKJEFF15 (create)
GTPB	IKJGTPB	GETLINE Parm Block	EXEC LINK/LOADGO OPERATOR	IKJCT430 (create) AKJLKMSG (create) IKJEE150 (create)

Acronym	Macro	Common Name	Command Processor	Module/Access
HISTORY	IKJEFFHT	SUBMIT Internal History Table	SUBMIT	IKJEFF04 (create) IKJEFF05 (alter) IKJEFF07 (alter) IKJEFF08 (alter) IKJEFF09 (alter) IKJEFF13 (alter) IKJEFF15 (alter) IKJEFF20 (alter)
IKJWHEN	IKJWHEN	WHEN Common Data Area	WHEN/END	IKJEFE11 (create) IKJEFE15 (alter)
IOPL	IKJIOPL	I/O Service Routine Parameter List	ATTRIB	IKJEFATT (create)
			CALL	IKJEFG00 (create)
			EXEC	IKJCT430 (create)
			FREE	IKJEFD20 (create)
			HELP	IKJEFH01 (create)
			LINK/LOADGO	AKJLKL01 (create)
			LISTBC	IKJEES70 (create)
			OPERATOR	IKJEE100 (create) IKJEE150 (create) IKJEE1A0 (create)
			OUTPUT	IKJCT460 (create) IKJCT466 (create) IKJCT467 (create) IKJCT469 (create) IKJCT472 (create)
			PROFILE	IKJEFT82 (create)
			PROTECT	IKJEHPRO (create)
			RENAME	IKJEHREN (create)
			RUN	IKJEFR00 (create)
			SEND	IKJEES10 (create)
			SUBMIT	IKJEFF02 (create)
			TERMINAL	IKJEFT80 (create)
			WHEN/END	IKJEFE11 (create) IKJEFE15 (create)
LSD	IKJLSD	List Source Descriptor	EXEC	IKJCT431 (create)
			RUN	IKJEFR00 (create)
			WHEN/END	IKJEFE11 (create)

Acronym	Macro	Common Name	Command Processor	Module/Access
MSGTABLE	IKJEFFMT	Message Issuer (IKJEFF02) Parameter List	ALLOCATE	IKJEFD30 (create)
				IKJEFD34 (alter)
				IKJEFD35 (alter)
				IKJEFD36 (alter)
				IKJEFD37 (alter)
			CANCEL/STATUS	IKJEFF49 (alter)
				IKJEFF50 (create)
				IKJEFF51 (alter)
				IKJEFF52 (alter)
				IKJEFF54 (alter)
				IKJEFF56 (create)
				IKJEFF57 (create)
			EXEC	IKJCT430 (create)
				IKJCT431 (create)
				IKJCT432 (create)
			HELP	IKJEFH01 (create)
				IKJEFH02 (alter)
				IKJEFH03 (alter)
			SUBMIT	IKJEFF01 (create)
				IKJEFF02 (alter)
				IKJEFF04 (create)
				IKJEFF05 (alter)
				IKJEFF08 (alter)
				IKJEFF09 (alter)
				IKJEFF13 (alter)
				IKJEFF15 (alter)
				IKJEFF16 (alter)
				IKJEFF19 (create)
				IKJEFF20 (alter)
OLDMAP	IKJOLD	Output Line Descriptor	CALL	IKJEFG00 (create)
			OUTPUT	IKJCT467 (create)
				IKJCT472 (create)
			RUN	IKJEFR00 (create)
			SUBMIT	IKJEFF02 (create)
			WHEN/END	IKJEFF15 (create)
OUTCOMTB	IKJOCMTB	Output Communications Table	OUTPUT	IKJCT460 (alter)
				IKJCT462 (alter)
				IKJCT463 (alter)
				IKJCT464 (alter)
				IKJCT466 (create)
				IKJCT467 (alter)
				IKJCT469 (create)
				IKJCT470 (alter)
				IKJCT471 (alter)
				IKJCT472 (alter)
				IKJCT473 (alter)
PAPL	IKJPPL	Parse Parameter List	ALLOCATE	IKJEFD30 (create)
				IKJEFD37 (create)
			ATTRIB	IKJEFATT (create)
			CALL	IKJEFG00 (create)
			CANCEL/STATUS	IKJEFF50 (create)
				IKJEFF57 (create)
			EXEC	IKJCT430 (create)
				IKJCT431 (create)
			FREE	IKJEFD20 (create)
			HELP	IKJEFH01 (create)
			LINK/LOADGO	AKJLKL01 (create)
			LISTALC	IKJEHAL1 (create)
			LISTBC	IKJEES70 (create)
			LISTDS	IKJEHDS1 (create)

Acronym	Macro	Common Name	Command Processor	Module/Access
PAPL (Continued)			OUTPUT	IKJCT463 (create) IKJCT469 (create)
			PROFILE	IKJEFT82 (create)
			PROTECT	IKJEHPRO (create)
			RENAME	IKJEHREN (create)
			RUN	IKJEFR00 (create)
			SEND	IKJEES10 (create)
			SUBMIT	IKJEFF04 (create)
			TERMINAL	IKJEFT80 (create)
			WHEN/END	IKJEFE11 (create)
PARML	IKJEFFIE	FIB Installation Exit Parameter List	CANCEL/STATUS	IKJEFF51 (create) IKJEFF53 (alter)
			OUTPUT	IKJCT469 (create)
PARMLIST	IKJEFFPT	CANCEL/STATUS Internal Parameter List	CANCEL/STATUS	IKJEFF50 (create) IKJEFF51 (alter) IKJEFF52 (alter) IKJEFF54 (alter) IKJEFF57 (create)
PGPB	IKJPGPB	PUTGET Parameter Block	LINK/LOADGO	AKJLKMSG (create)
			OPERATOR	IKJEE100 (create)
			OUTPUT	IKJCT467 (create)
			SUBMIT	IKJEFF02 (create)
PTPB	IKJPTPB	PUTLINE Parm Block	OPERATOR	IKJEE100 (create) IKJEE150 (create) IKJEE1A0 (create)
			SUBMIT	IKJEFF02 (create)
			TERMINAL	IKJEFT80 (create)
			WHEN/END	IKJEFE15 (create)
RPL	IFGRPL	VSAM Request Parameter List	OUTPUT	IKJCT462 (create) IKJCT470 (alter)
			SUBMIT	IKJEFF05 (alter) IKJEFF15 (create)
SDWA	IHASDWA	System Diagnostic Work Area	OUTPUT	IKJCT460 (alter)
			SUBMIT	IKJEFF02 (alter)
SNTAB	IKJEXEC	Symbolic Name Table	EXEC	IKJCT431 (create) IKJCT432 (alter)
SSOB	IEFJSSOB	Subsystem Options Block	CANCEL/STATUS	IKJEFF49 (create) IKJEFF52 (create) IKJEFF54 (create)
			OUTPUT	IKJCT462 (alter) IKJCT464 (alter) IKJCT469 (create)
STPB	IKJSTPB	Stack Parameter Block	ATTRIB	IKJEFATT (create)
			CALL	IKJEFG00 (create)
			CANCEL/STATUS	IKJEFF56 (create)
			EXEC	IKJCT430 (create)
			HELP	IKJEFH01 (create)
			LINK/LOADGO	AKJLKL02 (create)
			LISTBC	IKJEES70 (create)
			OPERATOR	IKJEE100 (create) IKJEE150 (create)
			OUTPUT	IKJCT466 (create) IKJCT472 (create)

Acronym	Macro	Common Name	Command Processor	Module/Access
STPB (Continued)			PROTECT	IKJEHPRO (create)
			RENAME	IKJEHREN (create)
			RUN	IKJEFR00 (create)
			SEND	IKJEES10 (create)
			SUBMIT	IKJEFF01 (create)
			TERMINAL	IKJEFT80 (create)
			WHEN/END	IKJEFE11 (create)
STPL	IKJSTPL	Stack Parameter List	ALLOCATE	IKJEFD30 (create)
			ATTRIB	IKJEFATT (create)
			CALL	IKJEFR00 (create)
			CANCEL/STATUS	IKJEFF56 (create)
			EXEC	IKJCT430 (create)
			HELP	IKJEFH01 (create)
			LINK/LOADGO	AKJLKL02 (create)
				AKJLKMSG (create)
			OUTPUT	IKJCT460 (create)
				IKJCT466 (create)
				IKJCT467 (create)
				IKJCT469 (create)
				IKJCT472 (create)
			RUN	IKJEFR00 (create)
			SUBMIT	IKJEFF01 (create)
			WHEN/END	IKJEFE11 (create)
SVTAB	IKJEXEC	Symbolic Value Table	EXEC	IKJCT431 (create)
SWAEPA	IEFZB505	SWA Manager Parameter List	SUBMIT	IKJEFF04 (create)
S99RB	IEFZB4D0	Dynamic Allocation Request Block	ALLOCATE	IKJEFD30 (create)
				IKJEFD32 (alter)
				IKJEFD34 (alter)
				IKJEFD36 (alter)
			ATTRIB	IKJEFATT (create)
			FREE	IKJEFD20 (create)
			OUTPUT	IKJCT462 (create)
				IKJCT464 (create)
			SUBMIT	IKJEFF15 (create)
TCB	IKJTCB	Task Control Block	OUTPUT	IKJCT463 (alter)
UPT	IKJUPT	User Profile Table	PROFILE	IKJEFT82 (alter)
WPL	IEZWPL	WTO/WTOR/MLWTO/WTP Parameter List	SUBMIT	IKJEFF02 (create)
ZB831	IEFZB831	GENTRANS Parameter List	ALLOCATE	IKJEFD30 (create)
				IKJEFD32 (alter)
				IKJEFD34 (alter)
			ATTRIB	IKJEFATT (create)
			FREE	IKJEFD20 (create)

Started Task Control (STC), passes control to LOGON Initialization, IKJEFLA. Here, the various control blocks required for LOGON and the terminal session are initialized, the ESTAE recovery routine, IKJEFLS, is established, Master Scheduler JCL, MSTRJCL, is searched to ensure that SYSLBC, System Broadcast Dataset, and SYSUADS, System User Attribute Dataset, are available to LOGON and the subsequent terminal session, and then LOGON Scheduler, IKJEFLB, is called.

IKJEFLB receives control from IKJEFLA during a LOGON, and receives control from the Job Scheduling Subroutine, JSS, during a re-LOGON and a LOGOFF. IKJEFLB invokes the LOGON Prompting Monitor, IKJEFLC, and then waits for notification to either continue with the LOGON by passing control to JSS, or in the case of a LOGOFF, IKJEFLB will terminate and pass control back to STC.

IKJEFLC passes control to the LOGOFF processor, IKJEFLL, in the case of a LOGOFF or a re-LOGON. Then IKJEFLC passes control to LOGON Verification, IKJEFLE, who parses the command to obtain the LOGON data and verify this data against the UADS, User Attribute Dataset. In the case of a LOGOFF, IKJEFLC, notifies IKJEFLB that LOGON should terminate and then IKJEFLC terminates. For a LOGON or a re-LOGON, IKJEFLC notifies IKJEFLB that it should pass control to JSS and then IKJEFLC passes control to IKJEFLH, the routine that invokes LISTBC, List Broadcast Dataset.

IKJEFLB passes control to JSS for the LOGON or the re-LOGON, and JSS eventually passes control to the Pre-TMP Exit, IKJEFLJ. IKJEFLJ notifies IKJEFLH that once LISTBC has completed, IKJEFLH and then IKJEFLC should terminate. After IKJEFLJ terminates, the TMP is invoked for the users terminal session.

When a LOGON command, referred to as re-LOGON, or a LOGOFF command is entered, the TMP terminates. JSS then passes control to the Post-TMP Exit, IKJEFLK, for some housekeeping. After JSS has completed its work it passes control to IKJEFLB who inturn invokes IKJEFLE to handle the LOGOFF or the re-LOGON.

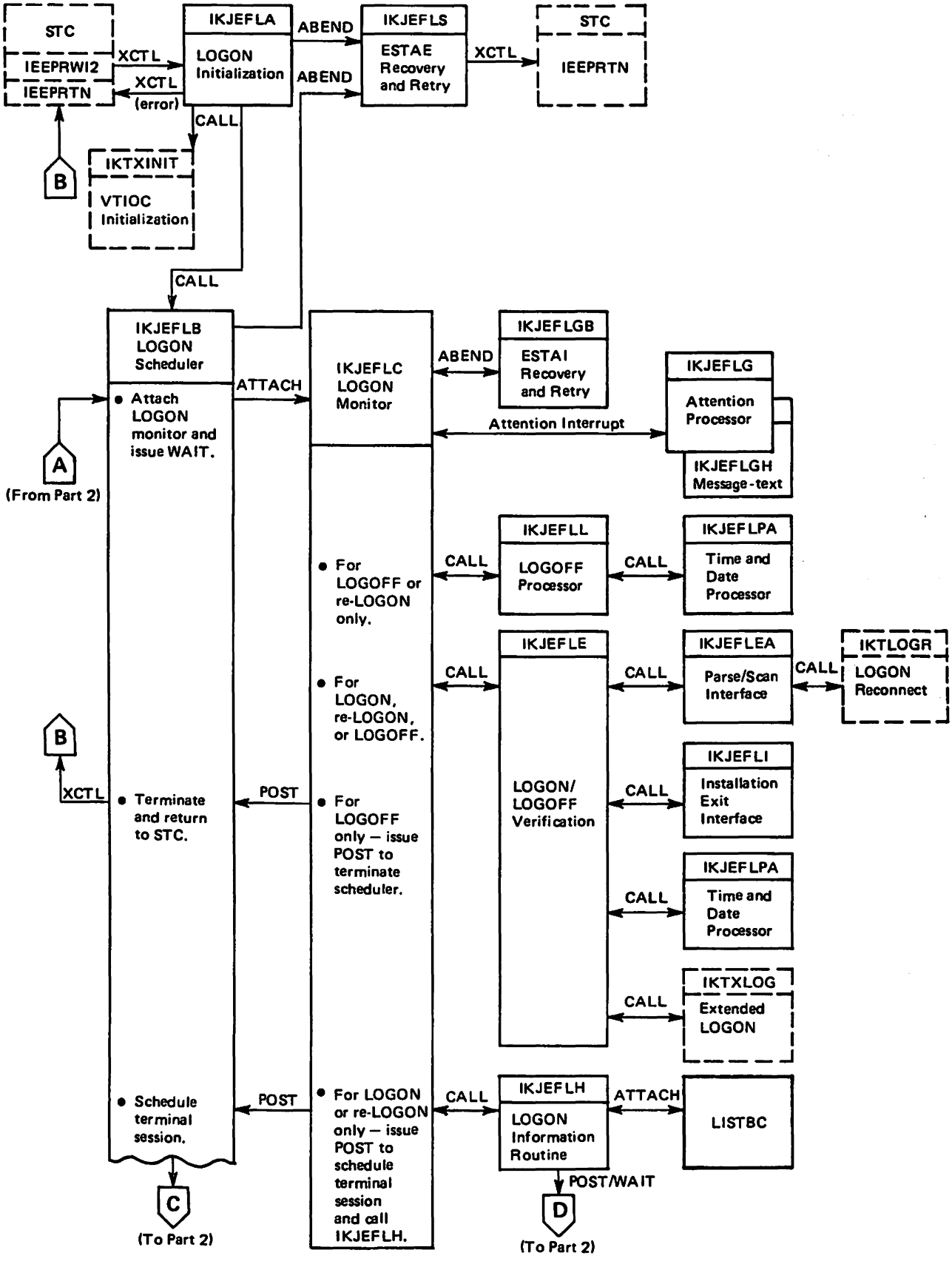


Figure 1. LOGON Scheduling Module Flow (Part 1 of 2)

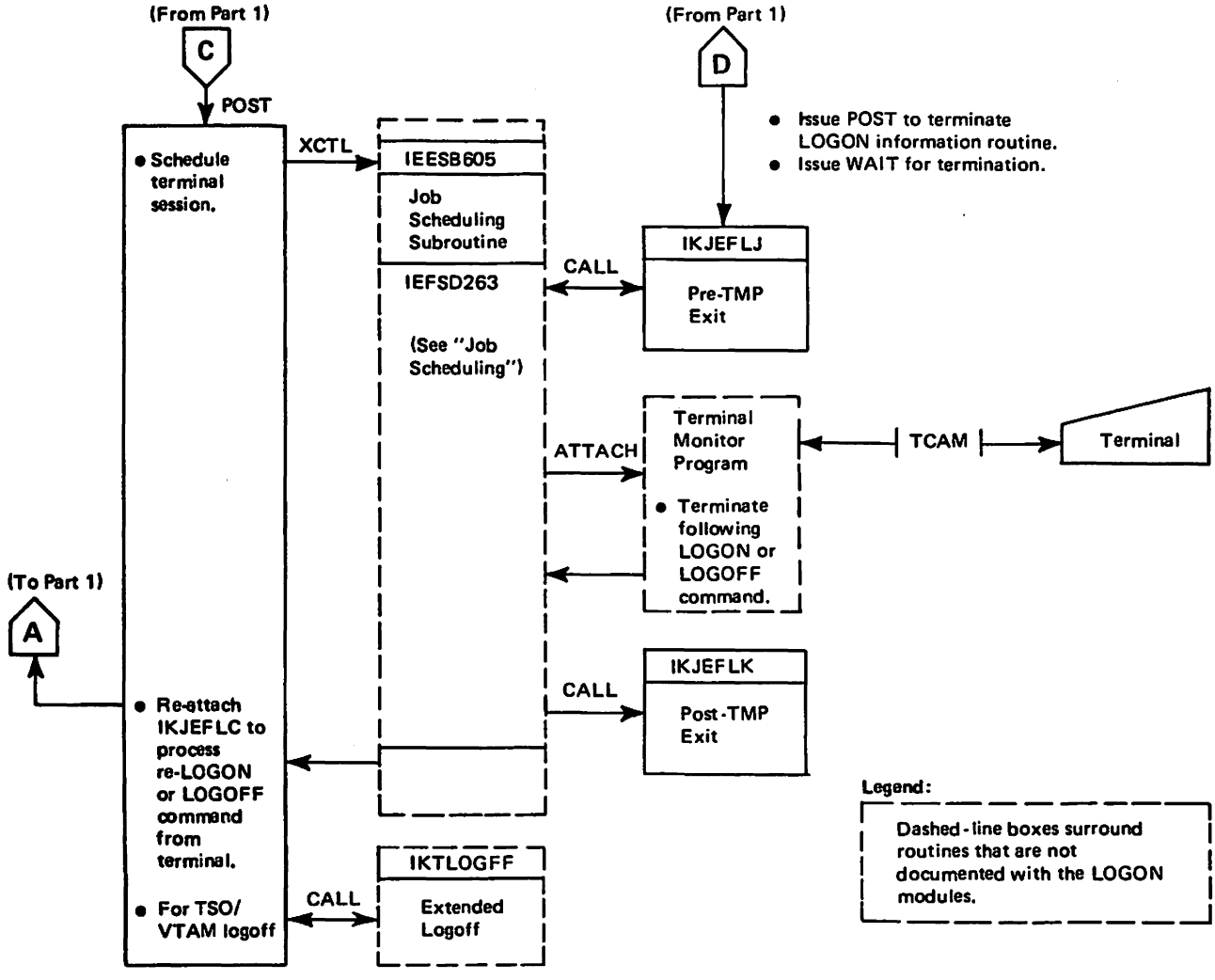


Figure 1. LOGON Schedule Module Flow (Part 2 of 2)

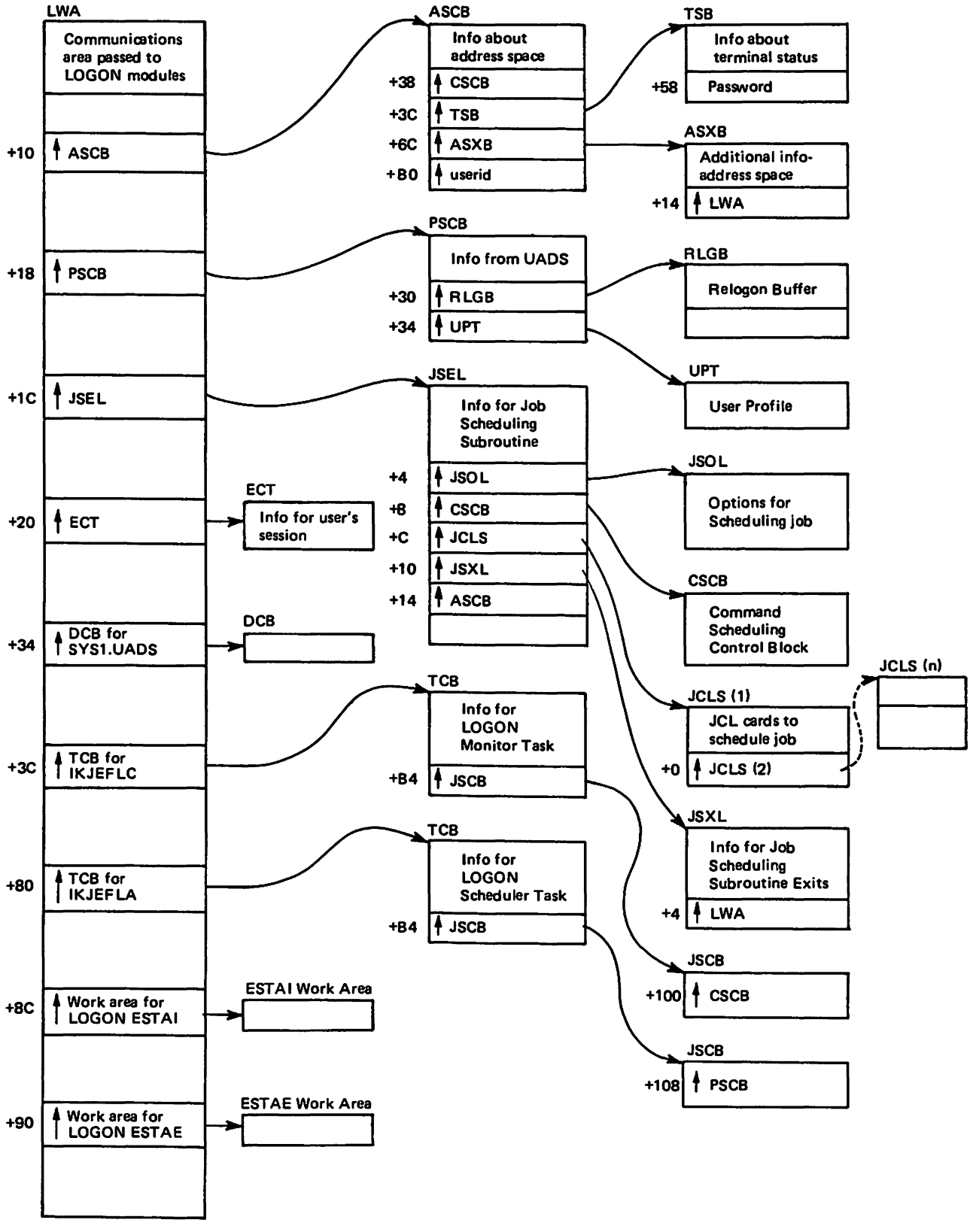


Figure 2. LOGON Scheduling Control Block Overview

Diagram 23.1 LOGON Initialization (IKJEFLA) (Part 1 of 2)

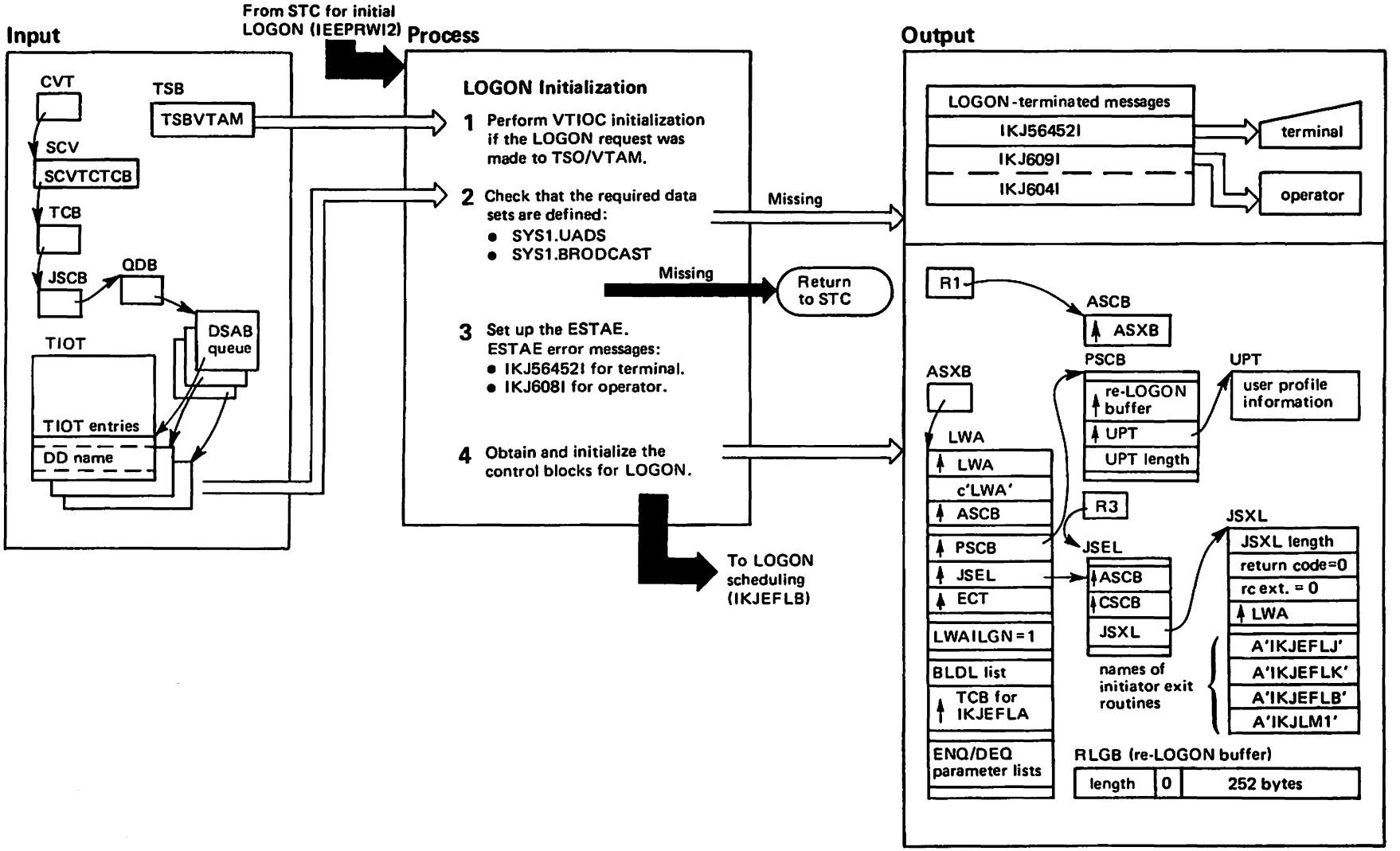


Diagram 23.1 LOGON Initialization (IKJEFLA) (Part 2 of 2)

Extended Description	Module	Label
LOGON initialization receives control from started task control (STC) to process an initial LOGON command from a terminal. The initialization functions are bypassed for a LOGOFF or reLOGON.	IKJEFLA	
1 IKTXINIT initializes VTAM control blocks and the TVWA, and transfers control (OPNDST PASS) of the terminal user's address space. An OPNDST RPL exit is then dispatched by VTAM to verify that the OPNDST was successful.	IKTXINIT	
2 Two TSO data sets—SYS1.UADS and SYS1.BRODCAST—must have been defined by master scheduler's JCL (MSTRJCL member of SYS1.LINKLIB). LOGON initialization checks for these data sets by searching the master scheduler's TIOT for the DD names SYSUADS and SYSLBC. If either of the names is missing, error messages are issued and LOGON is terminated.	IKJEFLA	
3 IKJEFLS is used as the ESTAE routine to protect IKJEFLA and IKJEFLB.		
4 LOGON initialization creates the control blocks that contain LOGON information needed by the various LOGON routines. LOGON initialization turns on the initial-LOGON bit (LWAILGN) to indicate that this is the first LOGON command to be processed for the current address space.	IKJEFLA	

Diagram 23.2 LOGON Scheduling (IKJEFLB) (Part 1 of 2)

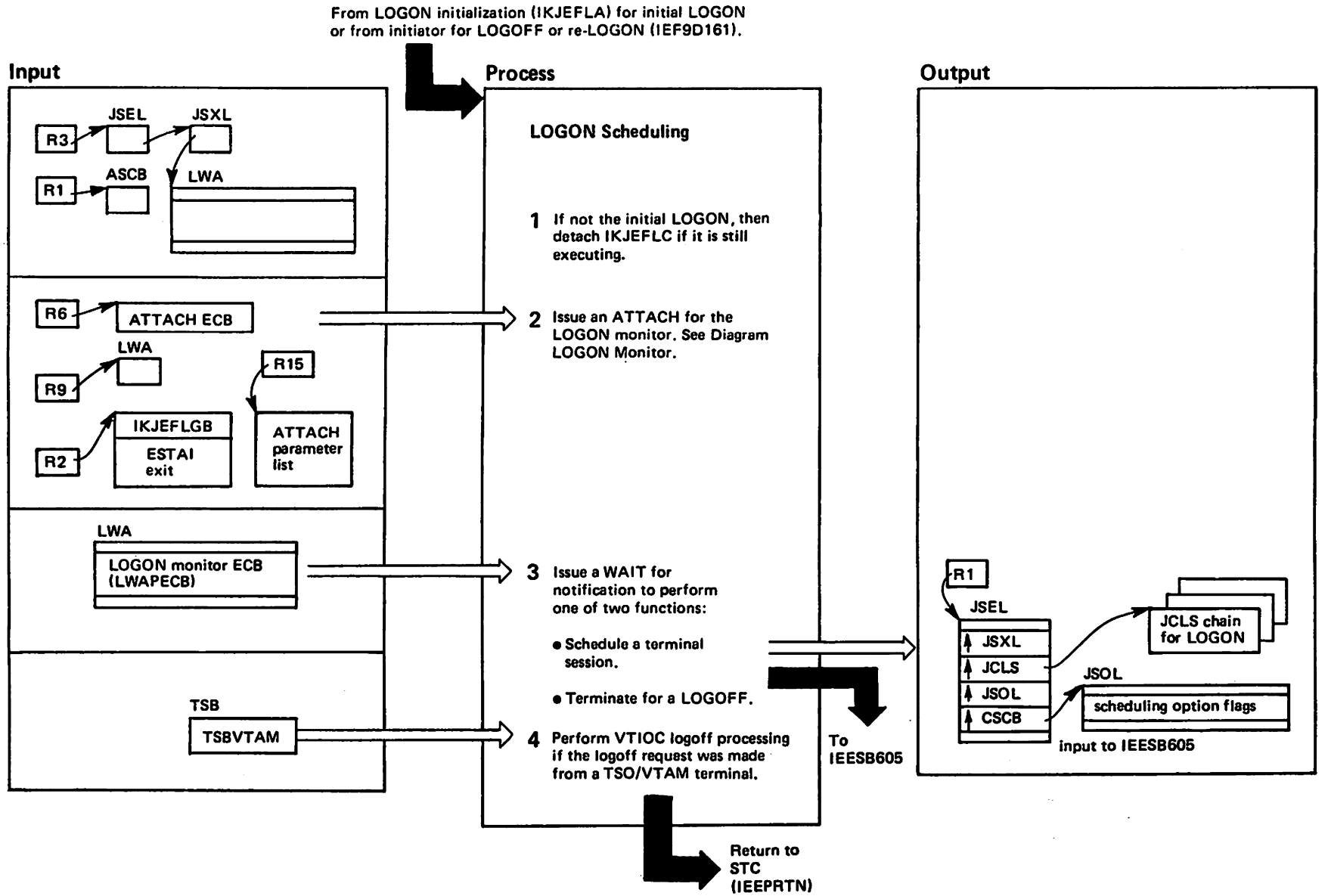


Diagram 23.2 LOGON Scheduling (IKJEFLB) (Part 2 of 2)

Extended Description	Module	Label
LOGON scheduling receives control from LOGON initialization or from the initiator at the end of the terminal session (for LOGOFF or re-LOGON). The new terminal session that is scheduled following a re-LOGON operates in the same address space as the initial terminal session. LOGON scheduling invokes the job scheduling subroutine. This subroutine interprets the JCL card images that define the terminal session and attaches the terminal monitor program (TMP), which processes commands from the terminal. The TMP remains active until it intercepts a LOGOFF or a re-LOGON command from the terminal. At that time, the TMP terminates and the initiator passes control back to LOGON scheduling to process the command.		
1 Upon receiving control from STC for a LOGOFF or re-LOGON, LOGON scheduling ensures that the LOGON monitor has already terminated. If the monitor is yet active, LOGON scheduling notifies the monitor (ILWASECB-post code 20) to terminate. Once the monitor has terminated (LWAPECB-post code 24) LOGON scheduling detaches it and sets the attach ECB (LWAAECB) to zero. LOGON scheduling then performs the attach of the LOGON monitor (Step 2) as usual.	IKJEFLB	
If the LOGON monitor posts LWAPECB with an invalid post code (other than 16 and 24), LOGON scheduling terminates as follows: • Detaches the LOGON monitor. • Cancels the ESTAE environment. • Places the address of the ASCB in register 1. • Returns to STC (IEEPRTN) for CSCB clean-up.	IKJEFLB	WAITUST BEXIT
But, if the LOGON monitor has caused an ABEND and recovery is to be attempted (LWABEND=1), LOGON scheduling does not terminate; it reissues the ATTACH of the LOGON monitor (returns to Step 2).		LCRESTRT

Extended Description	Module	Label
2 LOGON scheduling handles the initial LOGON, a LOGOFF, or a re-LOGON. First, it issues an ATTACH macro instruction to invoke the LOGON monitor (see Diagram "LOGON Monitor"). The monitor routine executes until it requires a function that LOGON scheduling performs. At that time, the monitor notifies LOGON scheduling via the LOGON monitor ECB (LWAPECB).	IKJEFLB	
3 When notified by the LOGON monitor, LOGON scheduling performs one of two functions; the function performed is determined by the post code located in the monitor's ECB: (LWAPECB).	IKJEFLB	WAITLIST
post code / **function performed by LOGON scheduling**		
16 — Schedules a terminal session as follows: • Notifies the LOGON monitor (LWASECB—post code 16) to invoke the LOGON information routine IKJEFLH. • Creates the job scheduling option list (JSOL) and chains it to the JSEL. The JSOL contains option flags that affect the scheduling of this terminal session. • Moves the JCL card image chain (created by either the LOGON monitor or the preprompt exit) from subpool 1 to subpool 253. • Invokes the initiator routine IEESB605 to schedule the terminal session.	IKJEFLB	
24 — Terminates LOGON scheduling as follows (performed following a LOGOFF command): • Notifies the LOGON monitor to terminate (LWASECB—post code 24). • Issues a DETACH macro instruction for the LOGON monitor. • Cancels the ESTAE environment protecting LOGON scheduling. • Transfers control to STC routine IEEPRTN for CSCB clean-up.	IKJEFLB	ENDJOB
4 VTIOC logoff processing is performed by IKTLOGFF.	IKTLOGFF	

Diagram 23.3 LOGON Initialization and Scheduling Recovery Routine (IKJEFLS) (Part 1 of 2)

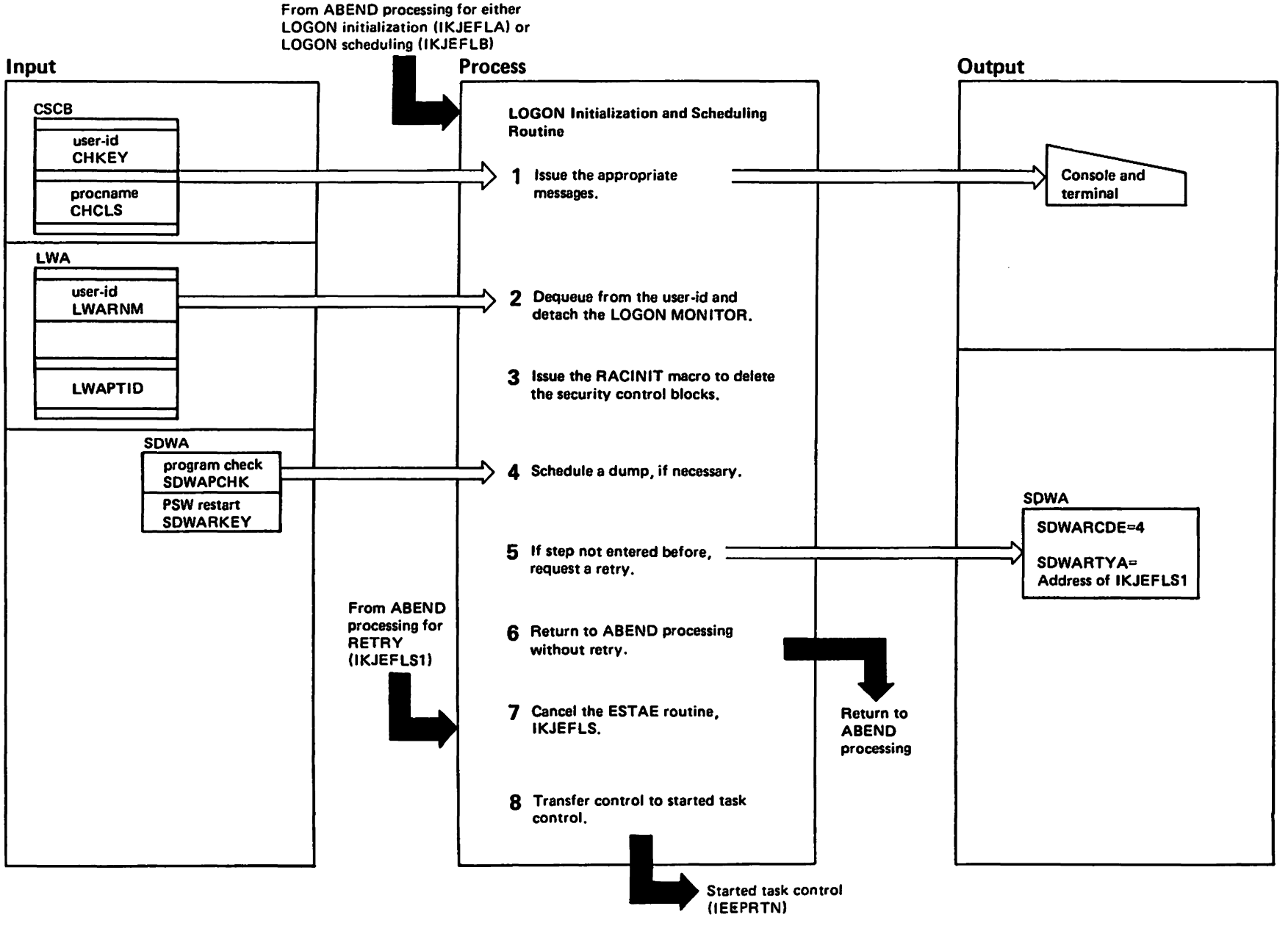


Diagram 23.3 LOGON Initialization and Scheduling Recovery Routine (IKJEFLS) (Part 2 of 2)

Extended Description	Module	Label
LOGON Initialization creates an ESTAE environment that handles abends that can occur during initialization and scheduling.	IKJEFLA	
1 Message IKJ601I is sent to the operator and message IKJ56452I is sent to the terminal.	IKJEFLS	
2 Dequeue from the user-id and detach the LOGON MONITOR. (The LWAPTID is the LOGON monitor TCB pointer.)		
3 If the user was in the RACF environment, IKJEFLS issues the RACINIT macro to delete the security related control blocks.		
4 Obtain a dump for a program check or PSW restart.		
5 If not a recursive abend, then indicate "RETRY" in the SDWA with the retry routine, IKJEFLS.		
6 Return to ABEND processing (IKJEFLS1) to possibly schedule a retry (see step 4).		
7 Cancel the ESTAE environment.	IKJEFLS1	
8 Transfer control to started task control, IEEPRTN, by using XCTL.		

Diagram 23.4 LOGON Monitor (IKJEFLC) (Part 1 of 4)

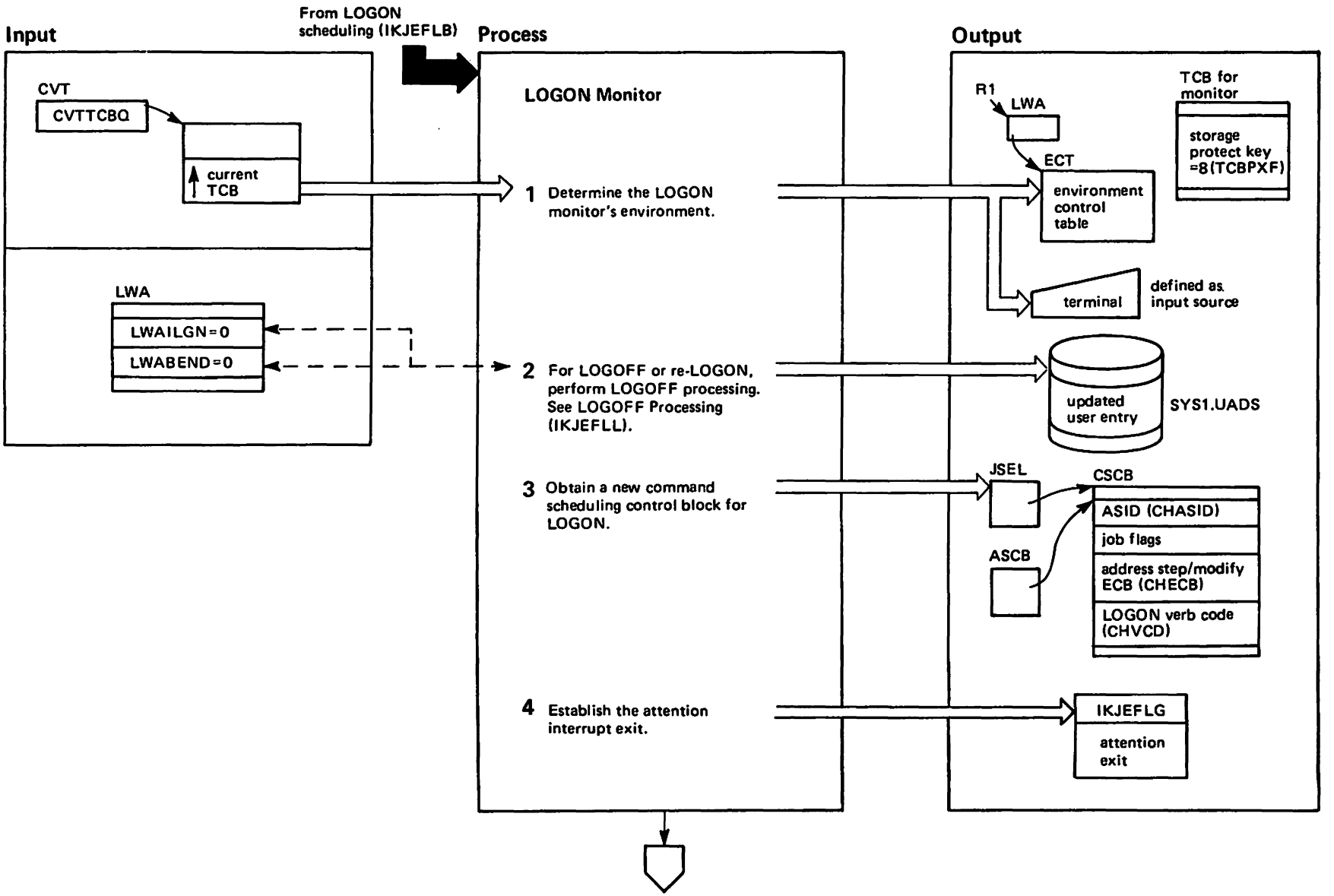


Diagram 23.4 LOGON Monitor (IKJEFLC) (Part 2 of 4)

Extended Description	Module	Label
The LOGON monitor controls the processing that verifies the LOGON or LOGOFF command, and the processing that issues informational and prompting messages to the terminal. It notifies LOGON scheduling to schedule a terminal session or, in the case of a LOGOFF, to terminate the LOGON scheduling task. Some of the informational messages (that is, mail, notices, and LOGON-proceeding messages) are issued in parallel with the scheduling of the terminal session. All LOGON monitor messages are issued by the message handler IKJEFLGM.	IKJEFLC	
1 The LOGON monitor creates the environment control table (ECT), which contains information about I/O service routines the monitor will use. Also, the monitor sets its own storage protection key to 8. This allows the storage obtained by the monitor to be referenced by programs not executing in privileged state (for example, LISTBC and the pre-prompt exit). Finally, the monitor issues a STACK macro instruction to define the terminal as the first source of input for time-sharing commands.	IKJEFLC	INITWKAR STACK

Extended Description	Module	Label
2 LOGOFF processing updates the terminal user's entry in SYS1.UADS and analyzes the return codes from the job scheduling subroutine and from the terminal session. LOGOFF processing is not performed for an initial LOGON (LWAILGN=1) or for recovery processing (LWABEND=1). For more detail, refer to the Diagram LOGOFF Processing.	IKJEFLL	
3 The LOGON monitor builds a new CSCB that contains the verb code for the LOGON command. This new CSCB replaces the one built for address space creation processing (START/LOGON/MOUNT) or, if this LOGON is a re-LOGON, replaces the CSCB previously created by the LOGON monitor. (It is important that LOGON establish a full size CSCB for all logons and re-logons before passing it to the initiator. The initiator, assuming the full size CSCB is passed, frees the second portion and uses only the first portion of the CSCB.)	IKJEFLC	CSCBINIT
4 The LOGON monitor issues a STAX macro instruction to establish a routine (IKJEFLG) that receives control when the terminal user causes an attention interruption by pressing the terminal's attention key. After causing the interruption, the terminal user may enter a question mark (?) to request second-level messages or may enter a new LOGON command to replace the one currently being processed.	IKJEFLC	TERMINAL

Diagram 23.4 LOGON Monitor (IKJEFLC) (Part 3 of 4)

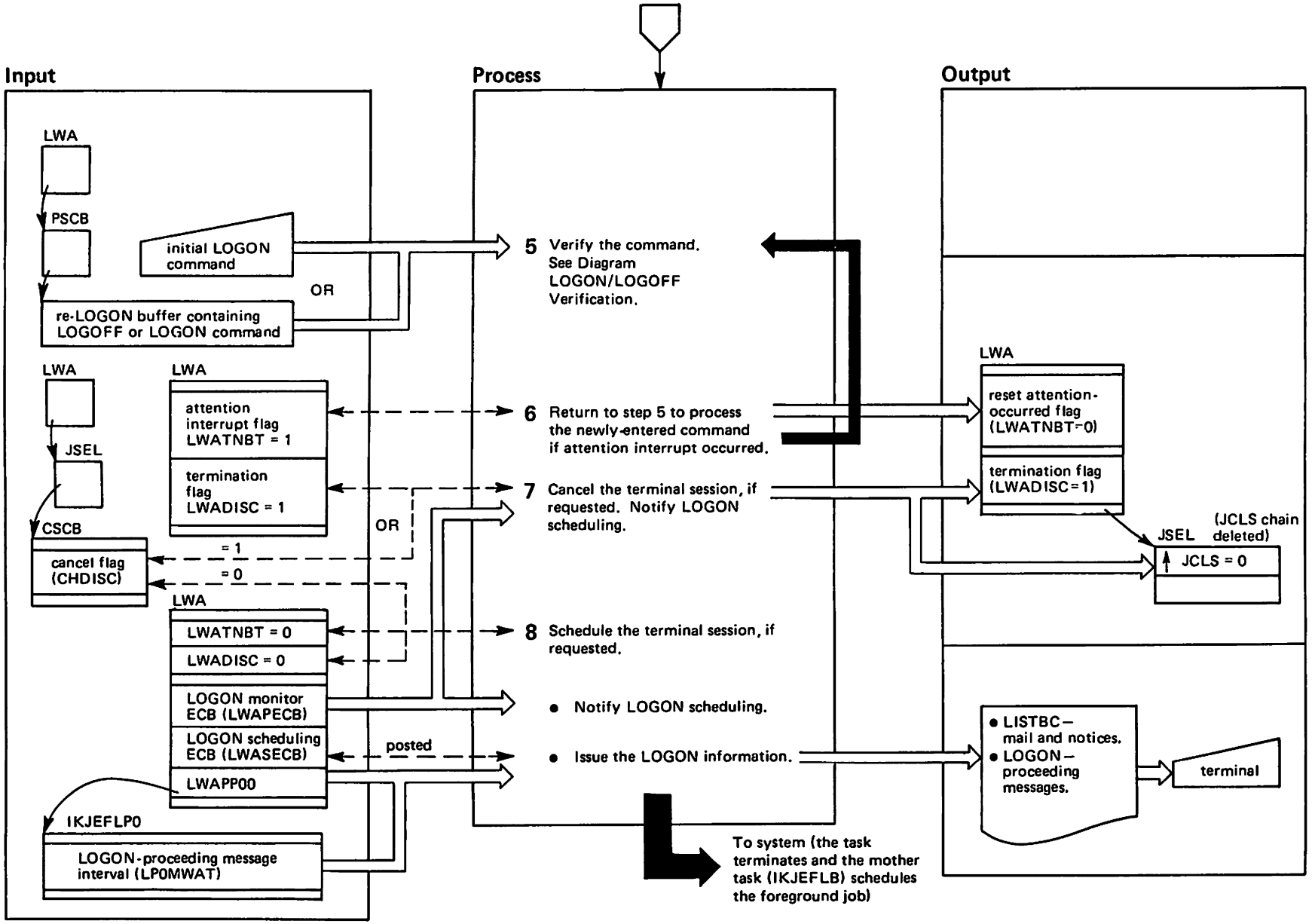


Diagram 23.4 LOGON Monitor (IKJEFLC) (Part 4 of 4)

Extended Description	Module	Label
5 The LOGON monitor invokes LOGON/LOGOFF verification (IKJEFLE) to scan and parse the LOGON or LOGOFF command. For a LOGOFF or a re-LOGON, the command text is found in the re-LOGON buffer; otherwise, the command is obtained from the terminal. LOGON verification checks the user's authorization and LOGON parameters against the user information in SYS1.UADS (user attribute data set) and prompts the user to replace invalid or missing information. See Diagram "LOGON/LOGOFF Verification."	IKJEFLE IKJEFLEA	
6 If the user presses the terminal's attention key during LOGON processing, he may re-enter the LOGON command. In this case, the LOGON monitor re-invokes LOGON verification to analyze the newly-entered command. The attention interrupt flag is reset to zero to indicate that the interrupt has been completely processed.	IKJEFLC	GOTOLE
7 If the system operator cancels the terminal user, if the user has entered a LOGOFF command, or if the user has failed to enter a valid LOGON command, the LOGON monitor ends the terminal session as follows:	IKJEFLC	GOTOLE
• Issues an error messages (IKJ56453I) to the terminal for an operator cancel.	IKJEFLGM	
• Issues a null STAX macro instruction to cancel the LOGON attention exit.	IKJEFLC	
• Frees the environment control table (ECT).		
• Notifies LOGON scheduling to terminate (LWAPECB—post code 24).		
• Waits for notification from LOGON scheduling to terminate (LWASECB—post code 24).		
• Returns to the operating system via SVC 3.		
8 • After LOGON verification has processed a valid LOGON command, the LOGON monitor notifies LOGON scheduling to schedule the terminal session (LWAPECB—post code 16). LOGON scheduling invokes the job scheduling subroutine of the initiator which attaches the terminal monitor program (TMP).	IKJEFLC	
•When LOGON scheduling is ready to invoke the job scheduling subroutine, it notifies the LOGON monitor to continue its operation. (LWASECB—post code 16). At that time, the LOGON monitor calls the LOGON information routine, allowing it to execute in parallel with the scheduling of the terminal session. The information routine attaches the LISTBC processor to issue mail and notices to the terminal user. Then the routine sets the timer to expire at the interval specified in the module IKJEFLP0. The LOGON-proceeding message is issued repeatedly to the terminal at this timed interval until the initiator is ready to attach the TMP. At that time, the pre-TMP exit (IKJEFLJ) notifies the information routine (LWASECB—post code 20) that the LOGON scheduling process is complete. The routine then cancels the timer and notifies the pre-TMP exit that LISTBC processing is completed (LWAPECB—post code 20).	IKJEFLH	
Finally, the LOGON monitor terminates as follows: —Issues a null STAX macro instruction to cancel the LOGON attention exit. (Pressing the terminal attention key no longer has any effect on LOGON processing.) —Deletes the environment control table (ECT). —Returns to the operating system via SVC 3.	IKJEFLC	CLEANUP
Error Processing		
LOGON scheduling establishes the LOGON monitor's ESTAI environment via a parameter on the ATTACH macro instruction. Since the LISTBC command processor is attached by the LOGON monitor task, it too is protected by the ESTAI environment. If the LOGON monitor task or the LISTBC task terminates abnormally, the ESTAI routine IKJEFLGB receives control. See Diagram "LOGON Monitor Recovery.	IKJEFLB IKJEFLGB IKJEFLC	
The LOGON monitor issues the STACK macro instruction to initialize the terminal as the source of input for commands. If this process encounters any errors, the LOGON monitor invokes the message handler to issue appropriate error messages to the terminal (IKJ56454I) or to the operator (IKJ608I). Also, the monitor turns on the LOGON-termination bit (LWADISC).	IKJEFLGM	
The LOGON monitor issues the MGCR macro instruction to chain a new CSCB. If this routine passes back a non-zero return code, the monitor issues error messages (IKJ56454I) to the terminal via the message handler. If the cancel bit is on (CHDISC field of the CSCB), a session-cancelled message (IKJ56453I) is issued by the message handler. In any case, the monitor ends the terminal session as in Step 6 of this diagram.	IKJEFLC IKJEFLGM	

Diagram 23.5 LOGOFF Processing (IKJEFLL) (Part 1 of 2)

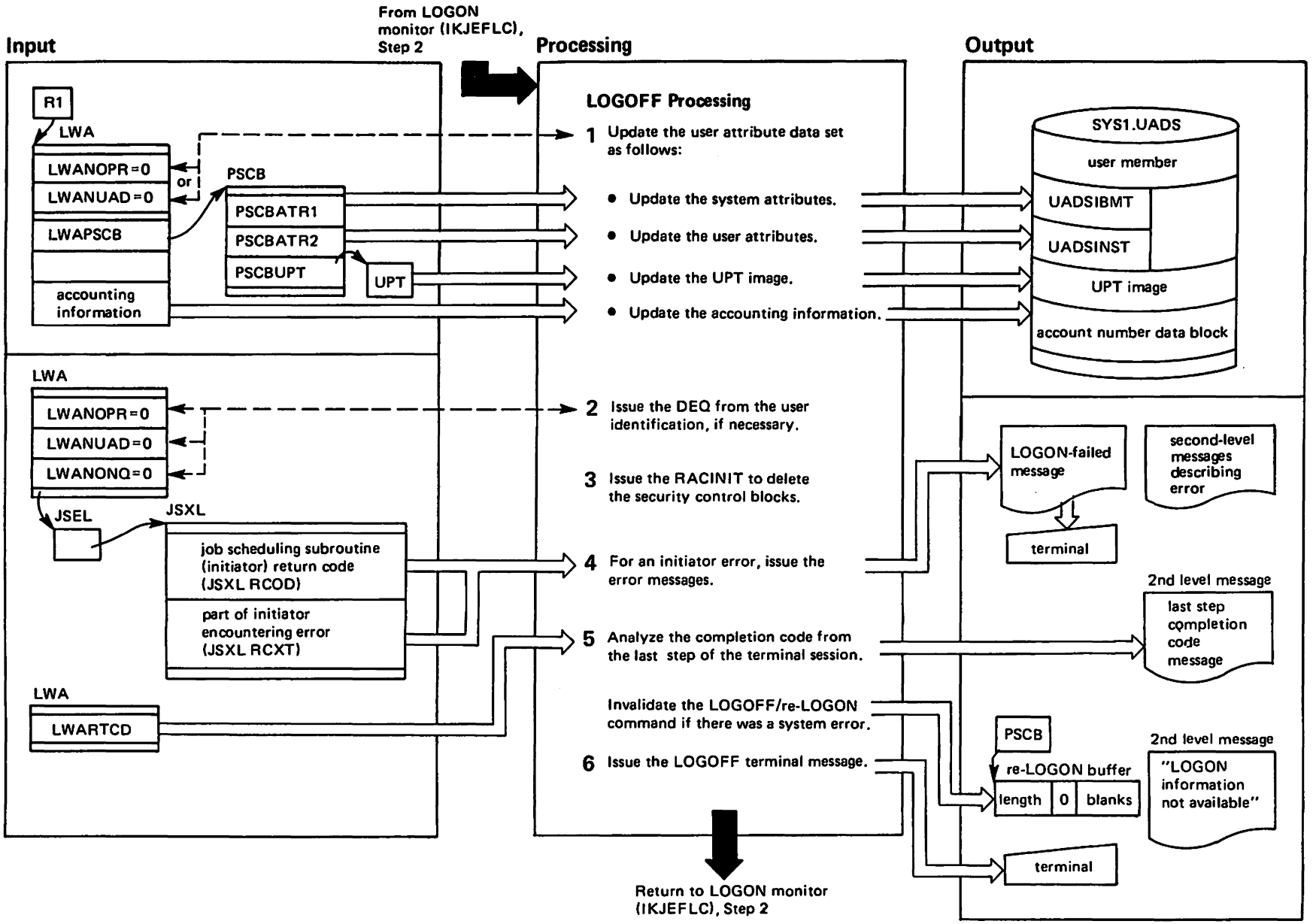


Diagram 23.5 LOGOFF Processing (IKJEFLL) (Part 2 of 2)

Extended Description	Module	Label
LOGOFF processing updates the terminal user's entry in SYS1.UADS and analyzes the return codes from the job scheduling subroutine (initiator) and from the last step of the terminal session. LOGOFF processing is performed for a LOGOFF command and for a re-LOGON. It is not performed for an initial LOGON (LWAILGN=1) or for recovery processing (LWABEND=1).	IKJEFLL	
1 Using the PROFILE command, the terminal user is able to change the attributes associated with his user identification. These attributes are supplied by a member of SYS1.UADS. LOGOFF processing must update this member at the end of the terminal session to reflect the changes made by the user. If the installation has supplied all of the LOGON information normally supplied by SYS1.UADS (LWANOPR=1 and LWANUAD=1), it is not necessary to update the user's member of SYS1.UADS. If any of the three bits LWAATR1, LWAATR2, and LWABUPT are off, the corresponding information (system attributes, user attributes, and the user profile, respectively) was not supplied by the installation. The information not supplied by the installation (and, therefore, subject to changes made via the PROFILE command) is updated by LOGOFF processing. If LWAACCT≠0, the user's accounting information in SYS1.UADS is also updated. Accounting information consists of the following items: the length of the terminal session, the amount of processor time used, and the number of service units used.	IKJEFLL	UPDTUADS
2 LOGOFF processing must release the user identification resource that was obtained during LOGON verification. LOGOFF issues the DEQ macro instruction. If the three bits LWANOPR, LWANUAD, and LWANONQ are turned off, an ENQ was never issued on the user identification. In this case, a DEQ is not necessary.	IKJEFLL	DEQUSER
3 If the user was in the RACF environment, IKJEFLL issues the RACINIT macro to delete the security related control blocks.		

Extended Description	Module	Label
4 If the job scheduling subroutine encountered an error (LWARTCD≠0), LOGOFF processing examines the field JSXLRCXT to determine what part of job scheduling failed. Next, it examines the fields JSXLRCOD and LWARCDE to determine the nature of the error. Finally, LOGOFF informs the message handler (IKJEFLGM) to build the appropriate second-level message (IKJ56457I to terminal).	IKJEFLL	
5 LOGOFF analyzes the return code from the last step of the terminal session (LWARTCD) and builds an appropriate second-level message (IKJ56470I to terminal) via the message handler. If the code is a system return code, the re-LOGON buffer is considered to be unusable and is filled with blanks. In this case, LOGON/LOGOFF verification must prompt the user for a LOGON or LOGOFF command. (See Diagram LOGON/LOGOFF Verification.) The exception is a system return code that was generated by attention exit processing (indicated by LWATNBT=1). The attention exit posts the cancel ECB in the CSCB with a system code of 622, so that the job scheduling subroutine terminates in the same way as for an operator cancel. In this case, there is no reason why the re-LOGON buffer would be unusable; therefore, the contents of the buffer are retained.	IKJEFLL	
6 LOGOFF calls the LOGON time and date processor (IKJEFLPA) to set up the date and time-of-day buffers for the logged-off message. Then LOGOFF invokes the message handler to issue the logged-off message to the terminal (IKJ56470I).	IKJEFLL	LGMSETUP
Error Processing If, at any time, LOGOFF processing encounters an I/O error, an OPEN error, or a service routine error, it issues an error message (IKJ56454I) to the terminal via the message handler and turns on the LOGON-termination bit.	IKJEFLL	

Diagram 23.6 LOGON/LOGOFF Verification (IKJEFLE and IKJEFLES) (Part 1 of 4)

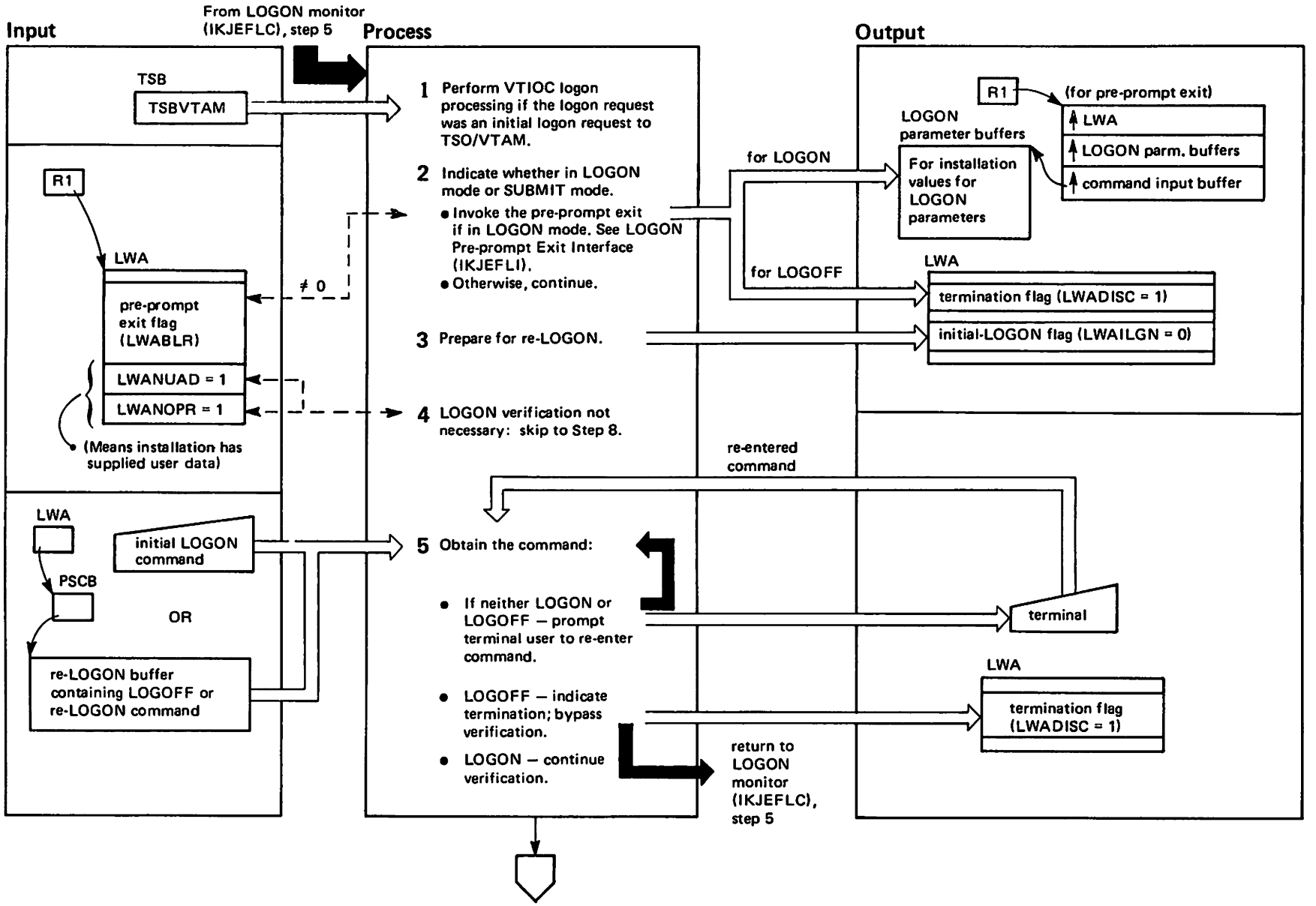


Diagram 23.6 LOGON/LOGOFF Verification (IKJEFLE and IKJEFLES) (Part 2 of 4)

Extended Description	Module	Label
LOGON/LOGOFF verification scans the LOGON or LOGOFF command and checks the LOGON parameters against the information in the user's member of the SYS1.UADS data set. As the verification process is checking LOGON parameters, it records valid LOGON information in various control blocks. An optional installation exit (pre-prompt exit IKJEFLD) can replace any part or all of the verification processing. If the LOGON is valid, JCL card images (JOB and EXEC) that define the terminal session are built. When SUBMIT enters LOGON verification, the LOGON command is parsed and the results are returned to SUBMIT (IKJEFF08). SUBMIT then builds JCL statements to execute commands in the background. The pre-prompt exit interface will not be invoked.	IKJEFLE	
1 VTIOC logon processing is done only for an initial logon to TSO/VTAM, not for a relogon or a logoff.	IKTXLOG	
2 If the VCON for the installation exit (IKJEFLD) is non zero (indicating an installation exit is present and link-edited into the LOGON load module), the interface routine IKJEFLI is invoked to initialize a parameter list for the exit. (See Diagram LOGON Pre-prompt Exit Interface.) The interface does not pass control to the pre-prompt exit (IKJEFLD) if the command is a LOGOFF or if LOGON is invoked from SUBMIT	IKJEFLE	GOTOIER
3 The initial-LOGON flag is turned off following the first GETLINE macro instruction issued by LOGON/LOGOFF verification. Any subsequent LOGON command entered by the terminal user for the current address space is considered to be a re-LOGON.	IKJEFLE	

Extended Description	Module	Label
4 LOGON/LOGOFF verification returns to the LOGON monitor if the termination flag is on (LWADISC) or if the cancel flag is on (CHDISC). If the pre-prompt exit has supplied all the LOGON information and indicates that no verification is necessary, the normal verification is bypassed.	IKJEFLE	
5 After the command scan service routine (IKJSCAN) scans the command for LOGON or LOGOFF, the verification process continues as follows:	IKJEFLEA	
• If neither command was found, the terminal user is prompted to enter LOGON or LOGOFF and the scan is repeated.		LOGONOFF
• If the command was a LOGOFF, the verification process returns control to the caller, the LOGON monitor. For a LOGOFF HOLD (TSBHLDL=1), terminal input/output control (TIOC) for TSO/VTAM keeps a line open to the terminal. If at any time a terminal line is accidentally disconnected TIOC or VTIOC retains, for a time specified in IKJPRM00 of SYS1.PARMLIB, the control blocks and the address space used for the current terminal session. If the terminal user then enters a LOGON RECONNECT command with the same user identification as the retained address space, TIOC or VTIOC reinstates the user in that address space.		
• If the command was a LOGON, the verification process continues (see Step 5).		

Diagram 23.6 LOGON/LOGOFF Verification (IKJEFLE and IKJEFLES) (Part 3 of 4)

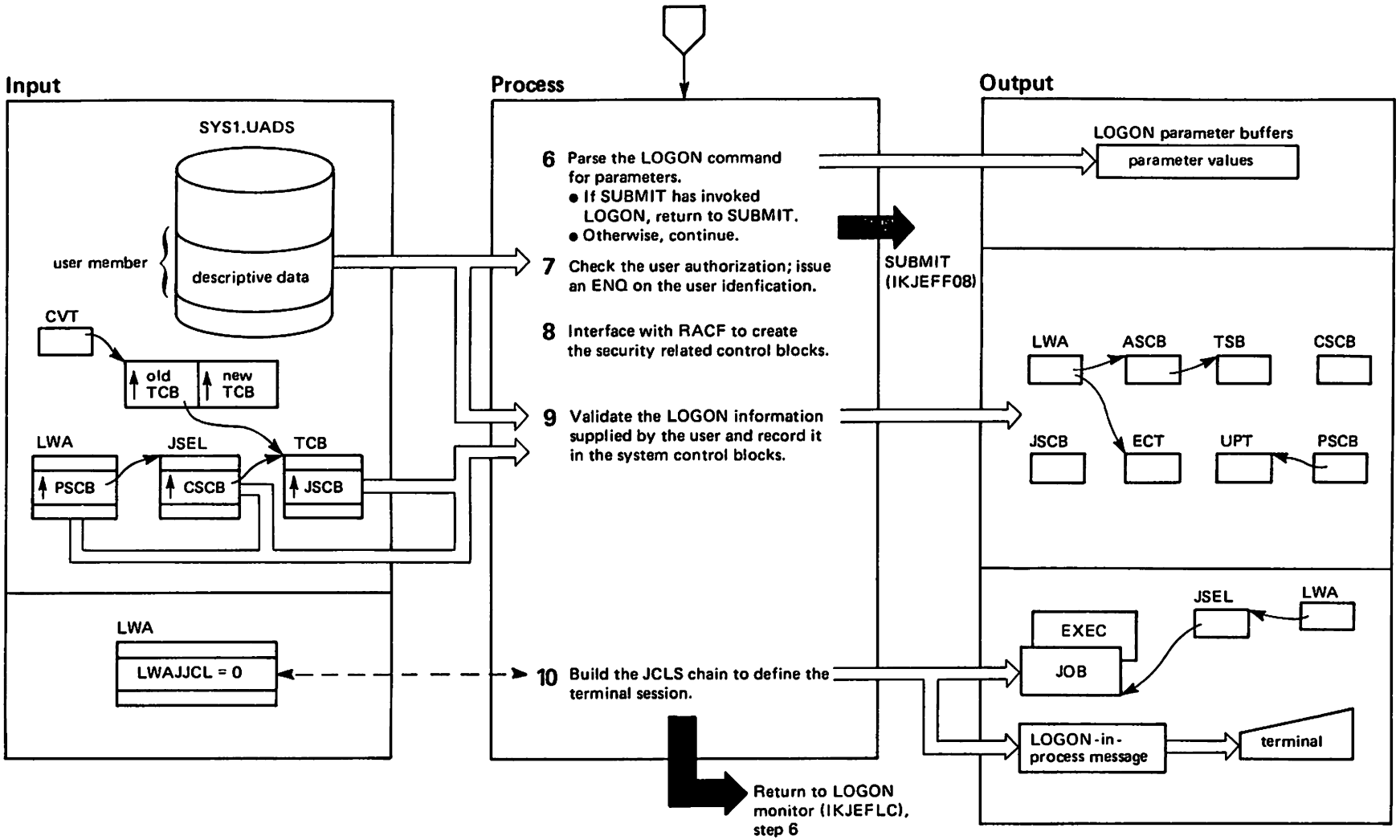


Diagram 23.6 LOGON/LOGOFF Verification (IKJEFLE and IKJEFLES) (Part 4 of 4)

Extended Description	Module	Label
6 The verification process invokes the parse service routine (IKJPARSE) to check the syntax of the LOGON command. If the command contains the RECONNECT parameter, LOGON determines whether the user identification is already assigned to an address space (one that TIOC or VTIOC retained following a disconnected line). If the user identification has an address space assigned to it, RACF is called to verify user and terminal access security and TIOC or VTIOC reinstates the user in the retained address space. If the user identification has no address space assigned to it, the LOGON RECONNECT is rejected.	IKJEFLE	TSBSRCH
7 LOGON verification opens the SYS1.UADS data set (user attribute data set) and copies into real storage the member associated with the user identification on the LOGON command and then ensures that the user identification is authorized. The user identification and its length are stored in the PSCB (protected step control block). Then LOGON issues an ENQ on the user identification resource. If the resource has already been obtained, LOGON verification reinvokes the pre-prompt exit if it exists. The installation can choose to authorize the user or to cancel the LOGON process.	IKJEFLE	OPEN
8 The RACINIT macro is issued by IKJEFLE causing RACF to create security related control blocks associated with the user identification and password.		
9 LOGON verification compares the LOGON parameter values with the user information in SYS1.UADS to check for the validity of the LOGON parameters. If parameters are invalid or missing, LOGON verification prompts the user for correct parameters. The user's reply is re-parsed and verified. Verification checks the user's password, account number, procedure name, region size, and performance group. The system resources manager checks that the group can be used at this time. The job entry subsystem verifies that the destination choice (DEST parameter) defines a valid device for SYSOUT data sets. If the user is RACF defined, then password verification with the UADS is bypassed. Both the password and group identification are verified by RACF. The remaining LOGON data is verified against the UADS.	IKJEFLE	

10 If LWAJJCL=I, the pre-prompt exit has supplied the JCL card images that define the terminal session. Otherwise, LOGON processing constructs the JCL card images as follows:
 	
`//userid JOB 'account #', REGION=region size`
`//procname EXEC procname,PERFORM=performance group`
 	
where the userid (user identification), account #, region size, and performance group are obtained form the LOGON parameters, from the user's member of SYS1.UADS, or from the pre-prompt exit.
	IKJEFLEA	BUILDJCL

Error Processing If the LOGON is an initial LOGON (LWAILGN=I), and the address of the terminal input line is zero, LOGON verification obtains a line from the terminal (issues a GETLINE for the terminal). LOGON verification is part of the LOGON monitor task and, therefore, is protected by the monitor's ESTAI environment in case of an ABEND.	IKJEFLE IKJEFLGB	

The following data areas contain TSO user information supplied by the SYS1.UADS data set, by the installation, or by the LOGON parameters:

Data Area Name	Field Name	Contents
ASCB	ASCBJBNS	Address of user identification.
CSCB	CHCLS	Procedure name for this LOGON.
	CHKEY	User identification.
ECT	ECT	Flags that control LISTBC processing.
EXEC card image		Procedure name for this LOGON.
		Performance group number.
JOB card image		Account number.
		Region size.
JSEL	JSEL	Address of JCL card images.
JSOL	JSOLDEST	Default destination for SYSOUT data sets.
LWA	LWACTLS	Control switches set by the installation exit.
	LWADEST2	Default destination for SYSOUT data sets.
	LWAACCT	Offset of accounting information in SYS1.UADS.
	LWATCPU	Total CPU time used.
	LWATSRU	Total service units used.
	LWATCON	Total time connected to the system.
	LWARTCD	Completion code for the last step of the terminal session.
PSCB	PSCBUSER	User identification.
	PSCBUSRL	Length of user identification.
	PSCBATR1	System attributes: switches that control use of OPERATOR, ACCOUNT, and SUBMIT commands, that indicate volume and mount authorization, and that define the attention key as the line-delete key.
	PSCBATR2	User attributes — reserved for installation use.
	PSCBGPNM	Generic unit name.
	PSCBRSZ	Region size.
TSB	TSBPSWD	Password.
UPT	UPTSWS	Environmental switches.
	UPTNPRM	No-prompting switch.
	UPTMID	Switch that controls printing of message identifiers.
	UPTNCOM	Switch that controls SEND command authorization.
	UPTPAUS	Switch that indicates whether to pause for a "?".
	UPTALD	Switch that defines the attention key as the line-delete key.
	UPTMODE	Switch that controls printing of mode messages.
	UPTWTP	Switch that allows the user to receive WTP messages.
	UPTCDEL	Character-delete character.
	UPTLDEL	Line-delete character.
	UPTPREFX	Data set name prefix.
	UPTPREFL	Length of data set name prefix.

Figure 3. Data Areas Containing LOGON User Information

Diagram 23.7 LOGON Pre-prompt Exit Interface (IKJEFLI) (Part 1 of 2)

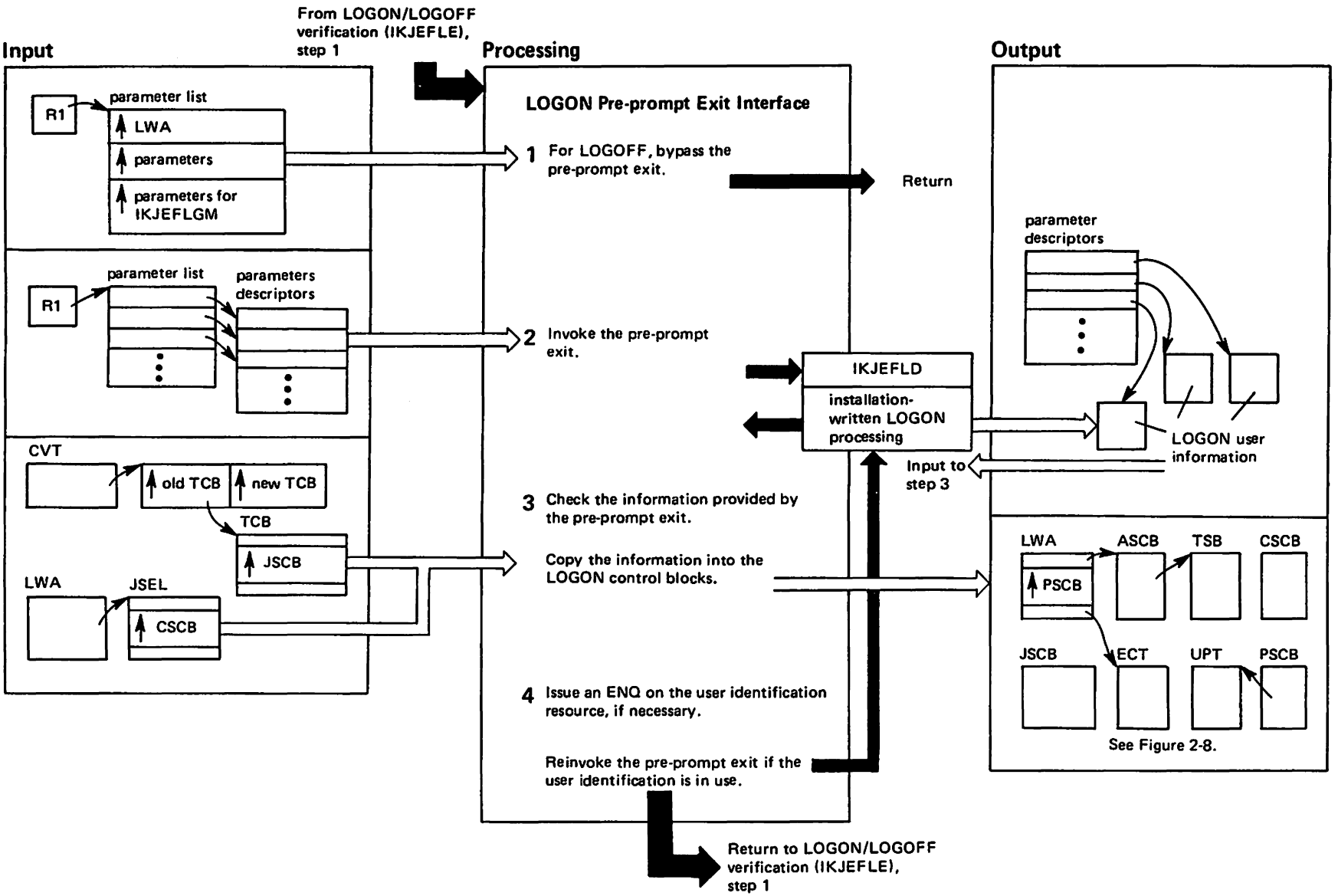


Diagram 23.7 LOGON Pre-prompt Exit Interface (IKJEFLI) (Part 2 of 2)

Extended Description	Module	Label
The LOGON pre-prompt exit interface invokes the LOGON pre-prompt exit which is a routine written by the installation. The pre-prompt exit can provide LOGON information on behalf of the terminal user, verify the user's LOGON command, and collect accounting information. Any user information provided by the pre-prompt exit overrides the information stored in the user's member of the SYS1.UADS data set. In fact, an installation can, if it wishes, replace all of the normal LOGON verification processing. For directions on writing the exit routine, refer to the topic, Writing a LOGON Pre-prompt Exit in the publication *OS/VS2 System Programming Library: TSO,* GC28-0629.	IKJEFLI	
1 The pre-prompt exit interface uses the command scan service routine (IKJSCAN) to determine if the command is a LOGON or LOGOFF. If it is a LOGOFF, the interface does not invoke the pre-prompt exit. Instead, it returns to its caller.	IKJEFLI	
2 The interface builds and passes to the pre-prompt exit a parameter list that defines those parameters the pre-prompt exit needs to verify the LOGON command and to provide LOGON information. Most of the addresses in the parameter list point to two-word descriptors. The first word of the descriptor contains the address of the actual parameter. The second word contains both the maximum length for the parameter and the actual length.		LI0100
3 After invoking the pre-prompt exit, the interface routine checks the parameter list for validity: • Ensures the parameter list is unchanged. • Ensures the parameter descriptors are unchanged, except for the field containing the actual length of the parameter. • Checks that the actual length of each parameter does not exceed the maximum length for the parameter.	IKJEFLI	
If errors are discovered, the interface invokes the message handler (IKJEFLGM) to issue error messages and terminates the terminal session (LWADISC=1). If no errors are found, the interface copies into the appropriate control blocks all user information provided by the pre-prompt exit. A control field in the LOGON work area (LWACTLS) contains bits that indicate what information the installation has provided.		LI800
4 If the pre-prompt exit has specified in the LOGON work area that the terminal user is not to be prompted (LWANOPR=1), that all LOGON information has been verified (LWANUAD=1), and that an ENQ is to be issued (LWANONQ=0), then the interface issues an ENQ on the user identification resource. If the resource is already in use, the pre-prompt exit is re-invoked to determine a course of action. The installation may choose to allow more than one user with the same user identification to be logged-on simultaneously (LWANONQ=1). In this case, the interface does not issue an ENQ on the user identification resource. Or, the installation may, instead, choose to terminate the session (LWADISC=1).	IKJEFLI	
Error Processing If either the LOGON pre-prompt exit interface (IKJEFLI) or the pre-prompt exit (IKJEFLD) cause an ABEND, the LOGON monitor's ESTAI routine IKJEFLGB is invoked by ABEND processing. In certain cases, the ESTAI routine schedules a re-attach of the LOGON monitor task. See Diagram, LOGON Monitor Recovery.	IKJEFLGB	

Diagram 23.8 LOGON Monitor Recovery (IKJEFLGB) (Part 1 of 2)

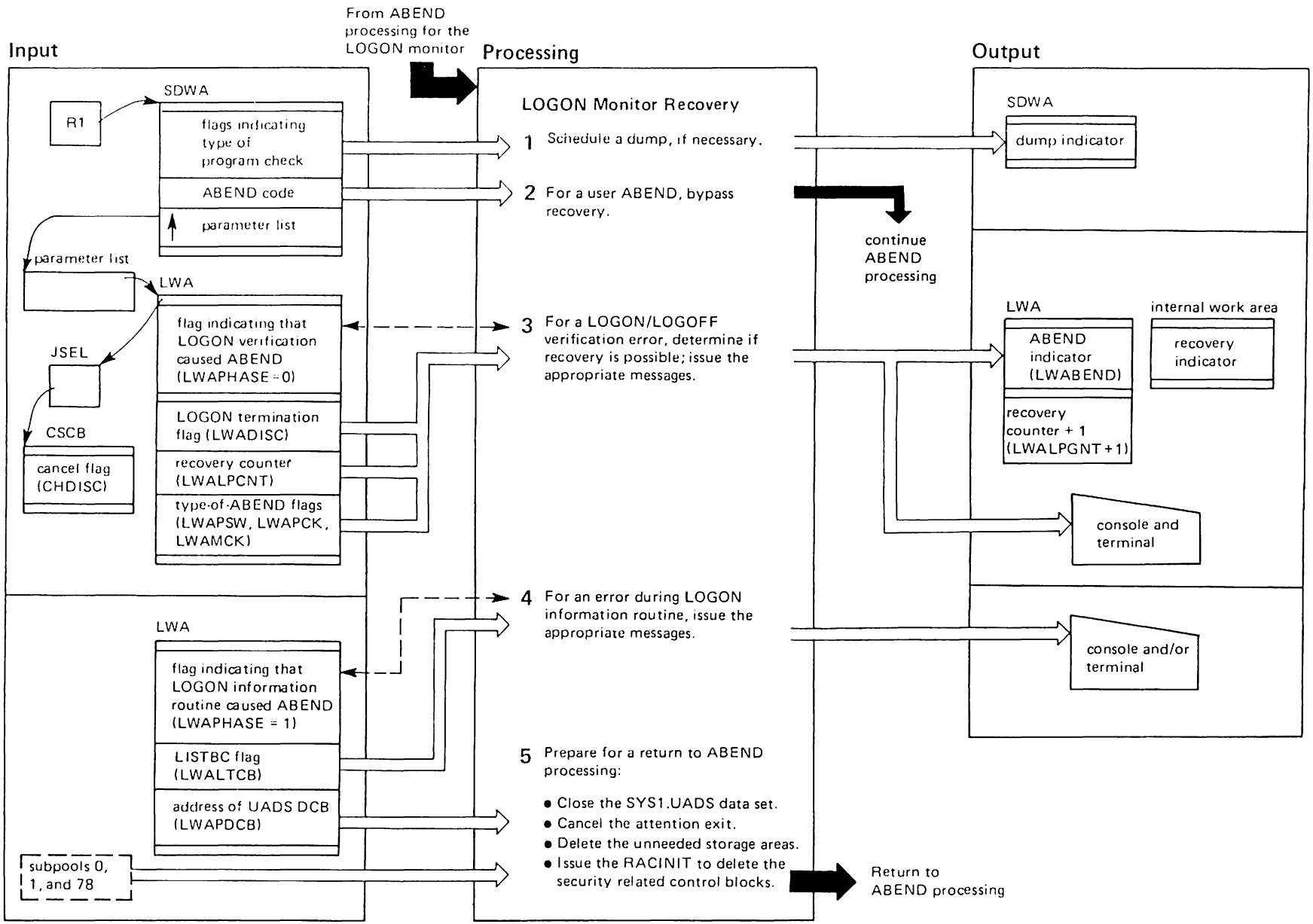


Diagram 23.8 LOGON Monitor Recovery (IKJEFLGB) (Part 2 of 2)

Extended Description	Module	Label
The LOGON monitor recovery routine receives control from ABEND processing following the abnormal termination of the LOGON monitor task. LOGON monitor recovery is an ESTAI routine that was specified on the ATTACH macro instruction when the LOGON monitor was attached by the LOGON scheduling task. If possible, a retry of the LOGON monitor is attempted by informing the LOGON scheduling task to re-attach the LOGON monitor (LWABEND = '1' B).	IKJEFLGB	
1 A dump is scheduled if the abnormal termination was the result of a program check or a PSW restart (an external interrupt from the operator).	IKJEFLGB	
2 If the ABEND code represents a user completion code, then recovery of the LOGON monitor task is not attempted. LOGON monitor recovery issues no error messages and passes control back to ABEND processing to continue the abnormal termination.	IKJEFLGB	
3 If the LOGON monitor abnormally terminated during LOGON/LOGOFF verification, recovery of the LOGON monitor task is scheduled (LWABEND=1). Recovery is not attempted in the following cases: • The system or the operator has canceled the terminal session (CHDISC=1). • The terminal session is scheduled for termination (LWADISC=1). • Four recoveries have already been attempted (LWALPCNT=4). • The current ABEND is the same type as the previous one (determined by checking bit settings in the LOGON work area: fields LWAPSW, LWAPCK, and LWAMCHK).	IKJEFLGB	PHASE1
LOGON monitor recovery builds and issues appropriate messages to the terminal and to the system operator. One set (IKJ56451I for the terminal and IKJ603I for the operator) is issued if the LOGON pre-prompt exit terminated abnormally (LWAINX1=1). Another set (IKJ56452I for the terminal and IKJ601I for the operator) is issued if LOGON/LOGOFF verification itself terminated abnormally (LWAINX1=0).		MSGINIT

Extended Description	Module	Label
4 If the ABEND occurred after the user's LOGON information has been processed and the terminal session has been scheduled (that is, LWAPHASE=1), recovery may not be necessary. If LWAPHASE=1, the ABEND occurred either during LISTBC command processing or during the issuing of the LOGON-proceeding messages (issued by LOGON module IKJEFLH). If LISTBC caused the ABEND (LWALTCB=1), LOGON monitor recovery issues an error message to the terminal (IKJ56406I) and the LISTBC task terminates. In this case, the scheduling of the terminal session proceeds normally. If the LOGON module IKJEFLH caused the ABEND, LOGON monitor recovery does not schedule a re-attach of the monitor (LWABEND=0) but does issue error messages to the terminal (IKJ56452) and to the operator (IKJ601).	IKJEFLGB	PHASE2
5 LOGON monitor recovery performs exit processing as follows:	IKJEFLGB	
• Closes the SYS1.UADS data set using the DCB address in the LOGON work area. If this address is zero, recovery does not issue the CLOSE macro instruction. Recovery also issues a DEQ on the SYS1.UADS directory resource.		CLOSUADS
• Issues a null STAX macro instruction to cancel the attention exit. Pressing the terminal attention key no longer has any effect on LOGON processing.		
• Frees the storage allocated to subpools 0, 1, and 78.		FREECORE
• If the user was running in the RACF environment, the RACINIT macro is issued to delete security related control blocks.		

Diagram 23.9 Pre-TMP Exit (IKJEFLJ) (Part 1 of 2)

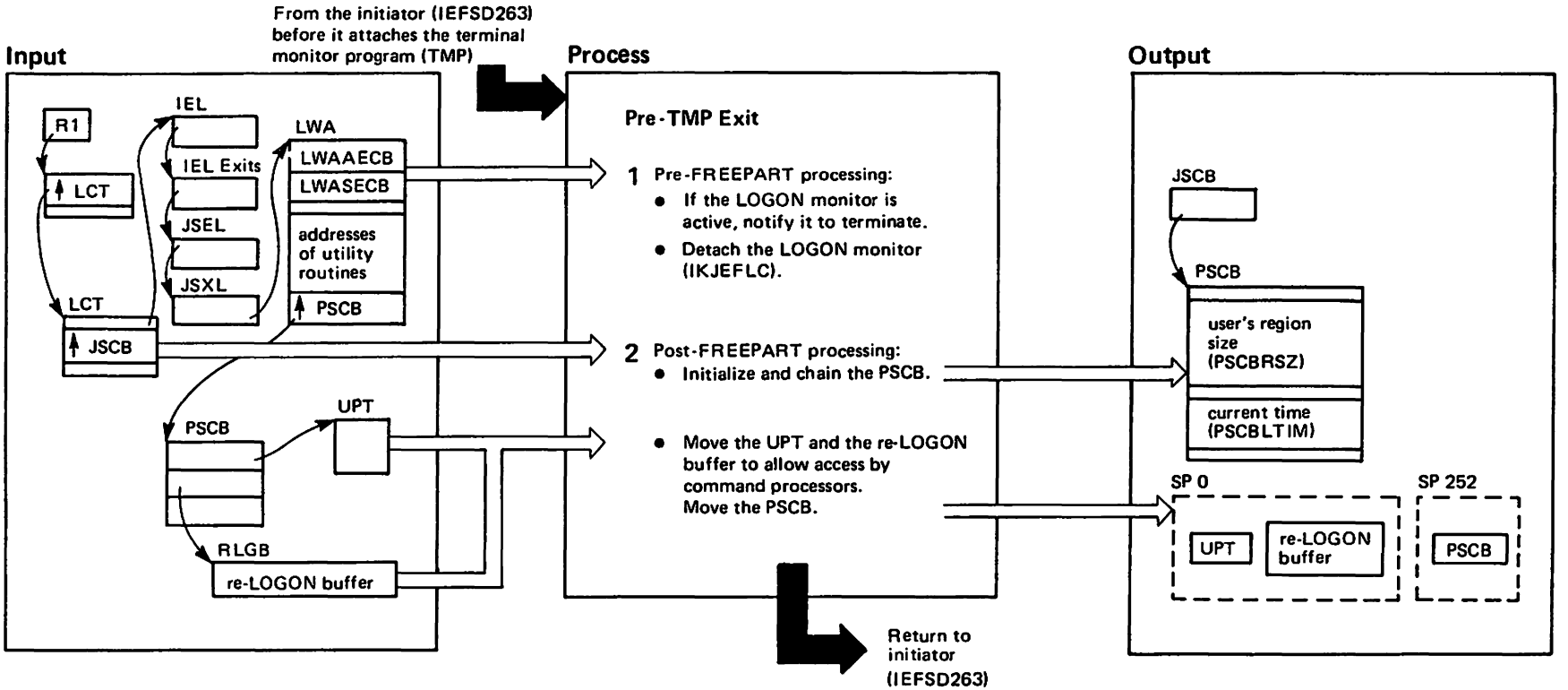


Diagram 23.9 Pre-TMP Exit (IKJEFLJ) (Part 2 of 2)

Extended Description	Module	Label
The initiator (IEFSD263) invokes the pre-TMP exit before attaching the terminal monitor program (TMP); it invokes the post-TMP exit after the TMP terminates. The pre-TMP exit prepares for the terminal session to begin by notifying the LOGON monitor task to terminate. The pre-TMP exit has two parts; an entry point name is assigned to each part. The first part is invoked before the initiator issues the FREEPART macro instruction (pre-FREEPART processing). The second part is invoked following the FREEPART (post-FREEPART processing).	IKJEFLJ	
1 This step represents pre-FREEPART processing. It is performed before the initiator issues the FREEPART macro instruction. Since the LOGON monitor task may still be active, the data areas it uses must not be deleted (by FREEPART) until the task is notified to terminate.	IKJEFLJ	IKJLM1
• Pre-FREEPART processing notifies the LOGON monitor task to terminate (LWASECB—post code 20). When the monitor task terminates, it notifies pre-FREEPART processing to continue (LWAPECB—post code 20). See LOGON Monitor (IKJEFLC).		
• The System Initiated Cancel (SIC) is notified that the TMP was executing when the line dropped or the user canceled. SIC will then notify the Post-TMP exit to free other users who are waiting on this memory. For example, SEND W/WAIT option sent to a canceled memory can cause the sender to wait forever unless the Post-TMP exit frees the sender.		
2 This step represents post-FREEPART processing. It is performed after the initiator issues the FREEPART macro instruction. Post-FREEPART processing now can move the UPT and the re-LOGON buffer to subpool 0 (which is deleted by the FREEPART).	IKJEFLJ	IKJLJ1
• Post-FREEPART processing invokes the SWA manager to obtain the user's region size from the step control block (SCB). The region size is stored in the protected step control block (PSCB). If the SCT indicates that the terminal session is a job with more than one step, post-FREEPART processing passes a non-zero return code back to the initiator, which then terminates the job. The current time of day is also stored in the PSCB for later use in computing the length of the terminal session.		
• The UPT and the re-LOGON buffer are moved to subpool 0 (a non-protected subpool) so that the command processors may alter them during the terminal session. The PSCB is moved to subpool 252; the command processors cannot alter data areas in subpool 252.		

Diagram 23.10 Post-TMP Exit (IKJEFLK) (Part 1 of 2)

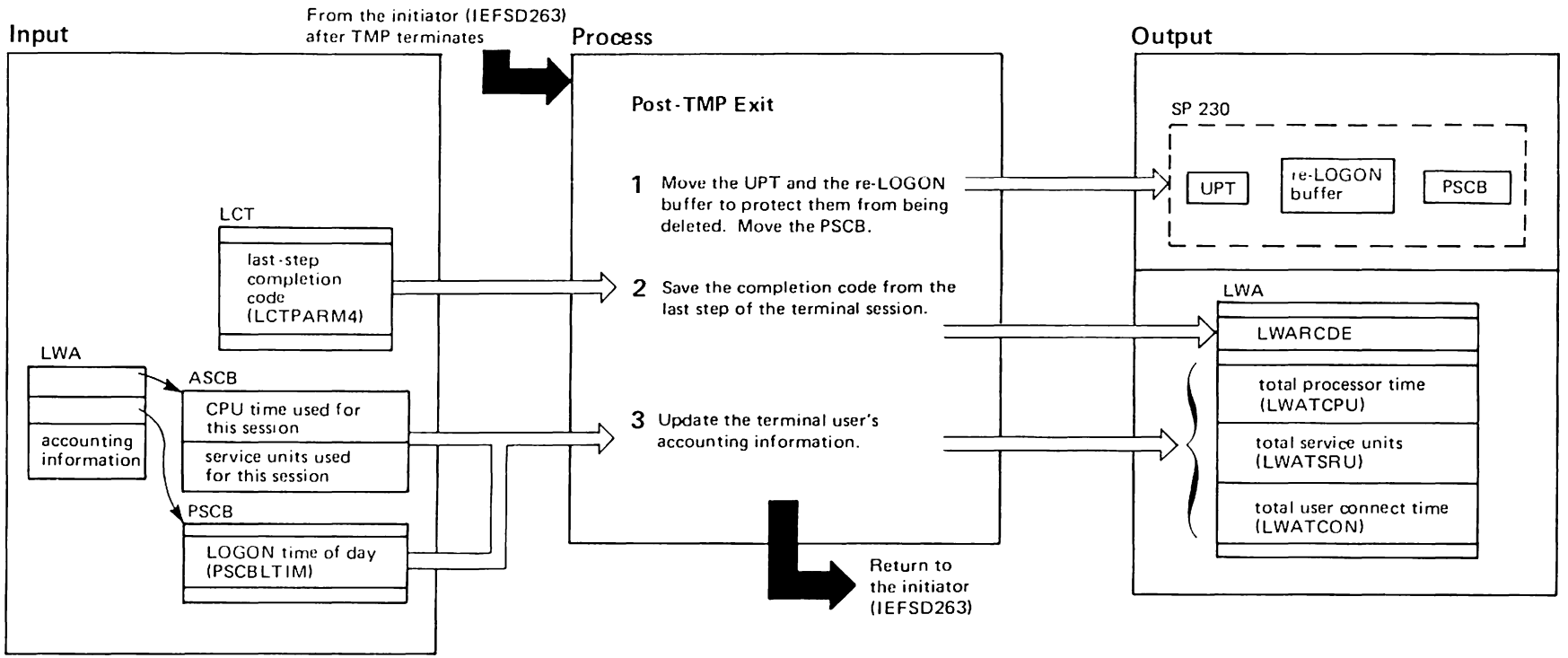


Diagram 23.10 Post-TMP Exit (IKJEFLK) (Part 2 of 2)

Extended Description	Module	Label
The initiator (IEFSD263) invokes the post-TMP exit after the TMP terminates. The post-TMP exit saves the completion code from the last step of the terminal session and updates the user's accounting information in the LOGON work area. Then, the initiator performs termination processing and passes control back to the LOGON scheduling task.		
1 The post-TMP exit moves the UPT and the re-LOGON buffer from subpool 0 to subpool 230 to prevent job scheduling from deleting them during job termination. The PSCB is also moved to subpool 230.	IKJEFLK	IKJLK1
2 The post-TMP exit saves the completion code from the last step of the terminal session, obtaining it from the linkage control table (LCT). The completion code is later analyzed by LOGOFF processing to determine if the terminal session terminated abnormally. See Diagram LOGOFF Processing.	IKJEFLK	IKJLK1
3 The post-TMP exit updates the accounting information in the LOGON work area to account for the system resources used during the terminal session that is now terminating.	IKJEFLK	
Error Processing If either the pre-TMP exit or the post-TMP exit causes an ABEND, LOGON scheduling's ESTAE routine IKJEFLS is invoked by ABEND processing. The function of this ESTAE routine is described under Error Processing in the diagram LOGON Initialization and Scheduling.	IKJEFLJ,K	

Index

OS/VS2 TSO
Command Processor
Logic Volume IV

SY28-0652-4

READER'S
COMMENT
FORM

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Note: ***Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.***

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name, company, mailing address, and date:

_____

_____

_____

_____

Note: Staples can cause problems with automated mail sorting equipment.
Please use pressure sensitive or other gummed tape to seal this form.

Cut or Fold Along Line

What is your occupation? _____

How do you use this publication? _____

Number of latest Newsletter associated with this publication: _____

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 or you may mail directly to the address in the Edition Notice on the back of the title page.)

OS/VS2 TSO Command Processor Logic Volume IV

SY28-0652-4 S370-39

Reader's Comment Form

Cut or Fold Along Line

Fold and tape Please Do Not Staple Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

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

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department D58, Building 921-2
PO Box 390
Poughkeepsie, New York 12602

Fold and tape Please Do Not Staple Fold and tape

Printed in U.S.A.

IBM®

System Library Supplement

This Supplement No. SD23-0299-0
Date March 1, 1985
File No. S370-39
For Base Publication SY28-0652-4, OS/VS2 TSO Command Processor Logic Volume IV

Prerequisites None

This supplement contains replacement pages for *CP Logic Volume IV*. It replaces Supplement LD23-0273-0.

Before inserting any of the attached pages into *CP Logic Volume IV*, read ***carefully*** the instructions on this cover. They indicate when and how you should insert pages.

Pages to be Removed	Attached Pages to be Inserted*
Cover - Edition Notice	Cover - Edition Notice
3 - 6	3 - 6
9 - 14	9 - 14
37 - 44	37 - 44
85 - 86	85 - 86
117 - 120	117 - 120
135 - 138	135 - 138
149 - 150	149 - 150
155 - 156	155 - 156
I1 - I2	I1 - I2

*If you are inserting pages from different Newsletters/Supplements and ***identical*** page numbers are involved, always use the page with the latest date (shown in the slug at the top of the page). The page with the latest date contains the most complete information.

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

Summary of Amendments

This supplement was reissued because of the major revision to base SY28-0652. The parallel TMP structure does not apply to the MVS/XA environment without TSO/E.

Note: Please file this cover letter at the back of the publication to provide a record of changes.

**IBM Corporation, Information Development, Dept. D58, Building 921-2,
P.O. Box 390, Poughkeepsie, New York 12602**

Printed in U.S.A.

OS/VS2 TSO Command Processor Logic Volume IV

SY28-0652-4

S370-39

Printed in U.S.A.

SY28-0652-04