ST75-0004-0
(formerly SY35-0004-0)

Systems

OS/VS2 TSO
Command Processor Logic
Volume III: TEST

Release 1

IBM

First Edition (September, 1972)

This edition applies to release 1 of OS/VS2, and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. Changes are continually made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest "IBM System/360 and System/370 Bibliography," GA22-6822, and the current SRL Newsletters.

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

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Programming Writing, Department 27D, P.O. Box 1900, Boulder, Colorado 80302.

Preface

This publication describes the internal logic of the TSO TEST Command Processor. It contains text introductions, method-of-operation diagrams, program organization descriptions, a module directory, and data area formats. This publication is written for persons who maintain or modify TSO TEST. It is not required by those who use TSO TEST to process programs or who write programs that are processed by TSO TEST.

This book is divided into seven sections, each containing a particular type of information, as follows:

- "Section 1: Introduction," which describes the TEST program in general and its relationships to other parts of the TSO control program, and tells the reader where he may find descriptions of different parts of the program. It includes an overview of TEST logic.
- "Section 2: Method of Operation," which describes graphically how each functional area of the program works. The emphasis here is on function: what is done, why it is done, what tables and control blocks are used, and what interrelationships exist. System interfaces are indicated.
- "Section 3: Program Organization," which contains module descriptions. The module descriptions are arranged alphabetically by module entry name. The module descriptions contain reference information, such as register usage, system routines called, and return codes.
- "Section 4: Directories," which contains alphamerically organized lists of important internal symbols, entry names, and module names. Each name is defined and cross-referenced to the object module that contains the named item.
- "Section 5: Data Areas," which contains a brief description and the format of each major table and control block used by the TEST program.
- "Section 6: Diagnostic Aids," which contains detailed information on how the TEST program handles STAE-intercepted abnormal terminations that occur in the TEST program.
- "Section 7: Appendix," which describes the setting of pseudo breakpoints by particular TEST modules.

Prerequisite Information

The information in the following manuals, as they apply to the TEST command, should be understood before you read this publication:

- *OS/VS2 TSO Command Language Reference*, GC28-0646, which describes the format and meaning of the TEST command and its subcommands.
- *OS/VS2 TSO Guide to Writing a Terminal Monitor Program or a Command Processor*, GC28-0648, which describes how to use the TEST command and its subcommands to test a new program.

The following three books should be available for reference, while you are reading this publication:

- *OS/VS2 TSO Terminal Monitor Program and Service Routines Logic*, SY28-0650, which describes linkages between the TEST program and the Terminal Monitor Program (TMP) and discusses in detail the TMP STAI Exit routine and the service routines. The service routines include: IKJPARS, IKJDAIR, IKJDELT, and the I/O service routines invoked by the macro instructions STACK, GETLINE, PUTLINE, and PUTGET.

- *OS/VS2 Supervisor Logic*, SY27-7244, which describes the logic of system services that the TEST program uses: exiting (SVC 3), dispatching, Contents Supervision, and the ABEND/STAE interface routines.
- *OS/VS2 TSO Control Program Logic*, SY28-0649, which describes the operation of the attention handler (modules IKJVAR04 and IKJVAR05).

Contents

Figures

Diagrams

Section 1: Introduction

The TEST command processor, referred to as TEST in this publication, permits you to test a program that was written in assembler language or some other language that produces an assembler language listing. To perform its functions, TEST uses subcommands and service routines which are explained in this section. TEST may be invoked in any one of three situations, each of which is also described in this section. Finally, the relationship of TEST to other Time Sharing Option (TSO) tasks is shown in this section, and the TEST system requirements are given.

Using TEST and Its Subcommands

You can test a problem program using basic debugging functions supplied by the following TEST subcommands:

- The TEST Assignment function, which changes the contents of specified program locations in main storage or the contents of specific registers.
- AT, which interrupts the problem program at a specified location or locations so that you can issue subcommands to be executed at these locations.
- CALL, which initializes registers and begins execution of the problem program at an address that you specify.
- COPY, which transfers data from storage to storage, from register to register, from register to storage, and from storage to register. It also allows you to place storage addresses in a register.
- DELETE, which removes a specified module from main storage.
- DROP, which removes one or more symbols from the symbol table for the problem program. Only those symbols previously established with the EQUATE subcommand can be removed using DROP.
- END, which frees TEST storage when a terminal session is complete.
- EQUATE, which assigns a symbol to an absolute or symbolic address in the problem program or data area. You specify as subcommand operands the symbol and the address to which it is to be assigned.
- FREEMAIN, which frees a specified number of previously acquired bytes of main storage as if a FREEMAIN macro instruction had been issued.
- GETMAIN, which acquires a specified number of bytes of main storage as if a GETMAIN macro instruction had been issued by the problem program.
- GO, which establishes parameters so that the machine module (IKJEGMNL) can either start or restart problem program execution at a particular address. If you specify an address, execution resumes at that address. If the problem program was interrupted by a breakpoint, execution resumes at the instruction following the breakpoint.
- LIST, which prints the contents of registers or of a main storage area either at a terminal or to a data set.
- LISTDCB, which prints the contents of a Data Control Block (DCB) either at a terminal or to a data set.

- LISTDEB, which prints the contents of the basic section and any direct access sections of a Data Extent Block (DEB) at a terminal or to a data set. You can receive the entire contents of those sections of the DEB or you can specify one or more fields within a DEB.
- LISTMAP, which prints information about a problem program region at a terminal or to a data set. The information includes region size; problem program Task Control Block (TCB) address; names, lengths, and locations of all programs running under the problem program TCB; type and identification of all active request blocks; and the number, location and size of user subpools.
- LISTPSW, which prints the contents of a Program Status Word (PSW) (eight bytes) at a terminal or to a data set. If you do not specify the address of the requested PSW, the LISTPSW subcommand retrieves the PSW from the most Request Block (RB) on your RB queue.
- LISTTCB, which prints the contents of a TCB at a terminal or to a data set. You can receive the entire contents of a TCB or you can specify one or more fields within a TCB.
- LOAD, which loads a module of the problem program.
- OFF, which removes user breakpoints no longer needed and replaces them with the original problem program instructions.
- QUALIFY, which allows you to specify the name of a program in main storage, and a Control Section (CSECT) within that program, or a TCB address from which relative addresses can be calculated.
- RUN, which causes the problem program to be executed without interruption by any previously established breakpoints. RUN causes the problem program to execute to completion from its beginning or from an address you specify. RUN then terminates TEST so that the TMP may dispatch your program as its subtask.
- WHERE, which prints at a terminal the absolute address of either a symbolic or relative address in the problem program, or the absolute address of either a load module or an entry point within a load module.

Service Routines Used by TEST

The TMP, TEST, and the problem program all use the same service routines to obtain TSO services. When a service routine has been invoked, it also resides in the user's region until the routine has completed execution, unless your installation has placed that routine in the time sharing link pack area.

The following service routines reside on SYS1.LINKLIB: PUTLINE, GETLINE, PUTGET, STACK (module IKJPTGT), PARSE (module IKJPARS), and Command Scan (module IKJSCAN).

The following service routines reside on SYS1.CMDLIB: Default Service Routine (module IKJEHDEF, alias IKJDFLT), and Dynamic Allocation Interface Routine (module IKJEFD00, alias IKJDAIR). The service routines are described in detail in *OS/VS2 TSO Terminal Monitor Program and Service Routines Logic*, SY28-0650.

Two TEST load modules are SVC routines and reside in virtual memory. When an SVC 97 instruction is issued, the OS/VS2 Supervisor fetches IGC0009G; when an SVC 61 instruction is issued, the Supervisor fetches IGC0006A. Each SVC routine is purged by the Supervisor when the routine exits.

Invoking TEST

TEST can be invoked when you want to test your command processor or problem program that is not currently executing, when you want to test an executing program that is abnormally terminating, or when you want to test your command processor or problem program before it has finished executing.

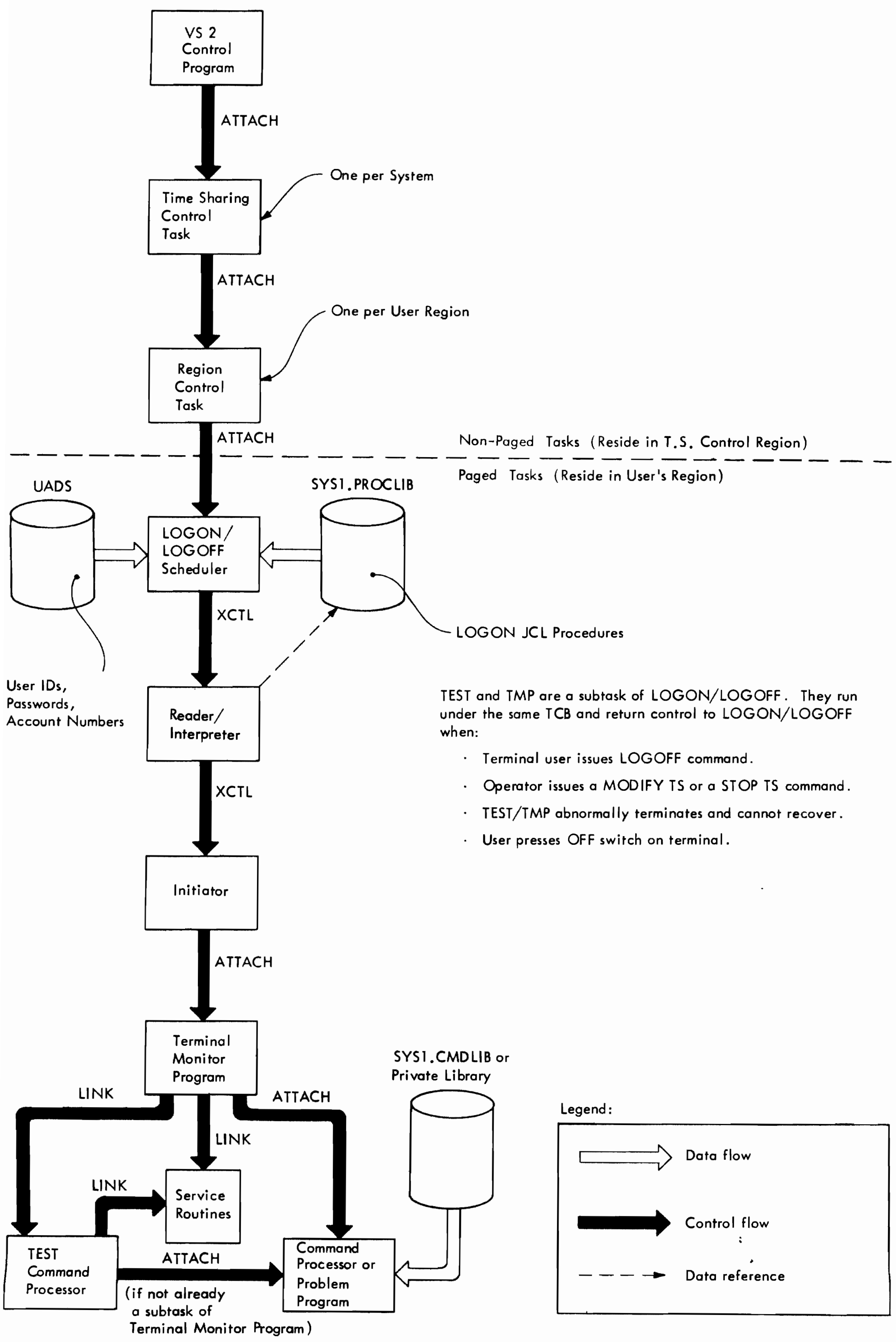
1. When you want to test your command processor or problem program that is not being currently executed, the TMP issues a READY message to request the next command. You respond with the command "TEST xxxx" (where xxxx is the name of your problem program).
2. When you want to test a program that is currently being executed and which has begun to abnormally terminate, you receive a diagnostic message from the TMP followed by a READY message. The TMP is in effect asking, "Do you want to terminate your program or test it?" If you respond with a null line (a carrier return), the TMP returns control to the supervisor's ABEND routine to abnormally terminate your program. If, however, you issue the TEST command with no operands, the TMP does not detach your currently executing command processor or problem program. Instead, it invokes TEST to debug the defective program. Since the problem program is already in memory, there is no need to fetch it.
3. When you change your mind before your command processor or problem program completes and you press the attention key, the TMP receives control after the time sharing control program processes the attention interruption. It then queries you by issuing its usual READY message. You respond by entering TEST, with or without the program name. If you omit the program name, the TMP assumes that you want to test your currently executing problem program or command processor as in the second situation. If, however, you enter a program name as a TEST command operand, the TMP assumes that you have changed your mind and want to test a program other than the one currently executing. The TMP detaches the currently executing program task and invokes TEST. TEST loads the problem program you want to debug as in the first situation. (If you enter the same program name as the one that was running, you will lose the copy of the program currently in memory, and TEST will fetch a new copy.)

When the TMP issues a LINK macro instruction to invoke TEST, the TEST initialization module (IKJEGINT) is fetched from SYS1.CMDLIB by Contents Supervision. IKJEGINT contains its own abnormal termination routines and the I/O module, IKJEGIO. IKJEGINT attaches the TEST load module (IKJEGLDR), if the problem program is not already a subtask of the TMP. If IKJEGLDR is attached, it fetches the problem program, then purges itself by issuing an XCTL macro instruction to the problem program's entry point.

The problem program is fetched in one of two ways:

- The TMP causes the program to be fetched by attaching the program directly or by attaching the program's invoker. An example of the latter is the attaching of the RUN command processor, which then attaches the problem program. In either case, the terminal user has not issued the TEST command.
- IKJEGINT attaches IKJEGLDR, which then directs the fetching of the problem program. The actual fetching is done by either Contents Supervision (through the program fetch routine) or the VS2 load module. The choice depends on the keyword, LOAD or OBJ, that the user specifies in the TEST command.

Relationship of TEST to Other TSO Tasks



TEST and TMP are a subtask of LOGON/LOGOFF. They run under the same TCB and return control to LOGON/LOGOFF when:

- Terminal user issues LOGOFF command.
- Operator issues a MODIFY TS or a STOP TS command.
- TEST/TMP abnormally terminates and cannot recover.
- User presses OFF switch on terminal.

Figure 1. Relationship of TEST to other TSO Tasks

Figure 1 illustrates the following points:

- The TMP is attached by an initiator after a reader/interpreter has read the LOGON procedure statements.
- Both TEST and the TMP are indirectly a subtask of LOGON/LOGOFF. Both programs return control to LOGOFF when the terminal user enters a LOGOFF command, the operator issues a MODIFY TS or STOP TS command, the TEST/TMP task abnormally terminates and cannot recover, or the terminal user presses the OFF switch on his terminal.
- TEST operates as part of the same task as the TMP; that is, both programs run under the same TCB.
- Either the TMP or TEST can attach a command processor or a problem program. The command processor or problem program then becomes a subtask of the TEST/TMP task.
- LOGON/LOGOFF, the TMP, TEST, and the command processor or problem program all normally reside in the user's time sharing region. An installation can, however, place LOGON/LOGOFF and the TMP in the time sharing link-pack area.

TEST System Requirements

TEST occupies approximately 90K bytes as it resides on SYS1.CMDLIB. IKJEGMNL occupies approximately 21K bytes. IKJEGMNL remains resident after once being fetched and contains the following:

- Attention Exit routine, IKJEGATN.
- STAE Exit routine, IKJEGSTA.
- A copy of the I/O module, IKJEGIO.
- Convert module, IKJEGCVT.
- The TEST subcommand name table, IKJEGSCD.
- The user subcommand name table, IKJEGSCU.
- The break element removal routine, IKJEGSRH.

Figure 2 gives the bytes of memory each TEST subcommand occupies. Four subcommands (Assignment, AT, END, and LIST) require the first subcommand load module to fetch and transfer control to a second load module, which then purges the invoking module.

Subcommand	Bytes of Memory Occupied
Assignment	5,806
AT	5,716
CALL	See GO
COPY	2,439
DELETE	See LOAD
DROP	3,588
END	664
EQUATE	See DROP
FREEMAIN	See LOAD
GETMAIN	See LOAD
GO	2,654
LIST	7,912
LISTDCB	3,800
LISTDEB	3,968
LISTMAP	2,044
LISTPSW	2,000
LISTTCB	6,154
LOAD	4,964
OFF	2,832
QUALIFY	3,340
RUN	See GO
WHERE	3,088

Figure 2. Memory Occupied by TEST Subcommands

Section 2: Method of Operation

This section describes TEST logic. It emphasizes functional descriptions of the TEST processor through method-of-operation diagrams and text, in some cases.

The Method-of-Operation diagrams in this section are designed to give you an overall understanding of the TEST command processor, its functions, and its subcommands. There are three levels of diagrams; each level directs you to the next lower level of diagrams for greater detail. Using these diagrams and the visual table of contents, you can quickly reach the level of detail you need at an particular time.

Level 1 The TEST Overview shows the major steps in the processing sequence of TSO TEST.

Level 2 The functional overviews, Diagrams 1, 2, and 3, divide the processing sequence into three functional areas: initializing TEST, processing under TEST control, and processing under problem program control. Initializing TEST establishes the TEST environment; the processing functions control the TEST environment that initialization established.

Level 3 The functional breakdowns, Diagrams 1.1 - 1.8 and 2.1 - 2.32 show in greater detail the functions described in the first two functional overviews. For greater detail of processing under problem program control, you are directed to specific diagrams in the first two functional areas.

These diagrams, at the end of this section, are graphically broken into three areas: input, processing, and output. An expanded description explaining each processing step is provided for each diagram. These descriptions also become more detailed at each succeeding level.

Arrows and other symbols are used to direct you through the diagrams.

➡ Heavy black arrow indicates diagram entry point.

⇨ White arrow indicates data movement.

▨ Hatched arrow indicates data alteration.

➡ Black arrow indicates control flow.

→ Thin black arrow indicates a pointer.

▷ Off-page connector leads to a related diagram.

'EST Initialization Function

The initialization function has three main parts: establishing the TEST environment, preparing a problem program for TEST, and inserting breakpoints. Method-of-Operation Diagram 1 is the initialization overview. Figure 3 is a summary of the initialization function.

Operation	Symbolic Name	Diagram Number
TEST initialization	IKJEGINT	1
Loading the problem program	IKJEGLDR	1.1, 1.2
Breakpoint processing	IKJEGAT IKJEGATD IG0009G IKJEGMNL IKJEGOFF IKJEGSRH	1.4 - 1.8

Figure 3. Initialization Function Summary

Establishing the TEST Environment

To establish the TEST environment, IKJEGINT performs the following tasks:

- Obtains storage for TEST tables and work areas.
- Initializes the tables.
- Assumes control of the problem program if the program is in the process of being abnormally terminated or if the terminal user has interrupted its processing with an attention interruption.
- Attaches IKJEGLDR to fetch either the problem program or the OS/VS load module, depending upon the presence of either the LOAD or OBJ keyword operand on the TEST command (LOAD is the default option).
- Issues an XCTL macro instruction to purge module IKJEGINT and fetch and transfer control to IKJEGMNL.

Preparing a Problem Program for TEST

If the problem program is not already a subtask of TEST when TEST is entered, it is brought into main storage by IKJEGLDR, which is attached by IKJEGINT, which attaches IKJEGLDR.

IKJEGLDR fetches the problem program in either of two ways. If the problem program has been link-edited, IKJEGLDR issues an XCTL macro instruction for Contents Supervision to fetch the specified load module and transfer control to it. The second way IKJEGLDR fetches the problem program is to invoke the OS/VS load module to load the specified object module and then issue an XCTL macro instruction to transfer control to the loaded problem program. Loading a problem program is shown in Diagram 1.1 and Diagram 1.2.

When a problem program is fetched, data set information about that program is saved by IGC0006A, whose main function is to store in an SVC information block certain information regarding the data set on which a fetched problem program resides. This information is needed to resolve symbolic addresses. Diagram 1.3 illustrates the main function of IGC0006A.

Inserting Breakpoints

A breakpoint is an SVC 97 instruction that causes a planned transfer of control from the problem program to TEST, which allows TEST to control the problem program. A breakpoint can be one of two types: pseudo or user. Two types of breakpoints are inserted: pseudo breakpoints or user breakpoints.

Pseudo Breakpoints

A pseudo breakpoint is inserted by a TEST module so the module can route control to either IKJEGMNL or IKJEGINT. The SVC 97 instruction is placed in the Test Communication Table (TCOMTAB) rather than in the problem program, and the problem program's restart address points to the SVC 97 instruction. Diagram 1.4 and Diagram 1.5 describe pseudo breakpoints. Figure 4 describes the conditions for which pseudo breakpoints are inserted and indicates which TEST module inserts the breakpoint.

Condition	TEST Module
To process a new subcommand.	IKJEGAT
When the problem program has been loaded by the OS/VS loader.	IKJEGLDR
When a load module of the problem program is to be invoked by the TEST load module.	IKJEGINT
When a subtask of TMP begins to terminate abnormally.	IKJEGINT
When a subtask of TMP is interrupted by an attention interruption.	IKJEGINT
When an ABEND occurs during the execution of the problem program or of the HELP command processor.	IKJEGMNL
When the user enters or simulates an attention interruption during the TEST session.	IKJEGATN
To force a task switch.	IKJEGLDF
To detect the completion of the problem program.	IKJEGINT
To force qualification of TCB at source level.	IKJEGQFY

Figure 4. Conditions Requiring a Pseudo Breakpoint

User Breakpoints

A user breakpoint is inserted in the problem program by the AT subcommand processor, IKJEGAT, when the user enters an AT subcommand. An immediate user breakpoint is one requested without the DEFER keyword. IKJEGAT inserts the SVC 97 instructions at specified locations in the problem program and saves or chains the breakpoint information in a break element it builds: See "Section 5: Data Areas" for the format and contents of a break element BRKELEM.

A deferred user breakpoint is one requested by an AT subcommand that contains the DEFER keyword. IKJEGAT does not place any SVC 97 instructions in the specified module for deferred user breakpoints, because the module is not yet in the user's region. IKJEGAT merely saves the information that describes the breakpoints in a defer element queue it builds. Later, when the specified module is available, IKJEGMNL uses IKJEGAT to activate the deferred breakpoint. User breakpoints are described in Diagrams 1.6, 1.7, and 1.8.

Processing Controlled by TEST

Once the TEST environment has been established, subcommands specified by the user are processed. (A subcommand summary is included in this section.) In processing subcommands TEST performs internal and exceptional operations discussed in this section. Subcommand processing continues until the user enters a GO, CALL, RUN, or END subcommand. When one of these four subcommands is entered, TEST either turns control over to the problem program or terminates the TEST session. The overview of processing controlled by TEST is described in Diagram 2.

Subcommand Processing

IKJEGMNL receives the subcommand the user entered and links to the associated subcommand processor when one of the following occurs:

- A breakpoint is reached in the problem program. IGC0009G issues a POST macro instruction to return control to IKJEGMNL.
- An attention request interrupts a TEST module.
- An ABEND in the problem program is intercepted by the TMP STAI Exit.

- Another subcommand processor executes a code of 0.
- An ABEND in a subcommand processor is intercepted by the TEST STAE Exit Routine.

Subcommand processing that is similar for all subcommands is described in Diagram 2.1. Specific subcommand processing is shown in Diagrams 2.2 through 2.23 and is summarized in Figure 5.

Operation	Symbolic Name	Diagram Number
Assigning value(s) to one or more locations	IKJEGPCH IKJEGASN	2.2
Specifying a point of interruption (breakpoint) in the problem program	IKJEGAT IKJEGATD	2.3
Specifying a restart address and a parameter list	IKJEGCAL entry point in IKJEGGO	2.4
Moving data fields or addresses	IKJEGCPY	2.5
Removing a module from main storage	IKJEGDEL entry point in IKJEGLDF	2.6
Removing one or more names from the symbol table in memory	IKJEGDRP entry point in IKJEGEQU	2.7
Ending the test session	IKJEGEND IKJEGOFF	2.8
Placing a name and its absolute address and attributes in the symbol table in memory	IKJEGEQU	2.9
Freeing an area of memory	IKJEGFRE entry point in IKJEGLDF	2.10
Allocating an area of memory	IKJEGGET entry point in IKJEGLDF	2.11
Specifying the restart address	IKJEGGO entry point in IKJEGGO	2.12
Displaying one or more locations or registers	IKJEGLST IKJEGLSA	2.13
Displaying a DCB	IKJEGDCB	2.14
Displaying a DEB	IKJEGDEB	2.15
Displaying an map of storage assigned to the problem program	IKJEGMAP	2.16
Displaying a PSW	IKJEGPSW	2.17
Displaying a TCB	IKJEGTCB	2.18
Loading a module	IKJEGLOD entry point in IKJEGLDF	2.19
Removing a breakpoint from the problem program	IKJEGOFF	2.20
Specifying the QUALIFY load module or entry name	IKJEGQFY	2.21
Specifying the restart address and the end of the test session	IKJEGRUN entry point in IKJEGGO	2.22
Determining the current load module, entry name and offset, or the current absolute address	IKJEGWHR	2.23

Figure 5. Summary of TEST Subcommands

Once a subcommand has been processed, a return code is issued to IKJEGMNL. Figure 6 lists these return codes, their meaning, and the action taken by IKJEGMNL.

Code	Description
0	Normal return from all subcommand processors, except GO, RUN, QUALIFY and the IKJEGLDF module. Processes another subcommand, unless the previously processed subcommand is RUN or END. END causes a return to the TMP. RUN causes a transfer of control to the problem program.
4	Normal return from the GO, CALL, and QUALIFY subcommand processors and from the IKJEGLDF module. This return code is also set when the AT subcommand processor activates a deferred breakpoint. Branches to location IKJEGCTL in IKJEGMNL to restart the SVC 97 routine or the SVC 61 routine. When the SVC 97 routine is restarted, the problem program is restarted.
12	Normal return from the RUN subcommand processor. This return code is also set when a critical error occurs in RUN processing. Branches to location RUNRTURN to remove all breakpoints, clean up TEST, post the subtask ECB, and return to the TMP. In this case, the problem program is not detached by the TMP. It will be restarted by the OS/VS Dispatcher and will run to completion after IKJEGMNL has returned to the TMP. If a critical error has occurred in RUN processing, normal END processing occurs and the problem program does not get dispatched.
16	Return from a subcommand processor that has been interrupted by an attention interruption from the terminal. IKJEGMNL branches to location SCREQ to process the new subcommand entered in response to the attention interruption. IKJEGMNL doesn't free the previously used input buffer, since the attention exit routine has already done so. IKJEGMNL gets the subcommand from the buffer pointed to by the IEBUF field of TCOMTAB.
20	Return from the retry routine of a subcommand processor that has prevented an ABEND. IKJEGMNL branches to location SCREQ1 to get a new subcommand. If END processing has started, IKJEGMNL finishes the termination processing.

Figure 6. Codes Returned to IKJEGMNL After Subcommand Processing

Internal TEST Operations

TEST performs certain internal operations during processing. Three of these operations are discussed in this section:

- Converting character codes.
- Translating symbolic addresses, determining CSECT names, and resolving entry names.
- Communicating with the terminal and creating the print data set.

Figure 7 lists the internal operations, names the modules involved with each operation, and gives the diagram number that describes the operation.

Operation	Symbolic Name	Diagram Number
Converting character codes	IKJEGCVT	2.24
Translating symbolic addresses, determining CSECT names, and resolving entry names	IKJEGSYM	2.25
Communicating with the terminal and creating a print data set	IKJEGIO	2.26
Invoking the HELP command	IKJEGCIV	2.27
Altering fields of the problem program's TCB or RB	IGC0009G	2.28
Searching for and removing specified breakpoints from the active queue	IKJEGSRH	2.29

Figure 7. Internal TEST Operations

Converting Character Codes

Certain TEST subcommand processors receive input addresses which they must convert to binary before they can process the addresses. Similarly, various subcommand processors must convert values to printable form in order to list the values in a message to the terminal or to include the values on the print data set. To convert addresses and values to binary or printable form, the subcommand processors invoke the convert module, IKJEGCVT. Diagram 2.24 describes IKJEGCVT.

IKJEGCVT is invoked both directly and indirectly by subcommand processors. It is invoked directly to convert one or more values to binary or printable form. It is invoked indirectly, through a validity check exit routine from IKJPARS, to convert one or more addresses to binary. Figure 8 shows the subcommand processors that invoke IKJEGCVT.

Subcommand Processor	Module Name	Invokes Directly	Invokes Indirectly
Assignment	IKJEGPCH	X	X
AT	IKJEGAT		X
CALL	IKJEGGO		X
EQUATE	IKJEGEQU		X
FREEMAIN	IKJEGLDF		X
GO	IKJEGGO		X
LIST	IKJEGLST	X	X
LISTDCB	IKJEGDCB	X	X
LISTDEB	IKJEGDEB	X	X
LISTPSW	IKJEGPSW	X	X
LISTTCB	IKJEGTCB	X	X
OFF	IKJEGOFF		X
QUALIFY	IKJEGQFY		X
RUN	IKJEGGO		X
WHERE	IKJEGWHR		X

Figure 8. Subcommand Processors That Invoke IKJEGCVT

Address Conversion: The processing for address conversions depends on the type of address being converted and whether the conversion is to binary or to printable form. Figure 9 lists the input parameters and the processing for each type of address. routine belonging to a subcommand processor. After processing, control is returned to the balidity check exit routine. The output of the conversion to binary form is located in the PSEUSER field of the PDE. The output of the conversion to printable form is located in a 32-byte work area pointed to by the CONAREA field of TCOMTAB.

Part 1: Binary Form

	Input Parameter			Processing									
Type of Address	Register 0	Register 1	PDEFLG4 Field of PDE	Invokes IKJEGSYM to get absolute address of symbol or entry name	Converts to packed decimal	Converts to Binary	Processes Relative Address	Processes Indirect Address(es)	Processes Address Expression	Gets information from symbol information block, if necessary	Indicates normal return	If range specified, processes other address(es) similarly	Returns to caller (validity check exit routine)
Absolute	Positive = Convert to Binary	negative = convert an Address Points to an Address-type PDE	ABSADDR '00'			X	X	X	X	X	X	X	X
Relative	Positive = Convert to Binary	Points to an Address type PDE	RELADDR '40'	X		X	X	X	X	X	X	X	X
Symbolic	Positive = Convert to Binary	Points to an Address type PDE	SYMADDR '80'	X				X	X	X	X	X	X
Register	Positive = Convert to Binary	Points to an Address type PDE	GENR '20'		X	X		X	X	X	X	X	X
Entry Name Only	positive = converts to binary	Points to an Address type PDE	CTONLY '04'	X							X	X	X

Figure 9. (Part 1 of 2) Address Conversion To Binary and Printable Form

Part 2: Printable Form										
	Input Parameters			Processing						
Type of Address	Register 0	Register 1	PDEFLG4 field of PDE	Subtracts base from input	If floating point specified, sets up special output format	Converts hex. to printable form	Adds hex. digits to each 4 bits to form printable character	Formats and moves to output area	Indicates normal return	Returns to validity check exit
Absolute	Negative = Convert to Printable	**negative - convert an Address** Points to an Address type PDE	ABSADDR '00'				X	X	X	X
Relative	Negative = Convert to Printable	Points to an Address type PDE	RELADDR '40'	X		X		X	X	X
Other Non-absolute Types	Negative = Convert to Printable	Points to an Address type PDE	Codes other than the above		X	X			X	X

Figure 9. (Part 2 of 2) Address Conversion To Binary and Printable Form

Value Conversion: The processing for value conversion depends on the type of value being converted and whether the conversion is to or from binary form. Figure 10 lists the input parameters and the processing for each type of value. IKJEGCVT performs the processing and receives control from a subcommand processor. After processing control is returned to the caller. The output of the conversion is located in a 32-byte work area pointed to by the CONAREA field of TCOMTAB. The length of the output is identified in the first byte.

Type of Conversion	Input Parameters			Processing									
	Register 0	Register 1	Hex code at offset +7 in PDE	Changes certain characters	Transfers each '1' and each '0' to output	Sets sign	Converts to packed decimal	Converts register and displacement to packed decimal	Converts to binary	Combines register and displacement	Moves to output work area	Indicates normal return	Returns to caller
From Hexadecimal to Binary	Positive = Convert to Binary	**Positive = Convert a Value** Points to a Value PDE	'04'	X			X				X	X	X
From Binary to Binary	Positive = Convert to Binary	Points to a Value PDE	'08'		X						X	X	X
From Fixed-point Decimal to Binary	Positive = Convert to Binary	Points to a Value PDE	'10' or '14'			X			X		X	X	X
From Constants: A, Y, V	Positive = Convert to Binary	Points to a Value PDE	A: '20' Y: '24' V: '2C'				X		X		X	X	X
From S-Constant to Binary	Positive = Convert to Binary	Points to a Value PDE	'28'					X	X	X	X	X	X
From Packed Decimal to Binary	Positive = Convert to Binary	Points to a Value PDE	'30'				X		X		X	X	X

Figure 10. (Part 1 of 2) Value Conversion From Binary Form and To Binary Form

Type of Conversion	Input Parameters			Processing							
	Register 0	Register 1	Hex code at offset +7 in PDE	Converts to hex, 4 bits at a time	Converts each type instruction via a separate routine	Converts to packed decimal	Checks validity of input	Converts to zoned decimal	Moves to output work area	Indicates normal return	Returns to caller
From Binary to Hexadecimal	Negative = Convert to Printable	Positive = convert a value Points to a Value PDE	'04'	X					X	X	X
From Binary to Instruction	Negative = Convert to Printable	Points to a Value PDE	'0C'		X				X	X	X
From Binary to FixedPoint Decimal	Negative = Convert to Printable	Points to a Value PDE	'10' or '14'			X		X	X	X	X
From Binary to Floating Point Decimal	Negative = Convert to Printable	Points to a Value PDE	'1C' or '18'					X	X	X	X
From Binary to Constants: A, Y, S, V	Negative = Convert to Printable	Points to a Value PDE	A: '20' Y: '24' S: '28' V: '2C'					X	X	X	X
From Binary to Packed Decimal	Negative = Convert to Printable	Points to a Value PDE	'30'				X	X	X	X	X

Figure 10. (Part 2 of 2) Value Conversion From Binary Form and To Binary Form

Translating Symbolic Addresses, Determining CSECT Names, and Resolving Entry Names

The symbol module (IKJEGSYM) translates an internal symbol, a module offset, or an entry name into an absolute main storage address or a CSECT name depending upon the request from the calling module. The symbol, offset, or entry name, which pertains to the program being tested. are specified in the QUALIFY or WHERE subcommand or in any subcommand that uses the convert module (IKJEGCVT). The Qualify, Where, or Convert module invokes IKJEGSYM to translate the input expression.

A module offset or an entry name can be obtained for both a load module fetched by Contents Supervision and an object module that has been loaded by the VS2 load module. An internal symbol, however, can be resolved only for a load module, because the OS/VS load module does not retain Assembler language symbol records. Internal symbols are therefore unavailable to IKJEGSYM, with the exception of a symbol the EQUATE subcommand places in TEST's symbol table residing in storage.

For each type of input, IKJEGSYM produces the output for the load module and the object module, subject to certain restrictions, as shown in Figure 11.

Symbol resolution is possible for a load module only if the TEST parameter was specified during both assembly and linkage editing. Otherwise, IKJEGSYM cannot resolve symbols created during the assembly of a load module. The user can circumvent this limitation by using the EQUATE subcommand during the TEST session to equate specified symbols with their relative offsets within the module. The equated symbols are placed in a symbol table in main storage. The symbol table remains for the remainder of the TEST session, and symbols in the table are available to IKJEGSYM.

Type of Input	Output	Availability of Output	Restrictions
Internal symbol	Absolute main storage address of symbol	Load module only, unless the symbol is in the symbol table built in memory by the EQUATE subcommand.	IGC0006A must have been invoked when the load module was fetched, and must have been able to determine and save the DDname by which the module was fetched, or have been able to identify the fetched data set as SYS1.LINKLIB.
Offset in module	CSECT in which offset lies	Load module Object module	See "Internal Symbol" The OS/VS load module must have had sufficient storage to build CESD entries
Entry name	Absolute main storage address for entry name	Load module Object module	See "Offset in Module"

Figure 11. Summary of the Input, Output, and Restrictions of Symbol Processing

If IKJEGSYM cannot resolve an input request by use of the symbol table, it tries to read SYM or CESD records from auxiliary storage (for a load module), or the CESD entries in main storage (for an object module).

If they are needed for a load module, the SYM records or the CESD records are read from the partitioned data set (PDS) from which the module was fetched. To read and use these records, IKJEGSYM needs the following information:

- The loaded address of the module.
- The DDname of the data set.
- The relative track and record address (TTR) of the PDS member.

IKJEGSYM obtains the foregoing information from an SVC information block (SIB) which is built by IGC0006A. IGC0006A is invoked by Contents Supervision when a load module has been fetched, unless the load module is a system module that is protected by the zero protection key IGC0006A. IGC0006A is invoked to build an SIB if either of the following took place:

- The fetched module was link-edited with the TEST attribute.
- The module was brought into main storage under a TCB that is a subtask of TSO.

This means that the module could have been fetched by the TEST load module (IKJEGLDR) or by the LOAD subcommand processor (IKJEGLDF). Contents Supervision tests the TCBTCP bit in the TCBABF field of the requestor's TCB; if the bit is 'on', the TCB represents a subtask of TEST.

If IGC0006A was not invoked for the module in question, IKJEGSYM cannot determine the DDname of the data set from which the load module was fetched. It cannot allocate the data set, and therefore it cannot read SYM or CESD records nor can it resolve the input symbol, offset, or entry name. In this case, IKJEGSYM assumes that the module is an object module, and it searches the entries in the contents directory and their associated extent lists, looking in main storage for a CESD entry for the module.

Refer to Diagram 2.25 and Figure 12 for more information about translating symbolic addresses, determining CSECT names, and obtaining entry point addresses.

Type of Request	Input Parameters in PDE	Search Argument	Processing	Output	Location of Output	Nearest Listing Symbol
1. Symbol: its address and attributes. **Note**: This request is possible only for a load module. The OS/VS Loader, when loading an object module, does not retain Assembler language symbol records. Therefore, the TEST Symbol module cannot obtain information about an object module's internal symbols. (The exceptions are internal symbols the user has placed in a in-core symbol table in main storage via the EQUATE subcommand.)	Symbol either alone or in combination with load name, and/or CSECT name. Thus, any of the following can be input: symbol or csect name symbol or loadname csectname symbol Loadname, if present, is pointed to by the load name pointer (PDELDNAM) in the PDE. CSECT name, if present, is pointed to by the entry name pointer (PDECTNAM). A flag (PDEFLG2) indicates which is present. Symbol is pointed to by the address string pointer (PDEADRPT). (A flag in the PDEFLG4 field indicates whether the symbol or an address string is pointed to.)	Symbol	a. If the symbol is not fully qualified, the program searches the symbol table built in main storage by the EQUATE subcommand for the symbol name. If the symbol is fully qualified, the program performs the processing in step c.	• Main storage address of symbol. • Symbol type. • Multiplicity factor. • Length of symbol.	Symbol information block (SIB) pointed to by the PDEUSER work in the PDE. (The PDRESYM flag [X '80'] in the PDE is set to indicate that the PDEUSER field Points to a SIB.)	SINCORES
			b. If the symbol is found, the program builds a symbol information block to contain the symbol information.			FOUNDSYM BUILDSIB
			c. If the symbol is not in the core symbol table, or if it is fully qualified, the program determines the search argument and prepares to read symbol records from the member of the PDS from wilch the module was loaded.			GETINFO
			Preparations consist of:			
			• Getting a DDname and a TTR from the SVC information block.			FOUND1
			• Opening a DCB and specifying the DDname.			OPENDCB
			• Issuing a type C find macro and specifying a TTR.			DOPOINT

Figure 12. (Part 1 of 5) Translating Symbolic Addresses, Determining CSECT Names, and Obtaining Entry Point Addresses

Type of Request	Input Parameters in PDE	Search Argument	Processing	Output	Location of Output	Nearest Listing Symbol
ntinued			d. The program reads symbol records from a member of the PDS, looking for the name of the CSECT in which the symbol is located. The program uses the CSECT name specified as the input parameter, or defaults to the currently qualified CSECT name stored in TCOMTAB.			FINDSYM
			e. When the Symbol module finds an entry in the symbol record that contains information about the specified symbol, it builds a symbol information block and places the symbol information into it. Symbol information includes: • Main storage address of symbol. • Symbol type. • Multiplicity factor. • Length of symbol.			FOUNDSYM
			f. The program sets a pointer to the symbol information block in the PDEUSER field of the PDE, and sets a flag (PDRESYM) to indicate that the field points to a symbol information block instead of to an entry point address.			BUILDSIB

Figure 12. (Part 2 of 5) Translating Symbolic Addresses, Determining CSECT Names, and Obtaining Entry Point Addresses

Type of Request	Input Parameters in PDE	Search Argument	Processing	Output	Location of Output	Nearest Listing Symbol
2. CSECT Name (requested only by IKJEGWHR) (If the module has not been link-edited, refer to number 4, "Object Module."	Load name (pointed to by the load name pointer field [PDELDNAM]).	Load name	a. The program searches the SVC Information blocks for the load name.	CSECT name	A work area, pointed to by the PDE entry name pointer (PDECTNAM)	GETIN
			b. When an SVC information block is found that contains the load name, the program gets the following information from the block: • Loaded address of the module. • DDname of the partitioned data set from which the module was fetched • Relative track and record address (TTR or the desired member of the partitioned data set.			FOUND1
			c. The program opens the DCB, specifying the DDname obtained from the SVC information block.			OPENDCB
			d. The program issues a FIND macro (specifying the TTR) to position the data set to the desired member.			DOPOINT
			e. The program reads the CESD records from the member.			FINDCESD
			f For each entry, the program computes the offset from the start of the member by adding the length of a CSECT to its relative address within the member. From the computation, the program finds the CSECT that contains the input offset.			SEARCH4
			g. The program places the found CSECT name in the work area belonging to IKJEGSYM and Points the PDE entry name pointer to the CSECT name in the work area.			FENTRY

Figure 12. (Part 3 of 5) Translating Symbolic Addresses, Determining CSECT Names, and Obtaining Entry Point Addresses

Type of Request	Input Parameters in PDE	Search Argument	Processing	Output	Location of Output	Nearest Listing Symbol
3. Load address associated with entry name.	Entry name: in entry name pointer field (PDECTNAM)	Load name initially, then entry name	a. Initial processing is the same as in steps a-e in "CSECT Name."	Main Storage address associated with entry name.	PDEUSER field of the input PDE	GETINFO FINDCESD
(If the module has not been link-edited, refer to number 4, "Object Module.")	Load name: pointed to by the load name pointer field, PDELDNAM If load name is missing, the symbol module uses as a default the currently qualified load name in the TCOMTAB (at TSTCURLD).		b. For each CESD record, the program examines entry name (the first field in the entry).			FINDNAME
			c. When the symbol module finds the specified entry name, it computes the entry point (loaded address) as follows:			INCORES
			• Gets the offset from the CESD entry.			
			• Adds the offset to the loaded address of the module, obtained from the SVC information block.			FOFFSET
			d. The program places the main storage address in the PDEUSER field of the input PDE.			FOFFSET
4. Object module Request for CSECT Name or Request for Loaded address associated with entry name **Note**: Internal symbols can't be resolved, because the OS Loader doesn't retain symbol records created during assembly. However, the symbol module can resolve any internal symbols that the user has placed in the symbol table in main storage, via the EQUATE subcommand.	Load name (pointed to by the load name pointer field, PDELDNAM) Offset from start of bad module (pointed to by address-string pointer field, PDEADRPT) Entry name (pointed to by the entry name pointer field, PDECTNAM) Module name (may or may not be present - pointed to by the load name pointer field PDELDNAM) **Note**: If the module name is missing, the symbol module uses as a default the currently qualified load name in TCOMTAB (at TSTCURLD).	Load name initially, then entry name	The following processing is the same for both types of request CSECT name of loaded address: a. The program searches SVC information blocks looking for a load name. This search will fail if the module is an object module. Since the module was not loaded by Contents Supervision, an SVC information block was not built. (If an SVC information block is found, the module must have been link-edited and is a load module.) The symbol module continues to search for the module as a load module.	CSECT name Main storage address associated with entry name	A work area, pointed to by the PDE entry name pointer (PDECTNAM) PDEUSER field of the PDE (see number 3)	GETINFO NOSVCINF GETINFO
			b. When the symbol module doesn't find the specified or defaulted load name in the SVC information blocks, it searches the contents directory entries (CDEs), starting with the contents directory entry for the currently qualified load module. (This CDE is queued from the request block pointed to by the PPRB field in TCOMTAB.) The symbol module compares the load name in the CDE with the specified or defaulted load name.			NOSVCINF

Figure 12. (Part 4 of 5) Translating Symbolic Addresses, Determining CSECT Names, and Obtaining Entry Point Addresses

Type of Request	Input Parameters in PDE	Search Argument	Processing	Output	Location of Output	Nearest Listing Symbol
4. Continued			c. If the symbol module locates the load name in the currently qualified CDE (step b) it examines the extent list associated with the CDE (see step d).			CDECHAIN
			If it does not find the load name in the currently qualified CDE, it searches for the load name in the CDE queue. (The queue origin is the TCBJPQ field in the TEST/TMP TCB.) If the load name is still not found, the symbol module issues an error message and returns control to the caller.			NOTFOUND
			d. If the symbol module finds a contents directory entry (CDE) that contains the desired load name, it determines that the CDE is for an object module. It then examines the associated extent list to determine if CESD information for the module exists in main storage.			XL
			Note: If the CDE is not found, the symbol module returns to the caller with an error code.)			
			e. The symbol module tests the second address entry in the extent list. If the second address entry is non-zero, the entry Points to a CESD record in main storage.			OSLOADER
			f. The symbol module then obtains the needed information from the CESD record in main storage in a manner similar to that used for a CESD record in auxiliary storage. (See load module request types number 2 and 3.)			OBJMOD
			g. Information is placed in either a work area or the PDEUSER field, as in load module request types 2 and 3.			

Figure 12. (Part 5 of 5) Translating Symbolic Addresses, Determining CSECT Names, and Obtaining Entry Point Addresses

Communicating with the Terminal and Creating the Print Data Set

The I/O module (IKJEGIO) handles I/O requests from any TEST module. Such I/O is an internal function of the TEST program; that is, the TEST program, rather than a terminal user, initiates the I/O.

The I/O module handles three types of I/O requests:

- GETLINE request, which is a request for input, either data or a subcommand. The data is obtained from the terminal; the subcommand is obtained from either the terminal or a subcommand list in main storage. (The list is built by the AT subcommand processor.)
- PUTGET request, which is a request to solicit new input from either the terminal or from a subcommand list in main storage. (The subcommand list, if it exists, was built by the AT subcommand processor.) The PUTGET request causes the I/O module to issue a message and to obtain the required input.
- PUTLINE request, which is a request to send either a line of data or a message to the terminal or to a print data set. When specifying the PRINT keyword, the LIST, LISTTCB, LISTDEB, LISTDCB, LISTPSW, and LISTMAP subcommands request output to a print data set.

Refer to Diagram 2.26 for more information about communicating with the terminal and creating the print data set.

Exceptional Operations

Exceptional operations in TEST consist of handling an attention request from the terminal and handling an abnormal termination in the TEST command processor. Figure 13 lists the modules involved in these operations and the diagrams that describe the operations.

Operation	Symbolic Name	Diagram Number
Handling an attention interruption from the terminal	IKJEGATN	2.30
Handling an abnormal termination in the TEST program	IKJEGSTA	2.31

Figure 13. Exceptional TEST Operations

Handling an Abnormal Condition in TEST

There are two types of abnormal situations the TEST command processor can encounter: an error in the problem program and an error in the TEST command processor itself.

An ABEND in the problem program is handled by the STAI Exit routine, IKJEFT04, of the TMP. The STAI Exit routine places the user in TEST mode. (For a detailed description of the TMP STAI Exit routine, see the OS/VS2 TSO Terminal Monitor Program and Service Routines Logic.)

An error in the TEST command processor itself is circumvented by the STAE Exit routine (IKJEGSTA) of TEST. This routine issues diagnostic messages and causes control to be passed first to an appropriate termination routine and then to IKJEGMNL to continue the test session.

Because of the similarity between STAI and STAE, remember that the STAI Exit routine handles a problem in the subtask (the problem program), where as the STAE Exit handles a problem in the parent task (the TEST command processor).

The STAE Exit routine's main functions consist of:

- Issuing a specific diagnostic message to the terminal for an abnormal condition that was predictable.
- Issuing a general purpose diagnostic message to the terminal for an abnormal condition that was not predictable.
- Requesting a 'retry' entry to a routine that frees a 104-byte STAE work area and resets certain switches.

There is a separate retry routine for each module in TEST. All such routines, except the one in IKJEGMNL, return control to IKJEGMNL so it can ask for a new subcommand and continue the TEST session. (IKJEGMNL's retry routine effectively executes an END subcommand by returning to the TMP.)

The processing of the STAE Exit routine in TEST is part of a sequence that begins in the first load module of the Supervisor's error routine (IGC0001C), continues through the Supervisor's ABEND/STAE Interface routines (IGC0B01C - IGC0E01C), and ends with execution of one of a number of retry routines within TEST. The address of the retry routine to be used for an error is placed in location TSTRETR in TCOMTAB by each TEST module when it gets control. IKJEGMNL places the address of the STAE Exit routine in the TSTSTAE field of TCOMTAB.

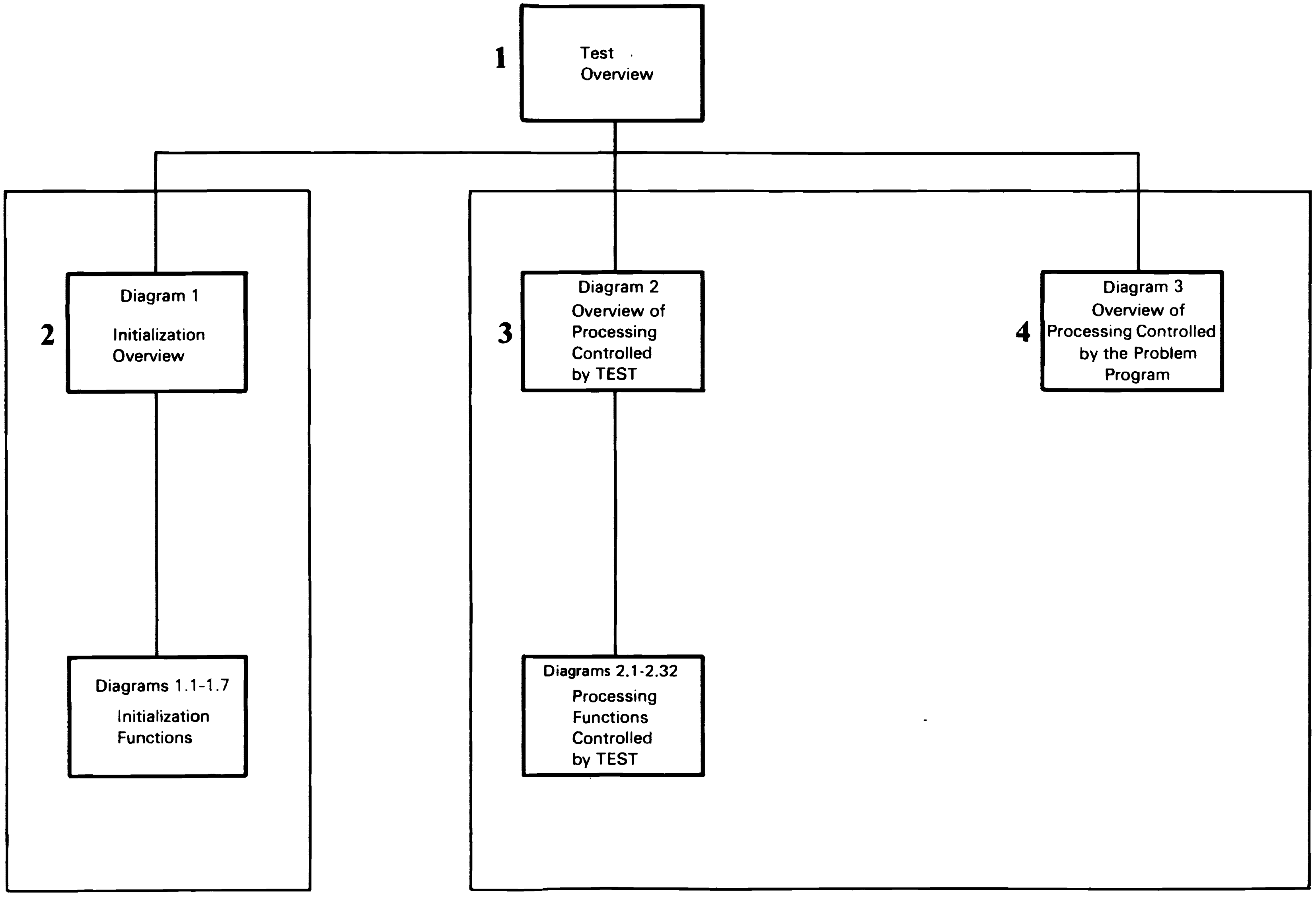
Refer to Diagram 2.31 for more information on handling an abnormal condition in TEST. (For detailed flowcharts of the ABEND/STAE Interface routines, see *OS/VS2 Supervisor Logic* .)

Processing Controlled by the Problem Program

Diagram 3 illustrates processing controlled by the problem program. IKJEGMNL starts the problem program after a subcommand issues a return code of 4. IKJEGMNL then issues a POST macro instruction to return control to IGC0009G, which exits. This forces the transfer of control to the problem program, which executes until a terminating condition occurs. Terminating conditions are:

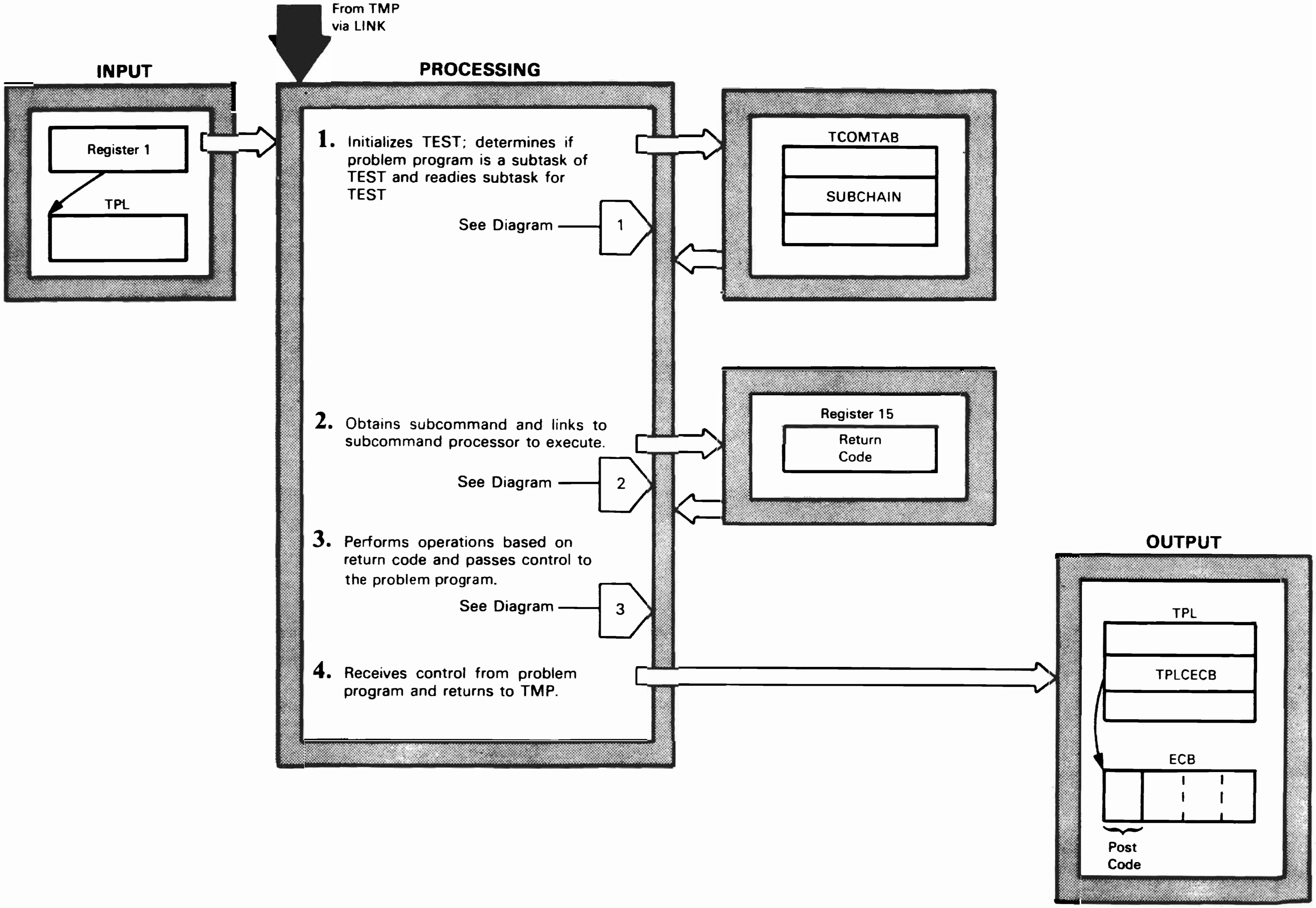
- The problem program issues an ATTACH, LINK, LOAD, or XCTL macro instruction to another module.
- The problem program reaches a breakpoint.
- The problem program receives an attention interruption.
- The problem program completes normally.
- The problem program begins an abnormal termination, and unprocessed statements remain.
- The problem program terminates abnormally, and no unprocessed statements remain. IKJEGCVT performs the processing and receives control from the validity check exit

TEST VISUAL TABLE OF CONTENTS


1
Test Overview
2
Diagram 1
Initialization Overview
Diagrams 1.1-1.7
Initialization Functions
3
Diagram 2
Overview of Processing Controlled by TEST
Diagrams 2.1-2.32
Processing Functions Controlled by TEST
4
Diagram 3
Overview of Processing Controlled by the Problem Program

Description	Module	Label
1. The TEST Overview provides a simple, step-by-step reference to the overall operation of the TEST command processor.		
2. Diagram 1, the Initialization Overview, illustrates how TEST prepares to receive the problem program. Diagrams 1.1 and 1.2 illustrate how a problem program is loaded. Diagram 1.3 shows how data set information about the problem program is saved. Diagrams 1.4 - 1.8 explain breakpoints.		
3. Diagram 2, the Overview of Processing Controlled by TEST, traces subcommand activity before control is passed to the problem program. Diagram 2.1 explains general subcommand processing; Diagrams 2.2 - 2.23 depict each subcommand separately. Diagrams 2.24 - 2.32 explain the routines TEST uses to perform internal functions.		
4. Diagram 3, the Overview of Processing Controlled by the Problem Program, depicts events occurring during execution of the problem program.		

TEST OVERVIEW



1. Initializes TEST; determines if problem program is a subtask of TEST and readies subtask for TEST

   See Diagram 1

2. Obtains subcommand and links to subcommand processor to execute.

   See Diagram 2

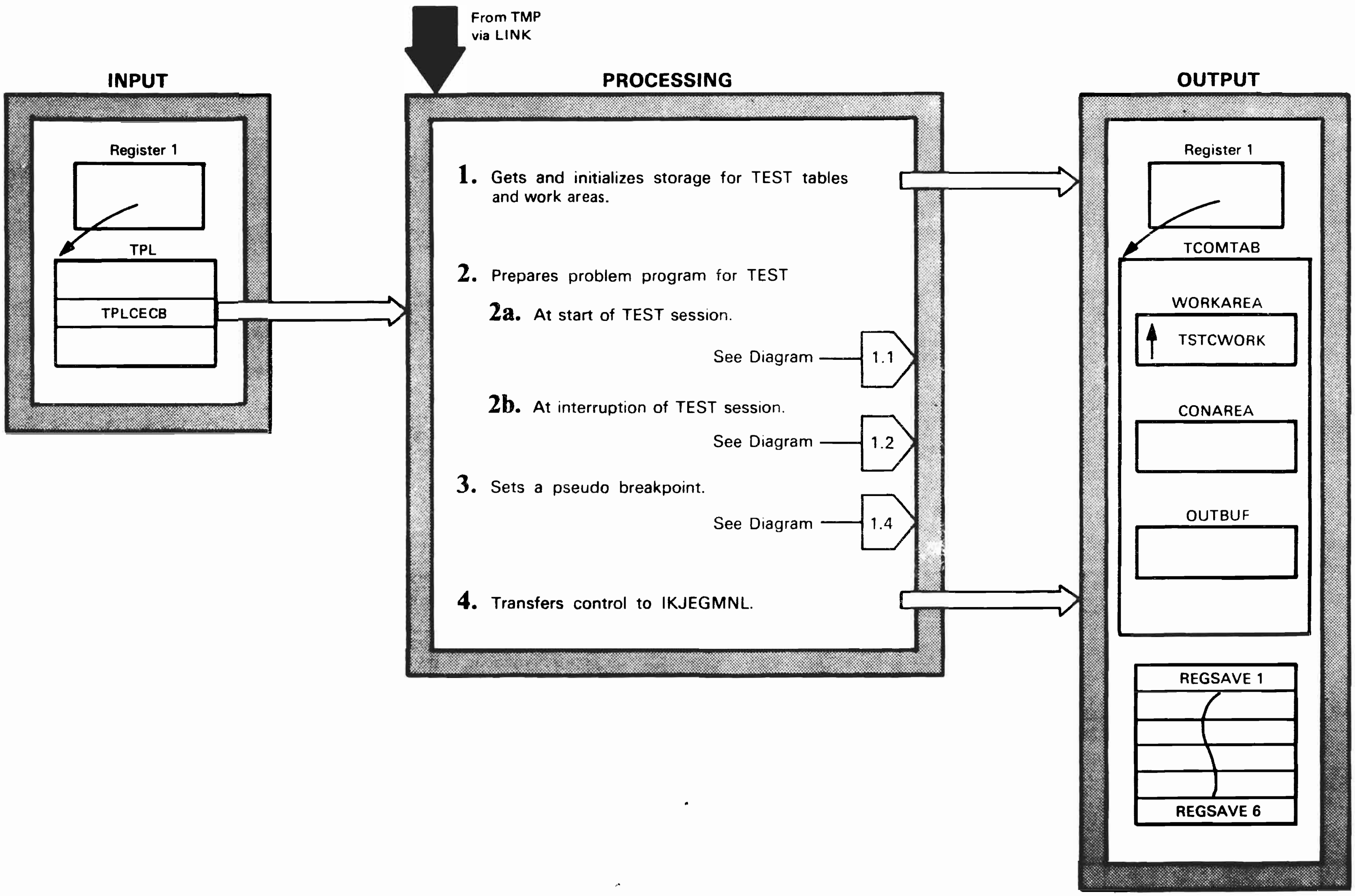
3. Performs operations based on return code and passes control to the problem program.

   See Diagram 3

4. Receives control from problem program and returns to TMP.

Description	Module	Label
1. IKJEGINT obtains space for TCOMTAB for the TEST modules' work area and for a register save area. IKJEGINT sets up a STAE Exit routine and obtains data for the TCOMTAB from the TPL and from addresses returned in storage allocation.	IKJEGINT	INT01010
If the problem program is already a subtask of TEST, it is readied to test, and control is passed to IKJEGMNL. If the problem program is not a subtask of TEST, it is fetched, readied to test, and control is passed to IKJEGMNL. IKJEGMNL finishes the TCOMTAB initialization and issues the STAX and STATUS STOP macro instructions.	IKJEGMNL	IKJEGMNL
2. IKJEGIO places the user's subcommand in the SUBCHAIN field of TCOMTAB, and IKJEGMNL links to the subcommand processor. If necessary, the subcommand processor writes a message to the terminal or to a data set using IKJEGIO. IKJPARS checks the validity of the subcommand, and, if valid, the subcommand is executed. Upon completion, the subcommand processor passes IKJEGMNL a return code.	IKJEGMNL IKJEGIO	MNLXX170 A00520
3. IKJEGMNL obtains additional subcommands from the SUBCHAIN field until a GO, CALL, RUN, or END subcommand is processed. The GO, CALL, RUN, and END subcommands perform operations different from the remaining TEST subcommands to pass control to the problem program.	IKJEGMNL	SCREQ1
4. When TEST returns control to the TMP, the TPLCECB field of the TPL points to the address of an ECB which contains the post code.	IKJEGMNL	RESTART4

DIAGRAM 1. Initialization Overview



1. Gets and initializes storage for TEST tables and work areas.

2. Prepares problem program for TEST

2a. At start of TEST session.

See Diagram 1.1

2b. At interruption of TEST session.

See Diagram 1.2

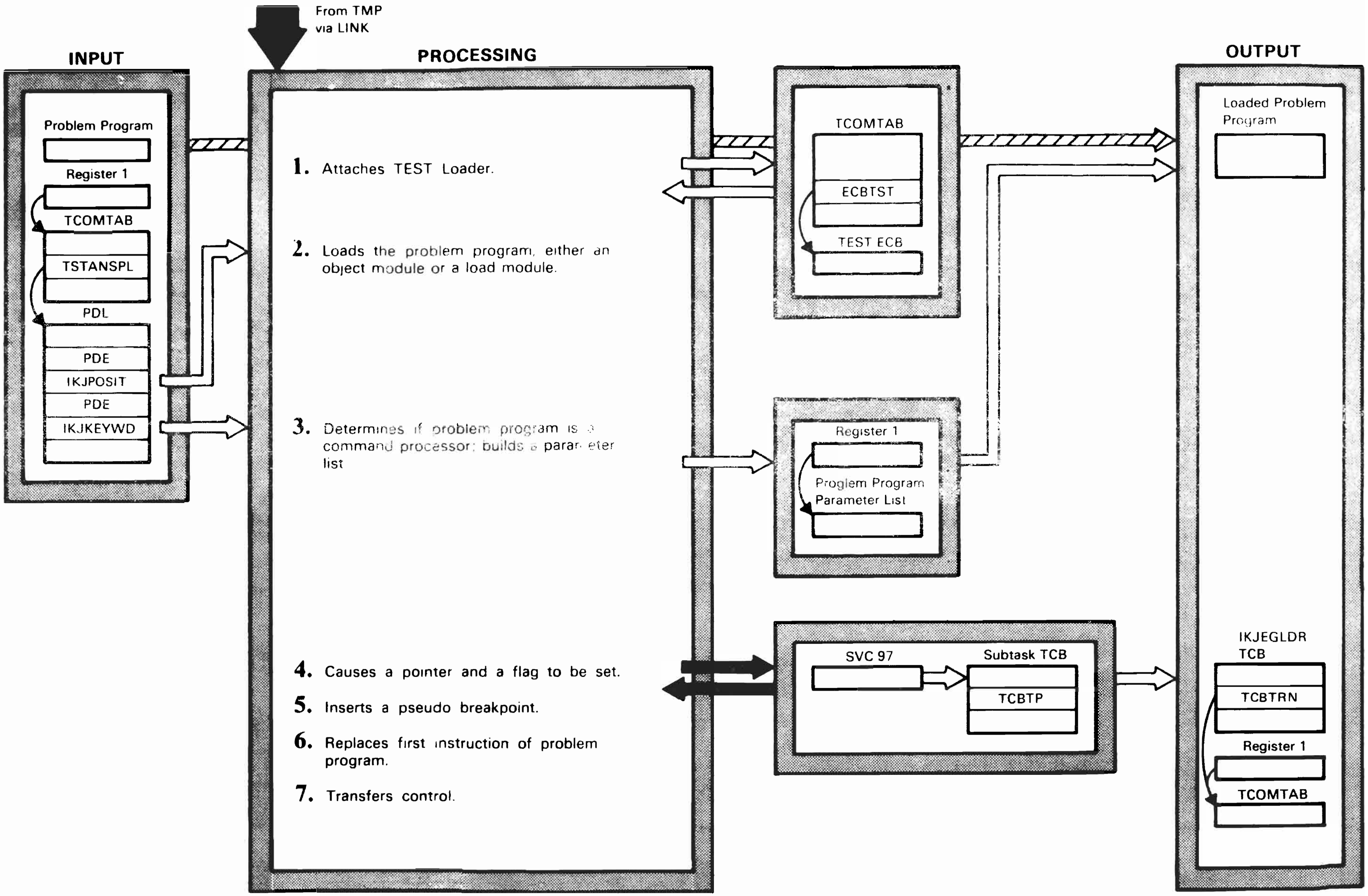
3. Sets a pseudo breakpoint.

See Diagram 1.4

4. Transfers control to IKJEGMNL.

Description	Module	Label
1. IKJEGINT issues a GETMAIN macro instruction to allocate storage for TCOMTAB, for TSTCWORK (a work area used by various TEST modules), and for register save areas. IKJEGINT invokes the SVC 97 routine to store the pointer to TCOMTAB in the TCB of TEST.	IKJEGINT	INT01010
2. IKJEGINT loads the problem program either at the beginning of a TEST session or after program execution is interrupted.	IKJEGINT	DAIR
2a. If the TEST session is just beginning, IKJEGINT loads the problem program. IKJEGLDR is used if the user specifies LOAD as his keyword. The OS/VS Loader is used if OBJ is the specified keyword.	IKJEGLDR OBJ LOAD	OSLOAD PARMSET
2b. If the problem program has been executing, IKJEGINT assumes control if the program is terminating abnormally or if the terminal user interrupted its processing with an attention interruption. In these cases, the problem program's TCB address is in the TPLCTCB field of the TPL.	IKJEGINT	INTCONT1
3. IKJEGINT sets a pseudo breakpoint whenever the following events occur to cause a task switch from the problem program's task to the TEST task:	IKJEGINT	INT06010
• When IKJEGINT receives control after an abnormal termination.	IKJEGINT	INT06010
• When IKJEGINT receives control after the program has been interrupted by an attention interruption.	IKJEGINT	INT06010
• When IKJEGINT is posted after attaching the TEST loader.	IKJEGINT IGC0006A	WAIT
4. IKJEGINT issues an XCTL macro instruction to fetch and transfer control to IKJEGMNL.	IKJEGINT	MODEGO

DIAGRAM 1.1. Preparing a Problem Program at the Start of a TEST Session



1. Attaches TEST Loader.

2. Loads the problem program, either an object module or a load module.

3. Determines if problem program is a command processor; builds a parameter list

4. Causes a pointer and a flag to be set.

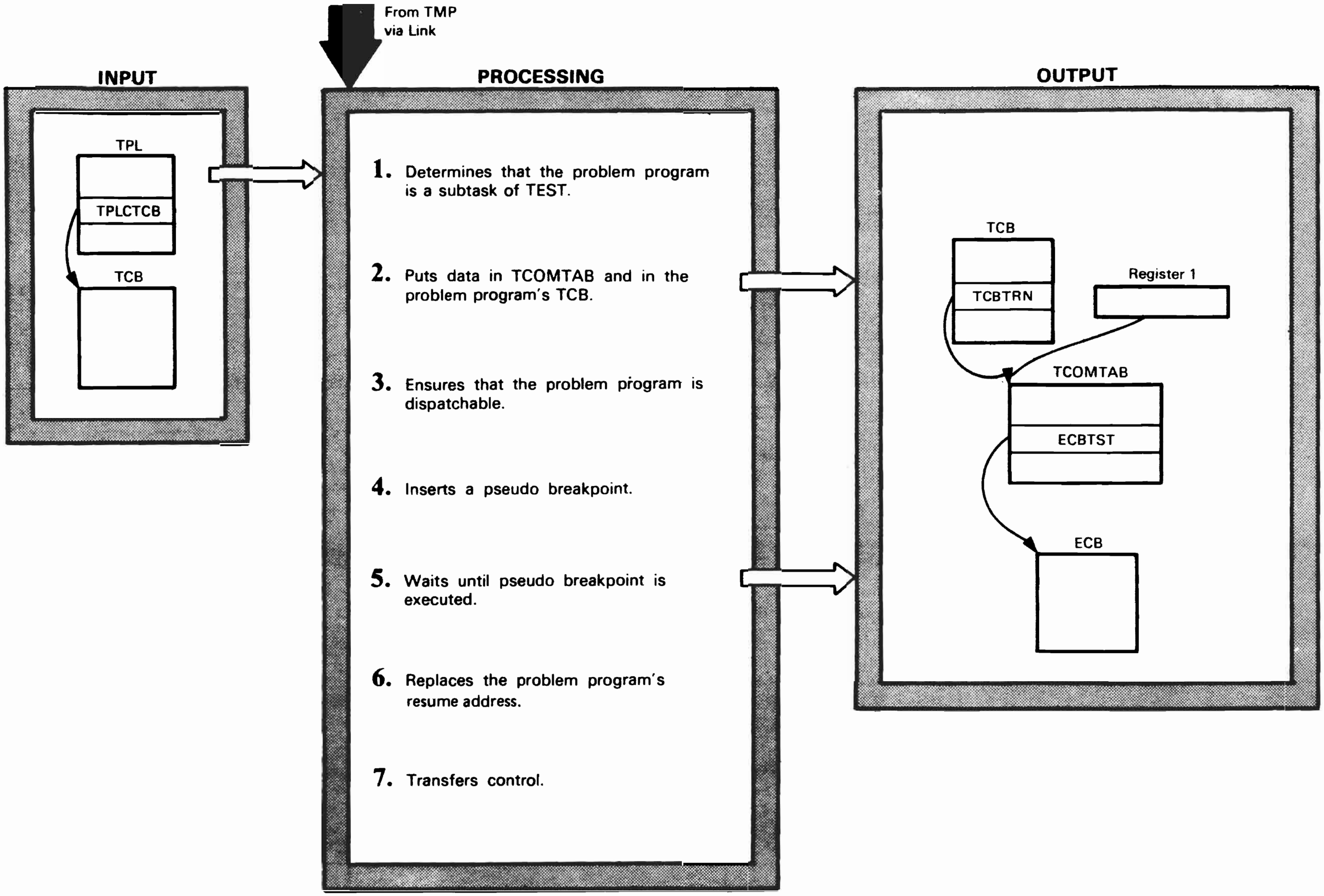
5. Inserts a pseudo breakpoint.

6. Replaces first instruction of problem program.

7. Transfers control.

Description	Module	Label
1. IKJEGINT attaches IKJEGLDR and waits until a POST macro instruction sets TEST's ECB, which is pointed to by the ECBTST field in TCOMTAB.	IKJEGINT	ATTACH
2. IKJEGLDR checks the IKJPOSIT PDE associated with the data set name used in the TEST command for the LOAD or the OBJECT keyword. If the problem program is a load module, Contents Supervision invokes the SVC 61 routine immediately after it fetches the load module. The SVC 61 routine issues a POST macro instruction to IKJEGINT, which inserts a pseudo breakpoint at the load module's entry point. IKJEGINT then issues a POST macro instruction to both the SVC 61 routine and contents supervision to complete to XCTL processing.	IKJEGLDR	PARMSET
If the problem program is an object module, IKJEGLDR issues a LOAD macro instruction to load the OS/VS loader and then calls the OS loader. When the OS loader successfully fetches the problem program, IKJEGLDR deletes the OS/VS loader.	IKJEGLDR	OSLOAD
3. IKJEGLDR examines the IKJKEYWD PDE associated with the command processor keyword used in the TEST command. If the keyword is CP, IKJEGLDR invokes IKJEGIO to request from the terminal and input command for the problem program. IKJEGLDR builds a CPPL (command processor parameter list) and a parameter list for IKJSCAN, the Command Scan routine. If the keyword is NOCP, IKJEGLDR checks for an input parameter stirng. If one is present, IKJEGLDR places it in the parameter list. If no parameter string is specified, IKJEGLDR builds an empty parameter list.	IKJEGLDR	CMDPRM
4. IKJEGLDR invokes the SVC 97 routine, which sets a pointer to TCOMTAB in the TCBTRN field of the TEST loader's TCB. The SVC 97 routine also sets the TCBTP flag in the subtask TCB to indicate that the TCB is a subtask of TEST.	IKJEGLDR	SETTRN
5. IKJEGLDR inserts a pseudo breakpoint at the entry point of the object module to cause a task switch to IKJEGINT and to create an SVRB in which program registers are saved. The SVC 97 routine determines that entrance was due to a pseudo breakpoint in the problem program, issues a POST macro instruction to set TEST's ECB and make IKJEGINT dispatchable, and waits for IKJEGMNL to issue a POST macro instruction to set the subtask ECB (for example, by an I/O request). When interrupted, control is passed to the Dispatcher.	IKJEGLDR IGC0009G	SETPSEUD QPSEUDO
6. IKJEGINT replaces the problem program's first instruction from the TSTGO field in TCOMTAB.	IKJEGINT	SPECIAL
7. The problem program is now a subtask of TEST, and control is passed to IKJEGMNL via an XCTL macro instruction.	IKJEGINT	MODEGO

DIAGRAM 1.2. Preparing a Problem Program During a TEST Session



1. Determines that the problem program is a subtask of TEST.

2. Puts data in TCOMTAB and in the problem program's TCB.

3. Ensures that the problem program is dispatchable.

4. Inserts a pseudo breakpoint.

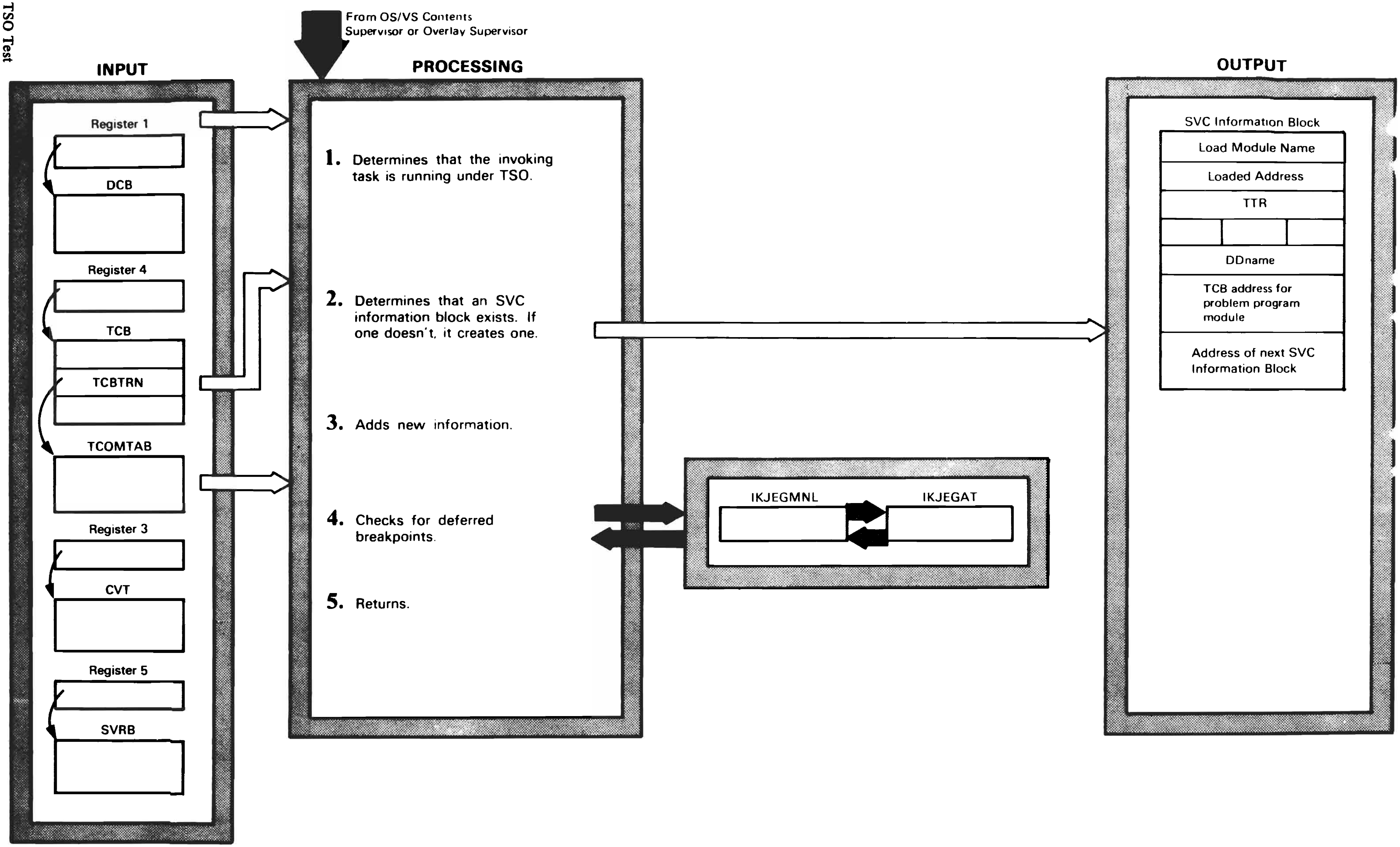
5. Waits until pseudo breakpoint is executed.

6. Replaces the problem program's resume address.

7. Transfers control.

Description	Module	Label
1. IKJEGINT examines the TPLCTCB field for a TCB address; if one is found, the problem program is already a subtask of TEST.	IKJEGINT	INTCONT1
2. IKJEGINT moves the queue origins of the SVC information chain from the problem program's TCB to TCOMTAB to make the chain readily accessible to IKJEGSYM, the TEST symbol module. IKJEGINT also flags the TCBTRNB field in each TCB of the problem program to point to TCOMTAB. IKJEGINT then invokes the SVC 97 routine to indicate in each problem program TCB that its tasks are subtasks of TEST.	IKJEGINT	INT05010
3. IKJEGINT ensures that the problem program is dispatchable by issuing a STATUS START macro instruction for the problem program TCB. A pseudo breakpoint can then be inserted.	IKJEGINT	STAIPOST
• If TEST is invoked when the problem program is in a permanent wait condition, IKJEGINT saves the problem program's RB wait count which is found in the most recently used PRB or IRB.	IKJEGINT	SETPSEUD
• If TEST is invoked via the TMP Attention Exit routine when it handled an attention interruption, IKJEGINT issues a STATUS macro instruction with a START operand to zero the TCBSTPCT field of the problem program's TCB. This field counts the number of STATUS STOP macro instructions issued against the TCB, and the problem program is dispatchable only when this field is zero.	IKJEGINT	SETPSEUD
• If TEST is invoked from the TMP when the problem program begins to terminate abnormally, IKJEGINT issues a POST macro instruction to set the ECB pointed to by the TPLNECB field in the TPL.	IKJEGINT	STAIPOST
4. IKJEGINT inserts a pseudo breakpoint:		
• When it receives control after an abnormal termination.	IKJEGINT	WAIT
• When it receives control after the problem program has been interrupted by an attention interruption.	IKJEGINT	WAIT
• When the SVC 61 routine issues a POST macro instruction after attaching the TEST loader.	IKJEGINT	WAIT
5. IKJEGINT waits until a POST macro instruction sets the ECB pointed to by the ECBTST field of TCOMTAB. The SVC 97 routine examines the breakpoint and issues the POST and WAIT macro instructions to restart IKJEGINT.	IGC0009G	TSTPOST
6. After the POST macro instruction sets the TEST ECB, IKJEGINT is dispatchable and replaces the problem program's resume address from the TSTGO field of TCOMTAB.	IKJEGINT	SETTSIP
7. IKJEGINT issues an XCTL macro instruction to transfer control to IKJEGMNL.	IKJEGINT	MODEGO

DIAGRAM 1.3. Saving Data Set Information about the Problem Program



1. Determines that the invoking task is running under TSO.

2. Determines that an SVC information block exists. If one doesn't, it creates one.

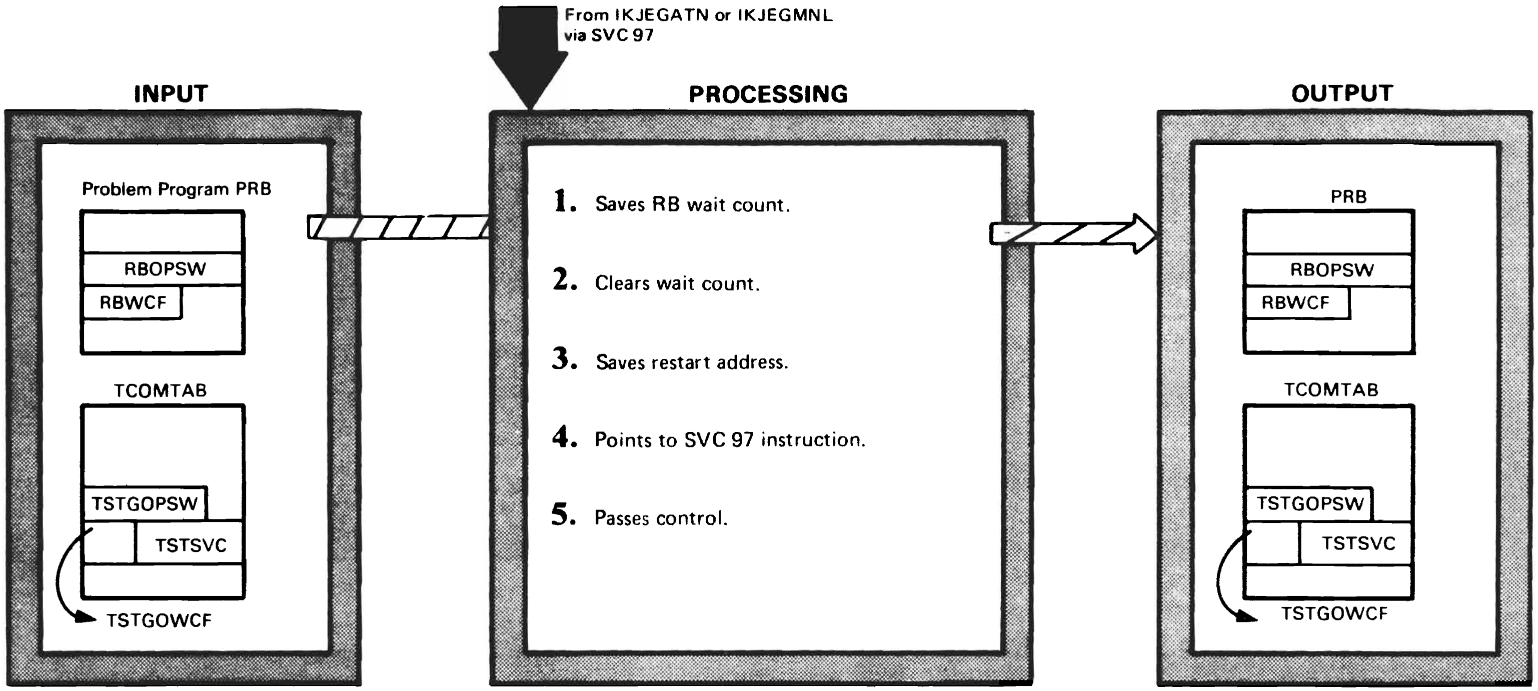
3. Adds new information.

4. Checks for deferred breakpoints.

5. Returns.

Description	Module	Label
The SVC 61 routine, IGC0006A, stores in an SVC information block information describing the data set from which the module was fetched. Deferred user breakpoints are activated. Entry is from either Contents Supervisor or Overlay Supervisor.	IGC0006A	GETSVCBK
1. IGC0006A checks the TSTTRN field of TCOMTAB. If TEST is running, this field points to the first SVC information block. If TEST is not running, the TCBTRN field of the current TCB points to the SVC information block.	IGC0006A	FORTEST
2. IGC0006A determines if one or more SVC information blocks exist. If IGC0006A was previously entered during this TEST session, an SVC information block exists. If at least one block exists, IGC0006A determines whether or not it contains the load module name of the module being fetched. If no block contains a load module name that agrees with the one in the BLDL entry, a new block is built.	IGC0006A	FORTEST
3. Information pertaining to the data set is entered into the block. This information includes: • Load module name. • Relative Track and record address. • Module attributes. • Loaded address. • DDname of data set. • TCB address of invoking task.	IGC0006A	FILLBLK
The block is queued from the previous one or from the TCOMTAB, if no previous block exists.	IGC0006A	CHAINBLK
4. Control is passed to IKJEGMNL, which activates any deferred user breakpoints by using IKJEGAT. See Diagram 1.7 on activating deferred user breakpoints. Control then returns to IGC0006A.	IGC0006A	POST WAIT
5. Return is to the caller.	IGC0006A	TOISSUER

DIAGRAM 1.4. Setting a Pseudo Breakpoint



1. Saves RB wait count.

2. Clears wait count.

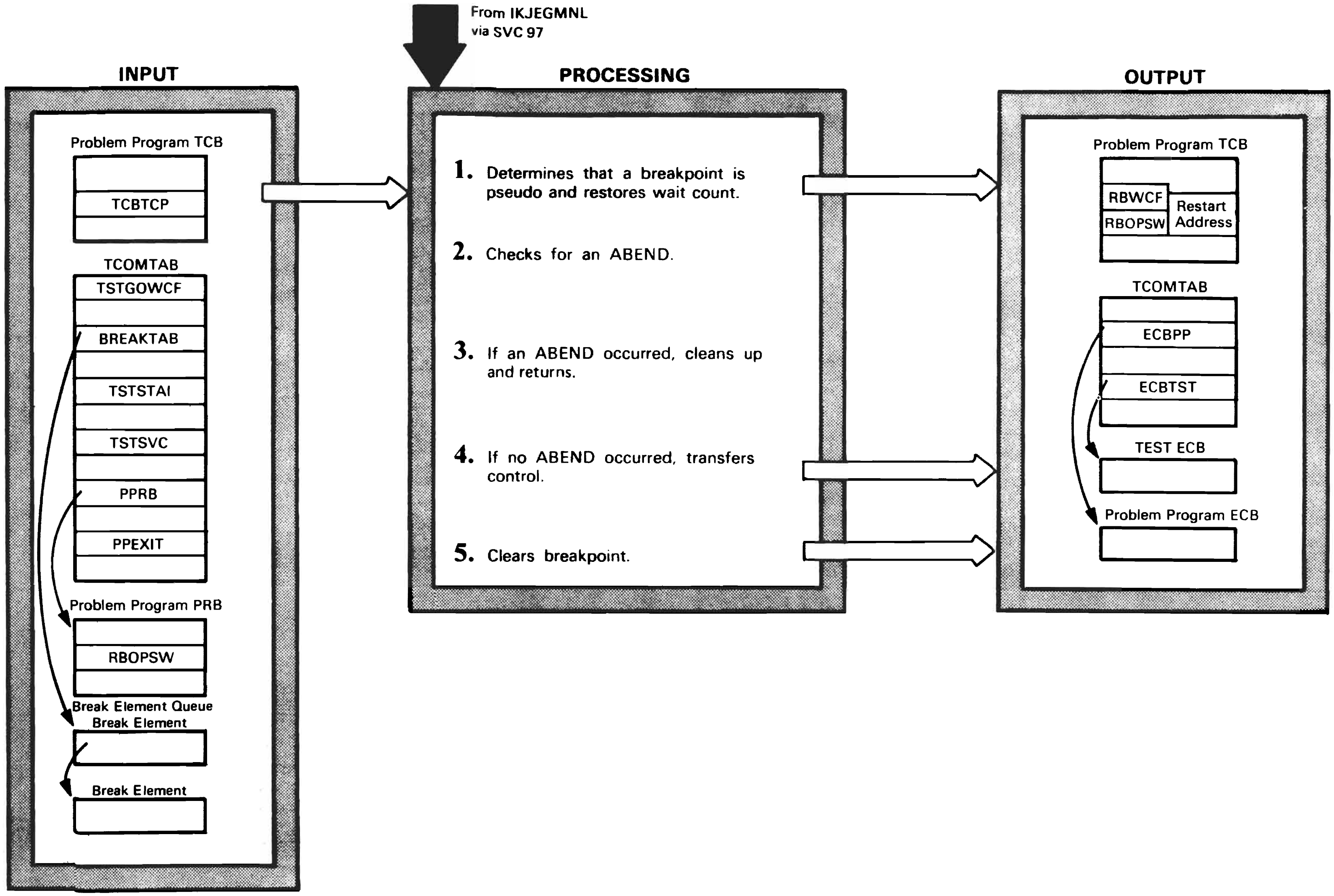
3. Saves restart address.

4. Points to SVC 97 instruction.

5. Passes control.

Description	Module	Label
1. The problem program's request block wait count (RBWCF) is copied into the TSTGOWCF field in TCOMTAB.	IKJEGINT	SETPSEUD
2. RBWCF is then set to zero.	IGC0009G	S9G20010
3. The problem program's restart address (RBOPSW +4) is copied into the TSTGOPSW field in TCOMTAB.	IKJEGINT	SETPSEUD
4. The problem program's restart address is then pointed to an SVC instruction in the TSTSVC field in TCOMTAB, the PPEXIT field in TCOMTAB, or the BRKINST +6 field in BRKELEM.	IKJEGINT	SETPSEUD
5. IGC0009G receives control to execute the pseudo breakpoint.	IGC0009G	PSEUDO

DIAGRAM 1.5. Executing a Pseudo Breakpoint



1. Determines that a breakpoint is pseudo and restores wait count.

2. Checks for an ABEND.

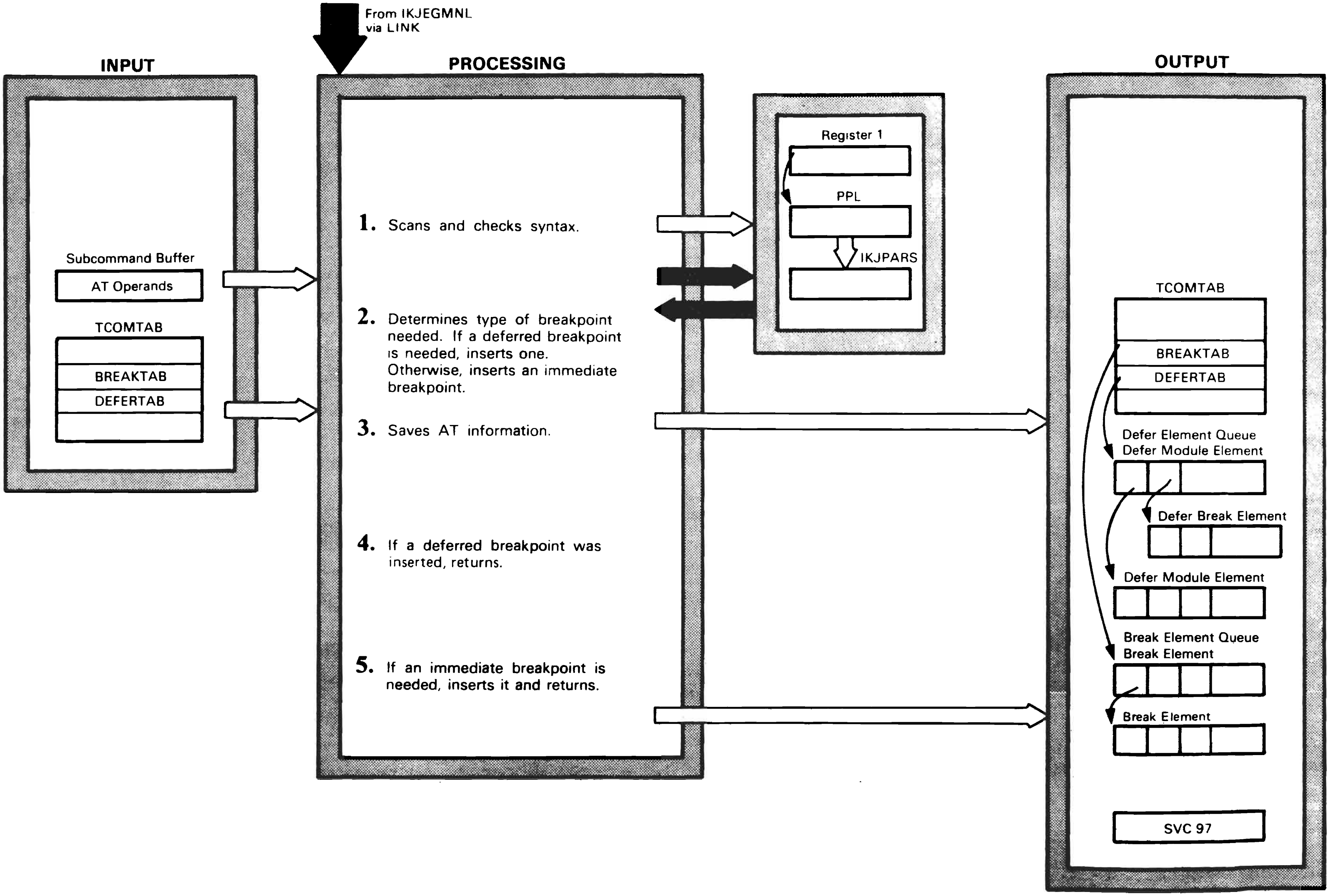
3. If an ABEND occurred, cleans up and returns.

4. If no ABEND occurred, transfers control.

5. Clears breakpoint.

Description	Module	Label
1. IGC0009G examines the problem program's TCB and the break element queues to determine that the breakpoint is a pseudo breakpoint. The TSTSVC or PPEXIT field in TCOMTAB or the BRKINST +6 field in BRKELEM are the fields inspected. The break element queue is pointed to by the BREAKTAB field in TCOMTAB. The request block wait count (RBWCF) is restored to the problem program's PRB from the TSTGOWCF field in TCOMTAB.	IGC0009G	QPSEUDO
2. IGC0009G checks TCOMTAB to determine if the TSTSTAI flag is set, which indicates an ABEND in the problem program.	IGC0009G	PSEUDO
3. If an ABEND occurred, the STAE work area is freed, the problem program's registers are restored, and control is transferred to IKJEGMNL or IKJEGINT. Breakpoint processing ends.	IGC0009G	PSEUDO1
4. If no ABEND occurred, control is transferred to IKJEGMNL or IKJEGINT. The ECBTST field in TCOMTAB points to the ECB where posting occurs.		
5. When IKJEGMNL or IKJEGINT finish processing, the problem program's restart address is restored and normal processing continues.		

DIAGRAM 1.6. Setting a User Breakpoint



1. Scans and checks syntax.

2. Determines type of breakpoint needed. If a deferred breakpoint is needed, inserts one. Otherwise, inserts an immediate breakpoint.

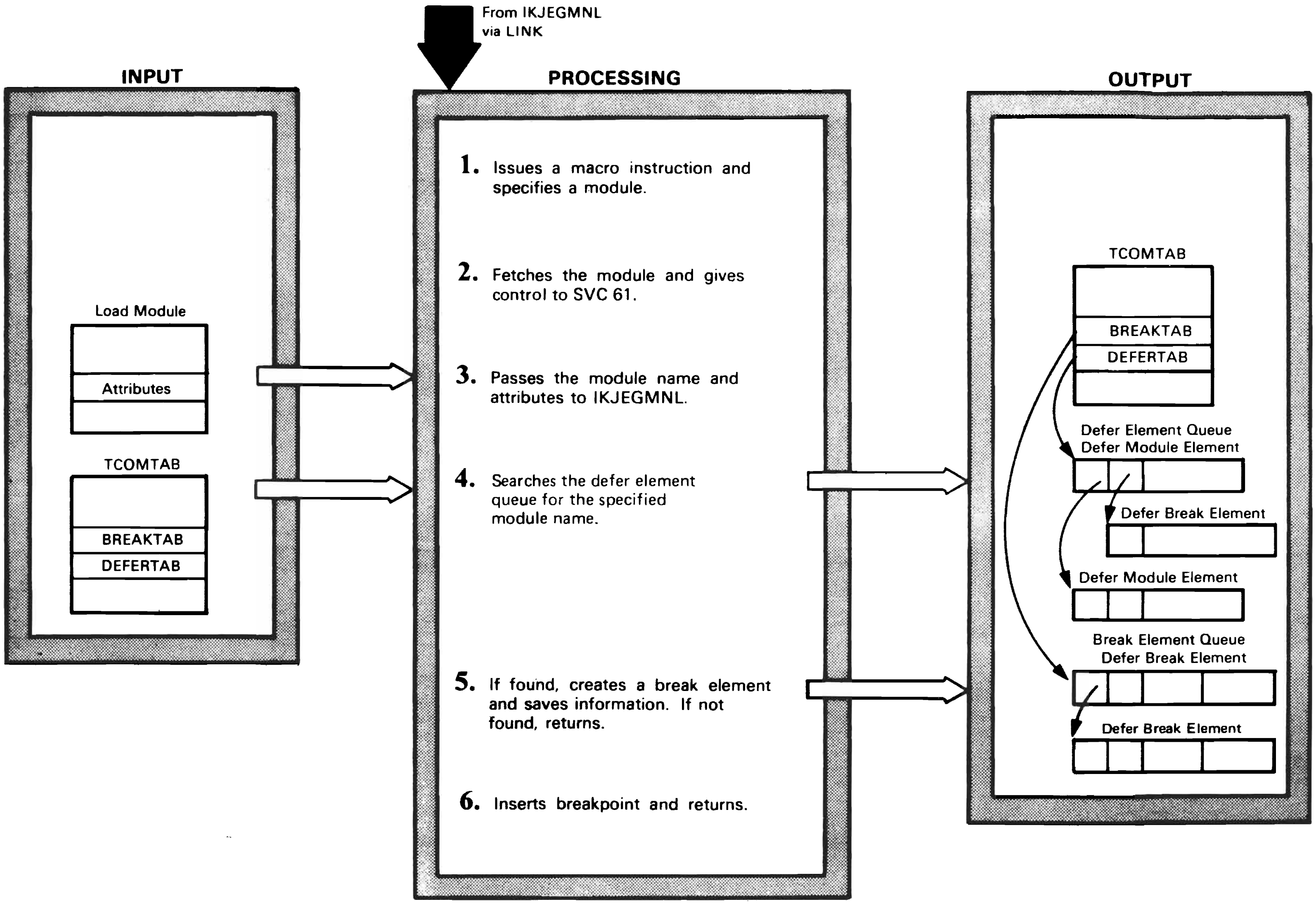
3. Saves AT information.

4. If a deferred breakpoint was inserted, returns.

5. If an immediate breakpoint is needed, inserts it and returns.

Description	Module	Label
1. IKJEGAT receives control from IKJEGMNL when an AT subcommand is entered. Control is passed to IKJPARS to check the syntax of the operands. (For IKJPARS processing see the Terminal Monitor Program and Service Routines PLM.) Upon completion, control returns to IKJEGAT.	IKJEGAT	CONTINUE
2. If the DEFER keyword was specified, the problem program may not yet be in the user's region, and a deferred breakpoint is inserted. If the DEFER keyword was not specified, immediate breakpoints are inserted in the problem program.	IKJEGAT (DEFERED) IKJEGAT (ACTIVE)	TRANSFER WASITCVD
3. To insert a breakpoint, the program first saves certain information about the subcommand. If a deferred breakpoint was inserted, information describing the breakpoint is saved in a defer element queue for use when the user's specified module is fetched into main storage. For the format and content of the defer element queue, see "Section 4: Data Areas".	IKJEGATD	NAMESOK
If an immediate breakpoint was inserted, information about it is inserted in the break element queue built by IKJEGAT. For the format and content of the break element queue, see "Section 4: Data Areas."	IKJEGAT	BLDBREAK
4. If a deferred breakpoint was inserted, IKJEGAT returns control to IKJEGMNL.	IKJEGAT	RETMAIN
5. If an immediate breakpoint was inserted, IKJEGAT inserts an SVC 97 instruction at each specified location in the problem program. The problem program instructions overlaid by the SVC 97 instructions are saved in the break element queue. IKJEGAT returns control to IKJEGMNL.	IKJEGAT	OPCODEOK

DIAGRAM 1.7. Activating a Deferred User Breakpoint



1. Issues a macro instruction and specifies a module.

2. Fetches the module and gives control to SVC 61.

3. Passes the module name and attributes to IKJEGMNL.

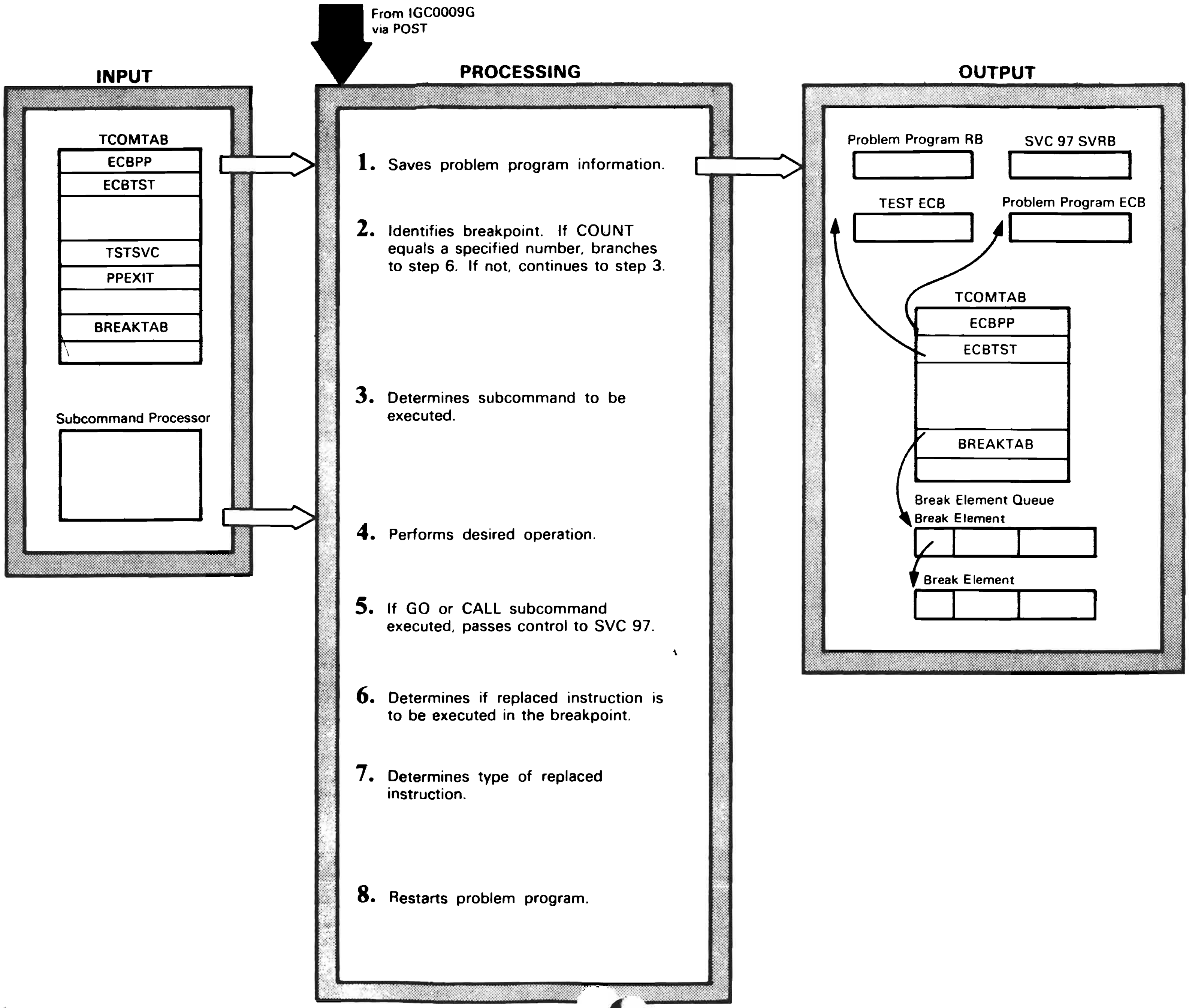
4. Searches the defer element queue for the specified module name.

5. If found, creates a break element and saves information. If not found, returns.

6. Inserts breakpoint and returns.

Description	Module	Label
1. The problem program issues a LINK, LOAD, ATTACH, or XCTL macro instruction and specifies a problem program module.	Problem Program	
2. Contents Supervision fetches the requested module and issues an SVC 61 instruction to transfer control to IGC0006A.	IGC00061	IGC00061
3. IGC0006A passes the load module name and attributes to IKJEGMNL.	IGC00061	POST WAIT
4. The DEFERTAB field in TCOMTAB is checked to determine if any deferred breakpoints were specified. If any were, control passes to IKJEGAT via a LINK macro instruction. IKJEGAT searches the defer element queue for the name of the fetched module. If no deferred breakpoints were specified, control passes back to IGC0006A.	IKJEGAT	HANDLED
5. If a defer break element containing the module name is found, breakpoint information is transferred from the defer element queue to the break element queue. If not, control passes to IGC0006A.	IKJEGAT	CHKMAT
6. An immediate breakpoint is inserted in the fetched module and control is passed to IKJEGMNL.	IKJEGAT	OPCODEOK

DIAGRAM 1.8. Executing a User Breakpoint



1. Saves problem program information.

2. Identifies breakpoint. If COUNT equals a specified number, branches to step 6. If not, continues to step 3.

3. Determines subcommand to be executed.

4. Performs desired operation.

5. If GO or CALL subcommand executed, passes control to SVC 97.

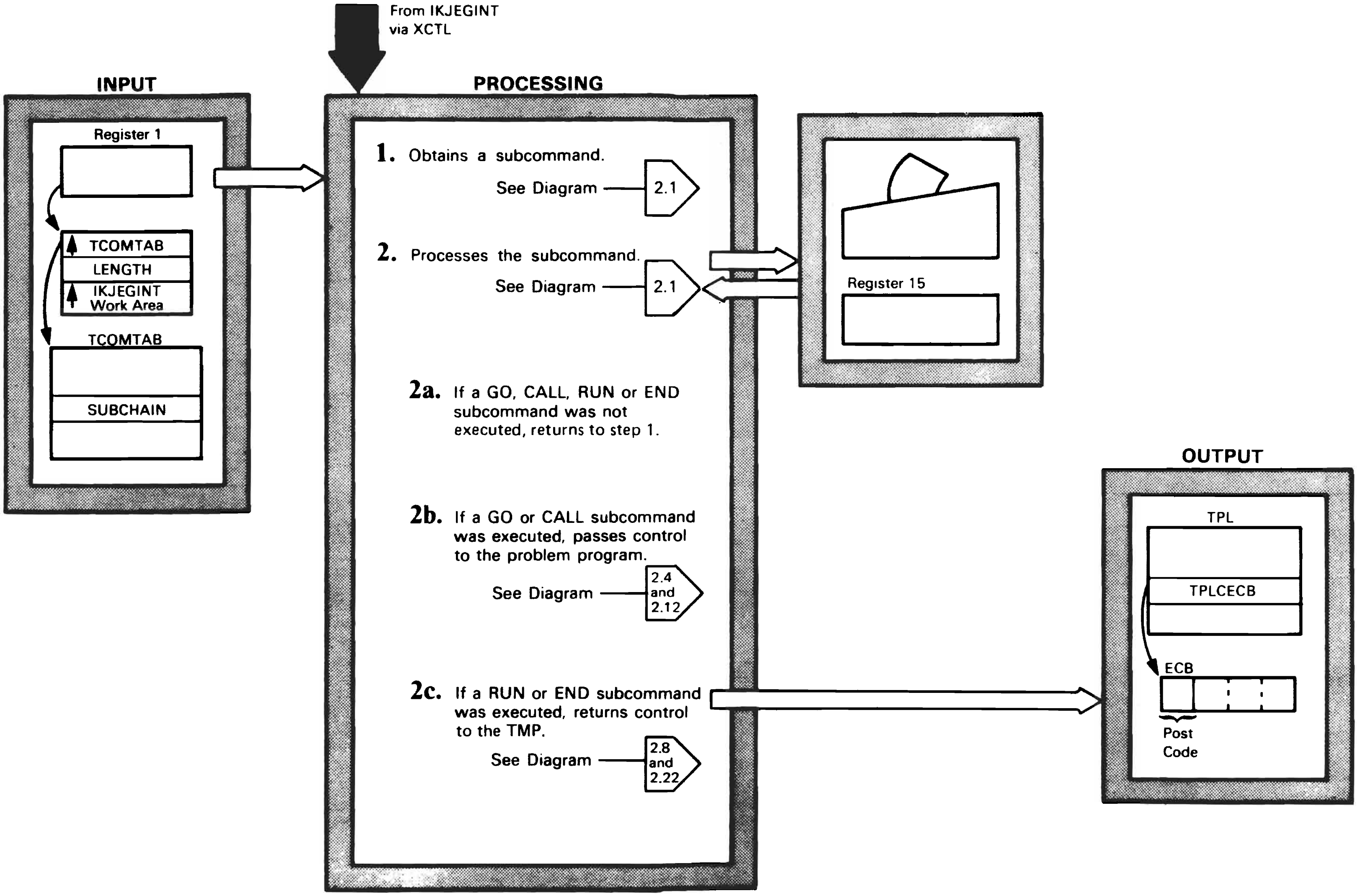
6. Determines if replaced instruction is to be executed in the breakpoint.

7. Determines type of replaced instruction.

8. Restarts problem program.

Description	Module	Label
1. The problem program executes until an SVC 97 instruction is encountered. Control passes to the SVC interruption handlers which save the program restart address in the problem program request block and the problem program register contents in the SVC 97 routine SVRB. Control passes to IGC0009G to process the breakpoint.		
2. IGC0009G determines if the current breakpoint is a user or a pseudo breakpoint by checking the TSTSVC field of the PPEXIT field in TCOMTAB or the BRKINST+6 field in BRKELEM. If any of the addresses of these fields match the address in the problem program PRB resume address field, the breakpoint is a pseudo breakpoint. If the breakpoint is a user breakpoint and the BRKCOUNT field of the break element is equal to zero, control passes to IKJEGMNL. Otherwise IGC0009G prepares to execute the replaced instruction and branches to step 6.	IGC0009G	QPSEUDO
3. The breakpoint allows a user to enter subcommands. Subcommands are obtained from a subcommand list specified in the AT subcommand or by issuing a TEST mode message to the terminal. Subcommand processors are accessible via a LINK macro instruction.		
4. Each subcommand specified is processed. Register contents are made to conform to those that were saved in the SVC 97 routine's SVRB. After a GO, CALL, LOAD, DELETE, GETMAIN, FREEMAIN, or QUALIFY (TCB) subcommand executes and returns control to IKJEGMNL, IKJEGMNL passes control to IGC0009G.		
5. If the subcommand entered is GO or CALL, IKJEGMNL sets the completion flag in the problem program's ECB.		
6-7. IGC0009G determines if the instruction saved in the break element is a BAL or a BALR. If the instruction is not a BAL or a BALR, it is executed in the breakpoint. If the instruction saved is a BAL or a BALR, execution is simulated and processing continues.	IGC0009G	BALBALR
IGC0009G tests the executed instruction and readjusts the restart address so that the problem program will resume executing at the next instruction after the replaced instruction.	IGC0009G	SETIC
8. The SVC 97 routine's SVRB is detected and the MVT Dispatcher restarts the problem program at the next instruction after the breakpoint.		

DIAGRAM 2. Overview of Processing Controlled by TEST



1. Obtains a subcommand.

See Diagram 2.1

2. Processes the subcommand.

See Diagram 2.1

2a. If a GO, CALL, RUN or END subcommand was not executed, returns to step 1.

2b. If a GO or CALL subcommand was executed, passes control to the problem program.

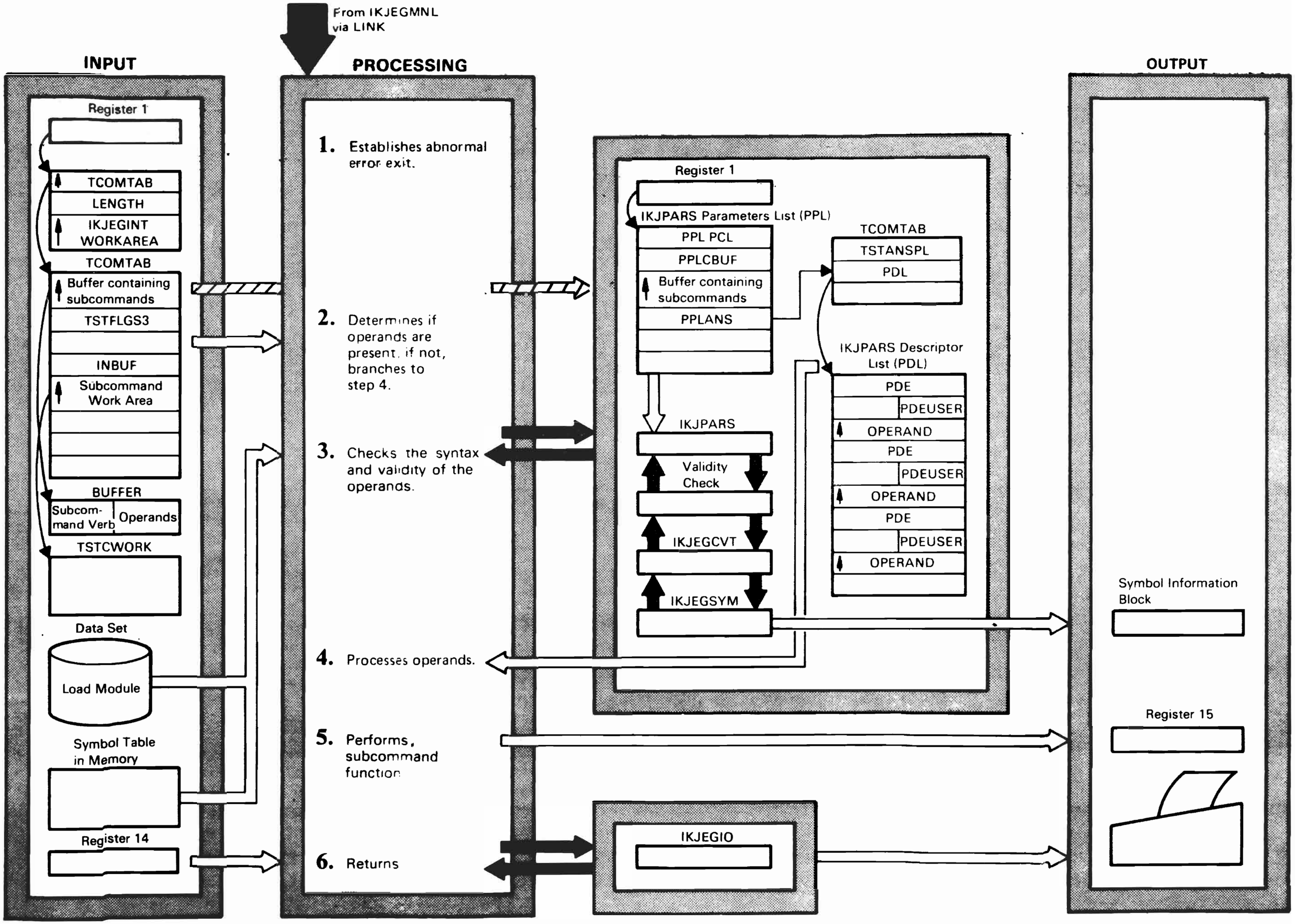
See Diagram 2.4 and 2.12

2c. If a RUN or END subcommand was executed, returns control to the TMP.

See Diagram 2.8 and 2.22

Description	Module	Label
1. Based on the contents of the SUBCHAIN field of TCOMTAB, a subcommand is obtained. If the SUBCHAIN field of TCOMTAB is non-zero, IKJEGIO obtains a subcommand from the subcommand chain built by the AT subcommand processor. If the SUBCHAIN field is zero, a GETLINE macro instruction is issued to obtain a subcommand from the terminal or the current source of input (for a discussion of the EXEC command see OS/VS2 TSO Command Language Reference).	IKJEGMNL IKJEGIO IKJEGIO IKJEGIO	SCREQ IKJEGGT A00700
2. IKJEGMNL passes control to a subcommand processor via a LINK macro instruction. The subcommand processor invokes IKJPARS to check the syntax of the subcommand and its operands. The subcommand processor performs its function by processing the operands. If necessary, the processor branches to IKJEGIO to issue data or diagnostic messages before returning control to IKJEGMNL. Upon completion of processing, the subcommand processor passes a return code to IKJEGMNL.	IKJEGMNL IKJEGXXX	MNLXX170
2a. When a subcommand other than GO, CALL, RUN or END is executed, IKJEGMNL receives a return code of 0 and returns to step 1 to obtain a new subcommand.	IKJEGMNL	SCREQ
2b. When either a GO or CALL subcommand is executed, IKJEGMNL receives a return code of 4 and passes control to the problem program to execute.	IKJEGMNL	IKJEGCTL
2c. When a RUN subcommand is executed, IKJEGMNL receives a return code of 12 and passes control to the TMP. The TMP sets the TPLCECB field of the TPL to point to the address of an ECB containing the post code. The TMP allows the problem program to execute when the RUN subcommand has been processed successfully.	IKJEGMNL	RUNRTURN

DIAGRAM 2.1. Processing a Subcommand



1. Establishes abnormal error exit.

2. Determines if operands are present. if not, branches to step 4.

3. Checks the syntax and validity of the operands.

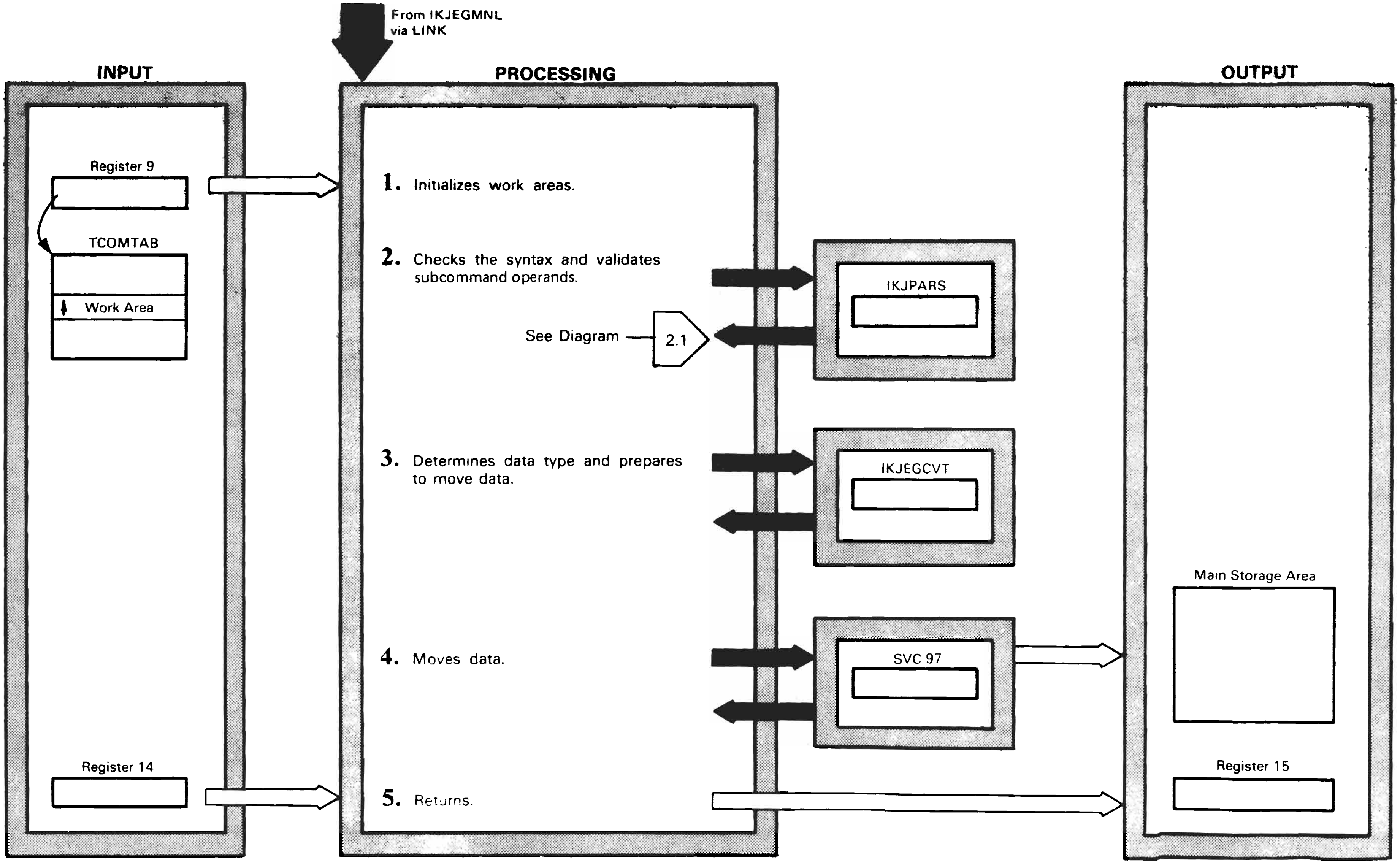
4. Processes operands.

5. Performs, subcommand function

6. Returns

Description	Module	Label
1. The subcommand processor invokes the STAE SVC routine to establish an abnormal error exit for subcommand processing.	IKJEGSTA	IKJEGSTA
2. The subcommand processor checks the NOPARMS flag in the TSTFLGS3 field in TCOMTAB to determine if operands are present. A '1' in the NOPARMS field indicates that there are no operands; a '0' indicates that operands exist. If there are no operands, the processor branches to step 5.		
3. IKJPARS checks the syntax of each subcommand operand found in the buffer pointed to by the INBUF field of TCOMTAB. IKJPARS then builds a PDE to contain the address of a valid operand. IKJPARS branches to the subcommand's validity check exit routine to check the value of the operands in the input buffer. The validity check routine does several general checks, then branches to IKJEGCVT, the convert module. IKJEGCVT converts the operand to hexadecimal and places the converted value in the PDEUSER field of the PDE. If the operand is a symbolic address or entry name, IKJEGCVT links to IKJEGSYM, the symbol module, to resolve the operand. IKJEGSYM evaluates the operand either by searching a symbol table built in main storage by the EQUATE subcommand or by reading symbol records or CESD records from auxiliary storage. When IKJEGSYM finds the symbol, it computes the hexadecimal value and places it in a symbol information block (SIB) that it builds. IKJEGSYM places a pointer to the SIB in the PDEUSER field of the PDE and flags the field. Control returns to IKJPARS if additional operands remain to be processed.	IKJEGCVT IKJEGSYM	TOBIN EXPADDR FINDSYM
4. If any routine -- IKJPARS, the validity check exit, IKJEGCVT, or IKJEGSYM -- finds an invalid operand or one it cannot convert to hexadecimal, it issues a diagnostic message and returns control to IKJPARS with an error code. IKJPARS prompts the terminal user to enter a new operand unless the user has previously specified the NOPROMPT option with the PROFILE command or has provided the TEST subcommand as part of a command procedure in storage.		
5. The subcommand processor performs its special function. Diagrams 2.2 through 2.23 show individual subcommand processors. The subcommand processor issues a return code indicating the status at the end of processing.		
6. When subcommand processing is complete, the subcommand processor may branch to IKJEGIO to issue a data message or a diagnostic message (if an error was encountered). The subcommand processor then returns control to IKJEGMNL by branching to the address specified in Register 14.		

DIAGRAM 2.2 Assignment Function Processor



1. Initializes work areas.

2. Checks the syntax and validates subcommand operands.

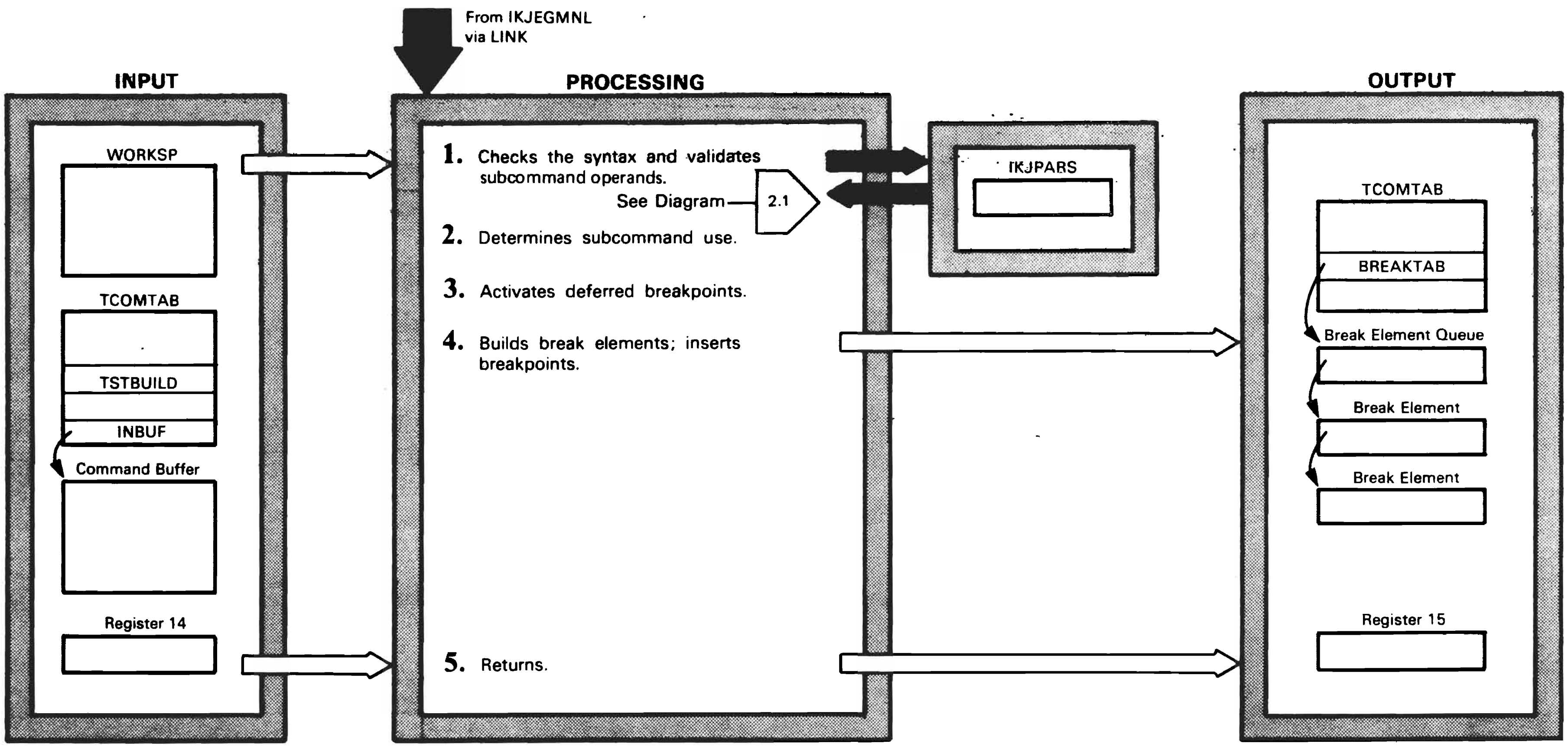
3. Determines data type and prepares to move data.

4. Moves data.

5. Returns.

Description	Module	Label
1. The assignment function is processed by modules IKJEGPCH and IKJEGASN. A general work area is established for use by IKJPARS.	IKJEGPCH	STAEOK
2. Control passes to IKJPARS via a LINK macro instruction to check the syntax of the operands and to build PDEs for each operand. IKJPARS returns control to IKJEGPCH.	IKJEGPCH	PARSEBLO
3. IKJEGPCH passes control to IKJEGASN via an XCTL macro instruction to determine the type of data the problem program has specified. If necessary, IKJEGCVT is used to convert data to hexadecimal.	IKJEGPCH	PCH018
4. The SVC 97 routine moves the data into the specified area or register.	IKJEGASN	REGINCRF
5. IKJEGASN issues a return code and returns control to IKJEGMNL by branching to the address specified in Register 14.	IKJEGASN	PCHRET1

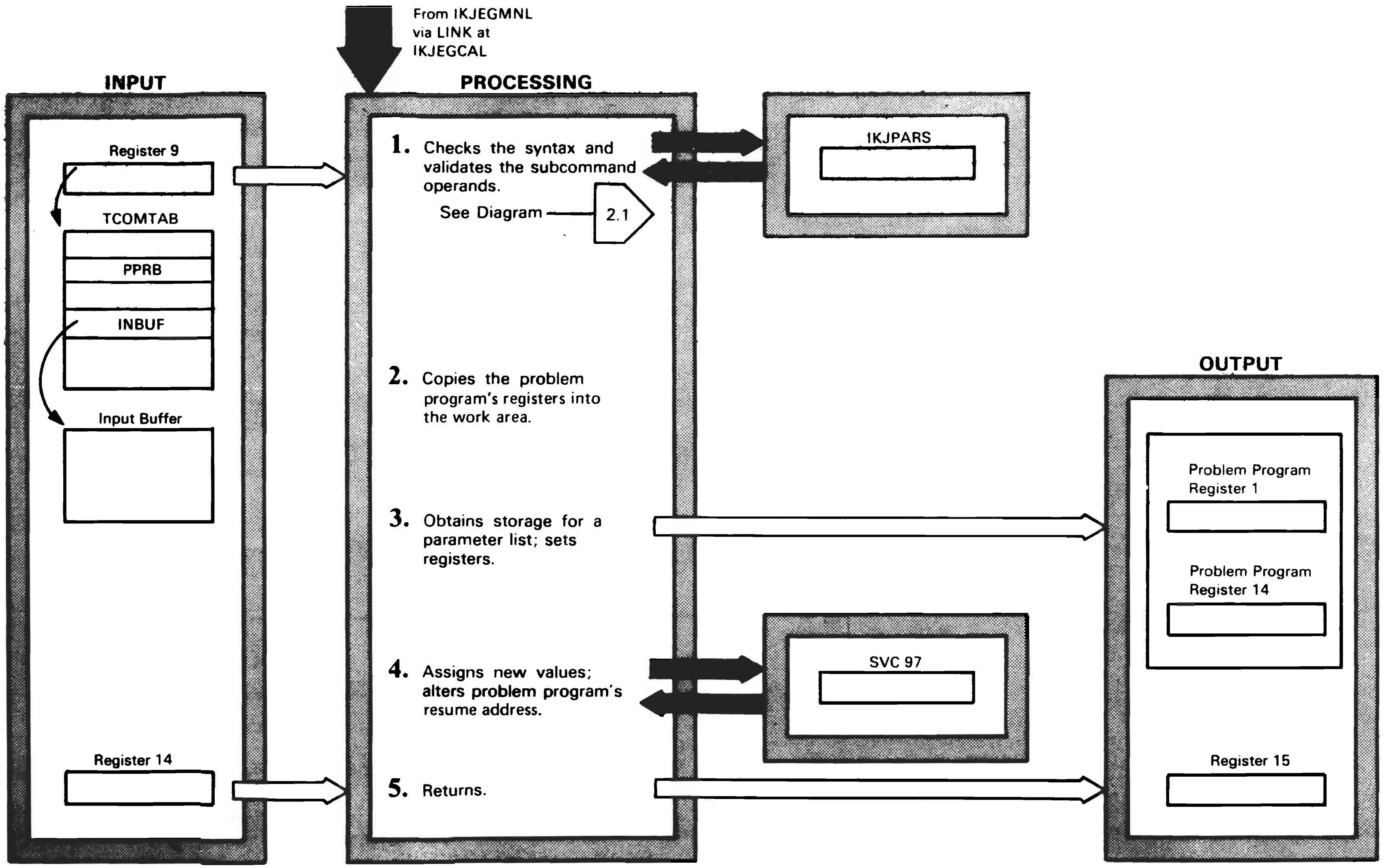
DIAGRAM 2.3. AT Subcommand Processor



1. Checks the syntax and validates subcommand operands.
   See Diagram 2.1
2. Determines subcommand use.
3. Activates deferred breakpoints.
4. Builds break elements; inserts breakpoints.
5. Returns.

Description	Module	Label
1. IKJEGAT invokes IKJPARS to check the syntax of the subcommand operands.		
2. IKJEGAT determines if the AT subcommand is for a DEFER request or for activating breakpoints. To activate a deferred breakpoint, the first two words of the WORKSP work area contain the name of the module for which deferred breakpoints are to be activated. The TSTBUILD switch in TCOMTAB is set to indicate that a new module is being fetched for the problem program. To set an active breakpoint or to store information about a deferred breakpoint, input parameters of the subcommand are in the command buffer, whose address is in the INBUF field of TCOMTAB. If the request is to set a DEFER breakpoint, this module transfers control to the second load module, IKJEGATD, via an XCTL macro instruction.	IKJEGAT	CONTINUE
3. IKJEGAT activates any deferred breakpoints that may have been specified for a newly fetched module. In this case, there is at least one element on the defer element queue. IKJEGAT builds one or more break elements and sets the corresponding breakpoints in the newly fetched module.	IKJEGAT	FROMDEFR
4. IKJEGAT builds break elements and inserts breakpoints at specified locations in the problem program.	IKJEGAT	BLDBREAK
5. IKJEGAT issues a return code and returns control to IKJEGMNL by branching to the address specified in Register 14.	IKJEGAT	RETMAIN

DIAGRAM 2.4. CALL Subcommand Processor



1. Checks the syntax and validates the subcommand operands.
   See Diagram 2.1

2. Copies the problem program's registers into the work area.

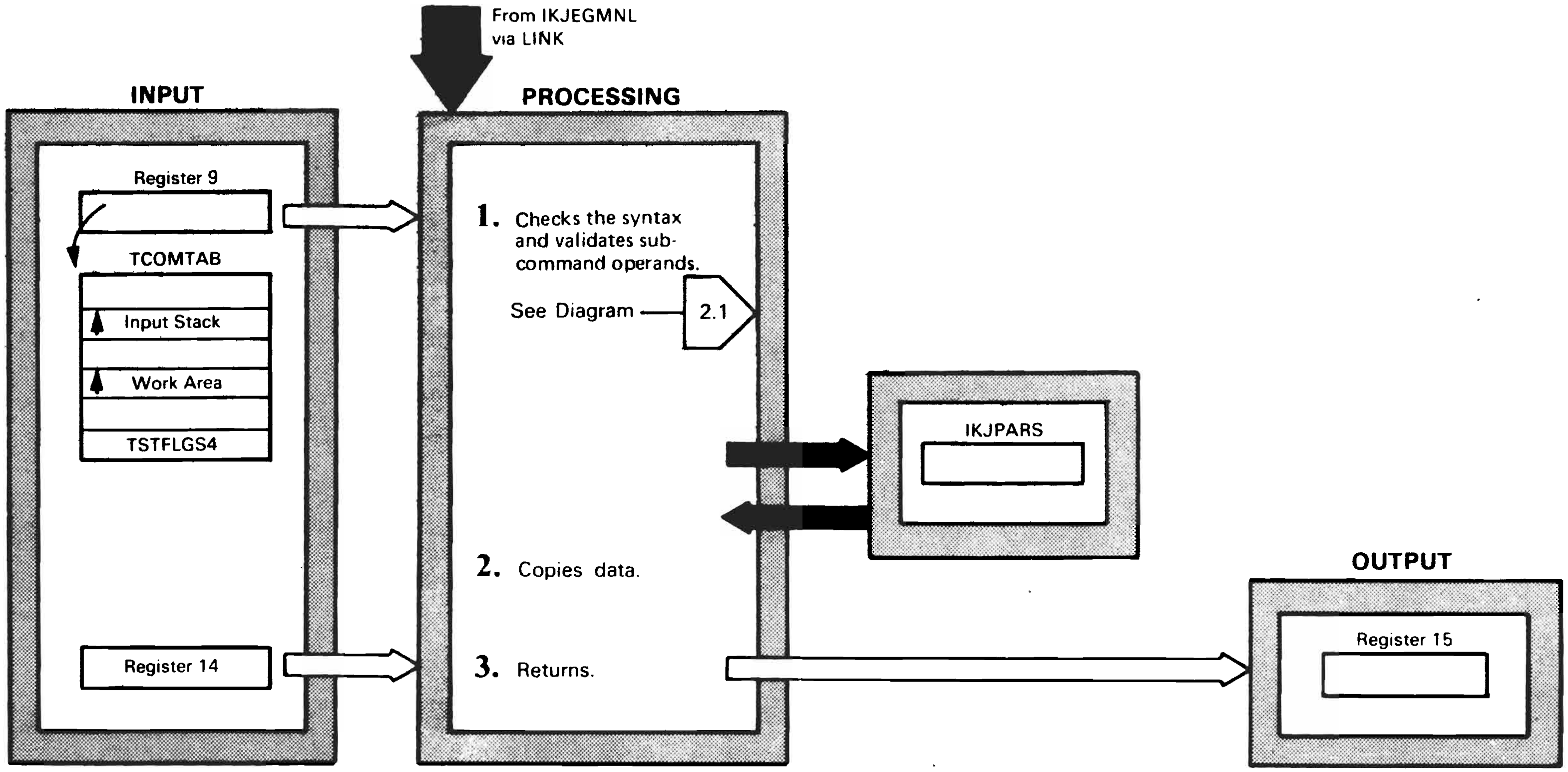
3. Obtains storage for a parameter list; sets registers.

4. Assigns new values; alters problem program's resume address.

5. Returns.

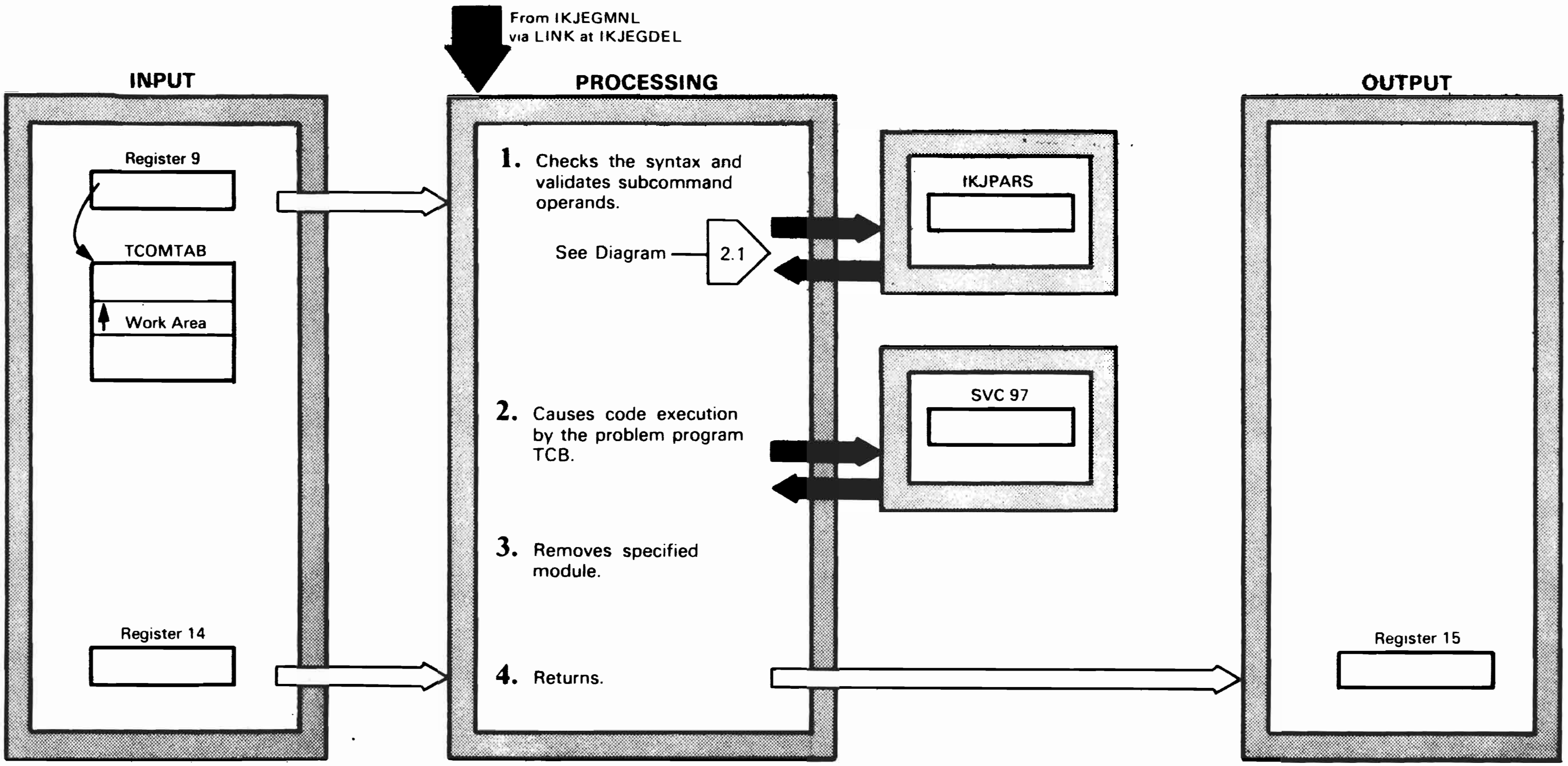
Description	Module	Label
1. The CALL subcommand is processed by IKJEGGO at entry point IKJEGCAL. IKJEGGO invokes IKJPARS to check the syntax of the subcommand operands.	IKJEGGO	PARSBILD
2. IKJEGGO copies the problem program's registers into its work area to assign new values to them.		
3. IKJEGGO issues a GETMAIN macro instruction to obtain main storage for a parameter list. If the user does not specify a parameter list, CALL builds a dummy parameter list. This parameter list is of variable length when the user specified the VL keyword operand, and IKJEGGO sets the high-order bit of the last full word in the parameter list to 1. The problem program's Register 1 points to the parameter list. IKJEGGO also places a user-specified address in the problem program's Register 14 if the user entered the RETURN keyword operand with an address following it. Otherwise, Register 14 points to the TSTSVC field in TCOMTAB.	IKJEGGO	BLDLST
4. IKJEGGO invokes the SVC 97 routine to modify the registers in its SVRB and to change the problem program's resume address as specified in the CALL subcommand.	IKJEGGO	ZAPREGS
5. IKJEGGO issues a return code and returns control to IKJEGMNL by branching to the address specified in Register 14.	IKJEGGO	RETURN

DIAGRAM 2.5. COPY Subcommand Processor



Description	Module	Label
1. IKJEGCPY invokes IKJPARS to build PDEs in a PDL and to check the syntax of the subcommand operands. IKJPARS then branches to check the validity of the exit routines, which check for certain specification errors (excessive length or a floating point register used as an address operand). If a validity check routine detects an error, it invokes IKJEGIO to issue an error message and requests IKJPARS to prompt the terminal for a new operand.	IKJEGCPY IKJEGCPY IKJEGCPY	PARSE IKJEGFM IKJEGTO IKJEGLN VTPUT
2. IKJEGCPY copies data specified by the subcommand operands.	IKJEGCPY	COPY0 COPY
3. IKJEGCPY issues a return code and returns control to IKJEGMNL by branching to the address specified in Register 14.	IKJEGCPY	MNLRET

DIAGRAM 2.6. DELETE Subcommand Processor



1. Checks the syntax and validates subcommand operands.

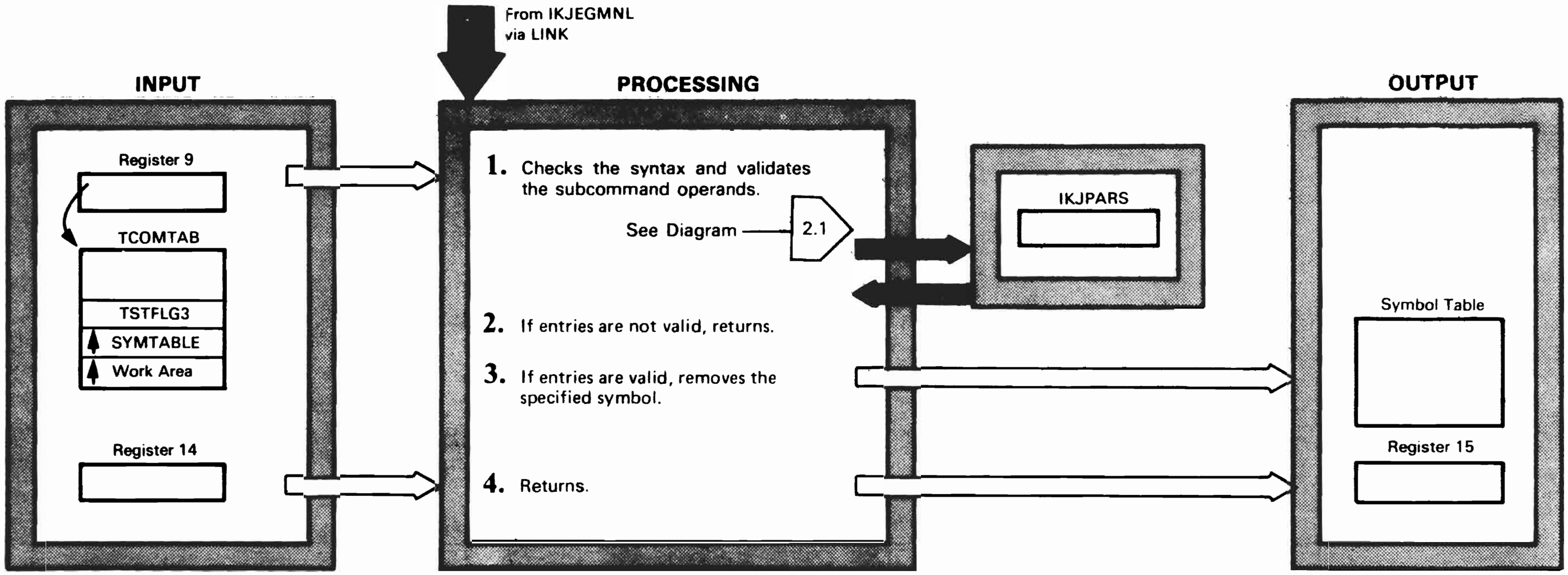
2. Causes code execution by the problem program TCB.

3. Removes specified module.

4. Returns.

Description	Module	Label
The DELETE subcommand is processed by IKJEGLDF, which receives control from IKJEGMNL at entry point IKJEGDEL.	IKJEGLDF	
1. IKJEGLDF passes control to IKJPARS to check the syntax of the operands and to build PDEs for each operand. Upon completion, IKJPARS returns control to IKJEGLDF.	IKJEGLDF	SETPARM
2. Using the SVC 97 routine, TASKSW (a closed subroutine) causes the execution of IKJEGLDF code to take place under the problem program TCB rather than the TEST TCB. This allows the execution of the DELETE macro instruction to take place under the problem program TCB.	IKJEGLDF	TASKSW
3. The specified module is removed using the DELETE macro instruction.	IKJEGLDF	CONTDEL
4. IKJEGLDF issues a return code and returns control to IKJEGMNL by branching to the address specified in Register 14.	IKJEGLDF	SMTDEL

DIAGRAM 2.7. DROP Subcommand Processor



1. Checks the syntax and validates the subcommand operands.

   See Diagram 2.1

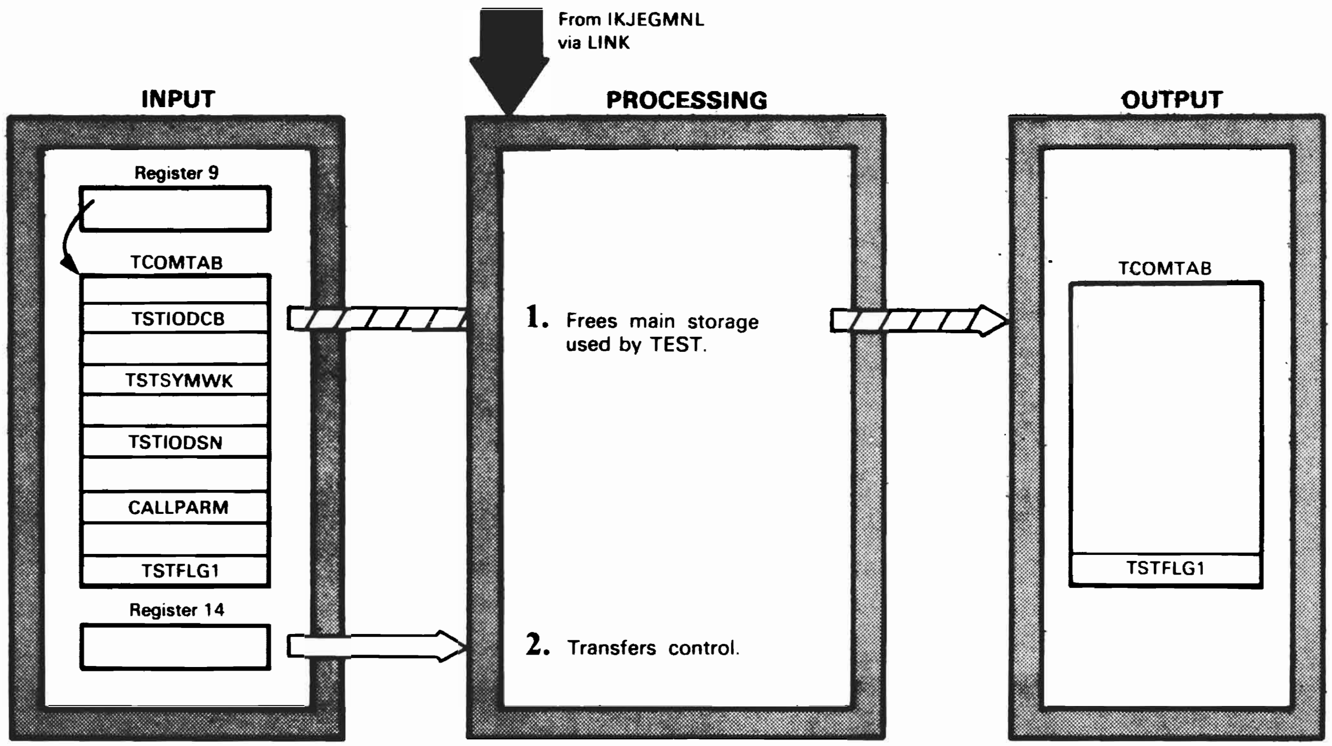
2. If entries are not valid, returns.

3. If entries are valid, removes the specified symbol.

4. Returns.

Description	Module	Label
1. IKJEGEQU passes control to IKJPARS to check the syntax of the operands and to build the PDEs in the work area for each operand. Upon completion, IKJPARS returns control to IKJEGEQU.	IKJEGEQU	IKJEGVDK
2. If the symbol table or the specified entry isn't valid, an error message is sent to the user and control passes to IKJEGMNL.		
3. If the specified entry is valid, pointers are reset, removing the entries from the chain as specified. If no operands are entered, the entire table is deleted. This is indicated by the NOPARMS flag in the TSTFLG3 field of TCOMTAB.		
4. IKJEGEQU issues a return code and returns control to IKJEGMNL by branching to the address specified in Register 14.		

DIAGRAM 2.8. END Subcommand Processor

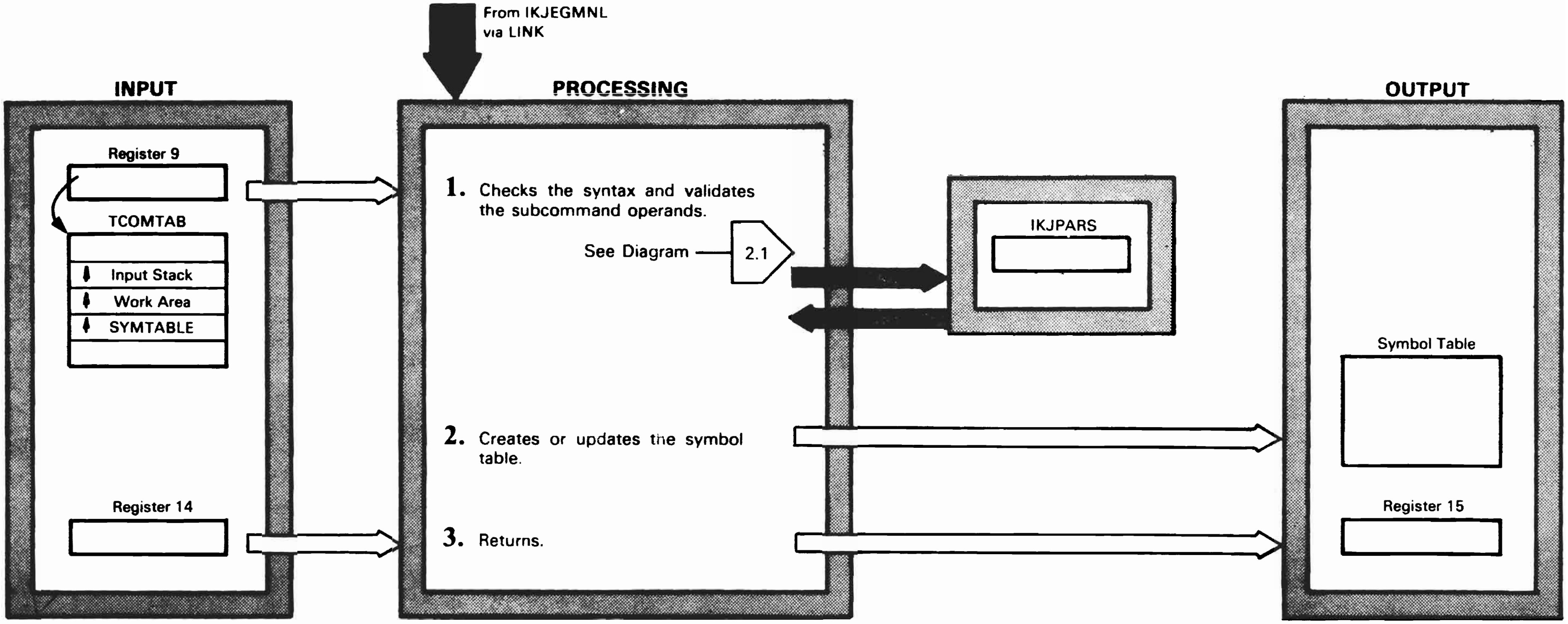


1. Frees main storage used by TEST.

2. Transfers control.

Description	Module	Label
1. IKJEGEND, the END subcommand module, frees the main storage used by TEST that is pointed to by fields in TCOMTAB.	IKJEGEND	END0000
• When the TSTIODCB field of TCOMTAB points to an open DCB, IKJEGEND closes and frees the DCB.		
• When the TSTSYMWK field of TCOMTAB points to an open DCB, IKJEGEND closes and frees the DCB.		END00020
• When the TSTIODSN field of TCOMTAB points to a data set name block, IKJEGEND frees the block.		END0060
• When the CALLPARM field of TCOMTAB points to CALL parameter blocks, IKJEGEND frees the blocks.		END0100
IKJEGEND sets the ENDSW in the TSTFLG1 field of TCOMTAB.	IKJEGEND	END0160
2. When the main storage used by TEST has been freed, IKJEGEND transfers control to IKJEGOFF via an XCTL macro instruction. If the STAE Exit routine is to be entered, IKJEGEND returns control to IKJEGMNL.		

DIAGRAM 2.9. EQUATE Subcommand Processor



1. Checks the syntax and validates the subcommand operands.

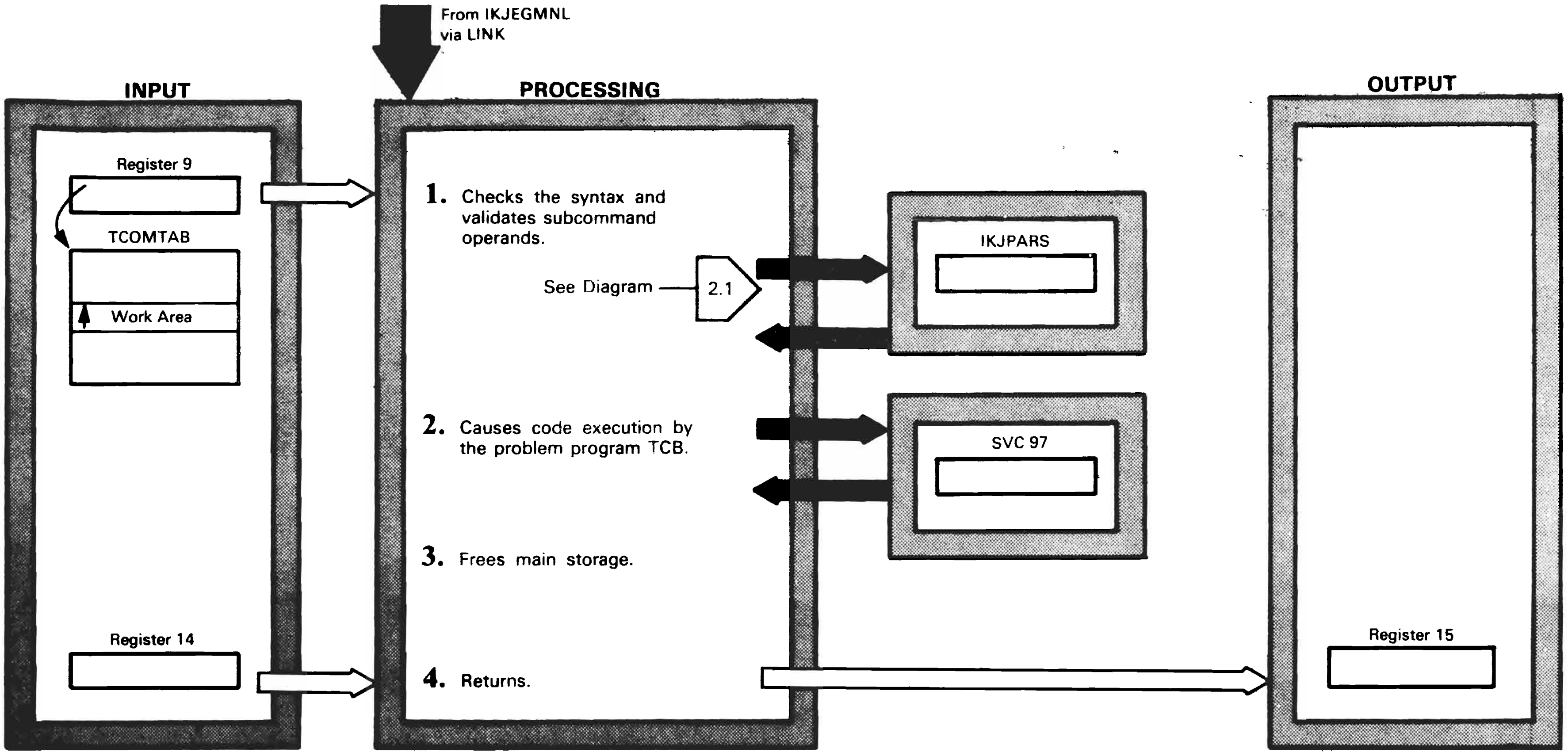
   See Diagram 2.1

2. Creates or updates the symbol table.

3. Returns.

Description	Module	Label
1. After preparing a parameter list, IKJEGEQU passes control to IKJPARS to check the syntax of the operands and to build PDEs for each operand. Upon completion, IKJPARS returns control to IKJEGEQU.	IKJEGEQU	IKJEGVDK
2. IKJEGEQU creates or updates an entry in the symbol table, the address of which comes from the PDE If a new entry in the symbol table is needed, pointers are updated to point to the created entry in the table. If necessary, IKJEGCVT is used to convert values to printable format.	IKJEGCVT	FROBIN
3. IKJEGEQU issues a return code and returns control to IKJEGMNL by branching to the address specified in Register 14.		

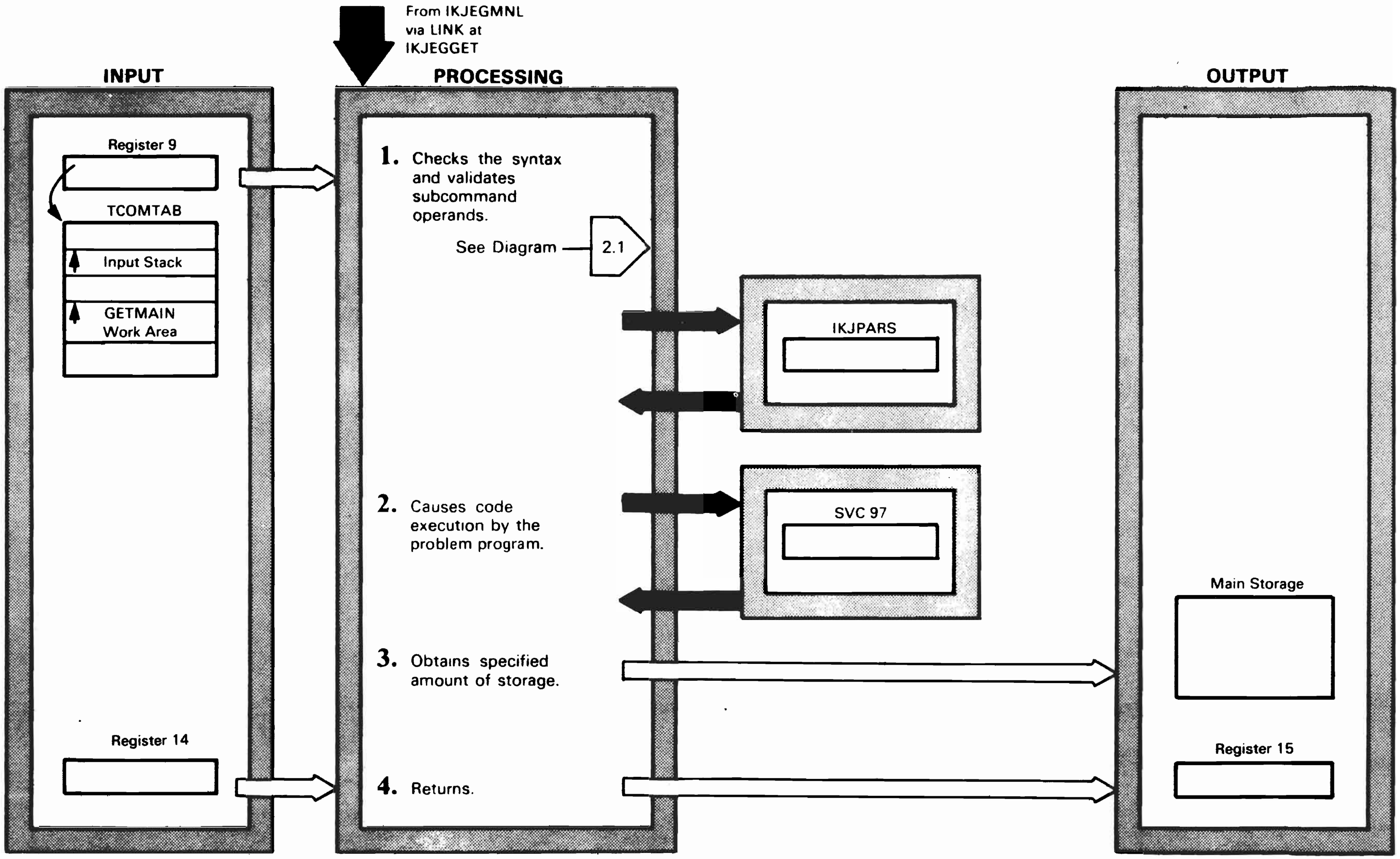
DIAGRAM 2.10. FREEMAIN Subcommand Processor



1. Checks the syntax and validates subcommand operands.
2. Causes code execution by the problem program TCB.
3. Frees main storage.
4. Returns.

Description	Module	Label
1. IKJEGLDF passes control to IKJPARS to check the syntax of the operands and to build PDEs for each operand. Upon completion, IKJPARS returns control to IKJEGLDF.	IKJEGLDF	SETPARM
2. Using the SVC 97 routine, TASKSW (a closed subroutine) causes the execution of IKJEGLDF code to take place under the problem program TCB rather than the TEST TCB. This allows the execution of the FREEMAIN macro instruction to take place under the problem program TCB.	IKJEGLDF	TASKSW
3. The specified area is freed using the FREEMAIN macro instruction.	IKJEGLDF	CONTFREE
4. IKJEGLDF issues a return code and returns control to IKJEGMNL by branching to the address specified in Register 14.	IKJEGLDF	STAEOFF

DIAGRAM 2.11. GETMAIN Subcommand Processor



1. Checks the syntax and validates subcommand operands.

   See Diagram 2.1

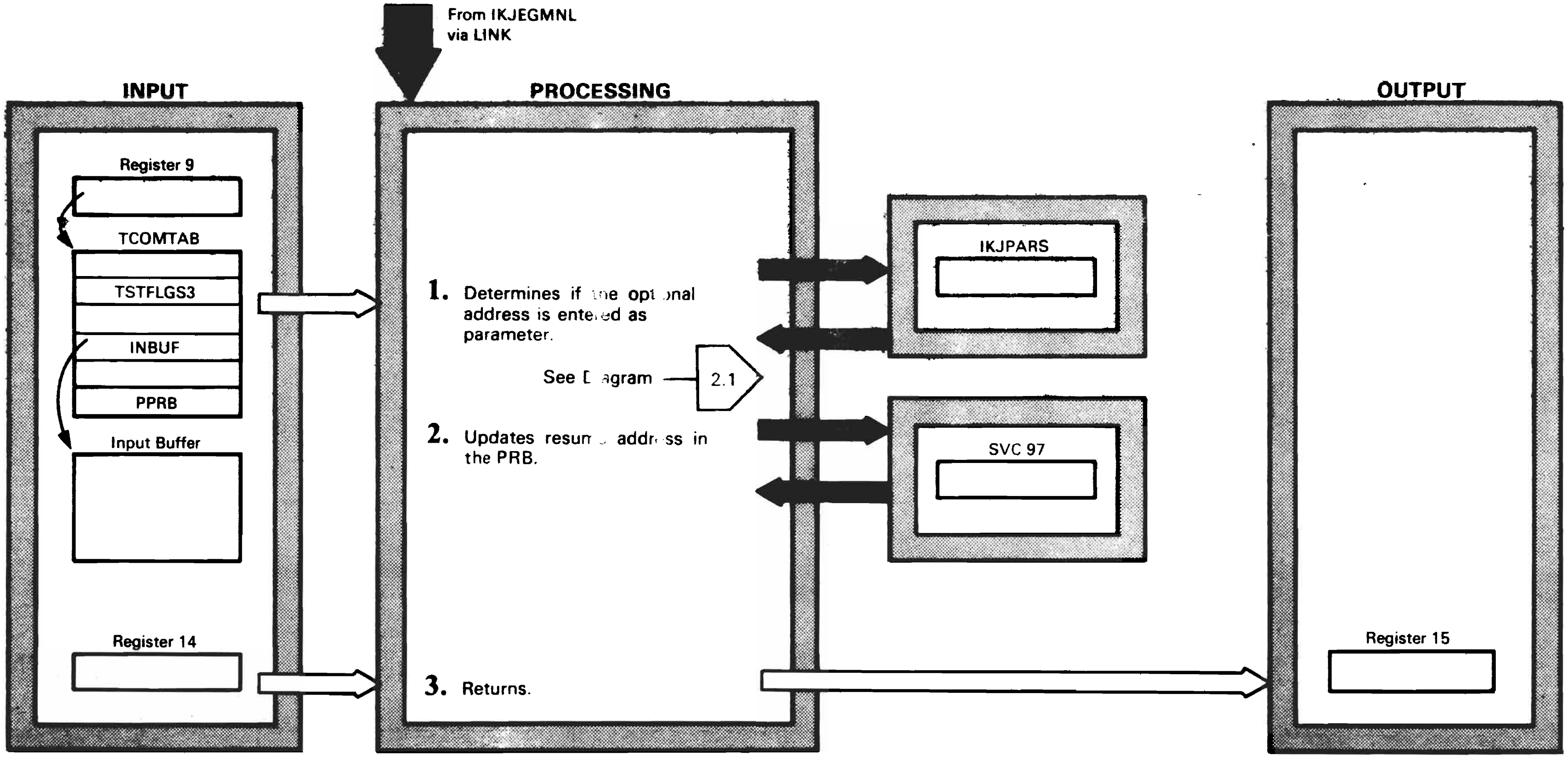
2. Causes code execution by the problem program.

3. Obtains specified amount of storage.

4. Returns.

Description	Module	Label
1. IKJEGLDF pâsses control to IKJPARS to check the syntax of the operands and to build PDEs for each operand. Upon completion, IKJPARS returns control to IKJEGLDF.	IKJEGLDF	SETPARM
2. Using the SVC 97 routine, TASKSW (a closed subroutine) causes the execution of IKJEGLDF code to take place under the problem program TCB (task control block) rather than the TEST TCB. This is accomplished by saving the problem program's registers to be restored later by IKJEGMNL. IKJEGLDF's registers are then placed in the problem program SVRB by the SVC 97 routine. The resume address in the problem program PRB is then altered to contain an address within IKJEGLDF. This allows the execution of the GETMAIN macro instruction to take place under the problem program TCB.	IKJEGLDF	TASKSW
3. The specified number of bytes of main storage are acquired using the GETMAIN macro instruction.	IKJEGLDF	CONTGET
4. IKJEGLDF issues a return code and returns control to IKJEGMNL by branching to the address specified in Register 14.	IKJEGLDF	DONTEQU

DIAGRAM 2.12. GO Subcommand Processor



1. Determines if the optional address is entered as parameter.

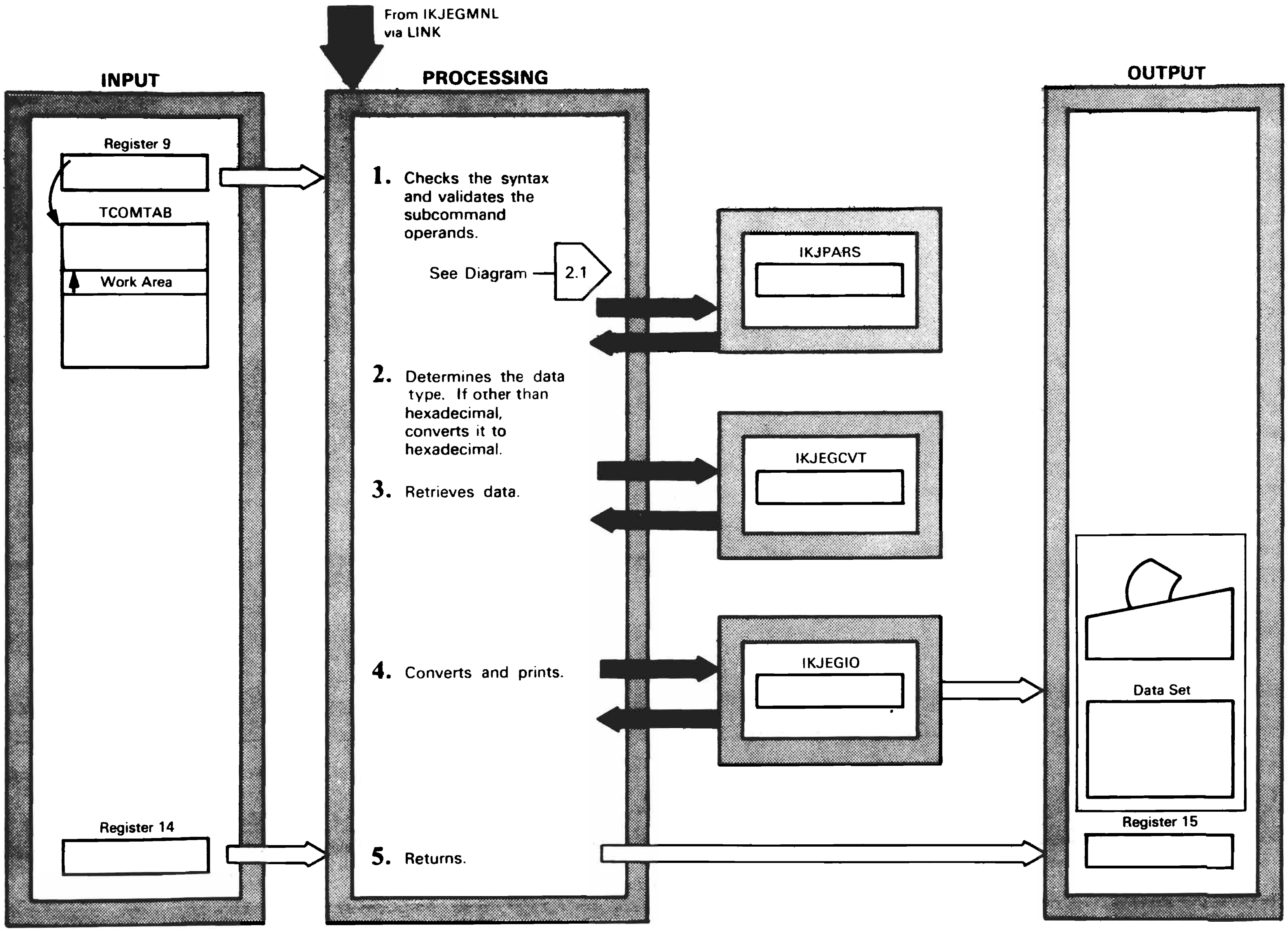
   See Diagram 2.1

2. Updates resume address in the PRB.

3. Returns.

Description	Module	Label
1. IKJEGGO receives control at entry point IKJEGGO. It checks the NOPARMS flag in the TSTFLGS3 field in TCOMTAB to determine whether the optional address was entered as a parameter. If so, IKJEGGO invokes IKJPARS to build a PDL. Otherwise the address in the current instruction counter is used as the address at which processing resumes.	IKJEGGO	PARSBILD
2. If the user specifies an address, the SVC 97 routine is invoked to update the resume address in the PRB.	IKJEGGO	PSWSET
3. IKJEGGO issues a return code and returns control to IKJEGMNL by branching to the address specified in Register 14.	IKJEGGO	EXIT

DIAGRAM 2.13. LIST Subcommand Processor



1. Checks the syntax and validates the subcommand operands.

   See Diagram 2.1

2. Determines the data type. If other than hexadecimal, converts it to hexadecimal.

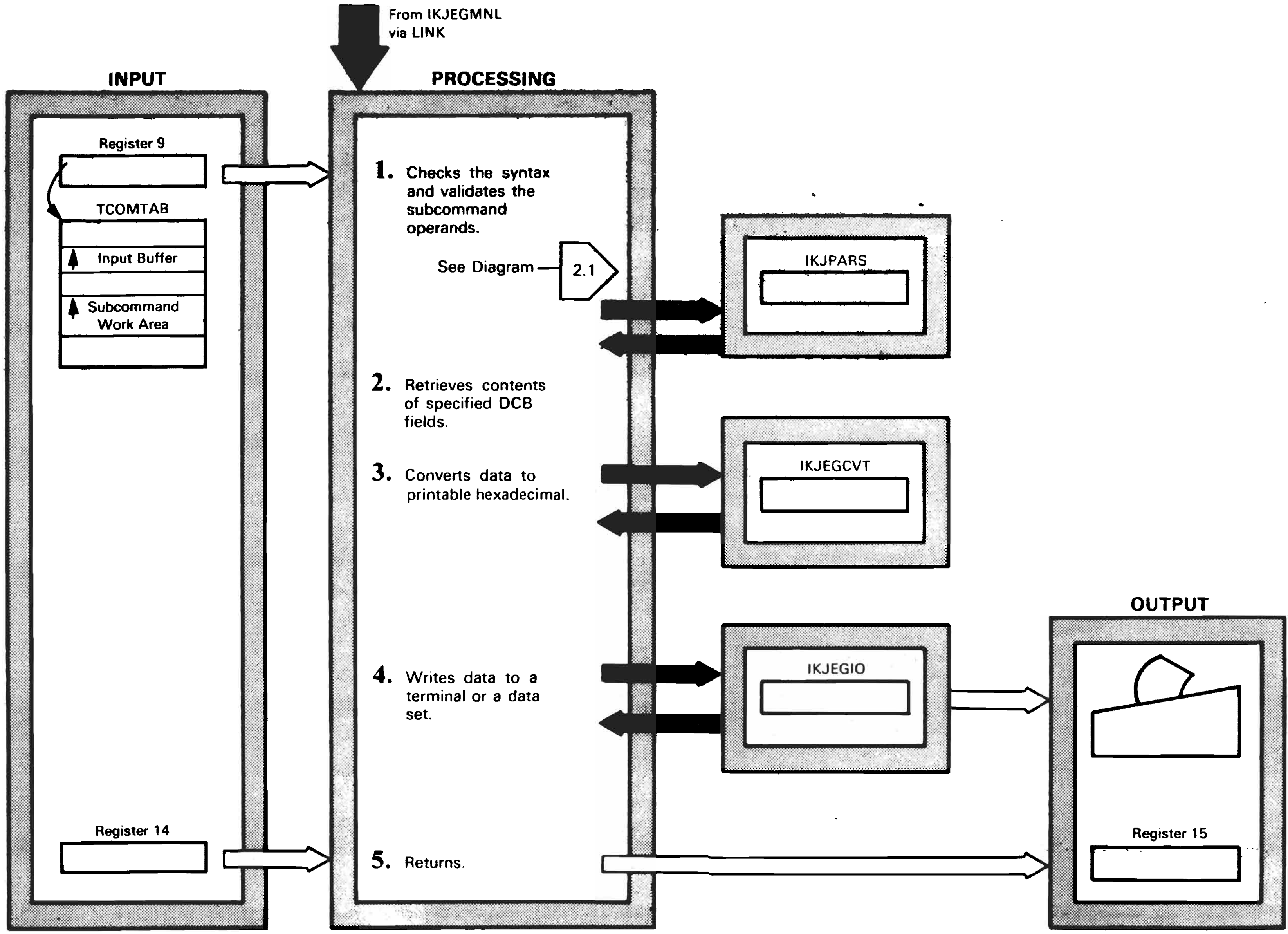
3. Retrieves data.

4. Converts and prints.

5. Returns.

Description	Module	Label
1. IKJEGLST invokes IKJPARS to check the syntax of the subcommand operands and to build PDEs for each operand. IKJPARS returns control to IKJEGLST.	IKJEGLST	SMTPARS
2. The data type is determined from the subcommand operands. If the data type is not hexadecimal or register, control passes to the second processing module, IKJEGLSA.	IKJEGLST IKJEGLSA	LIST
3. The requested data is retrieved from main storage or from registers.		
4. IKJEGCVT is used to convert the data to a printable format. IKJEGIO prints the data.	IKJEGCVT	FROBIN
5. Either IKJEGLST or IKJEGLSA issues a return code and returns control to IKJEGMNL by branching to the address specified in Register 14.		

DIAGRAM 2.14. LISTDCB Subcommand Processor



1. Checks the syntax and validates the subcommand operands.

   See Diagram 2.1

2. Retrieves contents of specified DCB fields.

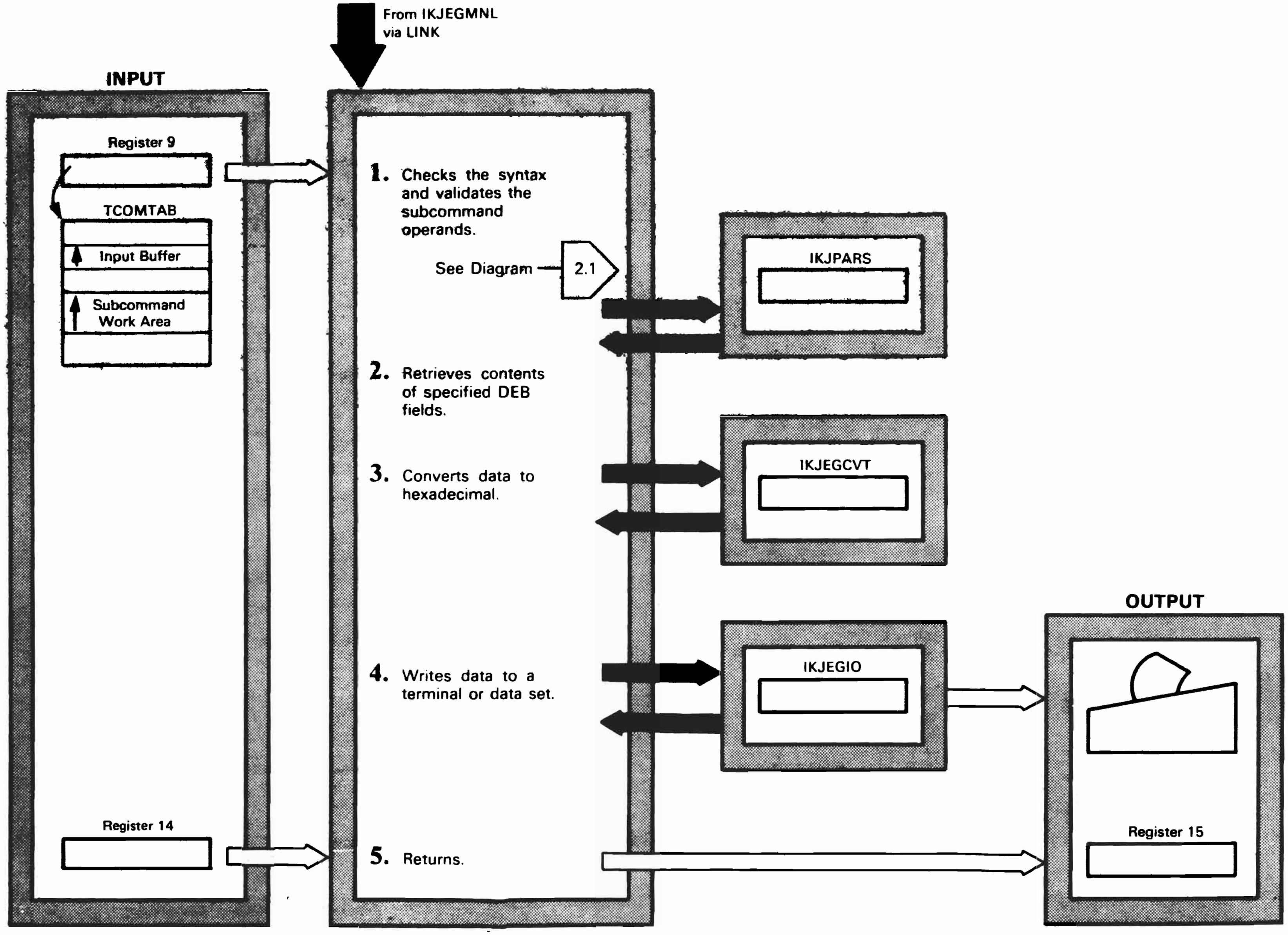
3. Converts data to printable hexadecimal.

4. Writes data to a terminal or a data set.

5. Returns.

Description	Module	Label
1. IKJEGDCB, the LISTDCB subcommand module, invokes IKJPARS to build PDEs in a PDL (one for each DCB field specified by the user) and to check the syntax of the subcommand operands. IKJPARS also sets a flag in the PDEs for each field requested by the user. IKJPARS branches to IKJEGDVK to determine the validity of the DCB address specified as an operand.	IKJEGDCB IKJEGDCB	PARSBLOW IKJEGDVK
2. IKJEGDCB retrieves the contents of the specified fields in the DCB. LISTDCB lists only specified fields within a DCB if the terminal user type the keyword "FIELD", followed by the names of the fields he wants listed.	IKJEGDCB	IKJEGDCB
3. IKJEGDCB branches to IKJEGCVT to convert the DCB data to printable hexadecimal.	IKJEGDCB IKJEGCVT	CONVTAB FROMBIN
4. IKJEGDCB branches to IKJEGIO to write the data either to a terminal or to a data set.	IKJEGIO	MSGIOCTD
5. IKJEGDCB issues a return code and returns control to IKJEGMNL by branching to the address specified in Register 14.	IKJEGDCB	RETURN

DIAGRAM 2.15. LISTDEB Subcommand Processor



1. Checks the syntax and validates the subcommand operands.

   See Diagram 2.1

2. Retrieves contents of specified DEB fields.

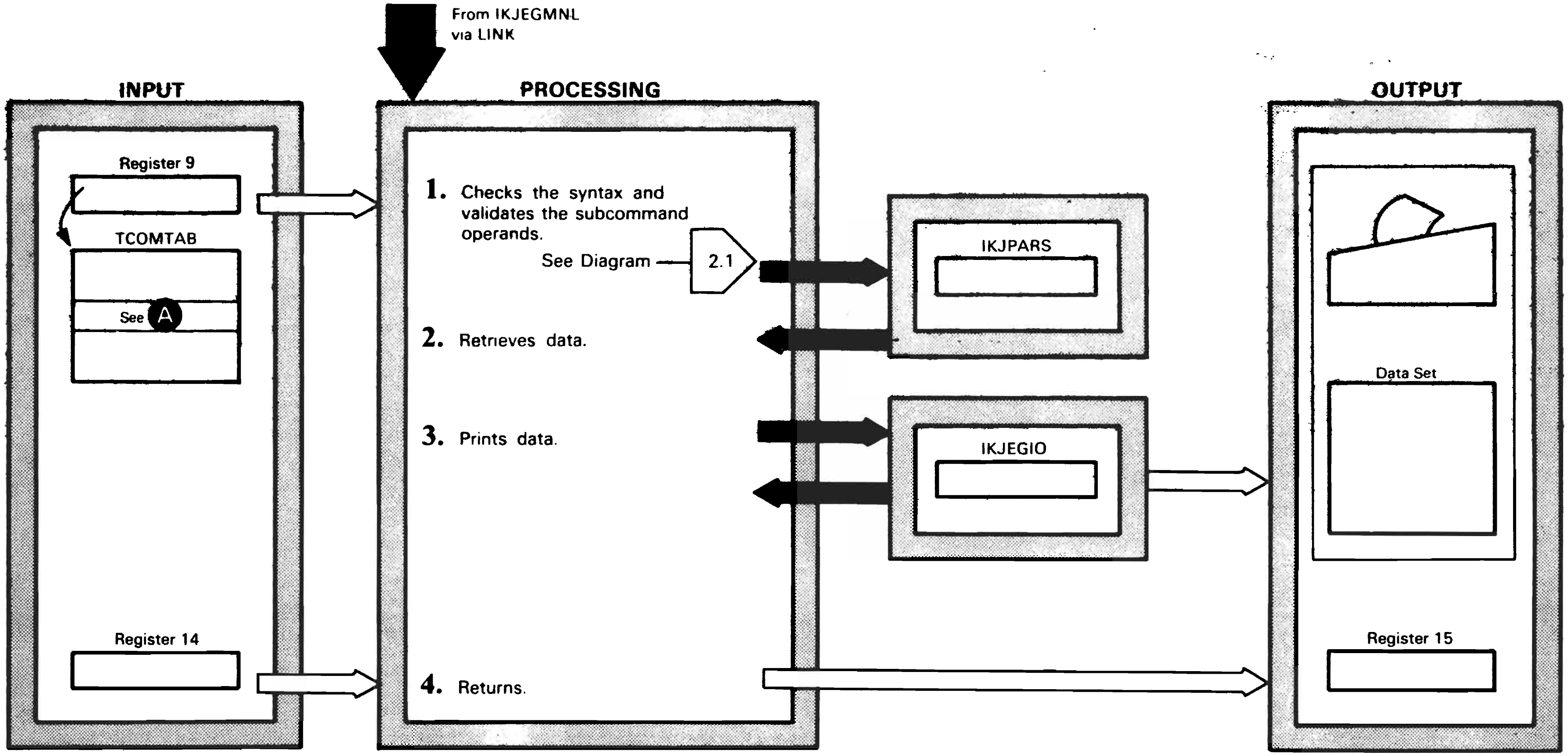
3. Converts data to hexadecimal.

4. Writes data to a terminal or data set.

5. Returns.

Description	Module	Label
1. IKJEGDEB invokes IKJPARS to check the syntax of the subcommand operands and to build PDEs in a PDL (one PDL for each DEB field specified by the user). IKJPARS also sets a flag in the PDEs for each field requested by the user. IKJPARS then branches to DEBVALCK to determine the validity of the DEB address specified as an operand.	IKJEGDEB IKJEGDEB	PADEND1 DEBVALCK
2. IKJEGDEB retrieves the contents of the specified fields in the DEB.		
3. IKJEGDEB branches to IKJEGCVT to convert the DEB data to printable hexadecimal.	IKJEGDEB IKJEGCVT	DEBCVT FROMBIN
4. IKJEGDEB branches to IKJEGIO to write the data either to a terminal or to a data set.	IKJEGDEB	DEBIO
5. IKJEGDEB issues a return code and returns control to IKJEGMNL by branching to the address specified in Register 14.	IKJEGDEB	RETURN

DIAGRAM 2.16. LISTMAP Subcommand Processor



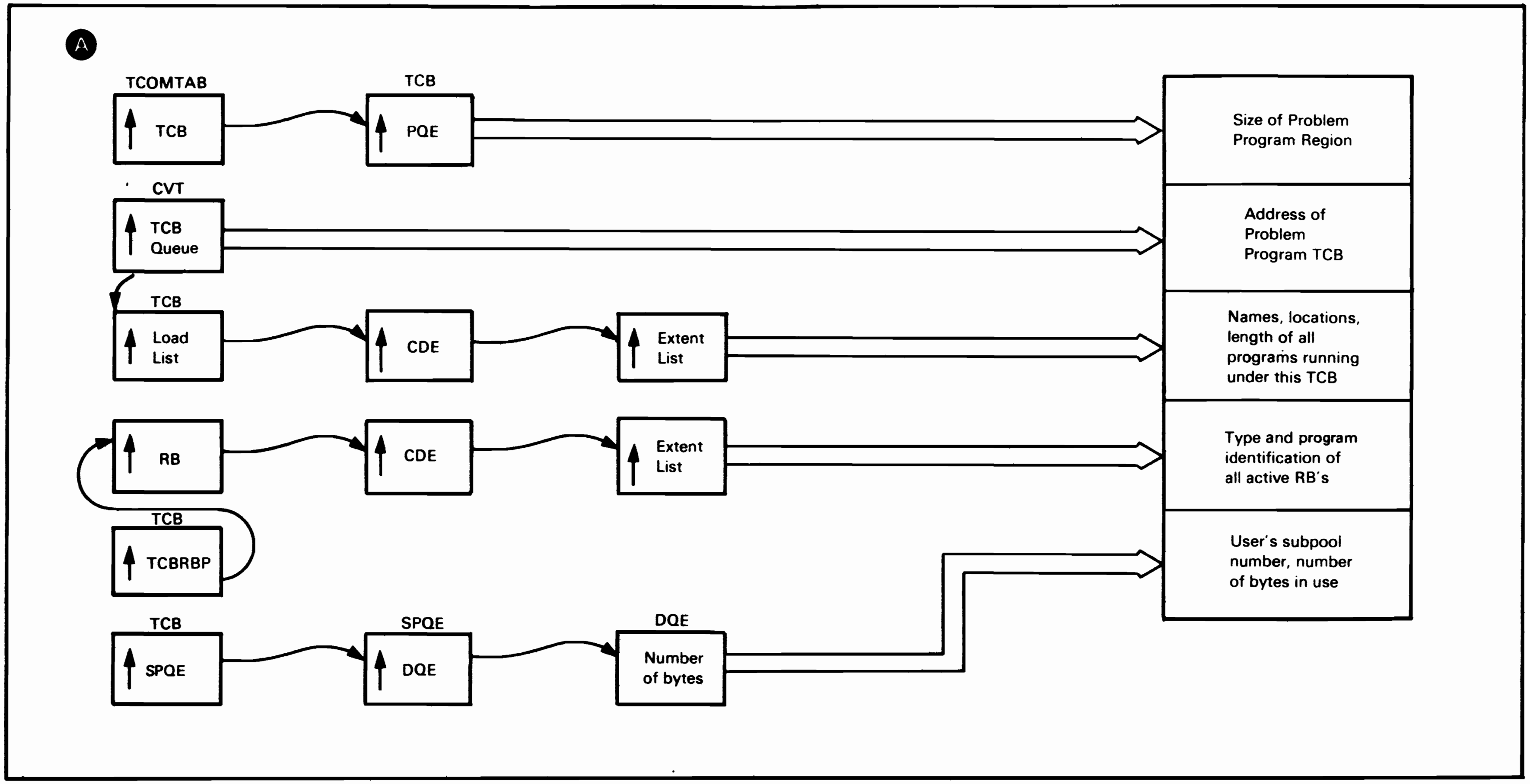
1. Checks the syntax and validates the subcommand operands.

   See Diagram 2.1

2. Retrieves data.

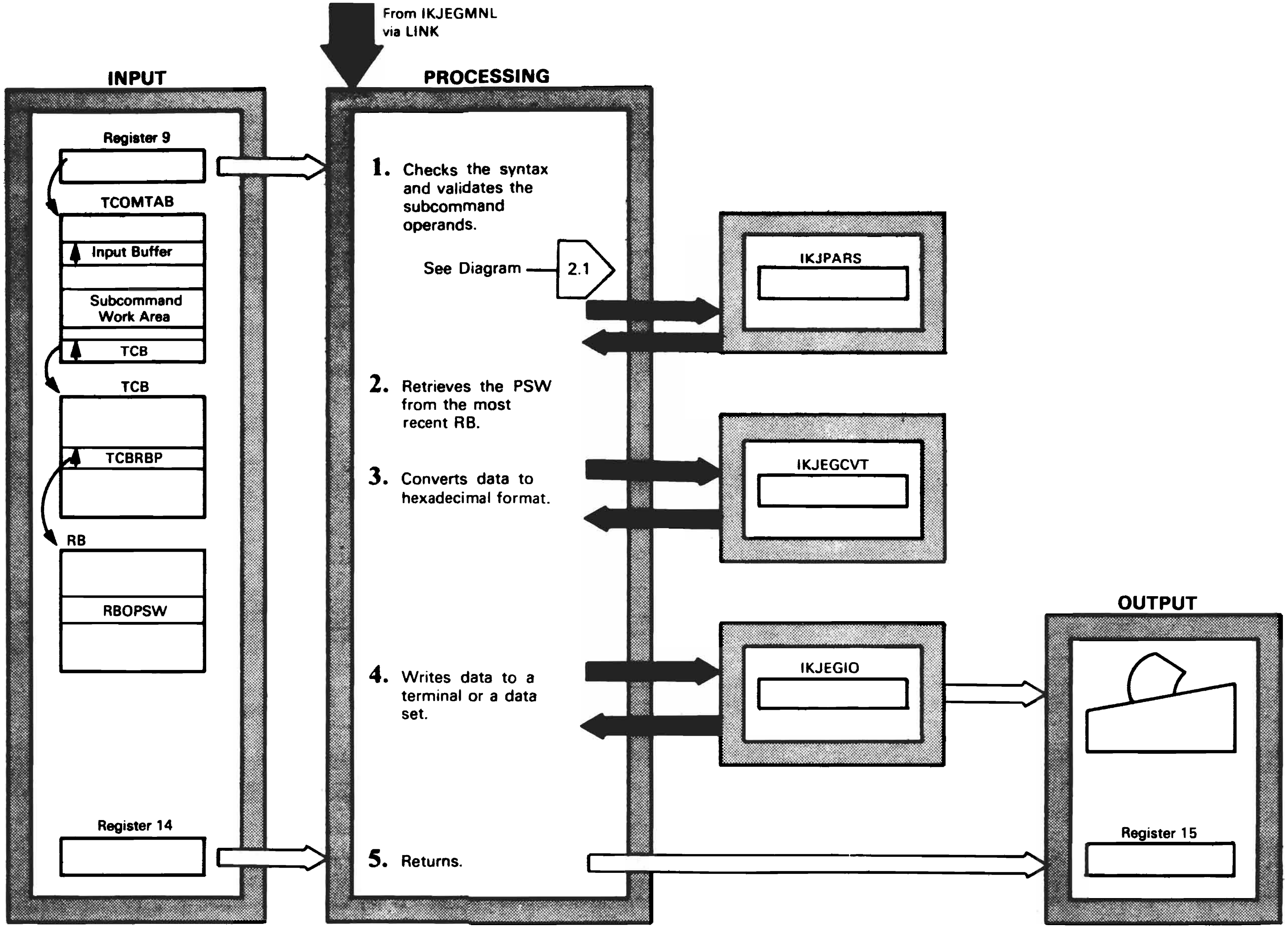
3. Prints data.

4. Returns.


A
TCOMTAB
TCB
TCB
PQE
CVT
TCB Queue
TCB
Load List
CDE
Extent List
RB
CDE
Extent List
TCB
TCBRBP
TCB
SPQE
SPQE
DQE
DQE
Number of bytes
Size of Problem Program Region
Address of Problem Program TCB
Names, locations, length of all programs running under this TCB
Type and program identification of all active RB's
User's subpool number, number of bytes in use

Description	Module	Label
1. IKJEGMAP passes control to IKJPARS to check the syntax of the subcommand operands and to build PDEs in the PDL if the PRINT operand was specified. Upon completion, IKJPARS returns control to IKJEGMAP.	IKJEGMAP IKJEGCVT	A0020 FROMBIN
2. Data requested by the user is gathered. Insert A shows the pointer scheme of the data.		
3. Data is converted to printable format using IKJEGCVT. IKJEGIO is called to write the data to a terminal or to a data set.		
4. IKJEGMAP issues a return code and returns control to IKJEGMNL by branching to the address specified in Register 14.		

DIAGRAM 2.17. LISTPSW Subcommand Processor



1. Checks the syntax and validates the subcommand operands.

   See Diagram 2.1

2. Retrieves the PSW from the most recent RB.

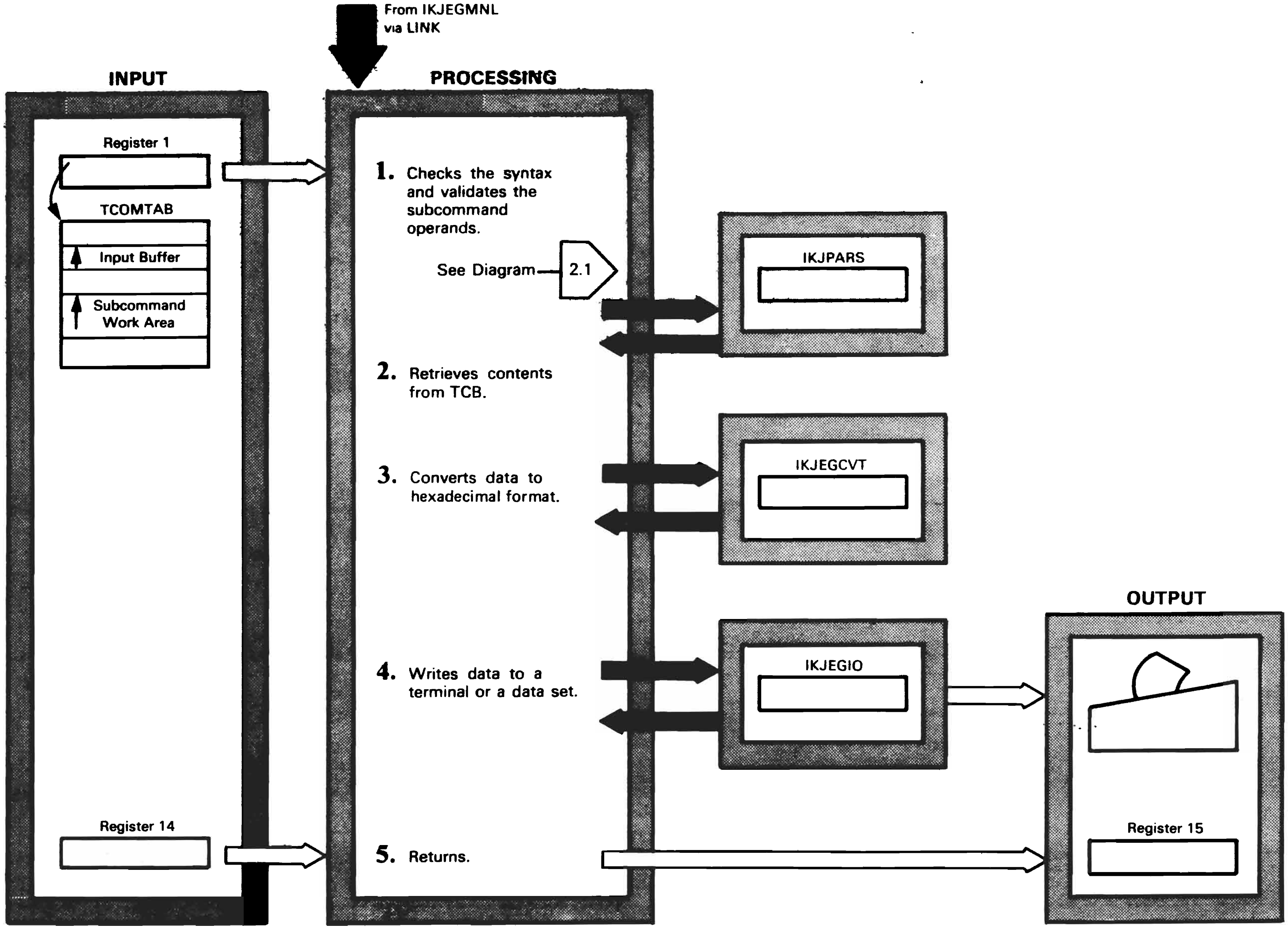
3. Converts data to hexadecimal format.

4. Writes data to a terminal or a data set.

5. Returns.

Description	Module	Label
1. IKJEGPSW invokes IKJPARS to check the syntax of the subcommand operands and to build PDEs in the PDL. IKJPARS then branches to PSWVALCK to determine the validity of the PSW address, if the address was specified as an operand.	IKJEGPSW	GOPARSE
2. IKJEGPSW retrieves eight bytes of data from the address specified by the user, or from the most recent RB if the user did not specify an address.	IKJEGPSW	PSWNOSYM
3. IKJEGPSW determines if the PSW is EC or BC mode by checking bit 13: a 0 indicates BC mode; a 1 indicates EC mode. IKJEGPSW formats the PSW according to mode and then branches to IKJEGCVT to convert the data to hexadecimal characters.	IKJEGPSW	PSWCONVT
4. IKJEGPSW branches to IKJEGIO to write the data to a terminal or to a data set.	IKJEGPSW	OUTPUTIT
5. IKJEGPSW issues a return code and returns control to IKJEGMNL by branching to the address specified in Register 14.	IKJEGPSW	RETURN

DIAGRAM 2.18. LISTTCB Subcommand Processor



1. Checks the syntax and validates the subcommand operands.

   See Diagram 2.1

2. Retrieves contents from TCB.

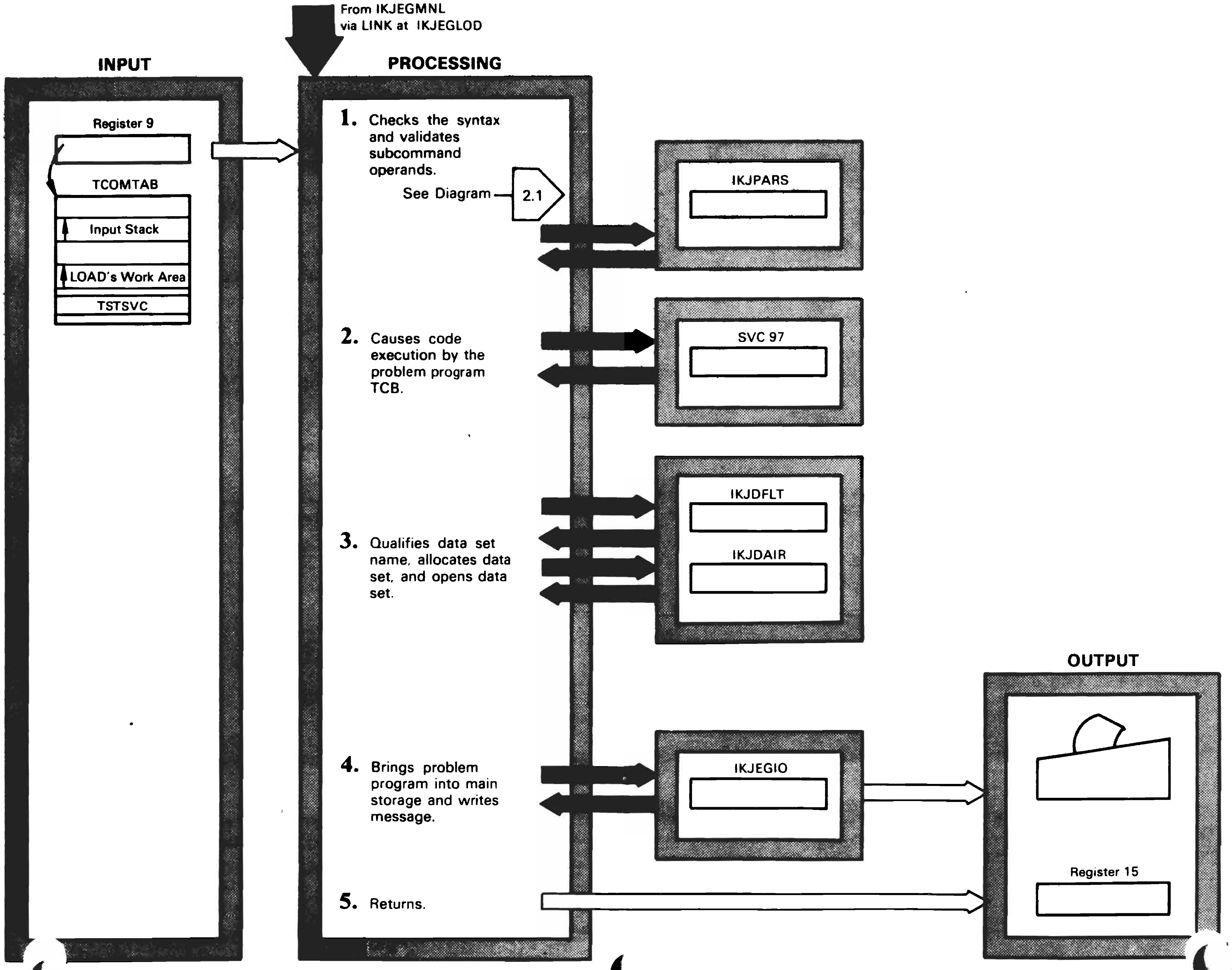
3. Converts data to hexadecimal format.

4. Writes data to a terminal or a data set.

5. Returns.

Description	Module	Label
1. IKJEGTCB invokes IKJPARS to check the symtax of the subcommand operands and to build PDEs in the PDL (one PDE for each TCB field specified by the user). IKJPARS also sets a flag in the PDEs for each field requested by the user. IKJPARS branches to IKJEGVCK to determine the validity of the TCB address specified as an operand.	IKJEGTCB	SETPARSE
2. IKJEGTCB retrieves the contents of the specified fields in the TCB.	IKJEGTCB	MOREROOM
3. IKJEGTCB branches to IKJEGCVT to convert the data to printable hexadecimal characters.	IKJEGTCB	PROTBLD2
4. IKJEGTCB branches to IKJEGIO to write the data to a terminal or to a data set.	IKJEGTCB	TCBIO1
5. IKJEGTCB issues a return code and returns control to IKJEGMNL by branching to the address specified in Register 14.	IKJEGTCB	RETURN

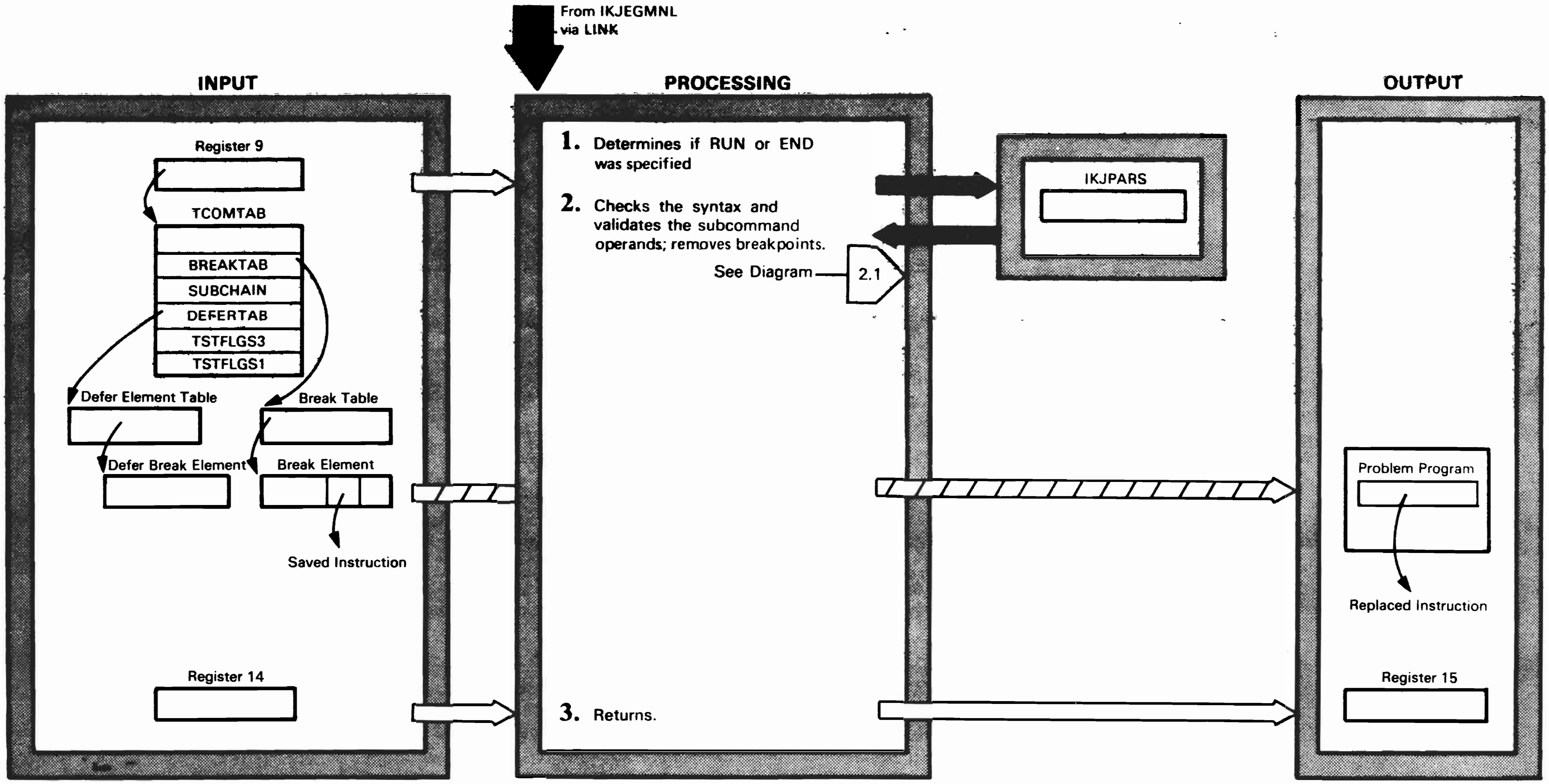
DIAGRAM 2.19. LOAD Subcommand Processor



1. Checks the syntax and validates subcommand operands.
   See Diagram 2.1
2. Causes code execution by the problem program TCB.
3. Qualifies data set name, allocates data set, and opens data set.
4. Brings problem program into main storage and writes message.
5. Returns.

Description	Module	Label
1. IKJEGLDF passes control to IKJPARS to check the syntax of the subcommand operands. IKJPARS then branches to LOADVLCK to determine whether the specified program is a valid member name. Upon completion, IKJPARS returns control to IKJEGLDF.	IKJEGLDF	SETPARM
2. Using the SVC 97 routine, TASKSW, a closed subroutine, causes the execution of IKJEGLDF code to take place under the problem program TCB rather than the TEST TCB. This is accomplished by saving the problem program's registers to be restored later by IKJEGMNL. IKJEGLDF's registers are then placed in SVC 97's SVRB by the SVC 97 routine. The resume address in the problem program PRB is then altered to contain an address within IKJEGLDF. This allows the execution of the LOAD macro instruction to take place under the problem program TCB.	IKJEGLDF	TASKSW
3. IKJDFLT, the Default Naming Service routine, fully qualifies the data set name specified by the user. IKJDAIR, the Dynamic Allocation Interface routine, allocates the specified data set, which is then opened by execution of an OPEN macro instruction.	IKJEGLDF IKJEGLDF IKJEGLDF	A10100 TODAIR A10200
4. IKJEGLDF issues a LOAD macro instruction to bring a usable copy of the problem program into main storage. IKJEGLDF then branches to IKJEGIO to write a message at the terminal, listing the address at which the problem program is loaded.		IKJEGLDF CONTLOAD
5. IKJEGLDF issues a return code and returns control to IKJEGMNL by branching to the SVC 97 instruction pointed to by the TSTSVC field in TCOMTAB.		IKJEGLDF STAEOFF

DIAGRAM 2.20. OFF Subcommand Processor



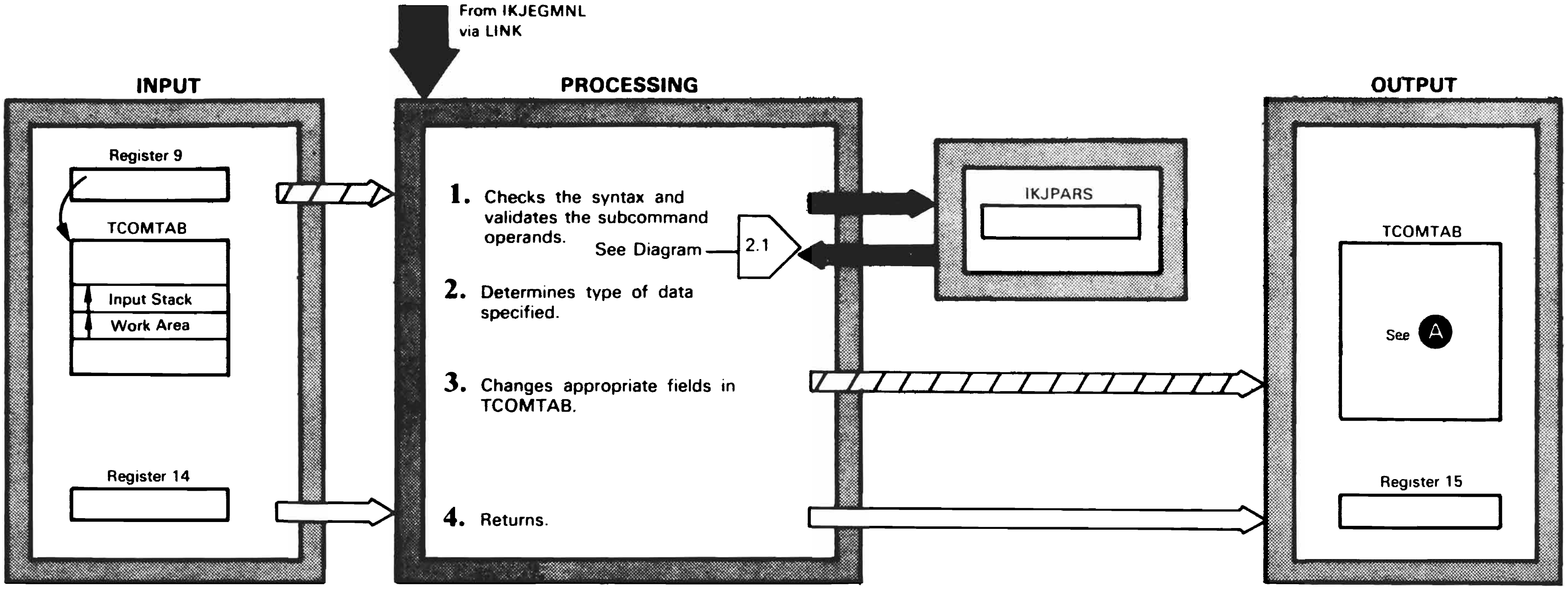
1. Determines if RUN or END was specified

2. Checks the syntax and validates the subcommand operands; removes breakpoints.
   See Diagram 2.1

3. Returns.

Description	Module	Label
Breakpoints are removed as a result of one of the following circumstances: • The terminal user enters an OFF subcommand. • The terminal user enters a RUN or an END subcommand. In this case, IKJEGEND sets the ENDSW in the TSTFLGS1 field in TCOMTAB and transfers control to IKJEGOFF. IKJEGEND takes this action when it prepares to return control to the TMP.	IKJEGOFF	
1. IKJEGOFF determines if a RUN or END subcommand was executed by examining the ENDSW and RUNSW flags in the TSTFLGS1 field of TCOMTAB and the NOPARMS flag in the TSTFLGS3 field.	IKJEGOFF	OFF0000
2. If a RUN or END subcommand was executed, or if an OFF subcommand has no operands, IKJEGOFF removes all breakpoints described in both the break table and the defer table, restores the original instructions, and issues a FREEMAIN macro instruction to free the break table, the defer table, and the subcommand chain.	IKJEGOFF	REMOVALL
If the OFF subcommand does contain operands, IKJEGOFF invokes IKJPARS to check the syntax of the operands. IKJEGOFF then removes the specified breakpoints from either the break table or the defer table and restores the original instructions.	IKJEGOFF	OFF0060
3. IKJEGOFF issues a return code and returns control to IKJEGMNL by branching to the address specified in Register 14.	IKJEGOFF	OFF0320

DIAGRAM 2.21. QUALIFY Subcommand Processor

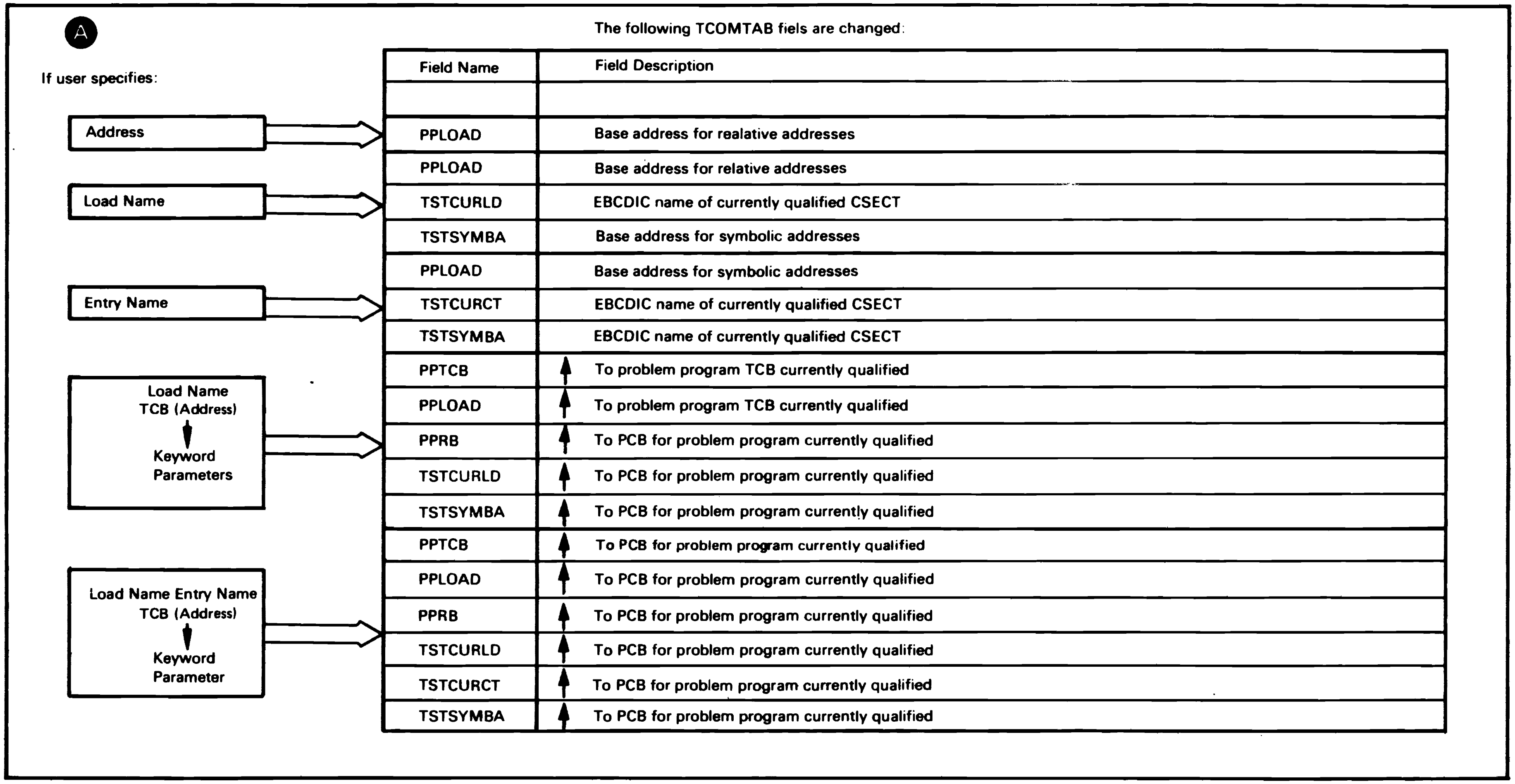


1. Checks the syntax and validates the subcommand operands. See Diagram 2.1

2. Determines type of data specified.

3. Changes appropriate fields in TCOMTAB.

4. Returns.

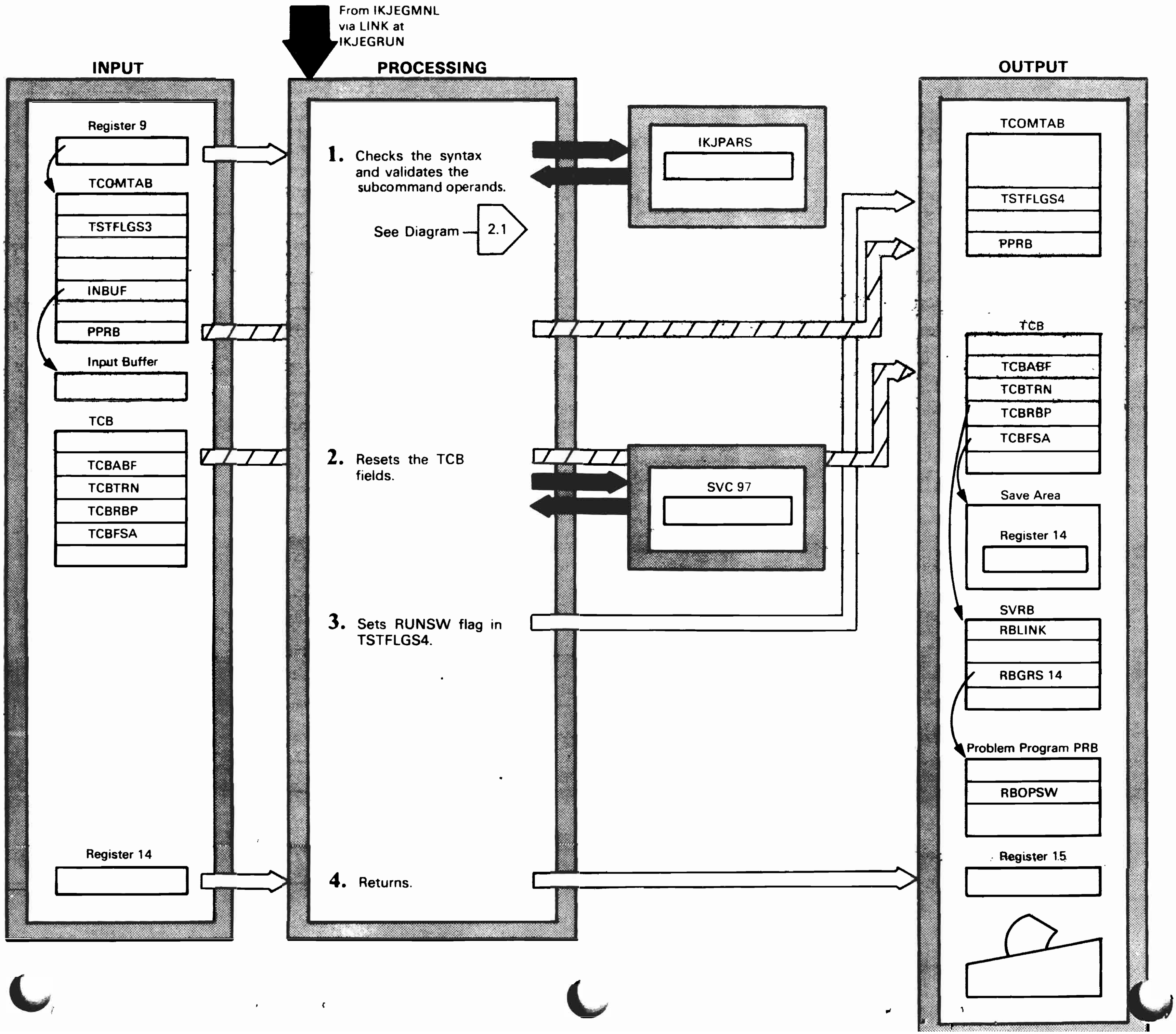

A
If user specifies:
Address
Load Name
Entry Name
Load Name TCB (Address) Keyword Parameters
Load Name Entry Name TCB (Address) Keyword Parameter

The following TCOMTAB fiels are changed:

Field Name	Field Description
PPLOAD	Base address for realative addresses
PPLOAD	Base address for relative addresses
TSTCURLD	EBCDIC name of currently qualified CSECT
TSTSYMBA	Base address for symbolic addresses
PPLOAD	Base address for symbolic addresses
TSTCURCT	EBCDIC name of currently qualified CSECT
TSTSYMBA	EBCDIC name of currently qualified CSECT
PPTCB	↑ To problem program TCB currently qualified
PPLOAD	↑ To problem program TCB currently qualified
PPRB	↑ To PCB for problem program currently qualified
TSTCURLD	↑ To PCB for problem program currently qualified
TSTSYMBA	↑ To PCB for problem program currently qualified
PPTCB	↑ To PCB for problem program currently qualified
PPLOAD	↑ To PCB for problem program currently qualified
PPRB	↑ To PCB for problem program currently qualified
TSTCURLD	↑ To PCB for problem program currently qualified
TSTCURCT	↑ To PCB for problem program currently qualified
TSTSYMBA	↑ To PCB for problem program currently qualified

Description	Module	Label
1. IKJEGQFY passes control to IKJPARS to check the syntax of the operands and to build a PDE for each operand. Upon completion, IKJPARS returns control to IKJEGQFY.	IKJEGQFY	SMTPARS
2. The type of data specified is determined from the PDEFLAGS in the PDEUSER field of the PDE.		
3. For each type of data specified, changes to TCOMTAB are made. Insert A shows the five possible entries and corresponding changes.		
4. IKJEGQFY issues a return code and returns control to IKJEGMNL by branching to an address specified in Register 14.	IKJEGQFY	QFYRET1

DIAGRAM 2.22. RUN Subcommand Processor



1. Checks the syntax and validates the subcommand operands.

   See Diagram 2.1

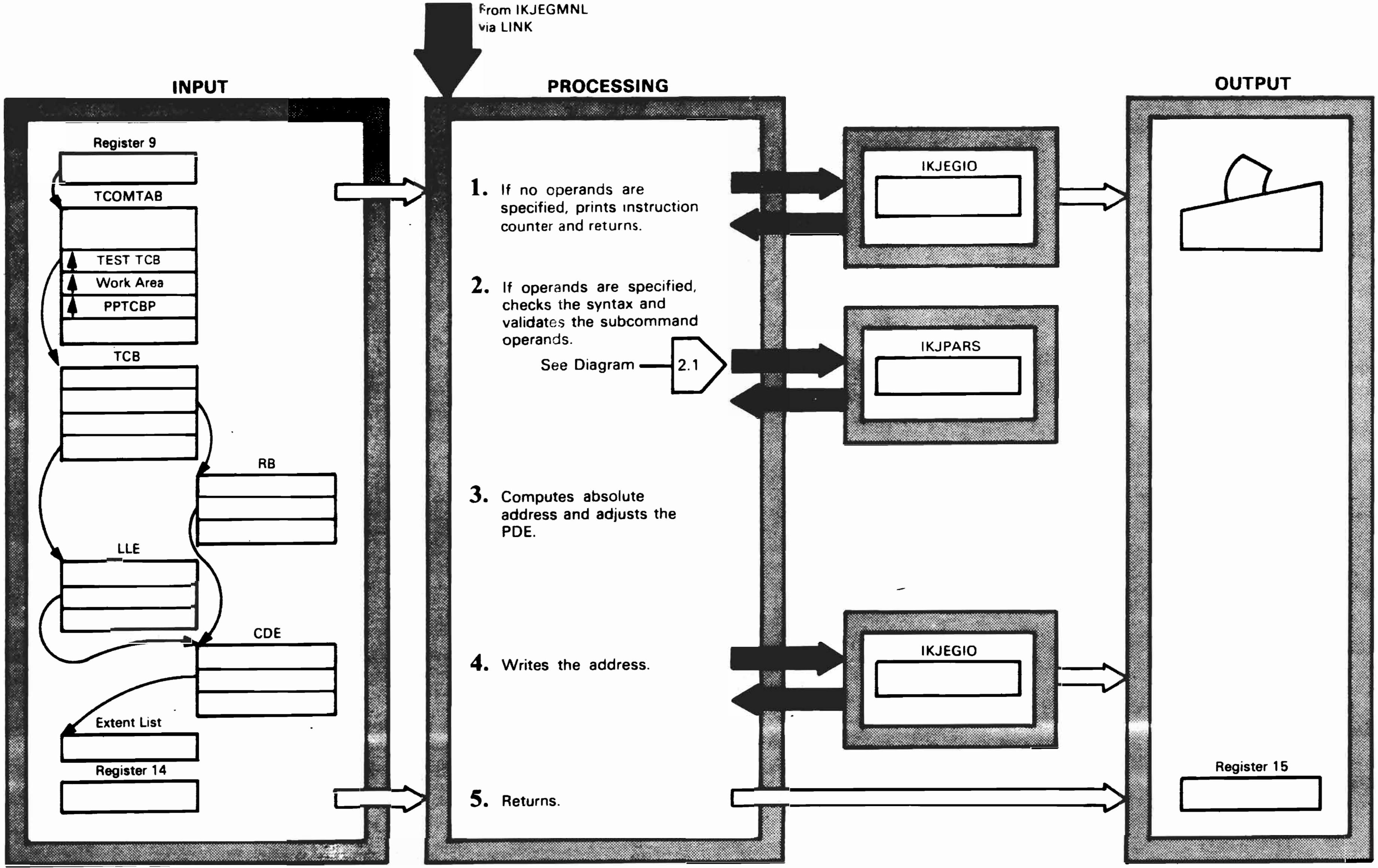
2. Resets the TCB fields.

3. Sets RUNSW flag in TSTFLGS4.

4. Returns.

Description	Module	Label
1. IKJEGGO invokes IKJPARS when the NOPARMS flag in the TSTFLGS3 field in TCOMTAB contains a zero. IKJPARS checks the syntax of the subcommand operands and builds a PDL.	IKJEGGO	PARSBILD
2. IKJEGGO invokes the SVC 97 routine four times to reset TCB fields:	IKJEGGO	PSWSET
• IKJEGGO invokes the SVC 97 routine to change the resume address in the RBOPSW field of the problem program PRB to point to the next sequential instruction (in the case where the program has been stopped at the breakpoint) or to the address specified in the RUN subcommand.		
• IKJEGGO changes Register 14 in the problem program's save area to point to an SVC 3 instruction in the system's CVT and then invokes the SVC 97 routine to reset the problem program's active Register 14 to point to an SVC 3 instruction in the CVT. These changes are made only if these registers point to the SVC 97 routine in the PPEXIT field in TCOMTAB.	IKJEGGO	CKACTIVE
• IKJEGGO invokes the SVC 97 routine to reset to zero the TCBTCP flag in the TCBABF field in all TCBs under TEST.	IKJEGGO	PROCESS
• IKJEGGO invokes the SVC 97 routine to set to zero the TCBTRN field in all TCBs under TEST.	IKJEGGO	PROCESS
The third and fourth changes indicate to the operating system that the task is no longer under TEST.		
3. IKJEGGO sets the RUNSW flag in the TSTFLGS4 field in TCOMTAB.	IKJEGGO	RUNETN
4. IKJEGGO issues a return code and returns control to IKJEGMNL by branching to the address specified in Register 14. After IKJEGMNL has invoked the END subcommand processor to free main storage for TEST, control returns to the TMP, which in turn passes control to the problem program.	IKJEGGO IKJEGMNL	EXIT TMPRTURN

DIAGRAM 2.23. WHERE Subcommand Processor



1. If no operands are specified, prints instruction counter and returns.

2. If operands are specified, checks the syntax and validates the subcommand operands.

   See Diagram 2.1

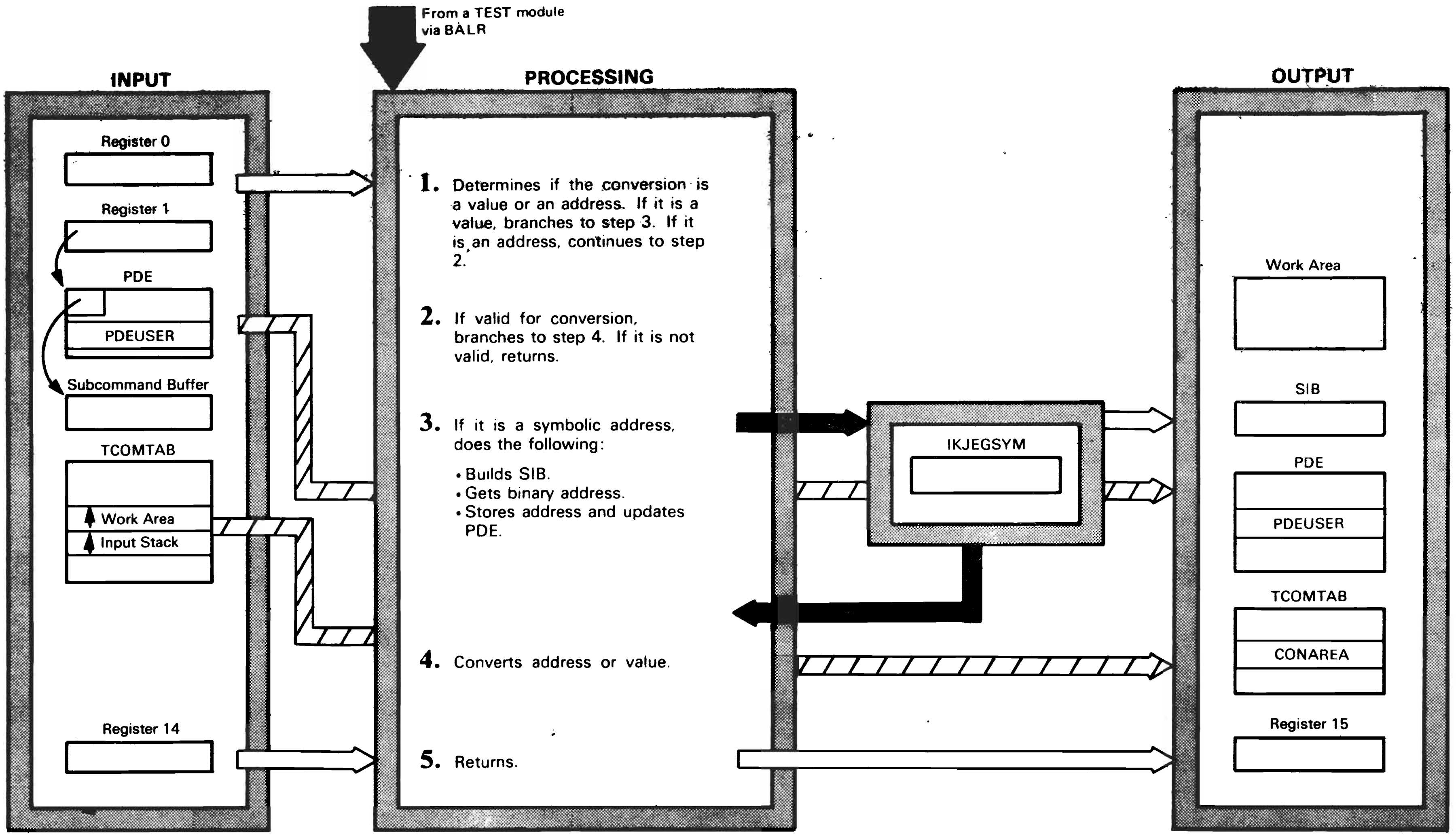
3. Computes absolute address and adjusts the PDE.

4. Writes the address.

5. Returns.

Description	Module	Label
1. If the user has not specified any operands, IKJEGWHR obtains the instruction counter in the current request block and writes it to the terminal using IKJEGIO. The PPTCBP field in TCOMTAB points to the problem program TCB, which contains a pointer to the request block. Return is to IKJEGMNL by a branch to the address specified in Register 14.	IKJEGWHR IKJEGWHR	COMMON RETCALL
2. If operands are specified, IKJEGWHR passes control to IKJPARS to check the syntax of the operands and to build a PDE for each operand. Upon completion, control returns to IKJEGWHR.	IKJEGWHR	SMTPARS
3. IKJEGWHR searches the extent list to determine whether or not a relative or symbolic address falls within the range of addresses associated with a load name. IKJEGCVT converts the address to printable format. When a load name and its associated address are known, IKJEGSYM locates each CSECT name associated with the load name and determines the relative address of the CSECT. IKJEGWHR can then calculate the absolute address by adding the load address of the problem program, the CSECT displacement into the problem program, and the relative address displacement into the CSECT.	IKJEGWHR IKJEGCVT IKJEGSYM IKJEGWHR	ADDRPROC FROMBIN FOFFSET ADDRPROC
4. IKJEGIO writes the requested address to the terminal.	IKJEGWHR	PUTC
5. IKJEGWHR issues a return code and returns control to IKJEGMNL by branching to the address specified in Register 14.	IKJEGWHR	RETCALL

DIAGRAM 2.24. Data Conversion Processing



1. Determines if the conversion is a value or an address. If it is a value, branches to step 3. If it is an address, continues to step 2.

2. If valid for conversion, branches to step 4. If it is not valid, returns.

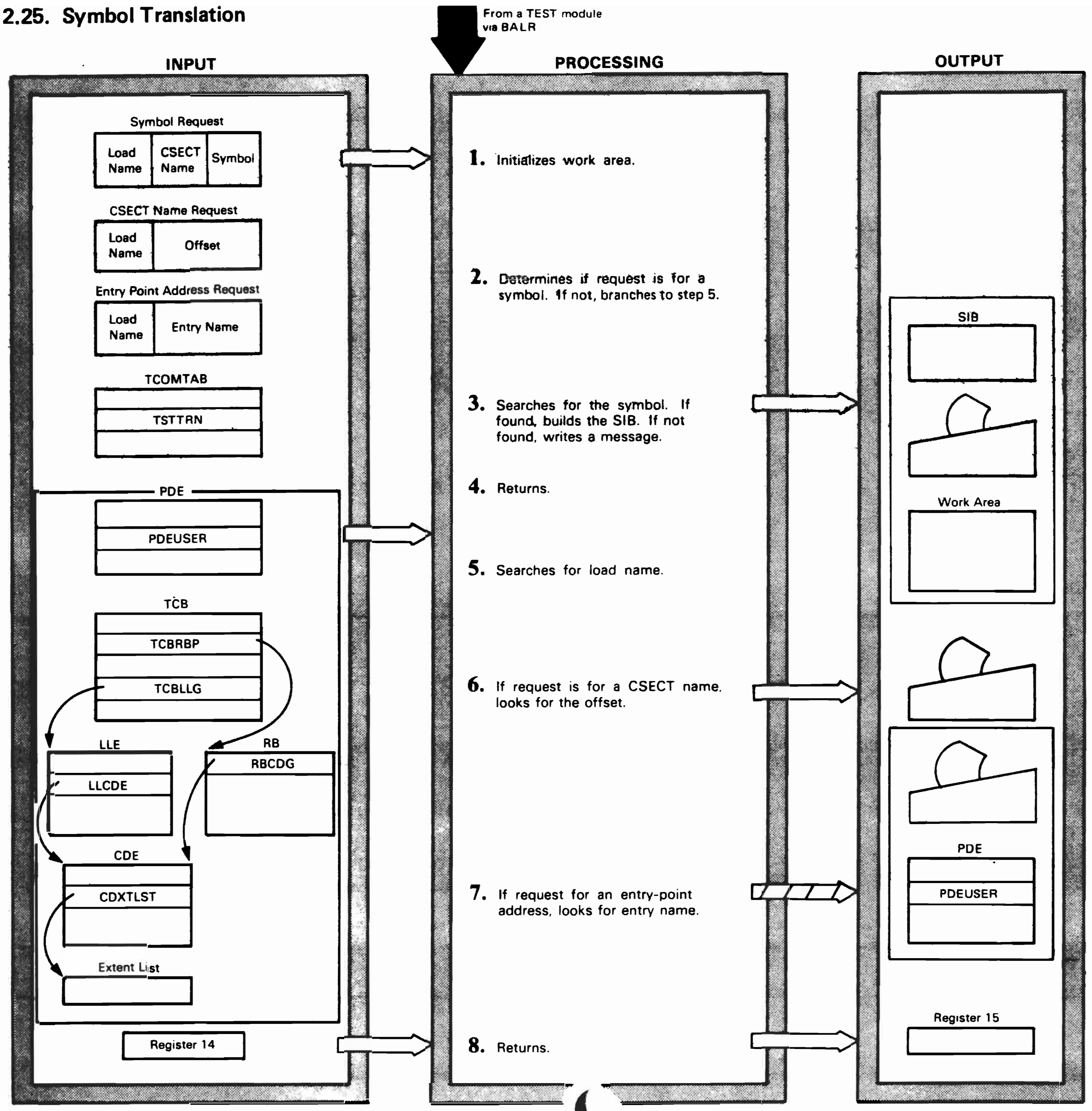
3. If it is a symbolic address, does the following:
   - Builds SIB.
   - Gets binary address.
   - Stores address and updates PDE.

4. Converts address or value.

5. Returns.

Description	Module	Label
Two modules, IKJEGCVT and IKJEGSYM, convert data. Certain TEST subcommand processors must convert input data to machine format before they can process the value. Similarly, some subcommand processors need to convert data to printable form to list the value in a message to the terminal user or to include the value in a data set.		
1. Upon entry, IKJEGCVT checks Register 1 to determine if the conversion is for a value or for an address. If Register 1 is positive, the conversion is for a value. If Register 1 is negative, the conversion is for an address.	IKJEGCVT	CVTBEGIN
2. IKJEGCVT examines the flags in the input PDE to determine if the data is acceptable for conversion. If it is not, an error message is sent to the terminal and control returns to the caller.	IKJEGCVT	VALCVT
3. If the address is a symbolic address, control passes to IKJEGSYM via a LINK macro instruction. IKJEGSYM builds a symbol information block (SIB) and puts the machine address of the symbol into it. The PDEUSER field of the PDE is updated to point to the SIB. Upon completion, control returns to IKJEGCVT.	IKJEGCVT	SYMLINK
4. The address or value is converted and put in one of two places. If converted to binary, the address is put in the PDEUSER field. All other data is placed in a 32-byte work area pointed to by the CONAREA field in TCOMTAB.	IKJEGSYM	GETINFO
5. IKJEGCVT issues a return code and returns control to the caller by branching to the address specified in Register 14.	IKJEGSYM	TOCALLER

DIAGRAM 2.25. Symbol Translation



1. Initializes work area.

2. ~~Determines~~ if request is for a symbol. If not, branches to step 5.

3. Searches for the symbol. If found, builds the SIB. If not found, writes a message.

4. Returns.

5. Searches for load name.

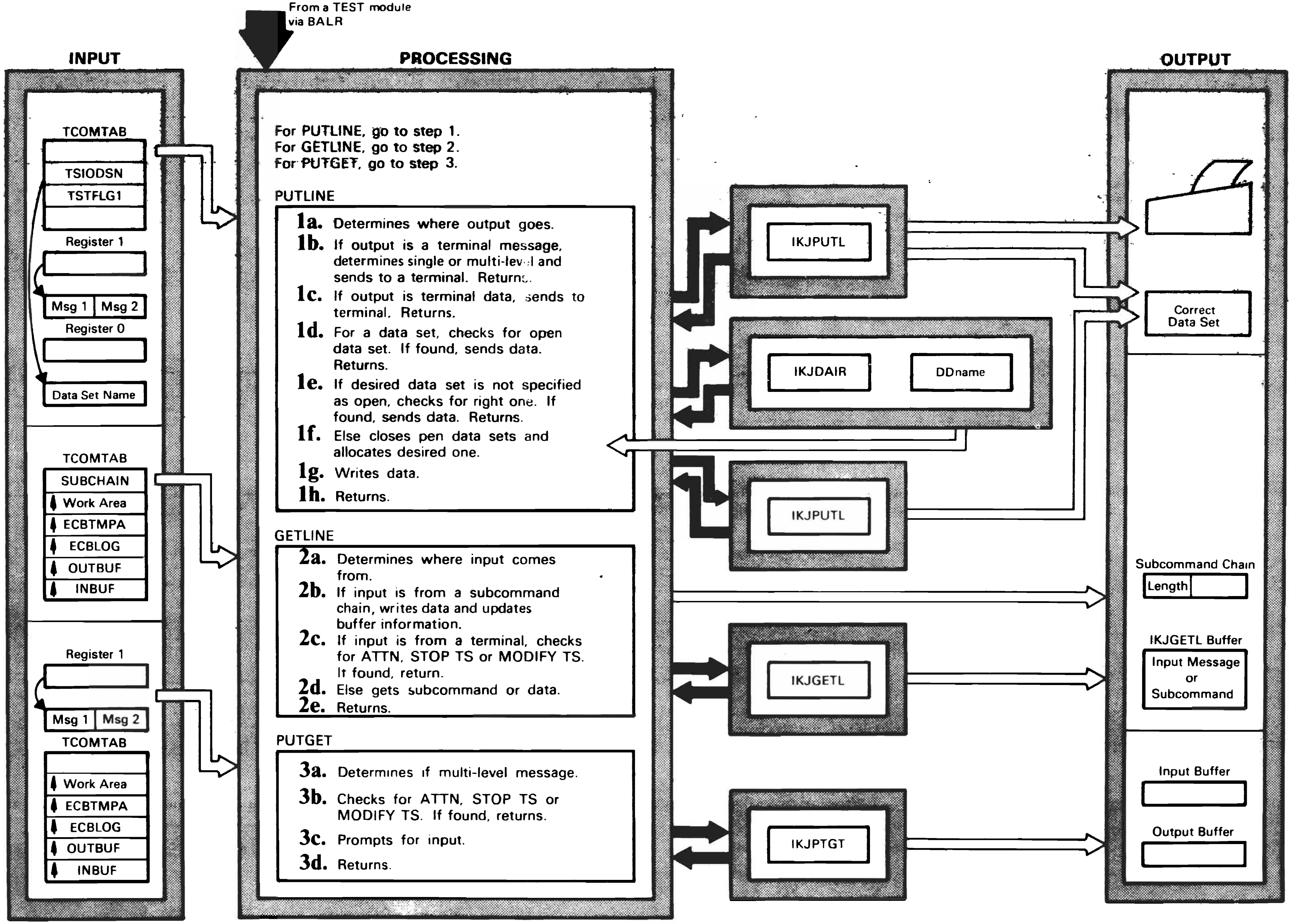
6. If request is for a CSECT name, looks for the offset.

7. If request for an entry-point address, looks for entry name.

8. Returns.

Description	Module	Label
IKJEGSYM translates an internal symbol, a module offset, or an entry name into an absolute main storage address or a CSECT name. The symbol offset or entry name pertains to the problem program and are specified by the user in the QUALIFY or WHERE subcommand or in any subcommand using the IKJEGCVT module. **1.** A work area of 800 bytes is obtained. The list form of macros and the DCB are brought into the work area.	IKJEGSYM	IKJEGSYM
2. IKJEGSYM determines if the request is for a symbol by examining PDEFLG4 in the PDE. If the flag is set, the request is for a symbol. If it is not set, IKJEGSYM branches to step 5.	IKJEGSYM	SMTSTAE
3. The symbol table built by the EQUATE subcommand processor is searched for the symbol name. If the symbol name is found a symbol information block (SIB) containing symbol attributes and addresses is built, and the SIB address is put into the PDE. If the symbol is not found, an error message is written to the terminal.	IKJEGSYM IKJEGSYM	FOUNDSY FOFFSET
4. Control returns to the caller.	IKJEGSYM	TOCALLER
5. The SVC information blocks are searched for a specified or currently qualified load name to get the information needed to ready the data set. SVC blocks are found through a pointer in the TSTTRN field in TCOMTAB.	IKJEGSYM	LE
6. If the request is not for a symbol, input from the parameter descriptor list is checked to determine if the request is for a CSECT name or an entry-point address. If offset is found, indicating request for CSECT name, it is placed in the work area and return is to the caller. If not found, an error message is written to the terminal and return is to the caller.	IKJEGSYM	FOFFSET
7. When the request is for an entry-point address, the CESD is searched for the entry name. If the entry name is found, the entry-point address associated with it is placed in the PDE. If it is not found, and error message is written to the terminal.	IKJEGSYM	FINDCESD
8. IKJEGSYM issues a return code and returns control to the caller by branching to the address specified in Register 14.	IKJEGSYM	TOCALLER

DIAGRAM 2.26. I/O Processing



For PUTLINE, go to step 1.
For GETLINE, go to step 2.
For PUTGET, go to step 3.

PUTLINE

1a. Determines where output goes.

1b. If output is a terminal message, determines single or multi-level and sends to a terminal. Returns.

1c. If output is terminal data, sends to terminal. Returns.

1d. For a data set, checks for open data set. If found, sends data. Returns.

1e. If desired data set is not specified as open, checks for right one. If found, sends data. Returns.

1f. Else closes pen data sets and allocates desired one.

1g. Writes data.

1h. Returns.

GETLINE

2a. Determines where input comes from.

2b. If input is from a subcommand chain, writes data and updates buffer information.

2c. If input is from a terminal, checks for ATTN, STOP TS or MODIFY TS. If found, return.

2d. Else gets subcommand or data.

2e. Returns.

PUTGET

3a. Determines if multi-level message.

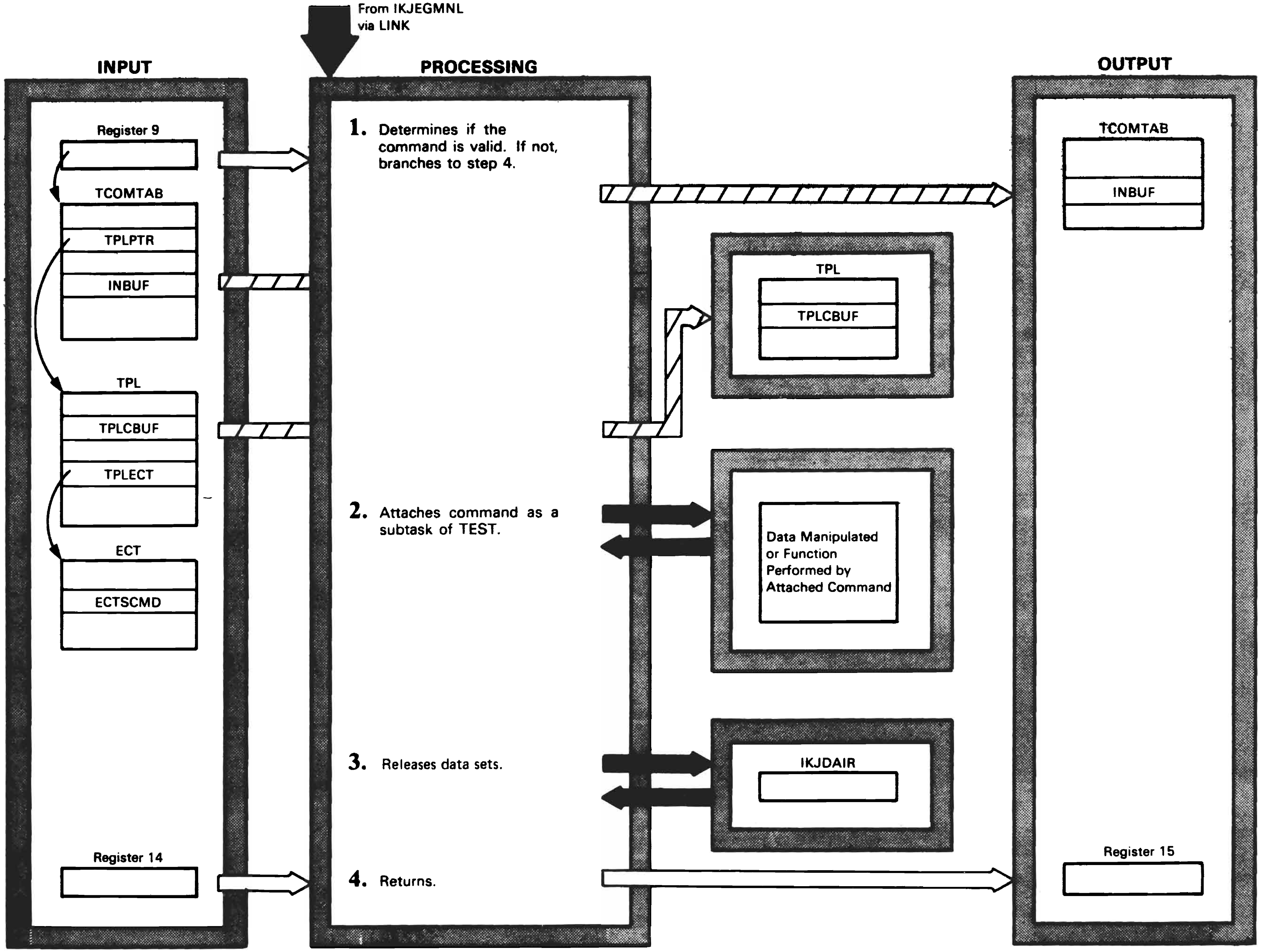
3b. Checks for ATTN, STOP TS or MODIFY TS. If found, returns.

3c. Prompts for input.

3d. Returns.

Description	Module	Label
IKJEGIO, the I/O module, handles three kinds of input/output requests: getting input from the terminal, processing input request from a TEST module, or sending output to either a terminal or to a data set. There are three entry points -- IKJEGPT, IKJEGGT, and IKJEGPG -- one for each type of request. An I/O parameter list is built in the work area. **1a.** If the request is for a PUTLINE macro instruction, IKJEGPT checks the TSTFLG1 field in TCOMTAB to find out if the input is for a terminal or for a data set. If the TSTPRINT flag is set, the output goes to a data set. If it is not set, output is for the terminal.	IKJEGIO	PUTLINE
1b. If the output is a terminal message, Register 1 contains a negative address and points to a double-word. A zero in the second word indicates the message is not multi-level. A PUTLINE macro instruction sends the message to the terminal. Control returns to the caller with a return code in Register 15 to indicate success or failure.	IKJEGIO	A00901
1c. If the output is terminal data, Register 1 contains a positive address. A PUTLINE macro instruction sends the data to the terminal. Control returns to the caller.	IKJEGIO	A00901
1d. When the output is for a data set and the data set is open to receive data, Register 0 contains a zero. The TSIODSN field in TCOMTAB contains a pointer to the data set. A PUT macro instruction sends the output. Control returns to the caller.	IKJEGIO	A01202
1e. If Register 0 contains a data set pointer, the data sets are compared to see if the desired one is open. If it is, a PUT macro instruction sends the output to the data set. Control returns to the caller.	IKJEGIO	A01400
1f. The open data set is closed if it is not the specified one. The specified data set name is fully qualified using the IKJDFLT routine, if necessary. The IKJDAIR routine allocates the specified data set. If no data set exists, IKJDAIR requests allocation with a disposition of NEW. A DDname is returned from IKJDAIR.	IKJEGIO	A01510
1g. The DCB is then opened. Output is written to the specified data set.	IKJEGIO	A01701
1h. Control returns to the caller.	IKJEGIO	A10000
2a. IKJEGGT determines if the input is from the terminal or from the subcommand chain by examining the SUBCHAIN field in TCOMTAB. If the field is zero, input is to come from the terminal.	IKJEGIO	IKJEGGT
2b. If input is to come from the subcommand chain, the SUBCHAIN field is updated, the length field in the subcommand buffer is updated, and the INBUF field in TCOMTAB is changed to the address of the new buffer. The return code is set and control returns to the caller.	IKJEGIO	A00520
2c. If the input is to come from the terminal, the ECBTMPA and ECBLOG fields of TCOMTAB are checked to determine if STOP TS, MODIFY TS or an attention interruption has been issued. If STOP TS or MODIFY TS was issued, return is to the TMP to cause the current user to be logged off. If an attention interruption was issued, IKJEGMNL handles a new subcommand or returns control to the TMP if the user wants to stop the session.	IKJEGIO	A80000
2d. If none of the above takes place, a GETLINE macro instruction is issued to obtain more input from the terminal.	IKJEGIO	A00700
2e. Control returns to the caller.	IKJEGIO	A10000
3a. Register 6 is checked to determine if the message is multi-level.	IKJEGIO	PUTGET
3b. If STOP TS, MODIFY TS, or an attention interruption has been issued, control returns to the caller.	IKJEGIO	A80000
3c. If none of the above takes place, a PUTGET macro instruction prompts the terminal user and obtains a new subcommand.	IKJEGIO	PUTGET
3d. Control returns to the caller.	IKJEGIO	A10000

DIAGRAM 2.27. Invoking the HELP Command



1. Determines if the command is valid. If not, branches to step 4.

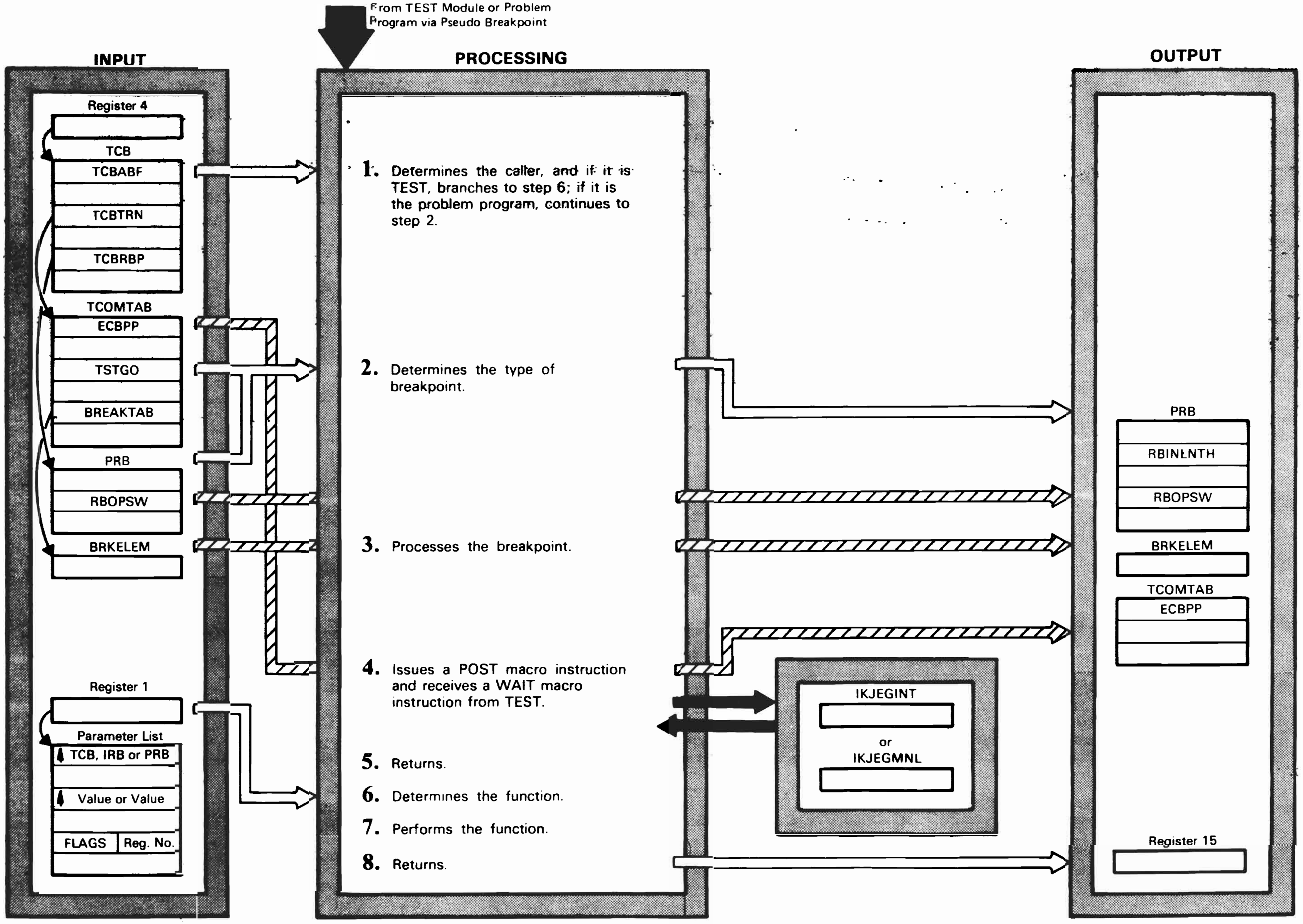
2. Attaches command as a subtask of TEST.

3. Releases data sets.

4. Returns.

Description	Module	Label
1. IKJEGCIV, the HELP command invoker, performs a BLDL on the specified command to determine if the command is valid. If the command is valid, IKJEGCIV moves the contents of the INBUF field in TCOMTAB to the TPLCBUF field in the TPL and zeroes the INBUF field. If the command is not valid, IKJEGCIV branches to step 4.	IKJEGCIV	CIV03010
2. IKJEGCIV issues an ATTACH macro instruction to make the command a subtask of TEST. The specified command manipulates data or performs its function and returns control to IKJEGCIV.	IKJEGCIV	CIV04010
3. IKJEGCIV invokes IKJDAIR to release the data sets used by the command.	IKJEGCIV	CIV04030
4. IKJEGCIV issues a return code and returns control to IKJEGMNL by branching to the address specified in Register 14.	IKJEGCIV	CIV07030

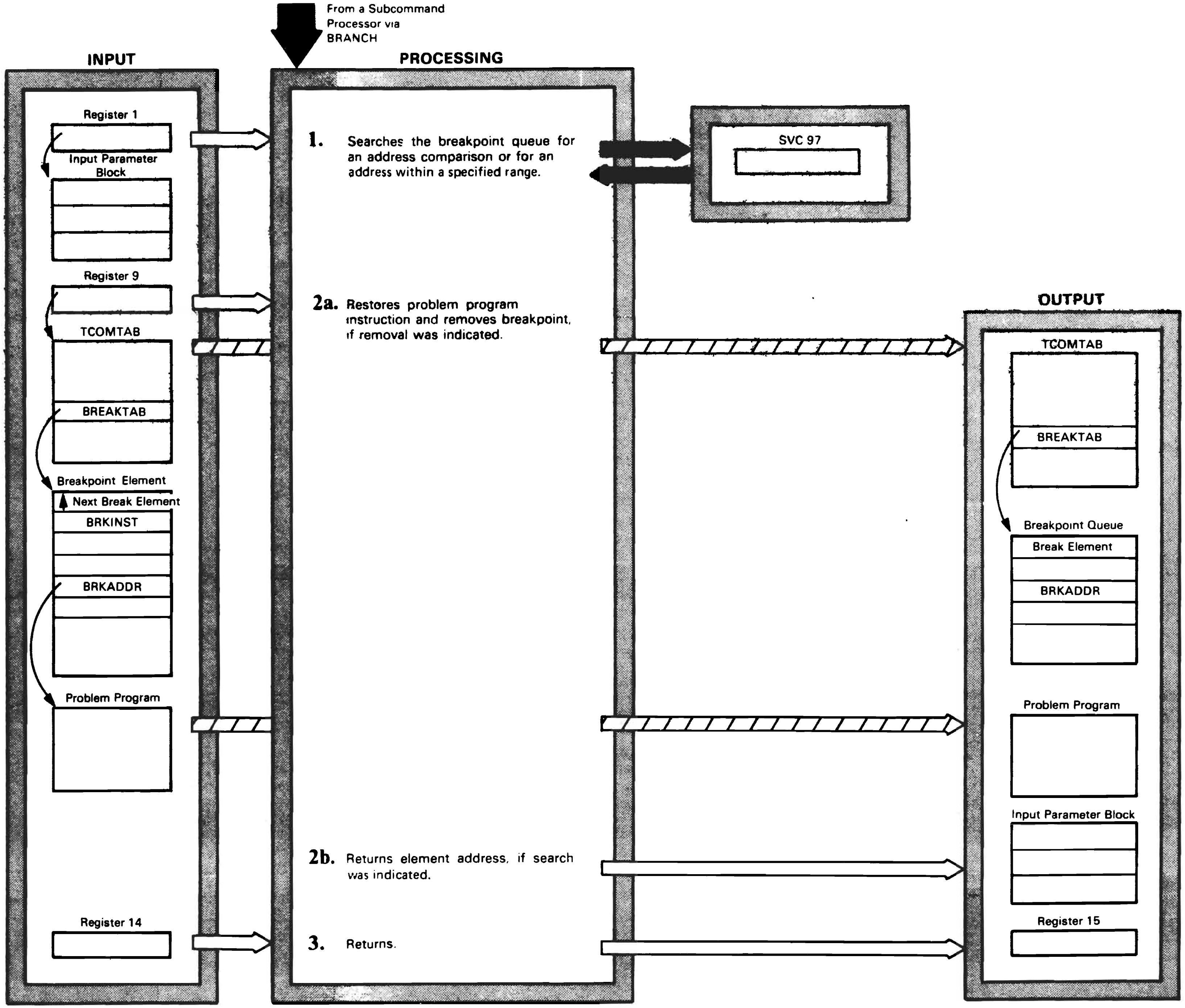
DIAGRAM 2.28. SVC 97 Processing



1. Determines the caller, and if it is TEST, branches to step 6; if it is the problem program, continues to step 2.
2. Determines the type of breakpoint.
3. Processes the breakpoint.
4. Issues a POST macro instruction and receives a WAIT macro instruction from TEST.
5. Returns.
6. Determines the function.
7. Performs the function.
8. Returns.

Description	Module	Label
1. IGC0009G, the SVC 97 routine, is used as a breakpoint handler by any module of the problem program and as a subroutine of TEST by any module of the TEST program. If the problem program is the caller, it continues to step 2; if TEST is the caller, it branches to step 6.	IGC0009G	QPSEUDO
2. When used as a breakpoint handler by the problem program, IGC0009G determines whether a pseudo breakpoint or a user breakpoint has been inserted. If a pseudo breakpoint was inserted, the resume address is updated to the next sequential instruction.	IGC0009G	TSTPOST
3. IGC0009G processes the breakpoint and updates BRKELEM if necessary. IGC0009G interprets BAL and BALR instructions.	IGC0009G	BRKSRCH
4. IGC0009G issues a POST macro instruction to IKJEGINT or IKJEGMNL. TEST issues a WAIT macro instruction to IGC0009G.	IGC0009G	GO
5. IGC0009G exits to the restart address of the problem program.	IGC0009G	RETURN1
6. When used as a subroutine of TEST, IGC0009G determines the service routine function to be performed (for example, updating a floating point register).	IGC0009G	S9G12010
7. IGC0009G performs the requested function.		
8. IGC0009G issues a return code and returns control to the calling TEST module.	IGC0009G	S9G23010

DIAGRAM 2.29. SEARCH Module



1. Searches the breakpoint queue for an address comparison or for an address within a specified range.

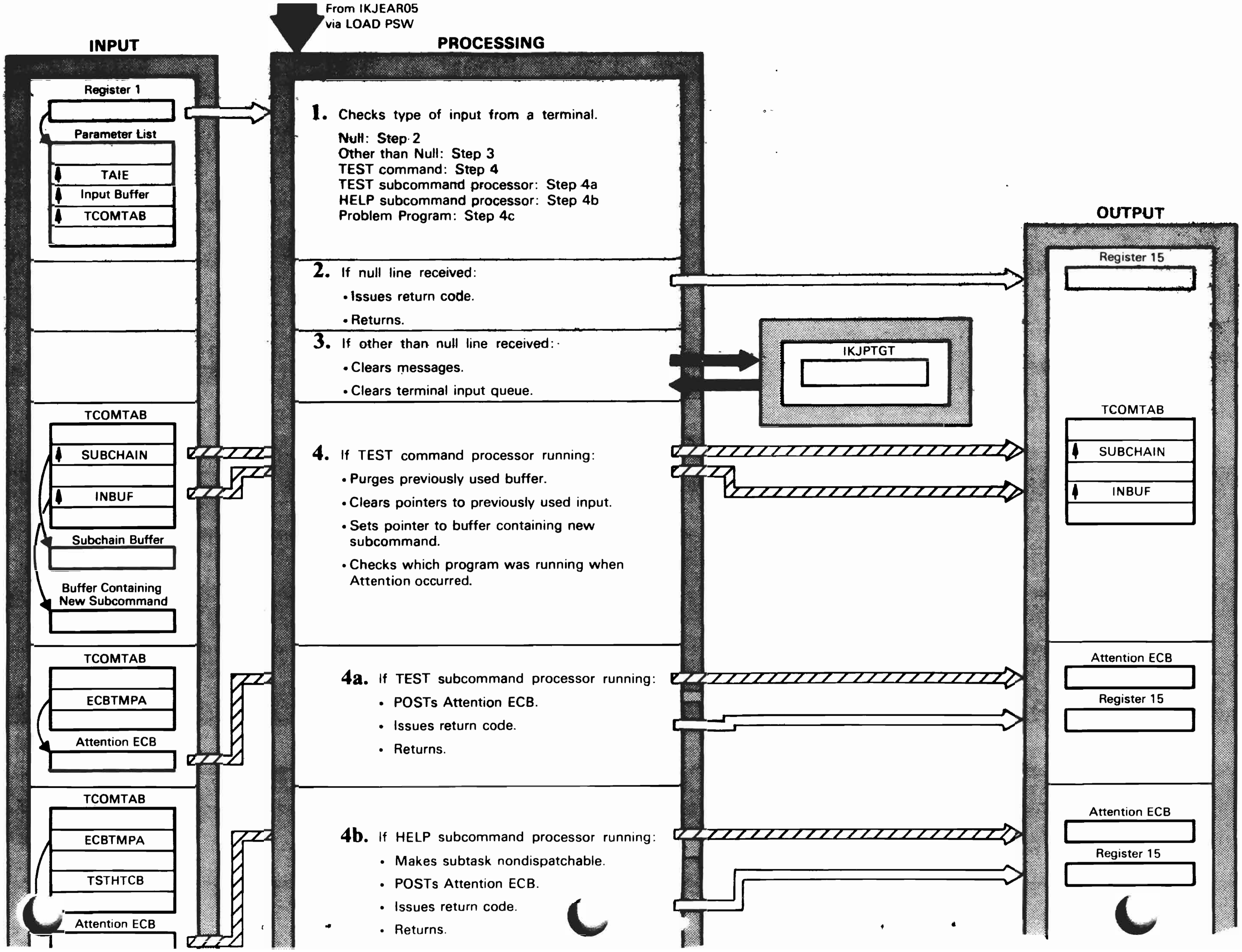
2a. Restores problem program instruction and removes breakpoint, if removal was indicated.

2b. Returns element address, if search was indicated.

3. Returns.

Description	Module	Label
1. IKJEGSRH is invoked by a subcommand processor to perform search functions based upon the contents of the three-word input parameter block pointed to by Register 1, which has the following contents:	IKJEGSRH	
• The first word contains flags. If the first word contains 00, IKJEGSRH searches for an address or range and deletes the breakpoint elements. If the first word contains 04, IKJEGSRH searches for an address and returns the address of the corresponding breakpoint queue element. If the first word contains 08, IKJEGSRH searches for an address or range and deletes the breakpoint element after backing up five bytes from the beginning of the range to include instruction overlap.		SRH0020
• The second word is an address of the two bytes containing the breakpoint or start of a range. • The third word is the end of a range or zero.	IKJEGSRH	SRH0340
IKJEGSRH invokes the SVC 97 routine to check the validity of the given addresses. Then it searches the active breakpoint queue for a match or for an address within the specified range.		SRH0360 (Range) SRH0100 (Single)
2a. IKJEGSRH tries to restore the saved problem program instruction from the BRKINST field in the break element. If removal was indicated, IKJEGSRH removes the breakpoint from the queue, and a FREEMAIN macro instruction is issued to free the queue element being removed.	IKJEGSRH	SRH0340
2b. If only search was indicated, the address of the element is returned to the caller in the second word of the input parameter block.	IKJEGSRH	SRH0180
3. IKJEGSRH issues a return code and returns control to the calling subcommand processor by branching to the address specified in Register 14.	IKJEGSRH	SRH0500

DIAGRAM 2.30. Processing an Attention Request



1. Checks type of input from a terminal.

Null: Step 2
Other than Null: Step 3
TEST command: Step 4
TEST subcommand processor: Step 4a
HELP subcommand processor: Step 4b
Problem Program: Step 4c

2. If null line received:

- Issues return code.
- Returns.

3. If other than null line received:

- Clears messages.
- Clears terminal input queue.

4. If TEST command processor running:

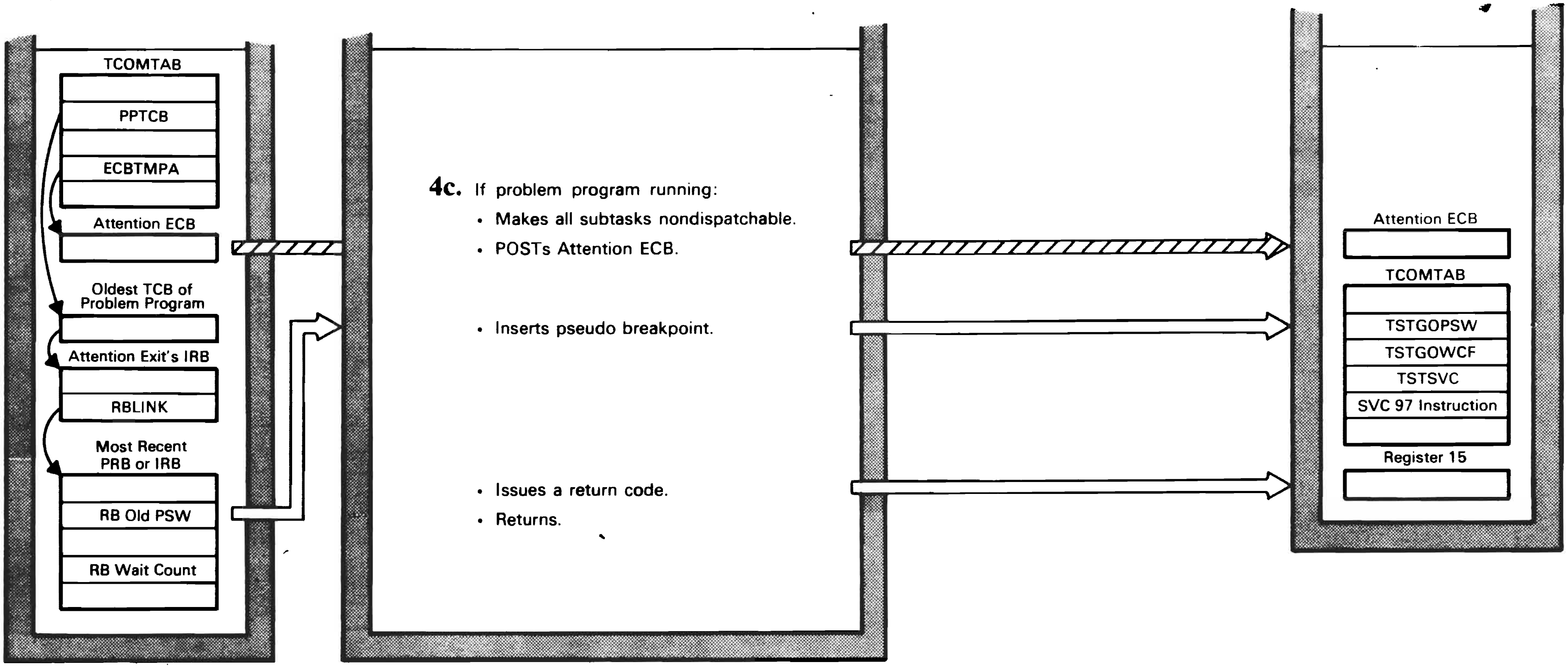
- Purges previously used buffer.
- Clears pointers to previously used input.
- Sets pointer to buffer containing new subcommand.
- Checks which program was running when Attention occurred.

4a. If TEST subcommand processor running:

- POSTs Attention ECB.
- Issues return code.
- Returns.

4b. If HELP subcommand processor running:

- Makes subtask nondispatchable.
- POSTs Attention ECB.
- Issues return code.
- Returns.

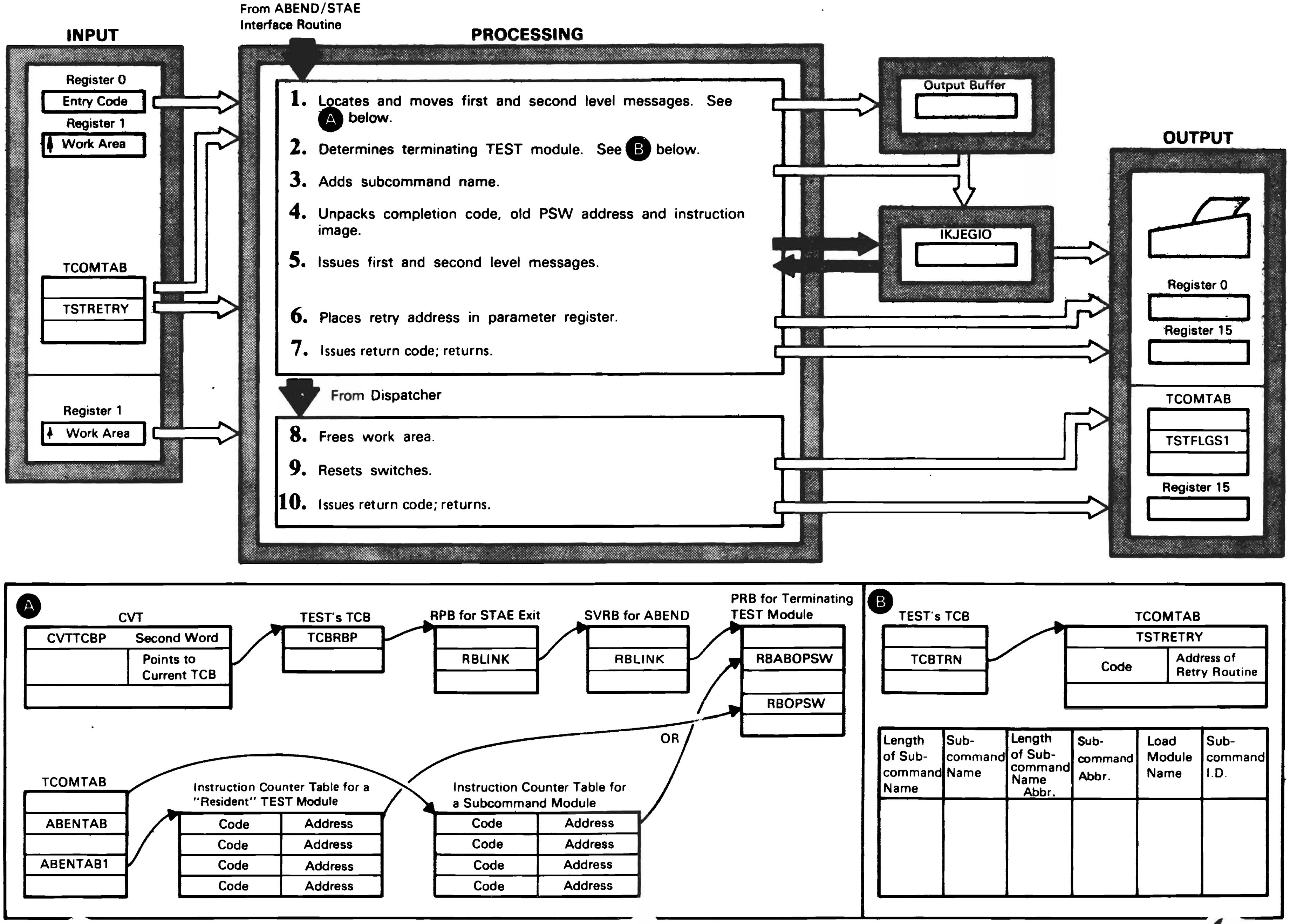


4c. If problem program running:

- Makes all subtasks nondispatchable.
- POSTs Attention ECB.
- Inserts pseudo breakpoint.
- Issues a return code.
- Returns.

Description	Module	Label
1. IKJEGATN examines the TAIE field in the parameter list to determine what kind of input was entered from the terminal: null, other than null, or the TEST command.	IKJEGATN	"UPON Entry"
2. If a null entry was made, IKJEGATN issues a return code and returns control to IKJEGMNL.	IKJEGATN	ATN03010
3. If the entry was other than null, IKJEGATN invokes IKJPTGT to clear any existing second level messages and to clear the terminal input queue.	IKJEGATN	ATN01050
4. If the entry was the TEST command processor, IKJEGATN purges the previously used buffer, clears pointers to previously used input, and sets a pointer to the buffer containing the new subcommand. IKJEGATN examines ECBTST to determine whether the TEST command or the problem program was running when the attention interruption occurred and examines TSTHTCB to determine if the HELP subcommand was running when the attention interruption occurred.	IKJEGATN	ATN01070
4a. If a TEST subcommand processor was running when the attention interruption occurred, IKJEGATN issues a POST macro instruction to set the attention ECB, issues a return code, and returns control to IKJEGMNL.	IKJEGATN	ATN01090
4b. If the HELP subcommand processor was running when the attention interruption occurred, IKJEGATN issues a STATUS STOP macro instruction for each subtask, issues a POST macro instruction to set the attention ECB, issues a return code, and returns control to IKJEGMNL.	IKJEGATN	ATN02010
4c. If the problem program was running when the attention interruption occurred, IKJEGATN issues a STATUS STOP macro instruction for each subtask, issues a POST macro instruction to set the attention ECB, inserts a pseudo breakpoint, issues a return code, and returns control to IKJEGMNL.	IKJEGATN	ATN02030

DIAGRAM 2.31. Processing an Abnormal Condition in TEST



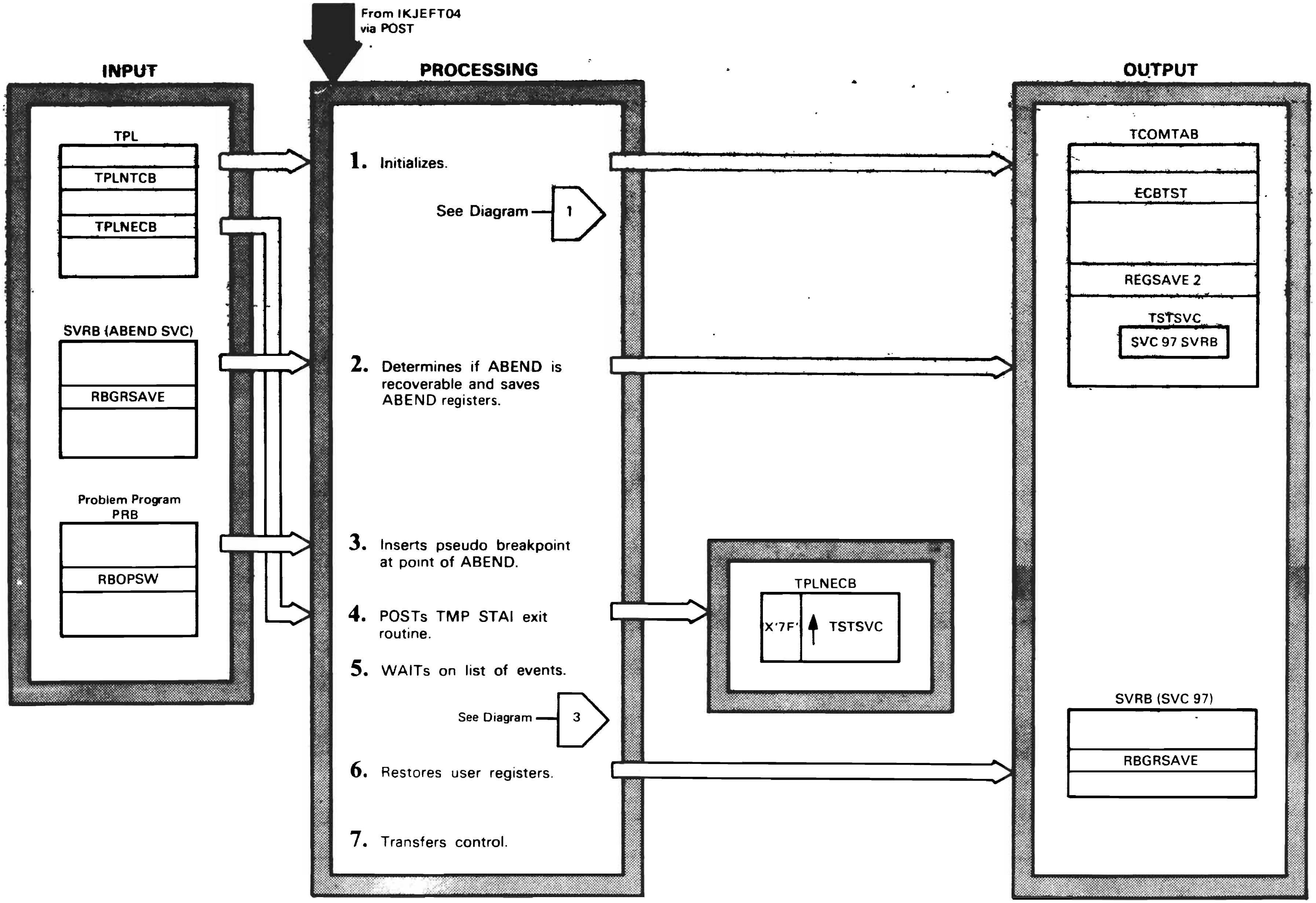
1. Locates and moves first and second level messages. See Ⓐ below.
2. Determines terminating TEST module. See Ⓑ below.
3. Adds subcommand name.
4. Unpacks completion code, old PSW address and instruction image.
5. Issues first and second level messages.
6. Places retry address in parameter register.
7. Issues return code; returns.

From Dispatcher

8. Frees work area.
9. Resets switches.
10. Issues return code; returns.

Description	Module	Label
1. IKJEGSTA, TEST's STAE exit routine, locates the first and second level message inserts belonging to the TEST module that caused the ABEND. IKJEGSTA compares the address fields RBABOPSW or RBOPSW with each address in the appropriate IC table until a match is found. The code found in the table entry is a displacement to a list of message inserts in the message CSECT, IKJEGMS1, in IKJEGSTA. IKJEGSTA then moves the first and second level messages to the output buffer.	IKJEGSTA IKJEGSTA	GETAPSW INSERT
2. IKJEGSTA determines the terminating TEST module name by comparing the code in the first byte of the TSTRETRY field in TCOMTAB with the ID in each entry of the subcommand name table.	IKJEGSTA	SEARCH
3. If a match is found, IKJEGSTA places the subcommand name in the first level message. If the code in TSTRETRY is 0, no subcommand is active, and "TEST" is used as a message insert.	IKJEGSTA IKJEGSTA	MATCH RETURN
4. IKJEGSTA unpacks the completion code, the old PSW address, and the instruction image (the twelve bytes preceding the old PSW address).		
5. IKJEGSTA invokes IKJEGIO to issue the first and second level messages.	IKJEGSTA	MSGIO
6. IKJEGSTA obtains the address of the retry routine from the TSTRETRY field in TCOMTAB and places it in the parameter register for use by the ABEND/STAE Interface routine.	IKJEGSTA	RETURN
7. IKJEGSTA issues a return code to indicate to the ABEND/STAE Interface routine that control should be given to the retry routine.	IKJEGSTA	RETURN
8. A typical retry routine first frees the 104 byte STAE work area, if it exists.		
9. The routine then resets critical switches in TCOMTAB, including the TSTPRINT and TSTFIRST portions of the TSTFLGS1 field.		
10. The routine then issues a return code and passes control to IKJEGMNL.		

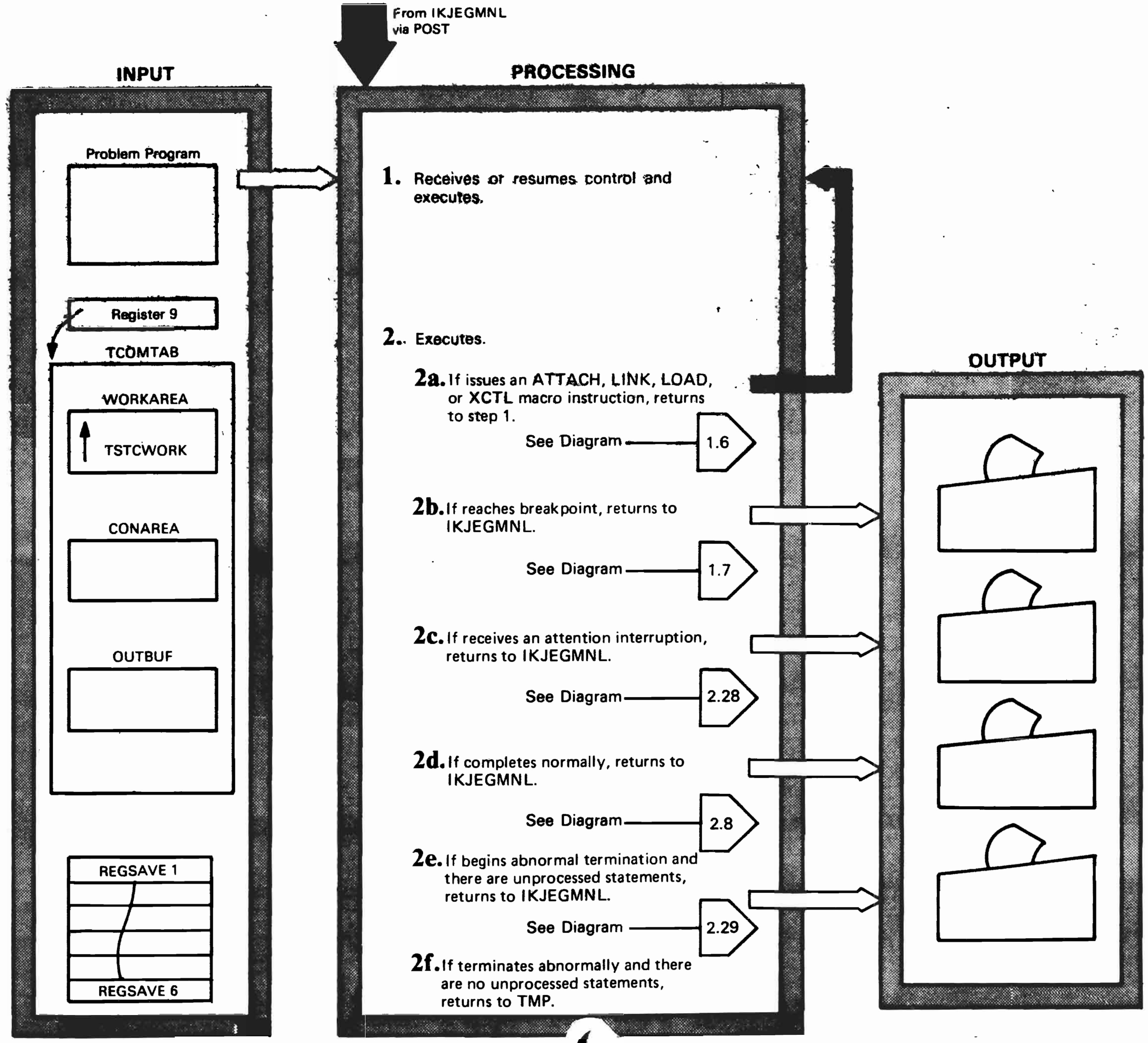
DIAGRAM 2.32. Processing an Abended TMP Subtask



1. Initializes.
2. Determines if ABEND is recoverable and saves ABEND registers.
3. Inserts pseudo breakpoint at point of ABEND.
4. POSTs TMP STAI exit routine.
5. WAITs on list of events.
6. Restores user registers.
7. Transfers control.

Description	Module	Label
1. IKJEGINT gains control from TMP's STAI Exit Routine when a subtask has terminated abnormally. IKJEGINT obtains memory for TCOMTAB, work areas, and buffers and initializes values in them.	IKJEGINT	INT01010
2. IKJEGINT determines that an abnormal termination has occurred by testing TPLNTCB in the TPL. If this field is not zero, IKJEGINT locates the ABEND SVRB, for the terminated task, and saves the registers in REGSAVE2, pointed to by TCOMTAB.	IKJEGINT	INTCONT1
3. IKJEGINT invokes the SVC 97 routine to update the terminated program's resume address to point to TSTSVC in TCOMTAB, a pseudo breakpoint.	IKJEGINT	INT01030
4. After the pseudo breakpoint is inserted, IKJEGINT issues a POST macro instruction to the TMP to allow the abended program to continue.	IKJEGINT	INT00010
5. IKJEGINT waits on a list of events to occur (see Diagram 3).	IKJEGINT	WAIT
6. When the SVC 97 routine posts IKJEGINT indicating a pseudo breakpoint, IKJEGINT invokes the SVC 97 routine to restore the problem program's registers from REGSAVE2.	IKJEGINT	AOK
7. When IKJEGINT processing is complete, IKJEGINT issues an XCTL macro to transfer control to IKJEGMNL.	IKJEGINT	MODEGO

DIAGRAM 3. Overview of Processing Controlled by the Problem Program



1. Receives or resumes control and executes.

2.. Executes.

2a. If issues an ATTACH, LINK, LOAD, or XCTL macro instruction, returns to step 1.

See Diagram 1.6

2b. If reaches breakpoint, returns to IKJEGMNL.

See Diagram 1.7

2c. If receives an attention interruption, returns to IKJEGMNL.

See Diagram 2.28

2d. If completes normally, returns to IKJEGMNL.

See Diagram 2.8

2e. If begins abnormal termination and there are unprocessed statements, returns to IKJEGMNL.

See Diagram 2.29

2f. If terminates abnormally and there are no unprocessed statements, returns to TMP.

Description	Module	Label
1. The problem program receives control upon execution of a GO or CALL subcommand. The problem program resumes control upon execution of an ATTACH, LINK, LOAD or XCTL macro instruction in the problem program.	Problem Program	
2. The problem program executes until a terminating condition occurs:	IGC0006A	
2a. When the problem program issues an ATTACH, LINK, LOAD, or XCTL macro instruction to another module, IKJEGAT activates any deferred breakpoints in the problem program and returns to step 1.	IKJEGMNL IKJEGAT	
2b. When the problem program reaches a breakpoint, control is returned to IKJEGMNL and message IKJ57024I (At address) is sent to the terminal user via IKJEGIO.	IGC0009G IKJEGMNL BRKCHK IKJEGIO	
2c. When the problem program is interrupted by an attention interruption, control is passed to IKJEGATN after the TEST mode message (TEST) is sent to the terminal user via the attention interruption processor.	IKJEGATN IGC0009G IKJEGMNL	
2d. When the problem program completes normally, control returns to IKJEGMNL and IKJEGIO sends a message to the terminal user. If a single program ends without a control transfer, message IKJ57023I (PROGRAM UNDER TEST HAS TERMINATED NORMALLY) is sent. If a single program ends with a control transfer, message IKJ57025I (PROGRAM UNDER TEST HAS TERMINATED) is sent. If a program attempts a control transfer to a non-resident program, message IKJ57028I (NO ACTIVE PROGRAM LEFT) is sent.	IGC0009G IKJEGMNL BRKCHK IKJEGIO	BRKCHK STAICHK
2e. When the problem program begins an abnormal termination and unprocessed statements remain, control returns to IKJEGMNL, and message IKJ56641I (ENDED DUE TO ERROR) is sent to the terminal user via IKJEFT04.	IKJEFT04 IKJEGMNL	
2f. When the problem program terminates abnormally and no unprocessed statements remain, IKJEGMNL removes all breakpoints inserted by TEST, releases TEST storage, and closes all open ECBs before returning control to the TMP.	IKJEFT04 IKJEGMNL IKJEGEND IKJEGOFF	

Section 3: Program Organization

This section contains module descriptions arranged alphabetically by module entry-point name. They contain reference information, such as register usage, system routines called, return codes, etc.

TEST uses the following 29 assembly modules, 11 macros, and 2 supervisor call routines.

MODULES	MACROS	SVCs
IKJEGASN	BRKELEM	IGC0006A—SVC 61
IKJEGAT	IKJEGDBE	IGC0009G—SVC 97
IKJEGATD	IKJEGDME	
IKJEGATN	IKJEGS6A	
IKJEGCIV	IKJEGS9G	
IKJEGCPY	IKJEGSUB	
IKJEGCVT	IKJEGSVB	
IKJEGDCB	IKJEGSVQ	
IKJEGDEB	IKJPARMA	
IKJEGEND	TCOMTAB	
IKJEGEQU		
IKJEGGO		
IKJEGINT		
IKJEGIO		
IKJEGLDF		
IKJEGLDR		
IKJEGLSA		
IKJEGLST		
IKJEGMAP		
IKJEGMNL		
IKJEGOFF		
IKJEGPCH		
IKJEGPSW		
IKJEGQFY		
IKJEGSRH		
IKJEGSTA		
IKJEGSYM		
IKJEGTCB		
IKJEGWHR		

IGC0006A–SVC 61 Routine

Diagram 1.3 contains the method-of-operation information for this routine.

Entry Information

IGC0006A is entered from Contents Supervision or the Overlay Supervisor when a module or overlay segment of the problem program is fetched. Entrance is from the SVC second-level interruption handler of the supervisor.

Input Parameters

Parameters received from Contents Supervision are:

Reg 1	Complement of the address of the DCB used to fetch the module.
Reg 3	Address of the supervisor's communication vector table (CVT).
Reg 4	Address cf the current TCB.
Reg 5	Address of the SVC 61 routine's supervisor request block (SVRB).
Reg 11	Address of the BLDL entry for the module in RBGRSAVE.
Reg 12	Address of the extent list for the module in RBGRSAVE.

Parameters received from the Overlay Supervisor are:

Reg 1	Zero.
Reg 3	Address of the supervisor's CVT.
Reg 4	Address of the current TCB.
Reg 5	Address of SVC 61 routine's SVRB.
Reg 12	Address of the module's segment table in RBGRSAVE. This is the same as the loaded address of the module contained in the extent list.

Function

The SVC 61 routine performs one main function and one auxiliary function. Its main function is to store information in an SVC information block that describes the data set from which the module was fetched. This information (load name, loaded address, and DDname) is used later by IKJEGSYM, the TEST Symbol module, when it tries to resolve internal symbols or entry names specified for the fetched module.

The routine's auxiliary function is to temporarily post control to either IKJEGINT or IKJEGMNL, depending on which TEST module is active when the SVC 61 routine is invoked. IKJEGINT, when posted, issues an XCTL macro instruction to fetch Mainline and give control to it. Mainline, when posted, links to the AT subcommand processor (if necessary) to activate deferred breakpoints that the user specified for the fetched module. Mainline then posts control back to the SVC 61 routine.

Data Areas

Created: The SVC information block is created by this module.

Updated: The TSTTRN field of TCOMTAB is set to point to the first SVC information block if TEST is running. If TEST is not running, the TCBTRN field of the current TCB is set to point to the first SVC information block.

Consulted: The following data areas are consulted by IGC0006A:

- The TCBTRN field of the TCB points to TCOMTAB if TEST is running. The TCBTIOT field of the TCB points to the task I/O table (TIOT).
- If the TIOT is available, it contains the DDname that was used to allocate the data set from which the module was fetched.

- An open DCB is used by Contents Supervision when it fetches the module, and by IKJEGSYM for symbol resolution.
- The BLDL entry for the PDS member from which the module was fetched is consulted.
- The extent list for the PDS member from which the module was fetched is consulted for pointers.
- The SVRB for the SVC 61 routine is consulted to get work areas and pointers.

Routines Called

TSO: No TSO routines are called by IGC0006A.

System: IGC0006A calls the following system routines:

- GETMAIN, to get space for an SVC information block and an ECB.
- POST, to make the IKJEGMNL or IKJEGINT module dispatchable.
- WAIT, to force control to IKJEGMNL or IKJEGINT by temporarily making the SVC 61 routine nondispatchable.

Register Usage during Module Execution

Reg 0 Used as a parameter register.

Reg 1 Used as a parameter register.

Reg 2 Points to the DCB used to fetch the problem module.

Reg 3 Points to the supervisor's CVT.

Reg 4 Points to the current TCB.

Reg 5 Points to the SVRB of the SVC 61 routine.

Reg 6 Used as a base register.

Reg 9 Points to TCOMTAB when TEST is running.

Reg 10 Used as a base for the dummy section (DSECT) of the SVC information block.

Reg 11 Points to the extended save area of the SVRB of the SVC 61 routine.

Reg 12 Points to the segment table if the SVC 61 routine is invoked from the Overlay Supervisor.

Exit Information

If the current task is running under TSO, exit is to the Contents Supervisor (or the Overlay Supervisor).

Output Parameters

If TEST is running, the origin of the chain of SVC information blocks is the TSTTRN field of TCOMTAB. If TEST is not running, the origin of the chain of SVC information blocks is the TCBTRN field of the current TCB.

Return Codes

0 Successful completion.

4 Error encountered.

Message CSECT

IGC0006A has no message CSECT.

IGC0009G–SVC 97 Routine

This module consists of two parts: a breakpoint handler whose entry point is IGC0009G and a subroutine or service routine of the TEST program, whose symbolic address is TSTSVC01. Diagrams 2.28 and 1.4 through 1.8 contain the method-of-operation information for this routine.

Entry Information

IGC0009G is entered from any module of the TEST program when used as a subroutine of TEST, or from any module of the problem program when used as a breakpoint handler.

Input Parameters

When IGC0009G is used as a breakpoint handler, the TCBTCP bit is "1" in the current TCB.

When IGC0009G is used as a subroutine of TEST, the TCBTCP bit is "0", and Register 1 points to a three-word parameter list.

Function

IGC0009G is invoked as a breakpoint handler in two cases. It is invoked when a user-inserted breakpoint (an SVC 97 instruction) is executed in the problem program, and it is invoked when a pseudo breakpoint is executed for a TEST module.

The TEST module saves the problem program's restart address in the TSTGO field in TCOMTAB, and either branches to an SVC 97 instruction or points the problem program's RBOPSW to an SVC 97 instruction. The SVC 97 instruction is stored at one of the following locations:

- TSTGO+6 in TCOMTAB, for most pseudo breakpoints.
- The problem program's entry point, if the TEST load module brought the program into main storage. This SVC 97 instruction is set by IKJEGINT for a load module and is set by the TEST load module for an object module.
- The PPEXIT field in TCOMTAB. IKJEGINT sets this breakpoint with a pointer to PPEXIT in Register 14 of the problem program. This action prepares for a pseudo breakpoint that will be taken when the problem program completes.

When invoked from a user-inserted breakpoint, IGC0009G:

1. Checks for an invalid caller. If the caller is invalid, the module exits; if the caller is valid, the module continues.
2. Checks if the count in the corresponding break element is exhausted. If it is, the module continues to Step 3; if it is not, the module branches to Step 4.
3. Points the instruction counter in the problem program's restart address (RBOPSW) to the SVC 97 instruction. Then it transfers control to IKJEGMNL and waits for IKJEGMNL to get and link to the subcommands desired by the terminal user. IKJEGMNL returns control to IGC0009G.
4. Either points the problem program's restart address (RBOPSW) to the saved program instruction in the break element or prepares for a simulated BAL or BALR instruction, depending on the type of instruction that was replaced by the breakpoint. If the routine cannot identify the breakpoint, it ignores the breakpoint and continues processing.
5. Restarts the problem program by issuing an SVC 3 instruction and exiting.
6. Sets the problem program's restart address (RBOPSW) to the next instruction after the breakpoint and exits. The supervisor's Exit routine and the VS2 Dispatcher restart the program. The preceding action occurs when control is received from the SVC 97 instruction in the break element. IGC0009G does not receive control if the executed instruction was a branch or an SVC instuction, or if it caused an ABEND.

When invoked from a pseudo breakpoint, this routine:

1. Checks for an invalid caller. If the caller is invalid, the routine exits; if the caller is valid, the routine continues.
2. Restores the saved RB wait count of the problem module to be given control. The count was saved at TSTGO+4 in TCOMTAB.
3. Checks the TSTSTAI bit in TCOMTAB to determine if the problem program terminated abnormally. If so, the routine frees the 104-byte STAE work area (if one exists), and restores the saved program registers.
4. Passes control to either IKJEGMNL or IKJEGINT and waits for that TEST module to execute and perform a function needed by the module that set the pseudo breakpoint. For example, IKJEGMNL gets control from module IKJEGLDF via a pseudo breakpoint to restore the problem program's registers in SVC 97's SVRB. IKJEGLDF cannot restore the registers, because it operates under the subtask and therefore cannot invoke SVC 97 as a subroutine.

When the SVC 97 routine receives control from IKJEGMNL, the routine resumes processing with the execution described in step 4 of the user-inserted breakpoint description, causing the problem program to resume execution.

When it is necessary to modify protected main storage, the subcommand processors of TEST call IGC0009G. The functions performed include setting fields in the TCB and the RB as well as validity-checking addresses.

Data Areas

Created: A queue of ECBs, chained from the TSTSVCQ field in TCOMTAB is used by IKJEGMNL to handle in first-in, first-out order concurrent POSTs from IGC0009G during multitasking.

Updated: As a breakpoint handler, IGC0009G updates the BRKCHAIN and BRKCOUNT fields of a break element, and the following fields in TCOMTAB: PPTCB, TSTMTASK, TSTSVCQ, ECBPP, and ECBTST.

As a subroutine of TEST, IGC0009G updates fields in the specified TCB or RB.

Consulted: As a breakpoint handler, IGC0009G consults the break element for the breakpoint that was encountered and the origin of the break element queue, location BREAKTAB in TCOMTAB.

As a subroutine of TEST, IGC0009G does not consult any data areas.

Routines Called

TSO: IGC0009G calls no system routines.

System: IGC0009G calls the following system routines:

- FREEMAIN, to free main storage used as work areas.
- GETMAIN, to obtain main storage for work areas.
- POST, to indicate completion of initialization processing.
- WAIT, to wait on event completion so that TEST processing can continue.
- SVC 3, to terminate SVRB processing.

Register Usage during Module Execution

Reg 1	Points to the input parameter list if the SVC 97 routine is used as a subroutine of TEST.
Reg 3	Points to the CVT.
Reg 4	Points to the current TCB.
Reg 5	Points to the SVRB for IGC0009G.

Reg 7 Points to the break element queue.

Reg 9 Points to TCOMTAB.

Reg 10 Used as a base register.

Reg 11 Used as a link register.

Reg 12 Used as an internal return register.

Reg 13 Used as a mask register for clearing high-order bytes.

Exit Information

When IGC0009G acts as a breakpoint handler, it exits to the restart address of the problem program.

When IGC0009G acts as a subroutine of TEST, it exits to the calling TEST module.

Output Parameters

IGC0009G has no output parameters.

Return Codes

Register 15 is unchanged with a successful return from use as a breakpoint handler.

0 Successful return from use as a TEST subroutine, except when used as an address validity checker.

0 Address is not write-protected from the user from use as an address validity checker.

4 Address is not read-protected from the user from use as an address validity checker.

8 Address is inaccessible to the user from use as an address validity checker.

Message CSECT

IGC0009G has no message CSECT.

IKJEGASN–Assignment Function Processor, Second Load Module

Diagram 2.2 contains the method-of-operation information for this module.

Entry Information

IKJEGASN is entered from IKJEGPCH via an XCTL macro instruction.

Input Parameters

IKJEGASN receives the input buffer contents, and Register 9 points to TCOMTAB.

Function

IKJEGASN is the second of two load modules that process the assignment function of TEST. The Assignment function changes values in main storage and in registers, according to the following user specifications.

- C, which specifies character data.
- X, which specifies hexadecimal data.
- B, which specifies binary data.
- H, which specifies fixed point binary (half-word) data.
- F, which specifies fixed point binary (full-word) data.
- E, which specifies floating point (single precision) data.

- D, which specifies floating point (double precision) data.
- P, which specifies packed decimal data.
- Z, which specifies zoned decimal data.
- A, which specifies address constant data.
- S, which specifies address (base plus displacement) data.
- Y, which specifies address constant (half-word) data.

IKJEGASN then converts the data to printable hexadecimal if necessary and formats it according to the specified type. IKJEGASN determines if the address specified for data modification is register type. If so, it causes a SVC 97 instruction to be issued which moves the changed data to the register address.

Data Areas

Created: IKJEGASN creates no data areas.

Updated: IKJEGASN updates the ABENTAB1, TSTFLGS1, TSTFLGS2, and TSTRTYPT fields in TCOMTAB.

Consulted: IKJEGASN consults BRKELEM, VALUPDE, TCB, TPL, CVT, the work area pointed to by TCOMTAB, and the following fields in TCOMTAB: CONAREAS, ECBLOG, ECBTMPA, OUTBUF, TSTFLGS1, PPTCB, REGSAVE2, REGSAVE3, TPUTADDR, TSTCONVT, TSTSTAE, and WORKAREA.

Routines Called

TSO: IKJEGASN calls the following TSO routines:

- IKJEGCVT, to convert values.
- IKJEGIO, to issue error messages.
- IKJEGSRH, to remove a break element and to see if the assignment will overlay any breakpoints.
- SVC 97, to modify the user's general purpose or floating point registers.

System: IKJEGASN calls the following system routines:

- STAE, to establish an abnormal termination exit.
- FREEMAIN, to free core used as work areas.

Register Usage during Module Execution

Reg 9	Points to TCOMTAB.
Reg 10	Points to PCHWORK, the work area.
Reg 11	Used as a base register.

Exit Information

IKJEGASN exits to IKJEGMNL.

Output Parameters

IKJEGASN has no output parameters.

Return Codes

0	Normal end processing continues.
16	Attention interruption encountered during processing.
20	STAE exit routine entered.

Message CSECT

IKJEGMSD is the message CSECT for IKJEGASN.

IKJEGAT–AT Subcommand Processor, First Load Module

Diagram 2.3 contains the method-of-operation information for this module.

Entry Information

IKJEGAT is entered from IKJEGMNL via a LINK macro instruction.

Input Parameters

Register 9 points to TCOMTAB.

To activate a deferred breakpoint, the first two words of the WORKSP work area contain the name of the module for which deferred breakpoints are to be activated. The TSTBUILD switch in TCOMTAB is set, to indicate that a new module is being fetched for the problem program.

To set an active breakpoint or store information about a deferred breakpoint, the input parameters of the subcommand are in the command buffer, whose address is the INBUF field of TCOMTAB.

Function

IKJEGAT has three functions:

- It determines if an AT subcommand was issued for a DEFER request or if it was issued to activate breakpoints. If the request is to set a DEFER breakpoint, this module transfers control to the second load module, IKJEGATD, via an XCTL macro instruction.
- It activates any deferred breakpoints that may have been specified for a newly fetched module. In this case, there is at least one element in the defer element queue. IKJEGAT is invoked by IKJEGMNL to build one or more break elements and set the corresponding breakpoints in the newly fetched module.
- It builds break elements and inserts breakpoints at specified locations in the problem program.

Data Areas

Created: IKJEGAT creates the BRKELEM, the subcommand chain save area, and the address string save area.

Updated: IKJEGAT updates the PDEUSER and PDE2USER fields in the parameter description entry (PDE), and the following fields in TCOMTAB: ABENTAB1, TSTRETRY, TSTFLGS1, TSTFLGS2, TSTFLGS4, and BREAKTAB.

Consulted: IKJEGAT consults the DME, the DBE, the TCB, the CVT, the PDL, and the following fields in TCOMTAB: OUTBUF, CONAREA, WORKAREA, REGSAVE2, REGSAVE3, PARMLIST, INBUF, TPUTADDR, TSTCONVT, TSTADDR, TSTSTAE, TSTFLGS1, DEFERTAB, OPCODTAB, TSTOPCD2, and TSTSRHRT.

Routines Called

TSO: IKJEGAT calls the following TSO routines:

- IKJPARS, to scan the subcommand and to check the syntax. Exit routines are used to check the validity of the IKJPARS parameters.
- IKJEGCVT, to convert addresses and values. At entry to convert an address, Register 1 contains the complement of the PDE address, indicating an address conversion, and Register 0 is set to zero to indicate converstion to binary.
- IKJEGIO, to issue error messages. At entry, Register 1 contains the complement of the pointer to the addresses of the length fields of the message.

- IKJEGSRH, to build a BRKELEM for an address for which a BRKELEM already exists, that is, a duplicate breakpoint. The old BRKELEM is removed by IKJEGSRH before the new BRKELEM is built. At entry, Register 1 points to a three-word parameter list. The first and third words are zero; the second word is the address of the breakpoint.
- IKJEGBLD, to construct the input address string. The address string is built either in the address save area, if needed in building a break element, or in the output buffer, if an error message is to be written. At entry, Register 0 points to the PDE address from which the information to construct the address string will be obtained. Register 1 points to the area in which the string will be built. If the string is to be built for the break element, Register 1 must be negative to indicate a maximum of 100 bytes. Otherwise, a maximum of 26 bytes is constructed.

System: IKJEGAT calls the following system routines:

- LINK, to invoke IKJPARS.
- XCTL, to transfer control to IKJEGATD.
- GETMAIN, to get space for a break element.
- FREEMAIN, to free the STAE work area, if it exists, and the current incomplete break element.
- STAE, to build a STAE control block for IKJEGSTA, the STAE exit routine for TEST.

Register Usage during Module Execution

Reg 0 Used as a parameter register.

Reg 1 Used as a parameter register.

Reg 3 Reserved for maintenance.

Reg 9 Points to TCOMTAB.

Reg 10 Points to the common work area described by TSTCWORK.

Reg 12 Used as a base register.

Exit Information

IKJEGAT exits to IKJEGMNL via an SVC 3 instruction pointed to by Register 14. If IKJEGATD was entered for an AT DEFER subcommand, this module exits to IKJEGATD via an XCTL macro instruction.

Output Parameters

The output parameter for IKJEGAT is the BREAKTAB field in TCOMTAB, which points to the break element queue.

Return Codes

0 Entered because of an AT subcommand.

4 Entered to activate a deferred breakpoint for a newly fetched module of the problem program.

16 The attention exit routine of TEST was entered to service an attention interruption from the terminal.

20 The retry routine belonging to IKJEGAT was executed to handle a STAE-intercepted ABEND.

Message CSECT

IKJEGATM is the message CSECT for IKJEGAT.

IKJEGATD-AT Subcommand Processor, Second Load Module

Diagram 2.3 contains the method-of-operation information for this module.

Entry Information

IKJEGATD is entered from IKJEGAT (the first load module of the AT subcommand processor) via an XCTL macro instruction whenever a deferred breakpoint is specified.

Input Parameters

Register 1 points to TCOMTAB. The PDEADDR field in the work area of IKJEGATD points to the first address PDE returned by IKJPARS.

The TSTANSPL field in TCOMTAB points to a parameter descriptor list (PDL) that has been parsed and checks for validity by IKJEGAT.

Function

IKJEGATD first checks that all input addresses are fully qualified and that they specify the same load module name.

The module then constructs a defer element queue in which it saves breakpoint information about a module that has not yet been fetched to memory. The defer element queue contains at least one defer module element, DME, and an associated defer break element, DBE. Each DME points to its DBE and contains the load name of the module for which breakpoint information is being saved. Each DBE contains a pointer to a parameter descriptor list built by IKJPARS, a pointer to the next DBE, and a pointer to an input buffer. The buffer contains the subcommand that specified the deferred breakpoint.

Data Areas

Created: IKJEGATD creates the DME and the DBE.

Updated: IKJEGATD updates the DEFERTAB, TSTANSPL, TSTFLGS2, RETRY1, and INBUF fields of TCOMTAB.

Consulted: IKJEGATD consults the CVT, the TCB, the DME, the DBE, and the following fields in TCOMTAB: REGSAVE2, TSTANSPL, DEFERTAB, INBUF, TPUTADDR, and TSTSTAE.

Routines Called

TSO: IKJEGATD calls the following TSO routine:

- IKJEGIO, to issue error messages.

System: IKJEGATD calls the following system routines:

- STAE, to build a STAE control block for IKJEGSTA, the STAE Exit routine for TEST.
- GETMAIN, to get space for a defer module element and its associated defer break element.
- FREEMAIN, to free the STAE work area, if it exists, using the retry routine (RETRY2).

Register Usage during Module Execution

Reg 1	Used as a parameter register.
Reg 3	Reserved for maintenance.
Reg 9	Points to TCOMTAB.
Reg 10	Points to common work area described by TSTCWORK.
Reg 12	Used as a base register.

Exit Information

IKJEGATD exits to IKJEGMNL via an SVC 3 instruction pointed to by Register 14.

Output Parameters

The DEFERTAB field in TCOMTAB points to the first defer module element in the defer element queue.

Return Codes

0	Normal return.
16	An attention interruption detected during module execution.
20	The module's retry routine, RETRY2, entered to handle a STAE-intercepted ABEND.

Message CSECT

IKJEGM2 is the message CSECT for IKJEGATD.

IKJEGATN-Attention Exit Routine

Diagram 2.30 contains the method-of-operation information for this module.

Entry Information

IKJEGATN is entered from the Attention Handler of IKJVAR05, the region control task, via a LOAD PSW instruction.

Input Parameters

Register 1 contains the address of a parameter list, which consists of the following pointers:

- A pointer to the Terminal Attention Interrupt Element (TAIE).
- A pointer to an input buffer containing new input from the terminal.
- A pointer to TCOMTAB.

Function

IKJEGATN performs the following functions:

- It checks for no terminal input (length equals zero). A null line indicates that the user wants control to return to the point of interruption.
- It invokes the PUTGET service routine to get a new input line and to process a question mark for second level messages.
- It posts control either to IKJEGMNL or to another interrupted TEST module.

Eventually, control is passed to IKJEGMNL, which gets the newly entered subcommand and links to the appropriate subcommand processor.

IKJEGBLD, a routine internal to IKJEGATN, builds an address string to a maximum of 100 bytes. As input, this subroutine requires a pointer in Register 0 to the address parameter description element PDE and a pointer in Register 1 to the area in which the string is to be built. Upon return to the caller, Register 0 contains the number of bytes constructed, and Register 1 contains a pointer to the byte following the last byte in the address string. If Register 1 is negative, the address string is built to only a maximum of 26 bytes.

Data Areas

Created: IKJEGATN creates a new input buffer, queued from the INBUF field of TCOMTAB.

Updated: IKJEGATN updates the work areas described by the WORKSP dummy section, the INBUF and TSTGO fields of TCOMTAB, and the high-order bit of the ECTSMSGF field in the environment control table ECT.

Consulted: IKJEGATN consults the TAIE, the input buffer, TCOMTAB, and the TPLCTCB field of the TEST parameter list.

Routines Called

TSO: IKJEGATN calls the following TSO routines:

- PUTGET, to issue messages to the terminal and gets terminal input.
- The SVC 97 routine, to change the restart address of the problem program.
- TCLEARQ, to clear the input buffers.
- STACK, to delete elements from the input stack.

System: IKJEGATN calls the following system routines:

- LINK, to provide linkage to the PUTGET service routine.
- FREEMAIN, to free space for the buffer(s).
- STATUS and POST, to make the subtasks dispatchable and nondispatchable.

Register Usage during Module Execution

Reg 8	Points to work space.
Reg 9	Points to TCOMTAB.
Reg 12	Used as a base register.

Exit Information

IKJEGATN exits to the Supervisor Exit routine via an SVC 3 instruction pointed to by Register 14.

Output Parameters

The INBUF field in TCOMTAB points to the buffer containing the new subcommand.

A post code of X'48' is placed in the event control block ECB whose address is in the ECBTMPA field of TCOMTAB. This code signifies that the TEST attention exit has been posted, rather than the attention exit routine of the TMP.

Return Codes

0 Processing complete.

Message CSECT

IKJEGATM is the message CSECT for this module.

IKJEGCIV–HELP Command Invoker

Diagram 2.27 contains the method-of-operation information for this module.

Entry Information

IKJEGCIV is entered from IKJEGMNL via a LINK macro instruction.

Input Parameters

Parameters received by IKJEGCIV include the subcommand name field ECTSCMD in the ECT and the INBUF field in TCOMTAB that points to the buffer containing the command and possible command operands. Register 9 points to TCOMTAB.

Function

IKJEGCIV calls the BLDL routine to determine that the specified command is valid. If it is valid, it is attached as a subtask of TEST. The subcommand that was entered in TEST mode and the command processor parameter list are passed to the command as input parameters.

Data Areas

Created: IKJEGCIV creates no data areas.

Updated: IKJEGCIV updates the TPLCBUF field in the TPL, the INBUF and TSTHTCB fields in TCOMTAB, and the ECTPCMD field in the ECT.

Consulted: IKJEGCIV consults the ECBTMPA, ECBTERM, and ECBLOG fields in TCOMTAB.

Routines Called

TSO: IKJEGCIV calls the following TSO routines:

- IKJDAIR, to release all data sets allocated by the subtask.
- IKJEGIO, to issue error messages to the terminal user.

System: IKJEGCIV calls the following system routines:

- STAE, to establish an abnormal termination exit.
- FREEMAIN, to free the main storage used for a terminal input buffer.
- BLDL, to determine if the command specified is valid.
- ATTACH, to create the subtask that processes the command.
- WAIT, to determine that one or more events has completed.
- DETACH, to terminate the subtask that was created to process the command.

Register Usage During Module Execution

Reg 2	Points to the ECT.
Reg 9	Points to TCOMTAB.
Reg 10	Points to the general work area of IKJEGCIV.
Reg 11	Points to the task parameter list (TPL).
Reg 12	Used as a base register.

Exit Information

IKJEGCIV exits to IKJEGMNL via an SVC 3 instruction pointed to by Register 14.

Output Parameters

IKJEGCIV has no output parameters.

Return Codes

0	Normal return.
16	Attention interruption occurred.
20	Abnormal termination recovery occurred.

Message CSECT

IKJEGCIM is the message CSECT for this module.

IKJEGCPY-COPY Subcommand Processor

Diagram 2.5 contains the method-of-operation information for this module.

Entry Information

IKJEGCPY is entered either from IKJEGMNL at entry point IKJEGCPY via a LINK macro instruction or from IKJPARS at entry point IKJEGFM, IKJEGTO, or IKJEGLN.

Input Parameters

Register 9 points to TCOMTAB, and the INBUF field in TCOMTAB points to the input buffer that contains the subcommand.

Function

IKJEGCPY copies data from storage to storage, from register to register, from register to storage, and from storage to register. It also obtains addresses, and it places these either in storage or in a register.

Data Areas

Created: IKJEGCPY creates WA, its work area.

Updated: IKJEGCPY updates the user registers and storage areas into which the data was copied, and it updates the ABENTAB1, TSTRETRY, and TSTFLGS4 fields in TCOMTAB.

Consulted: IKJEGCPY consults the WORKAREA field in TCOMTAB, the subtask for the problem program, and the CVT.

Routines Called

TSO: IKJEGCPY calls the following TSO routines:

- The SVC 97 routine, to check the validity of addresses and to copy register contents into the RB save area.
- IKJPARS, to check the syntax of the subcommand operands.
- IKJEGCVT, to convert values and addresses.
- IKJEGBLD, to attach addresses to error messages.
- IKJEGSRH, to locate or remove break elements within the range specified by the user.
- IKJEGIO is called to write data to a terminal or a data set.

System: IKJEGCPY calls the following system routines:

- STAE, to establish an abnormal termination exit.
- FREEMAIN, to free main storage used as work areas.

Register Usage during Module Execution

Reg 2 Points to the parameter descriptor list PDL used by the main routine of IKJEGCPY. It is also used by the validity check routine as a pointer to the PDE.

Reg 3 Used as a work register.

Reg 4 Used as a work register.

Reg 5 Used as a work register.

Reg 6 Used as a work register.

Reg 7 Used as a work register.

Reg 8 Points to the PDL passed from IKJPARS.

Reg 9 Points to TCOMTAB.

Reg 10 Points to the work area used by IKJEGCPY.

Reg 11 Used as an internal link register.

Reg 12 Used as a base register.

Exit Information

IKJEGCPY exits to the Supervisor Exit routine via an SVC 3 instruction pointed to by Register 14.

Output Parameters

IKJEGCPY has no output parameters.

Return Codes

0 Normal or error return.

16 Attention interruption encountered during processing.

20 STAE retry routine entered.

Message CSECT

IKJEGMGC is the message CSECT for this module.

IKJEGCVT-Convert Module

Diagram 2.24 contains the method-of-operation information for this module. Figure 8 and Figure 9 in Section 1 provide additional information for this module.

Entry Information

IKJEGCVT is invoked both directly and indirectly by subcommand processors. It is invoked directly to convert values to binary or printable form, and it is invoked indirectly, through a validity check exit routine from IKJPARS, to convert addresses to binary.

Subommand Processor	Module Name	Invokes Directly	Invokes Indirectly
Assignment	IKJEGPCH	X	X
AT	IKJEGAT		X
CALL	IKJEGGO		X
EQUATE	IKJEGEQU		X
FREEMAIN	IKJEGLDF		X
GO	IKJEGGO		X
LIST	IKJEGLST	X	X
LISTDCB	IKJEGDCB	X	X
LISTDEB	IKJEGDEB	X	X
LISTPSW	IKJEGPSW	X	X
LISTTCB	IKJEGTCB	X	X
OFF	IKJEGOFF		X
QUALIFY	IKJEGQFY		X
RUN	IKJEGGO		X
WHERE	IKJEGWHR		X

Input Parameters

IKJEGINT receives its input parameters from Register 0, Register 1, and a PDE that describes the address or value to be converted.

For address conversion to binary form, Register 0 is positive, indicating conversion to binary, and Register 1 is negative, indicating an address is to be converted. Register 1 also points to an address PDE. The type of address and the PDEFLG4 indicator in the PDE follow.

Type of Address	PDEFLG 4 in PDE
Absolute	ABSADDR - '00'
Relative	RELADDR - '40'
Symbolic	SYMADDR - '80'
Register	GENR - '20'
Entry Name Only	CTONLY - '04'

For address conversion to printable form, Register 0 is negative, indicating conversion to printable form, and Register 1 is negative, indicating an address conversion. Register 1 also points to an address PDI . The type of address and the PDEFLG4 indicator in the PDE follow.

Type of Address	PDEFLG 4 in PDE
Absolute	ABSADDR - 00'
Relative	RELADDR - '40'
Other Non-Absolute Types	Codes Other Than the Above

For value conversion from binary, Register 0 is negative, indicating conversion to printable form, and Register 1 is positive, indicating value conversion. Register 1 also points to a value PDE. The type of conversion and the hexadecimal code located at offset plus seven in the PDE follow.

Type of Conversion	Hex Code in PDE
Hexadecimal	'04'
Instruction	'0C'
Fixed Point Decimal	'10' or '14'
Floating Point Decimal	'1C' or '18'
Constants	A - '20' Y - '24' S - '28' V - '2C'
Packed Decimal	'30'

For value conversion to binary form, Register 0 is positive, indicating conversion to binary form, and Register 1 is positive, indicating a value conversion. Register 1 also points to a value PDE. The type of conversion and the hexadecimal code located at offset plus seven in the PDE follow.

Type of Conversion	Hex Code in PDE
Hexadecimal	'04'
Binary	'08'
Fixed Point Decimal	'10' or '14'
Constants	A '20' Y - '24' V '2C'
S Constant	'28'
Packed Decimal	'30'

Function

Certain TEST subcommand processors receive input addresses which they must convert to binary before they can process the addresses. Similarly, various subcommand processors wish to convert values to printable form in order to list the values in a message to the terminal or to include the values on the print data set. To convert addresses and values to binary or printable form, the subcommand processors invoke IKJEGCVT. This module is invoked directly by a subcommand processor for value conversion and indirectly, through a validity check exit routine from IKJPARS, for address conversion.

Data Areas

Created: IKJEGCVT creates no data areas.

Updated: IKJEGCVT updates the PDEUSER field of the input PDE and the 32-byte work area pointed to by the CONAREA field of TCOMTAB.

Consulted: IKJEGCVT consults the PDE and TCOMTAB.

Routines Called

TSO: IKJEGCVT calls the following TSO routines:

- IKJEGSYM, to resolve all symbolic addresses as well as for the symbolic part of address expressions. IKJEGSYM sets the first byte of the user word (PDEUSER) of the address PDE to X'80' and sets the remaining three bytes to point to a symbol information block (SYMINFO), whose first word contains the binary representation of the address.
- The SVC 97 routine, to validity check an address.

System: IKJEGCVT calls no system routines.

Register Usage during Module Execution

Reg 3 Points to the work area used in address conversion.

Reg 8 Used with IKJPARMA DSECT to address the fields in a PDE.

Reg 9 Points to TCOMTAB.

Reg 12 Used as a base register.

Exit Information

IKJEGCVT exits to the caller via a branch on Register 14.

Output Parameters

IKJEGCVT places the converted output in either of two locations, depending on the type of conversion. If it has converted an address to binary form, it places the binary address in the PDEUSER field of the input PDE. (See Section 5, "Data Areas" for the IKJPARMA DSECT.) All other converted outputs (addresses or values) are placed in a 32-byte work area that is pointed to by the CONAREA field of TCOMTAB.

Return Codes

0 Normal return.

4 Value not converted.

16 Attention interruption encountered during processing.

Message CSECT

IKJEGMSF is the message CSECT for this module and has four entry points: IKJEG1IA, IKJEGM01, IKJEGM02, and IKJEGM03.

IKJEGDCB–LISTDCB Subcommand

Diagram 2.14 contains the method-of-operation information for this module.

Entry Information

IKJEGDCB is entered from IKJEGMNL at entry point IKJEGDCB via a LINK macro instruction.

Input Parameters

IKJEGDCB receives the contents of the input buffer, and Register 9 points to TCOMTAB.

Function

IKJEGDCB processes the LISTDCB subcommand, which displays all or part of a specified data control block either to a user terminal or to a specified data set. After receiving control from IKJEGMNL, the module performs initialization processing. IKJEGDCB formats data from the DCB and converts data to hexadecimal via a branch to IKJEGCVT. IKJEGDCB then prints data via a branch to IKJEGIO and returns control to IKJEGMNL.

Data Areas

Created: IKJEGDCB creates DCBWKARA, its work area.

Updated: IKJEGDCB updates no data areas.

Consulted: IKJEGDCB consults TCOMTAB, the CVT, and the TCB of TEST.

Routines Called

TSO: IKJEGDCB calls the following TSO routines:

- IKJPARS, to check the syntax of the subcommand operands.
- IKJEGCVT, to convert values to printable characters and to convert addresses.
- IKJEGIO, to write data to a terminal or a data set.
- IKJEGBLD, to build a character symbolic address.

System: IKJEGDCB calls the following system routines:

- STAE, to establish an abnormal termination exit.
- FREEMAIN, to free main storage used as work areas.
- LINK, to invoke IKJPARS.

Register Usage during Module Executiion

Reg 3	Points to the DCB that is being processed.
Reg 5	Points to the work area of IKJEGDCB.
Reg 9	Points to TCOMTAB.
Reg 12	Used as a base register.

Exit Information

IKJEGDCB exits to IKJEGMNL via an SVC 3 pointed to by Register 14.

Output Parameters

The output from IKJEGDCB consists of lines to a terminal or a data set.

Return Codes

0	Normal return.
16	Attention interruption encountered during processing.
20	STAE retry routine entered.

Message CSECT

IKJEGMSG is the message CSECT for this module.

IKJEGDEB–LISTDEB Subcommand Processor

Diagram 2.15 contains the method-of-operation information for this module.

Entry Information

IKJEGDEB is entered from IKJEGMNL at entry point IKJEGDEB via a LINK macro instruction.

Input Parameters

IKJEGDEB receives the contents of the input buffer, and Register 9 points to TCOMTAB.

Function

IKJEGDEB processes the LISTDEB subcommand of TEST. The LISTDEB subcommand lists the basic section and any direct access sections of a DEB to a terminal or to a user specified data set. IKJEGDEB determines which of the fields in the basic section or direct access sections of a DEB has been requested by the user and then branches to a specific subroutine for that field. Each of these subroutines uses a common conversion and writing routine within IKJEGDEB.

Data Areas

Created: IKJEGDEB creates DBWKAREA, its work area.

Updated: IKJEGDEB updates no data areas.

Consulted: IKJEGDEB consults TCOMTAB, CVT, and the TCB of TEST.

Routines Called

TSO: IKJEGDEB calls the following TSO routines:

- IKJPARS, to syntax check the syntax of the subcommand operands.
- IKJEGCVT, to convert values to printable characters and to convert addresses.
- IKJEGIO, to write data to a terminal or a data set.

System: IKJEGDEB calls the following system routines:

- STAE, to establish an abnormal termination exit.
- LINK, to invoke IKJPARS.
- FREEMAIN, to free main storage used as work areas.

Register Usage during Module Execution

Reg 5	Points to DBWKAREA, the work area used by IKJEGDEB.
Reg 9	Points to TCOMTAB.
Reg 12	Used as a base register.

Exit Information

IKJEGDEB exits to IKJEGMNL via an SVC 3 instruction pointed to by Register 14.

Output Parameters

The output from IKJEGDEB consists of output lines to a terminal or a data set.

Return Codes

0	Normal return.
16	Attention interruption encountered during processing.
20	STAE retry routine entered.

Message CSECT

IKJEGDBM is the message CSECT for this module.

IKJEGEND-END Subcommand Processor

Diagram 2.8 contains the method-of-operation information for this module.

Entry Information

IKJEGEND is entered from IKJEGMNL via a LINK macro instruction.

Input Parameters

Register 9 points to TCOMTAB.

Function

IKJEGEND cleans up at the end of TEST processing and frees the Subpool1 main storage of TEST. IKJEGEND transfers control to IKJEGOFF to remove all breakpoints.

Data Areas

Created: IKJEGEND creates no data areas.

Updated: IKJEGEND updates the CALLPARM, TSTIODSN, TSTIODCB, TSTSYMWK, TSTFLG1, and TSTDCB fields in TCOMTAB.

Consulted: IKJEGEND consults TCOMTAB.

Routines Called

TSO: IKJEGEND calls the following TSO routine:

- IKJEGIO, to issue messages.

System: IKJEGEND calls the following system routines:

- STAE, to establish an abnormal termination exit.
- FREEMAIN, to free Subpool1 main storage of TEST.
- XCTL, to transfer control to IKJEGOFF.
- CLOSE, to close any open DCBs.

Register Usage during Module Execution

Reg 9	Points to TCOMTAB.
Reg 12	Used as a base register.

Exit Information

IKJEGEND transfers control to IKJEGOFF to deactivate all remaining breakpoints.

Output Parameters

IKJEGEND has no output parameters.

Return Codes

IKJEGEND has no return codes.

Message CSECT

IKJEGMSG is the message CSECT for this module.

IKJEGEQU-EQUATE and DROP Subcommand Processor

This module's alias is IKJEGDRP. Diagram 2.9 and Diagram 2.7 contain the method-of-operation information for this module.

Entry Information

IKJEGEQU is entered from IKJEGMNL via a LINK macro instruction at either entry point IKJEGEQU or IKJEGDRP.

Input Parameters

IKJEGEQU receives the contents of the input buffer, and Register 9 points to TCOMTAB.

Function

IKJEGEQU processes the EQUATE and DROP subcommands of TEST. The EQUATE subcommand builds and maintains the symbol table located in main storage. The DROP subcommand deletes a specified entry or entries from the table.

Data Areas

Created: IKJEGEQU creates a symbol table in main storage and AREAWORK, a work area.

Updated: IKJEGEQU updates the symbol table in main storage, if it already exists.

Consulted: IKJEGEQU consults TCOMTAB, CVT, and the TCB of TEST.

Routines Called

TSO: IKJEGEQU calls the following TSO routines:

- IKJPARS, to check the syntax of the subcommand operands.
- IKJEGCVT, to convert values to printable characters and to convert addresses.
- IKJEGIO, to write data to a terminal or a data set.
- IKJEGBLD, to build a character symbolic address.

System: IKJEGEQU calls the following system routines:

- STAE, to establish an abnormal termination exit.
- GETMAIN, to obtain main storage for the symbol table to be built.
- FREEMAIN, to free main storage used for the symbol table.

Register Usage during Module Execution

Reg 8	Points to a PDL.
Reg 9	Points to TCOMTAB.
Reg 10	Points to AREAWORK, the work area of IKJEGEQU.
Reg 11	Used as a base register.

Exit Information

IKJEGEQU exits to IKJEGMNL via an SVC 3 instruction pointed to by Register 14.

Output Parameters

IKJEGEQU has no output parameters.

Return Codes

0	Normal return.
16	Attention interruption encountered during processing.
20	STAE retry routine entered.

Message CSECT

IKJEGMSG is the message CSECT for IKJEGEQU.

IKJEGGO–GO Subcommand Processor

Diagrams 2.12, 2.4, and 2.22 contain the method-of-operation information for this module. This module's aliases are IKJEGRUN and IKJEGCAL

Entry Information

IKJEGGO is entered either from IKJEGMNL via a LINK macro instruction to entry point IKJEGGO, IKJEGRUN, or IKJEGCAL or from IKJPARS via a branch to entry point ADDRCHK.

Input Parameters

IKJEGGO recieves the contents of the input buffer, and Register 9 points to TCOMTAB.

Function

IKJEGGO processes the GO, CALL, and RUN subcommands of TEST. For all three subcommands, control blocks are altered to cause the problem program to receive control either at an address specified in the subcommand or at the next instruction in the problem program. For the RUN subcommand, the problem program is removed from the control of TEST. For the CALL subcommand, the problem program's Registers 1, 14, and 15 are altered to contain, respectively, the address of a parameter list, the address of a breakpoint SVC or the return address supplied in the subcommand, and the address at which control is passed to the problem program.

Data areas

Created: IKJEGGO creates the CALL parameter list which is chained from the CALLPARM field of TCOMTAB.

Updated: IKJEGGO updates the CALLPARM, ABENTAB1, RUNSW, TSTRTYPT, and TSTFLGS4 fields in TCOMTAB and the PDEUSER field in the PDE.

Consulted: IKJEGGO consults the BRKELEM, the TCB, the CVT, the RB, the PDL, and the following fields in TCOMTAB: TSTTCB, PPTCB, OUTBUF, WORKAREA, REGSAVE2, REGSAVE3, PARMLIST, TSTANSPL, INBUF, TPUTADDR, TSTCONVT, TSTADDR, TSTSTAE, TSTFLGS3, BREAKTAB, and PPRB.

Routines Called

TSO: IKJEGGO calls the following TSO routines:

- IKJPARS, to check the syntax of the subcommand operands.
- IKJEGIO, to write data to a terminal or a data set.
- IKJEGCVT, to convert values to printable characters and to convert addresses.
- The SVC 97 routine, to set the resume address and wait count.
- IKJEGBLD, to build a character symbolic address.

System: IKJEGGO calls the following system routines:

- STAE, to establish an abnormal termination exit.
- GETMAIN, to obtain main storage for work areas.
- FREEMAIN, to free main storage used as work areas.
- LINK, to invoke IKJPARS.

Register Usage during Module Execution

Reg 6 Points to the common work area described by TSTCWORK.

Reg 9 Points to TCOMTAB.

Reg 11 Reserved for maintenance.

Reg 12 Used as a base register.

Exit Information

IKJEGGO exits to IKJEGMNL.

Output Parameters

RUNSW is set on if processing completes normally. In all other cases, RUNSW is off.

Return Codes

0 System error; recoverable non-zero return code from the SVC 97 routine.

4 Normal end for GO and CALL.

12 Normal end for the RUN subcommand if the RUNSW is set; unrecoverable non-zero return code from the SVC 97 routine if RUNSW is off.

16 Attention interruption encountered during processing.

20 STAE retry routine entered.

Message CSECT

IKJEGMSG is the message CSECT for IKJEGGO.

IKJEGINT–Initialization Module

Diagram 1 contains the method-of-operation information for this module.

Entry Information

IKJEGINT is entered from one of the following:

- The TMP via a LINK macro instruction.
- The SVC 97 routine (IGC0009G) after a pseudo or user breakpoint has been encountered via POST and WAIT macro instructions.
- The ABEND/STAE Interface routine (IGC0D01C) via the VS2 Dispatcher after a STAE-intercepted ABEND in IKJEGINT.

Input Parameters

Register 1 contains the address of the TEST Parameter List (TPL), when entered from the TMP.

Function

IKJEGINT performs the following functions:

1. It allocates storage for tables, buffers, and work areas to be used during TEST execution.
2. It issues the STAE macro instruction for the STAE Exit routine to handle a possible ABEND during initialization.
3. It initializes part of TCOMTAB.
4. It determines if the problem program is already a subtask of TEST/TMP. If so, the module proceeds with step 5. If not, the module prepares to bring the program into main storage as a subtask of TEST. It attaches IKJEGLDR to fetch or load the program. IKJEGINT then branches to step 8. If IKJEGLDR is not attached, IKJEGINT does steps 5 through 7; otherwise, IKJEGLDR does Steps 5 through 7.

5. It moves the queue origin of the SVC Information Block chain (if it exists) from the problem program's TCB to TCOMTAB.
6. It ensures that the problem program is dispatchable by zeroing the program's RB wait count.
7. It inserts a pseudo breakpoint to cause linkage from the problem program, when it is dispatched, to IKJEGINT. IKJEGLDR sets this pseudo breakpoint if the VS2 Loader loads the problem program. IKJEGINT further ensures program dispatchability by issuing a STATUS START or IN macro instruction.
8. It waits to be posted from the SVC 97 routine (IGC0009G), after the problem program has been dispatched and has executed the pseudo breakpoint.
9. When posted from the SVC 97 routine, it restores the saved restart address for the problem program. When it inserted the pseudo breakpoint, IKJEGINT or IKJEGLDR saved this restart address in TCOMTAB.
10. It issues an XCTL macro instruction to purge IKJEGINT and the loading of IKJEGMNL and other TEST modules that are resident during the TEST session. The XCTL macro instruction transfers control to IKJEGMNL.

Data Areas

Created: IKJEGINT creates TCOMTAB, the control chain words for the queue of SVC information blocks built by the SVC 61 routine, and the following work areas: CONAREA, a communication area; OUTBUF; WORKAREA; and REGSAVE1 through REGSAVE6.

Updated: IKJEGINT updates the TPL, the TCB belonging to the problem program, the TCB of TEST, the program request block (PRB) or interruption request block (IRB) belonging to the problem program, and TCOMTAB.

Consulted: IKJEGINT consults no data areas.

Routines Called

TSO: IKJEGINT calls the following TSO routines:

- The SVC 97 routine, to put the address of TCOMTAB in the TCB.
- IKJPARS, to check the syntax of the TEST command operands.
- IKJDAIR, to allocate the data set specified in the TEST command.
- IKJDFLT, to fully qualify the data set name.
- IKJEGDDM, to generate the allocation failure message.
- IKJEGLDR, to attach the problem program.
- IKJEGIO, to write messages to the terminal user.
- TSEVENT, to record the start of a TEST session.
- IKJRLSA, to free the PDL of IKJPARS.

System: IKJEGINT calls the following system routines:

- STAE, to establish an abnormal termination exit.
- POST, to indicate completion of initialization processing to the SVC 61 routine or the SVC 97 routine.
- BLDL, to determine if the program name specified in the TEST command is valid.
- STATUS, to set the dispatchability of the problem program.
- CLOSE, to release the data set specified in the TEST command.
- XCTL, to transfer control to IKJEGMNL.

- WAIT, to wait on event completion so that IKJEGINT processing can continue.
- FREEMAIN, to free main storage used as work areas.
- LINK, to invoke another load module.
- GETMAIN, to obtain main storage for work areas.
- ATTACH, to create the problem program task.

Register Usage during Module Execution

Reg 9	Points to TCOMTAB.
Reg 10	Points to the TPL.
Reg 11	Used as a link register.
Reg 12	Used as a base register.

Exit Information

IKJEGINT exits to IKJEGMNL via an XCTL macro instruction if processing is normal or to the TMP if an error is detected (e.g., BLDL error, IKJDAIR error, GETMAIN error, etc.).

Output Parameters

Register 9 points to TCOMTAB.

Return Codes

0 Control passes to the TMP due to an error.

Message CSECT

Messages are contained in control section IKJEGXNT. This also is accessed by the STAE Exit routine of IKJEGINT to obtain inserts for second-level diagnostic messages.

ABENDLST contains addresses of retry locations to which control returns if IKJEGINT abnormally terminates and the STAE Exit routine intercepts the ABEND.

IKJEGIO–I/O Module

Diagram 2.26 contains the method-of-operation information for this module.

Entry Information

IKJEGIO is entered from any TEST module (except IKJEGATN) via a branch instruction to one of three entry points:

- IKJEGGT for input.
- IKJEGPT for output.
- IKJEGPG for output with a response requested.

Input Parameters

If IKJEGIO is entered at IKJEGPT and the TSTPRINT switch in TCOMTAB is set, output goes to a data set. If Register 0 is '0', the data set is already open. If Register 0 is not '0', it points to a data set name PDE for the data set to be opened. Register 1 points to a doubleword; the first word points to output data, and the second word contains '0'.

If the TSTPRINT switch in TCOMTAB is not set, Register 0 is not meaningful. However, if Register 1 is positive, the output is data. If Register 1 is negative, the output is a message. Register 1 also points to a doubleword; the first word points to the output data, and the second word points to a second-level message.

If the second word is zero, the output is not a message or there are no second-level messages.

If IKJEGIO is entered at IKJEGGT and the SUBCHAIN field in TCOMTAB is non-zero, the module gets a subcommand from the subcommand chain built in main storage by the AT subcommand processor. If the SUBCHAIN field is zero, a GETLINE macro instruction is issued to get a subcommand from the terminal.

If IKJEGIO is entered at IKJEGPG, Register 1 points to a doubleword; the first word points to a first-level message, and the second word points to a second-level message, if one exists. This second word contains zero if no second-level message exists.

Function

IKJEGIO handles three types of I/O request from any module except IKJEGATN.

For a GETLINE request, data is obtained from the terminal. A subcommand is obtained from the terminal, via a GETLINE macro instruction, if the SUBCHAIN pointer is zero. If, however, the SUBCHAIN pointer is not zero, IKJEGIO gets a subcommand from a subcommand chain built in storage by the AT subcommand.

For a PUTGET request, IKJEGIO issues a TEST message and obtains a subcommand, either from the terminal (via a PUTGET macro instruction), or from a list in storage, as with a GETLINE request.

For a PUTLINE request, the IKJEGIO prints a line of data or a message to the terminal. It can send output to a print data set, if such action is requested by one of the LIST subcommands. In the latter case, the PRINT keyword was specified in the subcommand.

Data Areas

Created: IKJEGIO creates no data areas.

Updated: IKJEGIO updates the INBUF and SUBCHAIN fields in TCOMTAB.

Consulted: IKJEGIO consults the TSTPRINT switch in the TSTFLGS1 field and the SUBCHAIN field of TCOMTAB, the ECBTMPA, and the ECBLOG.

Routines Called

TSO: IKJEGIO calls the following TSO routines:

- GETLINE, to get input from the terminal.
- PUTLINE, to issue a message or data to a terminal.
- PUTGET, to issue a mode message to a terminal and receives a response.
- IKJDAIR, to allocate the output data set.

System: IKJEGIO calls the following system routines:

- OPEN, to open the output data set.
- CLOSE, to close the output data set.
- PUT, to write variable blocked (VB) records to an output data set.

Register Usage during Module Execution

Reg 9	Points to TCOMTAB.
Reg 12	Used as a base register.

Exit Information

IKJEGIO exits to the caller via a branch on Register 14.

Output Parameters

IKJEGIO has no output parameters.

Return Codes

0 Normal return.

4 Error encountered.

16 Attention request or an operator STOP/MODIFY TS command encountered during processing.

Message CSECT

IOMSGCST is the message CSECT for IKJEGIO.

IKJEGLDF-DELETE, FREEMAIN, GETMAIN and LOAD Subcommand Processor

The aliases IKJEGLDF uses are IKJEGDEL, IKJEGFRE, IKJEGGET, and IKJEGLOD. Diagrams 2.6, 2.10, 2.11, and 2.19 contain the method-of-operation information for this module.

Entry Information

IKJEGLDF is entered via a LINK macro at one of four entry points - IKJEGLOD, IKJEGDEL, IKJEGGET, or IKJEGFRE.

Input Parameters

IKJEGLDF receives the contents of the input buffer, and Register 9 points to TCOMTAB.

Function

IKJEGLDF processes the following subcommands of TEST:

- DELETE, which removes the problem program.
- FREEMAIN, which frees a specified number of bytes for the problem program.
- GETMAIN, which acquires a specified number of bytes for the problem program.
- LOAD, which loads a specified problem program.

Data Areas

Created: IKJEGLDF creates the SIB.

Updated: IKJEGLDF updates the ABENTAB1, TSTRETRY1, TSTDCB and SYMTABLE fields in TCOMTAB.

Consulted: IKJEGLDF consults TCOMTAB and the Dynamic Allocation Interface routine (DAIR) parameter block.

Routines Called

TSO: IKJEGLDF calls the following TSO routines:

- IKJPARS, to check the syntax of the subcommand operands.
- IKJDAIR, to allocate the data set specified in the TEST command.
- IKJDFLT, to fully qualify the data set name.
- IKJEGCVT, to convert values to printable characters and to convert addresses.
- IKJEGIO, to write data to a terminal or a data set.
- The SVC 97 routine, to put the address of TCOMTAB in the TCB.

System: IKJEGLDF calls the following system routines:

- OPEN, to initialize system routines necessary to access a data set.

- CLOSE, to release the specified data set.
- LOAD, to bring a copy of a load module into main memory.
- DELETE, to delete a copy of a load module from main storage.
- GETMAIN, to obtain main storage for work areas.
- FREEMAIN, to free main storage used as work areas.
- STAE, to establish an abnormal termination exit.
- LINK, to invoke IKJPARS.
- BLDL, to determine if the program with the specified name exists.

Register Usage during Module Execution

Reg 9	Points to TCOMTAB.
Reg 11	Used as an error message pointer.
Reg 12	Used as a base register.

Exit Information

IKJEGLDF exits to IKJEGMNL via an SVC 3 instruction pointed to by Register 14.

Output Parameters

IKJEGLDF has no output parameters.

Return Codes

4	Normal return.
16	Attention interruption encountered during processing.
20	STAE retry routine entered.

Message CSECT

IKJEGMSG is the message CSECT for IKJEGLDF.

IKJEGLDR–Load Module

Diagram 1.1 contains the method-of-operation information for this module.

Entry Information

IKJEGLDR is entered from IKJEGINT via an ATTACH macro. Control is obtained from the VS2 Dispatcher when IKJEGINT waits.

Input Parameters

IKJEGLDR receives the following input parameters:

- Register 1 points to TCOMTAB.
- The address of the TPL, contained in the TPLPTR field of TCOMTAB.
- The DCB for the problem program's data set, queued from the TSTDCB field of TCOMTAB.
- The BLDL entry obtained by IKJEGINT and pointed to by BLDLAREA (equated to the CONAREA field in TCOMTAB).
- The PDL for the TEST command pointed to by the TSTANSPL field in TCOMTAB.

Function

This module performs the following functions:

1. It determines whether the problem program is a load module or an object module.
2. It sets a pointer to TCOMTAB in the TCB of the TEST Loader It also indicates in the TCB that the TEST Loader and the problem program are a subtask of TEST.
3. It loads the problem program using either Contents Supervision via an XCTL macro instruction or the VS2 Loader, IEWLOAD, as determined by step 1.
4. It builds an input parameter list for the problem program.
5. It prompts for an input line if the CP keyword was specified in the TEST command.
6. It transfers control via an XCTL macro instruction to the entry point of the problem program to execute a pseudo breakpoint. The pseudo breakpoint passes control, via the SVC 97 routine back to IKJEGINT.

Data Areas

Created: IKJEGLDR creates the parameter list for the problem program.

Updated: IKJEGLDR updates the TCOMTAB, CPPL, ECT, and TPL control blocks.

Consulted: IKJEGLDR consults the following control blocks: TCOMTAB, TSTCWORK, CDE, PDL, CVT, ECT, TCB, TPL, and LLE.

Routines Called

TSO: IKJEGLDR calls the following TSO routines:

- The SVC 97 routine, to issue an SVC to set fields within a TCB.
- IKJEGIO, to issue an error message.
- IKJSCAN, to scan the verb field of the command being sent to the command processor.

System: IKJEGLDR calls the following system routines:

- DELETE, to delete a copy of a load module from main storage.
- FREEMAIN, to free main storage used as work areas.

Register Usage during Module Execution

Reg 9	Points to TCOMTAB.
Reg 10	Points to the TPL.
Reg 11	Used as a link register.
Reg 12	Used as a base register.

Exit Information

IKJEGLDR normally exits to the entry point of the problem program via an XCTL macro instruction. When an error occurs, it exits to IKJEGINT via a branch on Register 14.

Output Parameters

Register 1 contains a pointer to the address of the parameter list for the problem program.

Return Codes

0	Normal return.
4	Terminal I/O error encountered during processing.

Message CSECT

IKJEGXDR is the message CSECT for this module.

IKJEGLSA–LIST Subcommand Processor, Second Load Module

Diagram 2.13 contains the method-of-operation information for this module.

Entry Information

IKJEGLSA is entered from IKJEGLST at entry point IKJEGLSA via an XCTL macro instruction.

Input Parameters

Register 9 points to TCOMTAB.

Function

IKJEGLSA is the second load module that processes the LIST subcommand. It is entered from IKJEGLST when the output data type requested by the user is one of the following:

A or V	Address constant.
B	Binary.
C	Character.
D or E	Floating point.
F or H	Fixed point.
I	Instruction.
P	Packed.
S	S-type address constant.
Y	Y-type address constant.
Z	Zoned decimal.

Data Areas

Created: IKJEGLSA creates no data areas.

Updated: IKJEGLSA updates MYWORK, a work area used by both IKJEGLST and IKJEGLSA.

Consulted: IKJEGLSA consults TCOMTAB, the CVT, and the TCB of TEST.

Routines Called

TSO: IKJEGLSA calls the following TSO routines:

- IKJEGCVT, to convert values to printable characters and to convert addresses.
- IKJEGIO, to write data to a terminal or a data set.

System: IKJEGLSA calls the following system routine:

- FREEMAIN, to free main storage used as work areas.

Register Usage during Module Execution

Reg 9	Points to TCOMTAB.
Reg 10	Used as a base register for MYWORK, the work area.
Reg 12	Used as a base register.

Exit Information

IKJEGLSA exits to IKJEGMNL via an SVC 3 instruction pointed to by Register 14 or to IKJEGLST via an XCTL macro instruction.

Output Parameters

IKJEGLSA issues output to a terminal or a data set.

Return Codes

0 Normal return.

16 Attention interruption encountered during processing.

20 STAE retry routine entered.

Message CSECT

IKJEGMSB is the message CSECT for IKJEGLSA.

IKJEGLST–LIST Subcommand Processor, First Load Module

Diagram 2.13 contains the method of operation information for this module.

Entry Information

IKJEGLST is entered from th: IKJEGMNL via a LINK macro instruction at entry point IKJEGLST. IKJEGLST is also entered from IKJEGLSA via an XCTL macro instruction.

Input Parameters

IKJEGLST receives the contents of the input buffer, and Register 9 points to TCOMTAB.

Function

IKJEGLST processes the LIST subcommand of TEST. This subcommand is used to write in hexadecimal format the contents of an area of main storage or the contents of registers.

IKJEGLST is the first of a double load module that processes the subcommand, and it performs three functions.

- It establishes limits and default values for the type of data requested by the user.
- It lists the contents of the general and floating point registers.
- It lists the requested contents of main storage in hexadecimal format.

Data Areas

Created: IKJEGLST creates MYWORK and the work areas used by both IKJEGLST and IKJEGLSA.

Updated: IKJEGLST updates no data areas.

Consulted: IKJEGLST consults TCOMTAB, the PDL, the TCB of TEST, and the CVT.

Routines Called

TSO: IKJEGLST calls the following TSO routines:

- IKJPARS, to check the syntax of the subcommand operands and to build a PDE for the PDL.
- IKJEGCVT, to convert values and addresses to printable form.
- IKJEGIO, to write data to a terminal or a data set.

System: IKJEGLST calls the following system routines:

- STAE, to establish an abnormal termination exit.
- LINK, to invoke IKJPARS.
- XCTL, to transfer control to IKJEGLSA.

Register Usage During Module Execution

Reg 9 Points to TCOMTAB.

Reg 10 Used as a base register for MYWORK, the work area.

Reg 12 Used as a base register.

Exit Information

IKJEGLST exits to IKJEGLSA via an XCTL macro instruction if the data being considered is not hexadecimal data or register information.

Upon completion of the needed function, exit is to IKJEGMNL via an SVC instruction pointed to by Register 14.

Output Parameters

IKJEGLST issues output to a terminal or a data set.

Return Codes

0 Normal return.

16 Attention interruption encountered during processing.

20 STAE retry routine entered.

Message CSECT

IKJEGMSA is the message CSECT for IKJEGLST.

IKJEGMAP-LISTMAP Subcommand Processor

Diagram 2.16 contains the method-of-operation information for this module.

Entry Information

IKJEGMAP is entered from IKJEGMNL by a LINK macro instruction.

Input Parameters

IKJEGMAP receives the contents of the input buffer, and Register 9 points to TCOMTAB.

Function

IKJEGMAP processes the LISTMAP subcommand of TEST. This subcommand is used to write to a terminal or to a data set the following information: the user's region size; address of the problem program's TCB; the names, addresses, and lengths of all programs running under the problem programs TCB; the type and program identification of all active RBs; and the user subpool numbers, locations, and number of bytes used.

IKJEGMAP gathers the specified information by means of a series of pointers. It formats the data and prepares it to be written.

Data Areas

Created: IKJEGMAP creates no data areas.

Updated: IKJEGMAP updates no data areas.

Consulted: IKJEGMAP consults TCOMTAB, the CVT, the TCB of TEST, the LLE (load list element), the CDE (contents directory entry), the RB, the SPQE (subpool queue element), the PQE (partition queue element), the DQE (descriptor queue element), and the XTLST (extent list).

Routines Called

TSO: IKJEGMAP calls the following TSO routines

- IKJPARS, to check the syntax of the subcommand operands and to build a PDE for the PDL.
- IKJEGCVT, converts values and addresses to printable characters.
- IKJEGIO, to write data to a terminal or a data set.

System: IKJEGMAP calls the following system routines:

- STAE, to establish an abnormal termination exit.
- FREEMAIN, to free main storage used as work areas.

Register Usage during Module Execution

Reg 9	Points to TCOMTAB.
Reg 12	Used as a base register.

Exit Information

IKJEGMNL exits to IKJEGMNL via an SVC instruction pointed to by Register 14.

Output Parameters

IKJEGMAP issues output to a terminal or a data set.

Return Codes

0	Normal return.
16	Attention interruption encountered during processing.
20	STAE retry routine entered.

Message CSECT

IKJEGMSG is the message CSECT for IKJEGMAP.

IKJEGMNL–Mainline Module

Diagrams 1, 2, and 3 contain the method-of-operation information for this module.

Entry Information

IKJEGMNL can be entered at six places:

- The initial entry to IKJEGMNL is for TEST initialization at location IKJEGMNL.
- Entry can be from the SVC 97 routine, IGC0009G, after a pseudo or user breakpoint has been encountered. Linkage is via the POST and WAIT macro instructions.
- Entry can be from the SVC 61 routine, IGC0006A, after a module of the problem program has been fetched for Contents Supervision or the Overlay Supervisor. Linkage is via the POST and WAIT macro instructions.
- Entry can be from the TMP STAI Exit routine, IKJEFT04, after the problem program has begun to terminate abnormally during the test session. Linkage is via the POST and WAIT macro instructions.
- Entry can be from the ABEND/STAE Interface routine, IGC0D01C, afer a STAE-intercepted ABEND in TEST. This is done via the OS/VS2 Dispatcher.

- Entry can be from the EOT routine of the Supervisor via the POST macro instruction.

Input Parameters

IKJEGMNL receives a parameter list from the TMP STAI exit routine, and Reigster 1 points to TCOMTAB.

Function

IKJEGMNL performs the following functions:

- It completes the initialization of TCOMTAB.
- It issues the STATUS macro instruction to stop and start the problem program.
- It qualifies the problem program automatically on initial entry and each time that IKJEGMNL receives control from the SVC 97 routine because of a breakpoint that was encountered in a new module. If the previous module of the problem program attaches, links, or transfers control to another module, and then encounters a breakpoint, IKJEGMNL qualifies to the second module. An ABEND or a attention interrupt that occurs during execution of the problem program can cause requalification, since these conditions result in a psuedo breakpoint.
- It gets a subcommand.
- It tests the return code from the subcommand processor that executed.
- It waits for an event to occur, as a result of the WAIT macro instruction.
- It makes decisions after an abnormal termination in the TEST command processor.
- It terminates TEST and returns to the TMP.

Data Areas

Created: IKJEGMNL creates the control chain word for the SVC 61 routine. The control chain word is queued from the TSTTRN field of TCOMTAB and is created only if the problem program has more than one task. The first control chain word is always built by IKJEGINT.

Updated: IKJEGMNL updates TCOMTAB, the TPL, and subtask TCBs; IKJEGMNL frees the PDL, SIB, the symbol table built in storage, the input buffer, the TSTDCB queue, WORKAREA, TCOMTAB, CONAREA, OUTBUF, and REGSAVE1 through REGSAVE6.

Consulted: IKJEGMNL consults TCOMTAB, the break element queue, the defer element queue, and the TPL.

Routines Called

TSO: IKJEGMNL calls the following TSO routines:

- IKJEGIO, to obtain subcommands and to issue messages.
- The SVC 97 routine, to alter request block fields.
- IKJDAIR, to unallocate data sets allocated by TEST modules.
- IKJRLSA, to free the PDL used by the previously executed subcommand processor.
- STACK, to add and remove a terminal element from the input stack.
- Subcommand processors of the TEST command.

System: IKJEGMNL calls the following system routines:

- STAE, to create and queue a STAE control block for possible scheduling of the STAE Exit routine of TEST during IKJEGMNL execution.

- STAX, to create and queue a terminal attention exit element for use in the scheduling of the TEST attention exit routine during IKJEGMNL execution.
- GETMAIN, to obtain main storage for a control element for use by the SVC 61 routine when it queues SVC information blocks.
- FREEMAIN, to free the input buffer before obtaining the next subcommand; also to release TEST's storage before IKJEGMNL retruns to the TMP.
- LINK, to invoke a subcommand processor; also to call IKJSCAN and IKJDAIR.
- POST, to prepare to return control to the SVC 97 routine, the SVC 61 routine, or the TMP's STAI exit routine when IKJEGMNL has received control via a POST macro instruction from one of these routines.
- STATUS, to start or stop a subtask.
- WAIT, to stop IKJEGMNL execution and force the OS/VS2 Dispatcher to give control to the subtask.
- DELETE, to remove the IKJEGLDF module after IKJEGMNL has restored the problem program's registers for IKJEGLDF.
- CLOSE, to close any open DCB after the LOAD subcommand has been invoked.

Register Usage during Module Execution

Reg 9 Points to TCOMTAB.

Reg 10 Points to the TPL.

Reg 11 Used as a link register.

Reg 12 Used as a base register.

Exit Information

IKJEGMNL returns control to the TMP when one of the following events occurs:

- The RUN or END subcommand has been specified.
- The problem program task has no remaining request blocks.
- The user has entered a new command after pressing the attention key twice.
- An unrecoverable error has been detected by a TEST module.
- A critical abnormal termination occurred in IKJEGMNL.

Output Parameters

If the problem program begins to terminate abnormally, IKJEGMNL returns the contents of the TPLNECB field of the TPL to the STAI Exit routine of the TMP. This field points to an ECB containing a retry POST code of X'7F' and the retry address.

Return Codes

0 Normal return.

Message CSECT

IKJEGXNL is the message CSECT for IKJEGMNL.

IKJEGOFF–OFF Subcommand Processor

Diagram 2.20 contains the method-of-operation information for this module.

Entry Information

IKJEGOFF is entered from IKJEGMNL via a LINK macro instruction or from IKJEGEND via an XCTL macro instruction.

Input Parameters

IKJEGOFF receives the contents of the input buffer, and Register 9 points to TCOMTAB.

Function

IKJEGOFF clears specified breakpoints from the problem program if address operands are included in the subcommand. All breakpoints are removed if no address operand is included, or if the end of TEST processing is indicated. The removed breakpoint is replaced with the original saved instruction.

Data Areas

Created: IKJEGOFF creates no data areas.

Updated: IKJEGOFF updates the break element queue and the defer element queue.

Consulted: IKJEGOFF consults the break element queue, the defer element queue, TCOMTAB, and the address of the PDE.

Routines Called

TSO: IKJEGOFF calls the following TSO routines:

- IKJPARS, to check the syntax of the subcommand operands.
- IKJEGIO, to issue messages.
- The SVC 97 routine, to alter the problem program's request blocks.
- IKJEGSRH, to search for and remove breakpoints from the active queue.
- IKJEGBDL, to add an address string to an error message.
- IKJRLSA, to free the PDL.
- IKJEGCVT, to convert values and addresses to printable characters.

System: IKJEGOFF calls the following system routines:

- FREEMAIN, to free queue elements being removed.
- STAE, to establish an abnormal termination exit.
- LINK, to invoke IKJPARS.

Register Usage during Module Execution

Reg 9	Points to TCOMTAB.
Reg 10	Used as a base register for the work area.
Reg 12	Used as a base register
Reg 13	Points to save area used by IKJEGOFF.

Exit Information

IKJEGOFF exits to IKJEGMNL. Register 14 points to an SVC instruction that allows indirect return.

Output Parameters

IKJEGOFF has no output parameters.

Return Codes

0	Normal return or an error condition exists.

16 An attention interruption occurred.

20 IKJEGOFF retry routine entered.

Message CSECT

IKJEGMSG is the message CSECT for IKJEGOFF.

IKJEGPCH-Assignment Subcommand Processor, First Load Module

Diagram 2.2 contains the method-of-operation information for this module.

Entry Information

IKJEGPCH is entered at any one of three locations. IKJEGMNL enters at entry point IKJEGPCH via a LINK macro instruction. IKJPARS enters at either PCHVALAD or PCHVALTP via a branch instruction.

Input Parameters

IKJEGPCH receives the contents of the input buffer, and Register 9 points to TCOMTAB.

Function

IKJEGPCH is the first of two load modules that process the assignment function of TEST. The assignment function changes values in memory and in registers. IKJEGPCH initializes the work area used by IKJEGASN, the second of the two load modules. IKJEGPCH issues a LINK macro instruction to IKJPARS to validate the operands. Upon completion, IKJEGPCH issues an XCTL macro instruction to IKJEGASN.

Data Areas

Created: IKJEGPCH creates PCHWORK, its work area.

Updated: IKJEGPCH updates TCOMTAB, the VALVPDE, the PDL, and the PPL.

Consulted: IKJEGPCH consults TCOMTAB, TSTCWORK, the VALVPDE, the TCB, the PDL, the CVT, and PCHWORK.

Routines Called

TSO: IKJEGPCH calls the following TSO routines:

- IKJPARS, to check the syntax of the subcommand operands.
- IKJEGCVT, to covert addresses to binary.
- IKJEGIO, to issue output to the terminal.

System: IKJEGPCH calls the following system routines:

- STAE, to establish an abnormal termination exit.
- LINK, to invoke IKJPARS.
- FREEMAIN, to free main storage used as work areas.
- XCTL, to transfer control to IKJEGASN.

Register Usage during Module Execution

Reg 9 Points to TCOMTAB.

Reg 11 Used as a base register.

Exit Information

IKJEGPCH returns to IKJEGMNL by a branch instruction, if an error occurs during IKJPARS or IKJEGCVT execution, or if an attention interruption occurs, or if the STAE retry routine is executed.

IKJEGPCH exits to IKJEGASN via an XCTL macro instruction.

Output Parameters

IKJEGPCH has no output parameters.

Return Codes

0 Normal return.

16 Attention interruption encountered during processing.

20 STAE retry routine entered.

Message CSECT

IKJEGMSG is the message CSECT for IKJEGPCH.

IKJEGPSW–LISTPSW Subcommand Processor

Diagram 2.17 contains the method-of-operation information for this module.

Entry Information

IKJEGPSW is entered by IKJEGMNL at entry point IKJEGPSW via a LINK macro instruction.

Input Parameters

IKJEGPSW receives the contents of the input buffer, and Register 9 points to TCOMTAB.

Function

IKJEGPSW processes the LISTPSW subcommand of TEST. The subcommand writes to a terminal or to a data set the contents of a program status word.

IKJEGPSW checks bit 12 of the PSW to determine the mode. If bit 12 contains a 1, the PSW is formatted in EC (Extended Control) mode. If bit 12 contains a 0, the PSW is formatted in BC (Basic Control) mode.

Data Areas

Created: IKJEGPSW creates PSWAREA, its work area.

Updated: IKJEGPSW updates no data areas.

Consulted: IKJEGPSW updates TCOMTAB.

Routines Called

TSO: IKJEGPSW calls the following TSO routines:

- IKJPARS, to check the syntax of the subcommand operands.
- IKJEGCVT, to convert values to printable characters and to convert addresses.
- IKJEGIO, to write data to a terminal or a data set.
- IKJEGBLD, to build a character symbolic address.

System: IKJEGPSW calls the following system routines:

- STAE, to establish an abnormal termination exit.
- LINK, to invoke IKJPARS.
- FREEMAIN, to free main storage used as work areas.

Register Usage During Module Execution

Reg 5	Points to PSWAREA.
Reg 9	Points to TCOMTAB.

Reg 12 Used as a base register.

Exit Information

IKJEGPSW exits to IKJEGMNL via an SVC 3 instruction pointed to by Register 14.

Output Parameters

IKJEGPSW issues output lines to a terminal or a data set.

Return Codes

0 Normal return.

16 Attention interruption encountered during processing.

20 STAE retry routine entered.

Message CSECT

IKJEGPWM is the message CSECT for IKJEGPSW.

IKJEGQFY-QUALIFY Subcommand Processor

Diagram 2.21 contains the method-of-operation information for this module.

Entry Information

IKJEGQFY is entered from IKJEGMNL via a LINK macro instruction at entry point IKJEGQFY.

Input Parameters

IKJEGQFY receives the contents of the input buffer, and Register 9 points to TCOMTAB.

Function

IKJEGQFY updates the fields in TCOMTAB specified by the user. Fields that can be updated include:

- PPTCB, which is the address of the current TCB.
- PPLOAD, which is the loaded address of the currently qualified load module.
- PPRB, which is the address of the current request block.
- TSTCURLD, which is the EBCDIC name of the currently qualified load module.
- TSTCURCT, which is the EBCDIC name of the currently qualified CSECT for the symbolic addresses.
- TSTSYMBA, which is the base address for the symbolic addresses.

Data Areas

Created: IKJEGQFY creates no data areas.

Updated: IKJEGQFY updates TCOMTAB.

Consulted: IKJEGQFY consults TCOMTAB.

Routines Called

TSO: IKJEGQFY calls the following TSO routines:

- IKJPARS, to check the syntax of the subcommand operands.
- IKJEGCVT, to convert values to printable characters and to convert addresses.
- IKJEGIO, to write data to a terminal or a data set.

System: IKJEGQFY calls the following system routines:

- LINK, to invoke IKJPARS.
- STAE, to establish an abnormal termination exit.

Register Usage during Module Execution

Reg 9	Points to TCOMTAB.
Reg 10	Points to the work area used by IKJEGQFY.
Reg 12	Used as a base register.

Exit Information

IKJEGQFY exits via an SVC 3 instruction pointed to by Register 14.

Output Parameters

IKJEGQFY has no output parameters.

Return Codes

0	Normal return.
4	Qualification is to a new TCB; control returns to the problem program.
16	Attention interruption encountered during processing.
20	STAE retry routine entered.

Message CSECT

IKJEGMSG is the message CSECT for IKJEGQFY.

IKJEGSRH–SEARCH Module

Diagram 2.29 contains the method-of-operation information for this module.

Entry Information

IKJEGSRH is entered from a number of subcommand processors via a branch instruction. Subcommand processors using IKJEGSRH include the AT, COPY, OFF, and Assignment subcommand processors.

Input Parameters

Reg 1 contains a pointer to three words. The first word contains flags. The flag 00 indicates to search for an address or range and delete the breakpoint elements. The flag 04 indicates to search for an address and return the address of the corresponding breakpoint queue element. The second word is an address of the two bytes containing the breakpoint or the start of a range. The third word is the end of a range or zero.

Function

When given an address or a range, IKJEGSRH searches the active breakpoint queue for a match or for an address within the range. If removal was indicated, the break element is removed from the queue and an attempt is made to restore the saved instruction into the problem program.

If only search was indicated, the address of this element is returned to the caller.

Data Areas

Created: IKJEGSRH creates no data areas.

Updated: IKJEGSRH updates the breakpoint element queue and TCOMTAB.

Consulted: IKJEGSRH consults the breakpoint element queue and TCOMTAB.

Routines Called

TSO: IKJEGSRH calls the following TSO routine:

- The SVC 97 routine, to check the validity of an address.

System: IKJEGSRH calls the following system routine:

- FREEMAIN, to free queue elements being removed.

Register Usage During Execution

Reg 9	Points to TCOMTAB.
Reg 8	Used as a base register for the break element queue.
Reg 12	Used as a base register.

Exit Information

IKJEGSRH returns to the caller via a branch to the address in Register 14.

Output Parameters

When only search is indicated, IKJEGSRH returns the element address in the second word of the input parameter block.

Return Codes

0	Normal return.
4	No elements found.
8	Invalid range detected.
12	STAE Retry routine invoked.

Message CSECT

IKJEGSRH has no message CSECT.

IKJEGSTA-STAE Exit Routine

Diagram 2.31 contains the method-of-operation information for this module.

Entry Information

IKJEGSRH is entered from the ABEND/STAE Interface routine (IGC0B01C).

Input Parameters

Reg 1 points to the work area if space was available.

Reg 0 contains a X'12' if no space was available for a work area.

The fields TSTRETRY, ABENTAB, and ABENTAB1 in TCOMTAB also contain input parameters and can be found from TEST's TCB which has a pointer to TCOMTAB.

Function

IKJEGSTA determines the terminating TEST module and prints a standard first-level and second-level diagnostic message. The first level message contains either the name of the terminating subcommand, or "TEST" if a resident TEST module is terminating.

IKJEGSTA adds an error description to the second-level message. It gets the error description by using a table of predictable ABEND addresses to index a list of error descriptions in a message CSECT.

IKJEGSTA obtains the name of the terminating subcommand. by examining the high order byte of the TSTRETRY field of TCOMTAB. If zero, a resident TEST module is terminating. If not zero, the field is used as a search argument for a search of the module list CSECTs. Two module list CSECTs exist, one for IBM subcommands and one for user subcommands. The list for IBM subcommands is searched first. If a match is found in either list, IKJEGSTA places the found subcommand name in the first level message.

IKJEGSTA obtains the address of the termination routine belonging to the terminating TEST module and passes the address to the ABEND/STAE Interface routine. The TSTRETRY field contains the address of the termination routine. Linkage is then established to the termination routine.

Data Areas

Created: IKJEGSTA creates no data areas.

Updated: IKJEGSTA updates no data areas.

Consulted: IKJEGSTA consults the following data areas:

- The ABENTAB and ABENTAB1 fields in TCOMTAB are consulted.
- The TCB of TEST is used to get a pointer to TCOMTAB. It is also used to get the address of the request block.
- The request block is used to find the PSW address associated with the ABEND condition.
- Module List CSECTs are used to determine the name of the terminating subcommand.
- The instruction counter table is used to determine the error description to be added to the standard second-level diagnostic message.

Routines Called

TSO: IKJEGSTA calls the following TSO routine:

- IKJEGIO, to write data to a terminal or a data set.

System: IKJEGSTA calls no system routines.

Register Usage during Module Execution

Reg 1 Used as a parameter register when IKJEGSTA invokes the I/O routine, IKJEGIO; points to an output buffer from which diagnostic messages are issued.

Reg 9 Points to TCOMTAB.

Reg 10 Points to a work area.

Reg 11 Used as a base register.

Reg 14 Used as an internal link register.

Exit Information

IKJEGSTA exits to the ABEND/STAE Interface routine via an SVC 3 instruction pointed to by Register 14.

Output Parameters

Reg 0 Contains the address of the retry routine to be given control by the ABEND/STAE Interface routine.

Reg 1 Contains the address of the STAE work area, or zero if no work area was made available to IKJEGSTA by the ABEND/STAE Interface routine.

Reg 14 Contains the return address set by the ABEND/STAE Interface routine.

Reg 15 Contains a return code of four. This indicates that the ABEND/STAE Interface routine is to give control to a cleanup (retry) routine whose address is in Register 0.

Return Codes

4 Indicates that the ABEND/STAE Interface routine is to give control to a termination routine whose address is in Register 0.

Message CSECT

IKJEGMS1 is the message CSECT for IKJEGSTA.

IKJEGSYM–Symbol Module

Diagram 2.25 contains the method-of-operation information for this module. Figures 11 and 12 provide additional information.

Entry Information

IKJEGSYM is entered from the Where, Qualify, or Convert modules.

Input Parameters

Register 1 points to a parameter descriptor entry (PDE) that contains:

Loadname: (In PDELDNAM)	Indicates request
CSECT name: (In PDECTNAM)	for symbol
Symbol: (In PDEADRPT)	information
Loadname: (In PDELDNAM)	Indicates request
Offset: (In PDEADRPT)	for CSECT name
Loadname: (In PDELDNAM)	Indicates request
Entryname: (In PDECTNAM)	for entry point address

Function

IKJEGSYM obtains symbol information by first searching the symbol table in memory built by the EQUATE subcommand. If this search fails, the assembler SYM records on the data set from which the specified module was fetched are searched.

IKJEGSYM obtains a CSECT name or an entry point address by searching CESD records on auxiliary storage for a load module or CESD records in memory for an object module.

Data Areas

Created: IKJEGSYM creates the symbol information block and the internal work area.

Updated: IKJEGSYM updates the PDE and the internal work areas.

Consulted: IKJEGSYM consults a contents directory entry, the extent list, TCOMTAB, symbol table elements in memory, the PDE, the SVC information block, SYM and CESD records on auxiliary memory, and CESD records in main memory.

Routines Called

TSO: IKJEGSYM calls the following TSO routines:

- IKJEGIO, to issue error messages to the terminal.
- IKJDAIR, to allocate SYS1.LINKLIB if the problem program was fetched from that data set.

System: IKJEGSYM calls the following system routines:

- GETMAIN, to allocate storage for the work area, symbol information block, and IKJDAIR parameter list.
- FREEMAIN, to free the IKJDAIR parameter list and the STAE work area.
- OPEN, to open BPAM DCB to read SYM records.

- CLOSE, to close BPAM DCB after reading SYM records.
- READ, to read SYM records.
- NOTE, to obtain the TTR of the first CESD record in the member associated with the load name.
- FIND, to locate symbol records.
- SYMADAF, in case of an I/O error to get the system information message.
- SYNADRLS, to release the SYMADAF work area.

Register Usage during Module Execution

Reg 2 Point to the address of the passed PDE.

Reg 8 Used as a base register for the message section.

Reg 9 Points to TCOMTAB.

Reg 10 Used as a base register for the information block.

Reg 11 Used as a base register for IKJEGMYS.

Reg 12 Used as a base register for IKJEGSYM.

Reg 13 Used as a base register for the SYM work area.

Exit Information

IKJEGSYM returns control to the caller.

Output Parameters

The PDEUSER field of the PDE contains symbol information and the address of the entry name. The PDECTNAM field of the PDE points to the CSECT name in the work area.

Return Codes

The return codes for this module follow.

0 Normal return.

4 Input symbolic name is invalid. Either loadname, entryname, symbol, or insufficient memory causes this.

8 Unable to resolve symbol at this time, cannot open DCB, or cannot allocate SYS1.LINKLIB.

12 An I/O error has occurred while reading a SYM record.

16 An attention interruption occurred, or the LOGON/LOGOFF ECB issued a POST macro instruction.

20 An ABEND occurred and was intercepted by IKJEGSTA.

Message CSECT

IKJEGMSG is the message CSECT for IKJEGSYM.

IKJEGTCB–LISTTCB Subcommand Processor

Diagram 2.18 contains the method-of-operation information for this module.

Entry Information

IKJEGTCB is entered from IKJEGMNL at the entry point IKJEGTCB via a LINK macro instruction.

Input Parameters

IKJEGTCB receives the contents of the input buffer, and Register 9 points to TCOMTAB.

Function

IKJEGTCB lists the contents of a Task Control Block to a terminal or to a user-specified data set. To do this, IKJEGTCB performs the following functions:

- Issues a LINK macro instruction to IKJPARS to create a parameter descriptor entry for each TCB field requested by the user.
- Retrieves the data requested by the user.
- Converts data to printable hexadecimal and prints it.
- Returns control to IKJEGMNL.

Data Areas

Created: IKJEGTCB creates its work area, MYWKAREA.

Updated: IKJEGTCB updates no data areas.

Consulted: IKJEGTCB consults TCOMTAB, the CVT, the TCB of TEST, and the TCB specified for the list function (default is to the subtask's TCB).

Routines Called

TSO: IKJEGTCB calls the following TSO routines:

- IKJPARS, to check the syntax of the subcommand operands.
- IKJEGCVT, to convert values to printable characters and to convert addresses.
- IKJEGIO, to write data to a terminal or a data set.
- IKJEGBLD, to build a character symbolic address.

System: IKJEGTCB calls the following system routines:

- LINK, to invoke IKJPARS.
- STAE, to establish an abnormal termination exit.
- FREEMAIN, to free the STAE work area.

Register Usage during Module Execution

Reg 3	Points to the TCB being processed.
Reg 5	Points to MYWKAREA.
Reg 9	Points to TCOMTAB.
Reg 12	Used as a base register.

Exit Information

IKJEGTCB exits to IKJEGMNL via an SVC 3 instruction pointed to by Register 14.

Output Parameters

Output from IKJEGTCB consists of data issued to a terminal or to a data set.

Return Codes

Return codes for this module follow.

0	Normal return.
16	Attention interruption encountered during processing.
20	STAE retry routine entered.

Message CSECT

IKJEGMSG is the message CSECT for IKJEGTCB.

IKJEGWHR–WHERE Subcommand Processor

Diagram 2.23 contains the method-of-operation information for this module.

Entry Information

IKJEGWHR is entered from IKJEGMNL at entry point IKJEGWHR via a LINK macro instruction.

Input Parameters

IKJEGWHR receives the contents of the input buffer, and Register 9 points to TCOMTAB.

Function

IKJEGWHR processes the WHERE subcommand of TEST. This processor obtains the absolute address either of a symbolic or a relative address in a problem program or the absolute address of an entry point in a load module or CSECT. IKJEGWHR also obtains the absolute address of the problem program interruption.

Data Areas

Created: IKJEGWHR creates no data areas.

Updated: IKJEGWHR updates no data areas.

Consulted: IKJEGWHR consults TCOMTAB.

Routines Called

TSO: IKJEGWHR calls the following TSO routines:

- IKJPARS, to check the syntax of the subcommand operands.
- IKJEGCVT, to convert values to printable characters and to convert addresses.
- IKJEGIO, to write data to a terminal or a data set.
- IKJEGSYM, to access a CSECT name in the indicated load module and to search the symbol table for a given load name CSECT.

System: IKJEGWHR calls the following system routines:

- LINK, to invoke IKJPARS.
- STAE, to establish an abnormal termination exit.
- FREEMAIN, to free main storage used as work areas.

Register Usage during Module Execution

Reg 9	Points to TCOMTAB.
Reg 12	Used as a base register.

Exit Information

IKJEGWHR exits to IKJEGMNL via a branch to the address in Register 14.

Output Parameters

Output from IKJEGWHR consists of lines written to a terminal.

Return Codes

4	Normal return.
16	Attention interruption encountered during processing.
20	STAE retry routine entered.

Message CSECT

IKJEGWHR has no message CSECT.

Section 4: Directory

This section contains an alphamerically organized list of entry names and CSECT names. Each name is defined and cross-referenced to its assembly module and load module.

Module Cross Reference Directory

Entry Name or CSECT Name	Assembly Module Name	Load Module Name	Assembly Modules That Refer to Entry Name or CSECT Name	Definition of Entry Name, CSECT Name, or Associated Code
ADDRCHK	IKJEGGO	IKJEGGO	IKJPARS	Validity check exit routine.
IGC0006A	IGC0006A	IGC0006A	IKJEGMNL Contents Supervision (IEAQLK00)	Major entry point of SVC 61 routine.
IGC0009G	IGC0009G	IGC0009G	Any module of TEST, when SVC 97 routine is used as a service subroutine. Any module of tested program.	Major entry point of SVC 97 routine.
IKJEGAPL	IKJEGAT	IKJEGAT	IKJPARS	Parameter control list CSECT.
IKJEGASN	IKJEGASN	IKJEGASN	IKJEGSCD IKJEGMNL	Major CSECT of second load module of Assignment subcommand processor. Is entered from IKJEGPCH by an XCTL macro instruction.
IKJEGAT	IKJEGAT	IKJEGAT	IKJEGMNL IKJGGSCD	Major CSECT of module IKJEGAT. Sets active breakpoints in problem program.
IKJEGATD	IKJEGATD	IKJEGATD	IKJEGAT	Major CSECT of module IKJEGATD. Receives control from module IKJEGAT, via an XCTL macro instruction, to set deferred breakpoints.
IKJEGATM	IKJEGAT	IKJEGAT	IKJEGMNL	Message CSECT of module IKJEGAT.
IKJEGATN	IKJEGATN	IKJEGATN	IKJEAR05 (Attention Handler in region control task)	Major CSECT of module IKJEGATN.
IKJEGCAL	IKJEGGO	IKJEGGO	IKJEGMNL IKJEGSCD	Entry point of the CALL subcommand processor.
IKJEGCDL	IKJEGCVT	IKJEGMNL	IKJEGCVT	Contains list of valid two-byte OP codes. Is used for validity checking.
IKJEGCOP	IKJEGCVT	IKJEGMNL	IKJEGCVT	Contains list of valid OP codes. Is used for validity checking. One byte.

Module Cross Reference Directory

Entry Name or CSECT Name	Assembly Module Name	Load Module Name	Assembly Modules That Refer to Entry Name or CSECT Name	Definition of Entry Name, CSECT Name, or Associated Code
IKJEGCPY	IKJEGCPY	IKJEGCPY	IKJEGMNL IKJEGSCD	Major entry point for the COPY subcommand processor.
IKJEGCVT	IKJEGCVT	IKJEGMNL	IKJEGMNL IKJEGSCD	Major entry point of module IKJEGCVT. Converts addresses to binary and printable characters. Converts values to printable and binary format.
IKJEGCVX	IKJEGCVT	IKJEGMNL	IKJEGCVT	CSECT in module IKJEGCVT. Converts instructions to printable characters.
IKJEGDA1	IKJEGWHR	IKJEGWHR	IKJEGWHR	Contains external character data used to build a data message.
IKJEGDA2	IKJEGWHR	IKJEGWHR	IKJEGWHR	Contains external character data used to build a data message.
IKJEGDA3	IKJEGWHR	IKJEGWHR	IKJEGWHR	Contains external character data used to build a data message
IKJEGDA4	IKJEGWHR	IKJEGWHR	IKJEGWHR	Contains external character data used to build a data message.
IKJEGDBM	IKJEGDEB	IKJEGDEB	IKJEGDEB	Message CSECT.
IKJEGDBP	IKJEGDEB	IKJEGDEB	IKJPARS	Parameter control list CSECT.
IKJEGDCB	IKJEGDCB	IKJEGDCB	IKJEGMNL IKJEGSCD	Major entry point of LISTDCB subcommand processor.
IKJEGDDM	IKJEGIO	IKJEGMNL	IKJEGMNL IKJEGINT IKJEGLDF IKJEGIO	CSECT used to create diagnostic messages after an unseccessful return from IKJDAIR.
IKJEGDEB	IKJEGDEB	IKJEGDEB	IKJEGMNL IKJEGSCD	Major entry point of LISTDEB subcommand processor.
IKJEGDEL	IKJEGLDF	IKJEGLDF	IKJEGMNL IKJEGSCD	Entry point of DELETE subcommand processor.
IKJEGDLT	IKJEGLDF	IKJEGLDF	IKJPARS	Parameter control list CSECT for DELETE subcommand processor.

Module Cross Reference Directory

Entry Name or CSECT Name	Assembly Module Name	Load Module Name	Assembly Modules That Refer to Entry Name or CSECT Name	Definition of Entry Name, CSECT Name, or Associated Code
IKJEGDMG	IKJEGIO	IKJEGMNL	IKJEGMNL	Message CSECT.
IKJEGDRP	IKJEGEQU	IKJEGEQU	IKJEGMNL IKJEGSCD	Entry point of the DROP subcommand processor.
IKJEGDVK	IKJEGDCB	IKJEGDCB	IKJPARS	Validity check routine.
IKJEGEND	IKJEGEND	IKJEGEND	IKJEGMNL IKJEGSCD	Entry point for the END subcommand processor.
IKJEGEQU	IKJEGEQU	IKJEGEQU	IKJEGMNL IKJEGSCD	Entry point of EQUATE subcommand processor.
IKJEGFLT	IKJEGCVT	IKJEGMNL	IKJEGCVT	Converts floating point data to and from binary.
IKJEGFM	IKJEGCPY	IKJEGCPY	IKJPARS IKJEGCVT	Entry point for the validity check routine that tests the "from" address (address1) in the COPY subcommand processor.
IKJEGFRE	IKJEGLDF	IKJEGLDF	IKJEGMNL IKJEGSCD	Entry point of FREEMAIN subcommand processor.
IKJEGFRL	IKJEGLDF	IKJEGLDF	IKJPARS	Parameter control list CSECT for FREEMAIN subcommand.
IKJEGGET	IKJEGLDF	IKJEGLDF	IKJEGMNL	Entry point of the GETMAIN subcommand processor.
IKJEGGO	IKJEGGO	IKJEGGO	IKJEGMNL IKJEGSCD	Entry point of the GO subcommand processor.
IKJEGGT	IKJEGIO	IKJEGMNL	Any TEST module, except IKJEGATN	Entry point to obtain input from the terminal for any TEST module except IKJEGATN
IKJEGGTM	IKJEGLDF	IKJEGLDF	IKJPARS	Parameter control list subcommand.
IKJEGINT	IKJEGINT	IKJEGINT	IKJEFT02 (TMP Mainline) IGC0009G	Major entry point of TEST Initialization module.
IKJEGIO	IKJEGIO	IKJEGMNL	IKJEGMNL	Major entry point of I/O module of TEST.
IKJEGLDF	IKJEGLDF	IKJEGLDF	IKJEGMNL IKJEGSCD	Major CSECT of module IKJEGLDF.
IKJEGLDP	IKJEGLDF	IKJEGLDF	IKJPARS	Parameter control list CSECT for LOAD subcommand.

Module Cross Reference Directory

Entry Name or CSECT Name	Assembly Module Name	Load Module Name	Assembly Modules That Refer to Entry Name or CSECT Name	Definition of Entry Name, CSECT Name, or Associated Code
IKJEGLDR	IKJEGLDR	IKJEGLDR	IKJEGINT	Major entry point of module IKJEGLDR.
IKJEGLN	IKJEGCPY	IKJEGCPY	IKJPARS IKJEGCVT	Entry point of the validity check routine that tests the LENGTH keyword of the COPY subcommand.
IKJEGLOD	IKJEGLDF	IKJEGLDF	IKJEGMNL IKJEGSCD	Entry point of LOAD subcommand processor.
IKJEGLSA	IKJEGLSA	IKJEGLSA	IKJEGSCD IKJEGMNL IKJEGLST	Major entry point of second load module of LIST subcommand processor.
IKJEGLST	IKJEGLST	IKJEGLST	IKJEGSCD IKJEGMNL IKJEGLSA	Major entry point of first load module of LIST subcommand processor.
IKJEGMAP	IKJEGMAP	IKJEGMAP	IKJEGMNL IKJEGSCD	Major entry point of LISTMAP subcommand processor.
IKJEGMGC	IKJEGCPY	IKJEGCPY	IKJEGCPY	Message CSECT.
IKJEGMNL	IKJEGMNL	IKJEGMNL	IKJEGINT	Major entry point of TEST IKJEGMNL module.
IKJEGMSA	IKJEGLDF	IKJEGLDF	IKJEGLDF	Message CSECT.
IKJEGMSA	IKJEGLST	IKJEGLST	IKJEGLST	Message CSECT.
IKJEGMSA	IKJEGOFF	IKJEGOFF	IKJEGOFF	Entry name in message CSECT in module IKJEGOFF.
IKJEGMSB	IKJEGLSA	IKJEGLSA	IKJEGLSA	Message CSECT.
IKJEGMSB	IKJEGLDF	IKJEGLDF	IKJEGMNL IKJPARS IKJDAIR IKJDFLT	Entry name in message CSECT in module IKJEGLDF.
IKJEGMSC	IKJEGLDF	IKJEGLDF	IKJEGLDF	Entry name in message CSECT in module IKJEGLDF.
IKJEGMSC	IKJEGPCH	IKJEGPCH	IKJEGPCH	Message CSECT.
IKJEGMSD	IKJEGASN	IKJEGASN	IKJEGASN	Message CSECT.
IKJEGMSD	IKJEGLDF	IKJEGLDF	IKJEGLDF	Message CSECT in module IKJEGLDF.
IKJEGMSE	IKJEGLDF	IKJEGLDF	IKJEGLDF	Entry name in message CSECT in module IKJEGLDF.
IKJEGMSF	IKJEGLDF	IKJEGLDF	IKJEGLDF	Entry name in message CSECT.
IKJEGMSG	IKJEGGO	IKJEGGO	IKJEGGO	Message CSECT.

Module Cross Reference Directory

Entry Name or CSECT Name	Assembly Module Name	Load Module Name	Assembly Modules That Refer to Entry Name or CSECT Name	Definition of Entry Name, CSECT Name, or Associated Code
IKJEGMSG	IKJEGCVT	IKJEGMNL	IKJEGCVT	Message CSECT.
IKJEGMSG	IKJEGDCB	IKJEGDCB	IKJEGDCB	Message CSECT.
IKJEGMSG	IKJEGEND	IKJEGEND	IKJEGEND	Message CSECT.
IKJEGMSG	IKJEGEQU	IKJEGEQU	IKJEGEQU	Message CSECT.
IKJEGMSG	IKJEGLDF	IKJEGLDF	IKJEGLDF	Message CSECT.
IKJEGMSG	IKJEGMAP	IKJEGMAP	IKJEGMAP	Message CSECT.
IKJEGMSG	IKJEGOFF	IKJEGOFF	IKJEGOFF	Message CSECT.
IKJEGMSG	IKJEGTCB	IKJEGTCB	IKJEGTCB	Message CSECT.
IKJEGMSG	IKJEGWHR	IKJEGWHR	IKJEGWHR	Message CSECT.
IKJEGMSH	IKJEGLDF	IKJEGLDF	IKJEGLDF	Entry name in message CSECT.
IKJEGMSJ	IKJEGLDF	IKJEGLDF	IKJEGLDF	Entry name in message CSECT.
IKJEGMSK	IKJEGLDF	IKJEGLDF	IKJEGLDF	Entry name in message CSECT.
IKJEGMSL	IKJEGLDF	IKJEGLDF	IKJEGLDF	Entry name in message CSECT.
IKJEGMSM	IKJEGLDF	IKJEGLDF	IKJEGLDF	Entry name in message CSECT.
IKJEGMSN	IKJEGLDF	IKJEGLDF	IKJEGLDF	Entry name in message CSECT.
IKJEGMSO	IKJEGLDF	IKJEGLDF	IKJEGLDF	Entry name in message CSECT.
IKJEGMSP	IKJEGLDF	IKJEGLDF	IKJEGLDF	Entry name in message CSECT.
IKJEGMSQ	IKJEGLDF	IKJEGLDF	IKJEGLDF	Entry name in message CSECT.
IKMEGMSQ	IKJEGQFY	IKJEGQFY	IKJEGQFY	Message CSECT.
IKJEGMSR	IKJEGLDF	IKJDGLDF	IKJEGLDF	Entry name in message CSECT.
IKJEGMSS	IKJEGLDF	IKJEGLDF	IKJEGLDF	Entry name in message CSECT.
IKJEGMSX	IKJEGLDF	IKJEGLDF	IKJEGLDF	Entry name in message CSECT.
IKJEGMS1	IKJEGSTA	IKJEGMNL	IKJEGSTA	Message CSECT.
IKJEGMS4	IKJEGWHR	IKJEGWHR	IKJEGWHR	Entry name in message CSECT.
IKJEGMS7	IKJEGWHR	IKJEGWHR	IKJEGWHR	Entry name in message CSECT.
IKJEGMS8	IKJEGWHR	IKJEGWHR	IKJEGWHR	Entry name in message CSECT.
IKJEGMS9	IKJEGWHR	IKJEGWHR	IKJEGWHR	Entry name in message CSECT.
IKJEGMYS	IKJEGSYM	IKJEGSYM	IKJEGSYM	Second executable CSECT.

Module Cross Reference Directory

Entry Name or CSECT Name	Assembly Module Name	Load Module Name	Assembly Modules That Refer to Entry Name or CSECT Name	Definition of Entry Name, CSECT Name, or Associated Code
IKJEGM2	IKJEGATD	IKJEGATD	IKJEGATD	Message CSECT.
IKJEGLDF	IKJEGLDF	IKJEGLDF	IKJEGMNL IKJEGLDF	Major CSECT name.
IKJEGOFF	IKJEGOFF	IKJEGOFF	IKJEGMNL IKJEGSCD IKJEGEND	Major entry point of the OFF subcommand processor.
IKJEGPAR	IKJEGGO	IKJEGGO	IKJPARS	Parameter control list CSECT that describes the format of the parameters, for GO and RUN subcommands.
IKJEGPAR	IKJEGEQU	IKJEGEQU	IKJPARS	Parameter control list CSECT that describes the format of the parameters, for the EQUATE subcommand.
IKJEGPAS	IKJEGGO	IKJEGGO	IKJPARS	Parameter control list CSECT that defines the format of the CALL parameters.
IKJEGPCH	IKJEGPCH	IKJEGPCH	IKJEGSCD IKJEGMNL	Major CSECT of first load module of Assignment subcommand processor.
IKJEGPCL	IKJEGDCB	IKJEGDCB	IKJPARS	Parameter control list CSECT that describes the format of the LISTDCB parameters.
IKJEGPCL	IKJEGTCB	IKJEGTCB	IKJPARS	Parameter control list CSECT that describes the format of the LISTDCB parameters.
IKJEGPCL	IKJEGMAP	IKJEGMAP	IKJPARS	Parameter control list CSECT that describes the format of the LISTMAP parameters.
IKJEGPCL	IKJEGOFF	IKJEGOFF	IKJPARS	Parameter control list CSECT that describes the format of the OFF parameters.
IKJEGPCL	IKJEGQFY	IKJEGQFY	IKJPARS	Parameter control list CSECT that describes the format of the QUALIFY parameters.

Entry Name or CSECT Name	Assembly Module Name	Load Module Name	Assembly Modules That Refer to Entry Name or CSECT Name	Definition of Entry Name, CSECT Name, or Associated Code
IKJEGPCL	IKJEGTCB	IKJEGTCB	IKJPARS	Parameter control list that describes the format of the LISTTCB parameters.
IKJEGPG	IKJEGIO	IKJEGMNL	Any TEST module except IKJEGATN	One of three entry points to module IKJEGIO. This entry point is used to issue a PUTGET macro instruction to solicit input from the terminal.
IKJEGPMA	IKJEGLST	IKJEGLST	IKJPARS	Parameter control list CSECT that describes the format of the LIST paramaters.
IKJEGPMB	IKJEGPCH	IKJEGPCH	IKJPARS	Parameter control list CSECT that describes the format of the Assignment parameters.
IKJEGPRM	IKJEGCPY	IKJEGCPY	IKJPARS	Parameter control list CSECT for the COPY subcommand processor.
IKJEGPRS	IKJEGEQU	IKJEGEQU	IKJPARS	Parameter control list CSECT that describes the format of the DROP parameters.
IKJEGPRS	IKJEGWHR	IKJEGWHR	IKJPARS	Parameter control list CSECT that describes the format of the WHERE parameters.
IKJEGPSW	IKJEGPSW	IKJEGPSW	IKJEGMNL IKJEGSCD	Major entry point for the LISTPSW subcommand.
IKJEGPT	IKJEGIO	IKJEGMNL	Any TEST module except IKJEGATN.	One of three entry points to module IKJEGIO. This entry point is used to issue output to the terminal or to a print data set.
IKJEGPWM	IKJEGPSW	IKJEGPSW	IKJEGMNL IKJPARS IKJEGCVT	Message CSECT.
IKJEGPWP	IKJEGPSW	IKJEGPSW	IKJPARS	Parameter control list CSECT that describes the format of the LISTPSW parameters.
IKJEGQFY	IKJEGQFY	IKJEGQFY	IKJEGMNL IKJEGSCD	Major entry point for QUALIFY subcommand.

Entry Name or CSECT Name	Assembly Module Name	Load Module Name	Assembly Modules That Refer to Entry Name or CSECT Name	Definition of Entry Name, CSECT Name, or Associated Code
IKJEGRUN	IKJEGGO	IKJEGGO	IKJEGMNL IKJEGSCD	Major entry point for RUN subcommand.
IKJEGSCD	IKJEGSCD	IKJEGSCD	IKJEGINT IKJEGMNL IKJEGSTA	Entry name of subcommand name table. This table contains the valid subcommand names and their associated IDs, and the suffixes for most load module names.
IKJEGSMG	IKJEGSYM	IKJEGSYM	IKJEGSYM	Message CSECT.
IKJEGSRH	IKJEGSRH	IKJEGSRH	IKJEGOFF IKJEGAT IKJEGASN	Entry point for the Search module.
IKJEGSTA	IKJEGSTA	IKJEGMNL	IKJEGMNL ABEND/STAE Interface (IGC0B01C)	Major entry point of STAE Exit routine of TEST.
IKJEGSYM	IKJEGSYM	IKJEGSYM	IKJEGCVT IKJEGWHR IKJEGQFY	Major entry point of Symbol module of TEST.
IKJEGTCB	IKJEGTCB	IKJEGTCB	IKJEGMNL IKJEGSED	Major entry point for the LISTTCB subcommand.
IKJEGTO	IKJEGCPY	IKJEGCPY	IKJPARS IKJEGCVT	Entry point for the validity check routine that tests the "to"address (address2) in the COPY subcommand processor.
IKJEGVCK	IKJEGOFF	IKJEGOFF	IKJPARS	Validity check exit for the OFF subcommand.
IKJEGVCK	IKJEGTCB	IKJEGTCB	IKJPARS	Validity check exit for the LISTTCB subcommand.
IKJEGVDK	IKJEGEQU	IKJEGEQU	IKJPARS	Validity check exit for the EQUATE and DROP subcommands.
IKJEGWHR	IKJEGWHR	IKJEGWHR	IKJEGMNL IKJEGSCD	Major CSECT name for the WHERE subcommand processor.
IKJEGXDR	IKJEGLDR	IKJEGLDR	IKJEGLDR	Message CSECT.
IKJEGXNL	IKJEGMNL	IKJEGMNL	IKJEGMNL	Message CSECT.
IKJEGXNT	IKJEGINT	IKJEGINT	IKJEGINT	Message CSECT.
IKJEGYDR	IKJEGLDR	IKJEGLDR	IKJEGLDR	Parameter control list CSECT.
IKJEGYNT	IKJEGINT	IKJEGINT	IKJPARS	Parameter control list CSECT for TEST command before IKJEGMNL receives control.
IOMSGCST	IKJEGIO	IKJEGMNL	IKJEGIO	Message CSECT.

Section 5: Data Areas

This section contains information about the following control blocks, work areas, and tables built and used by the TEST command processor.

- Break element queue.
- Break element.
- Defer element queue.
- Defer module element.
- Defer break element.
- IKJPARMA.
- Entry for symbol table built in storage.
- Subcommand Name Table (IKJEGSCD).
- SVC Information Block.
- Symbol information block.
- TEST Communication Table (TCOMTAB).
- TEST's work area (TSTCWORK).

Break Element Queue

The AT subcommand processor (IKJEGAT) builds a break element for each address specified by the user in an AT subcommand. The queue of break elements is chained from the BREAKTAB field of TCOMTAB. Many test modules (the SVC 97 routine, IKJEGGO, IKJEGMNL, IKJEGQFY, IKJEGASN, and IKJEGLST, for example) refer to the break element queue.

IKJEGOFF purges break elements from the queue either because the user has entered an OFF subcommand or because IKJEGMNL linked to IKJEGOFF during END or RUN processing. Figure 14 shows a break element queue.

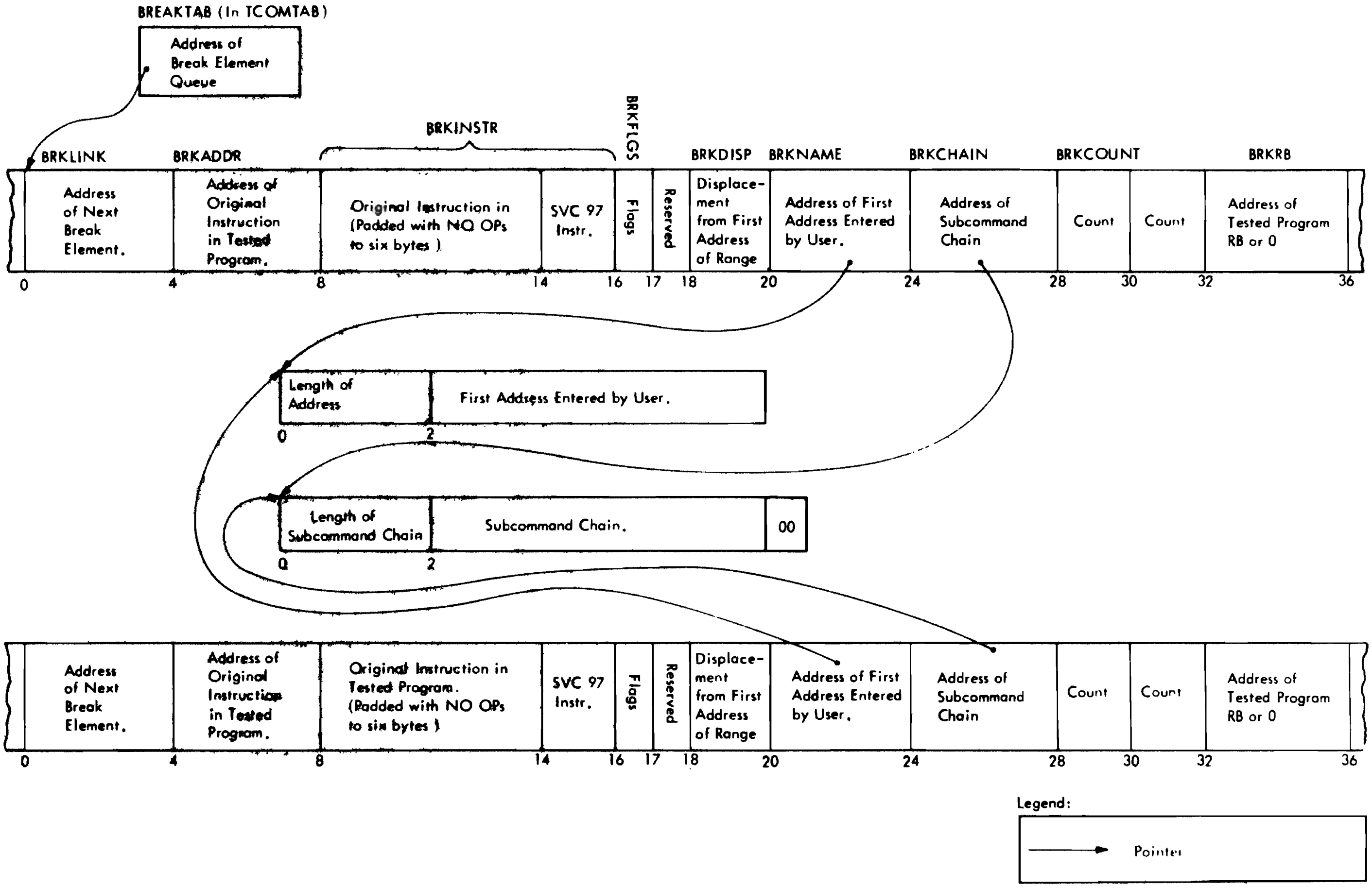
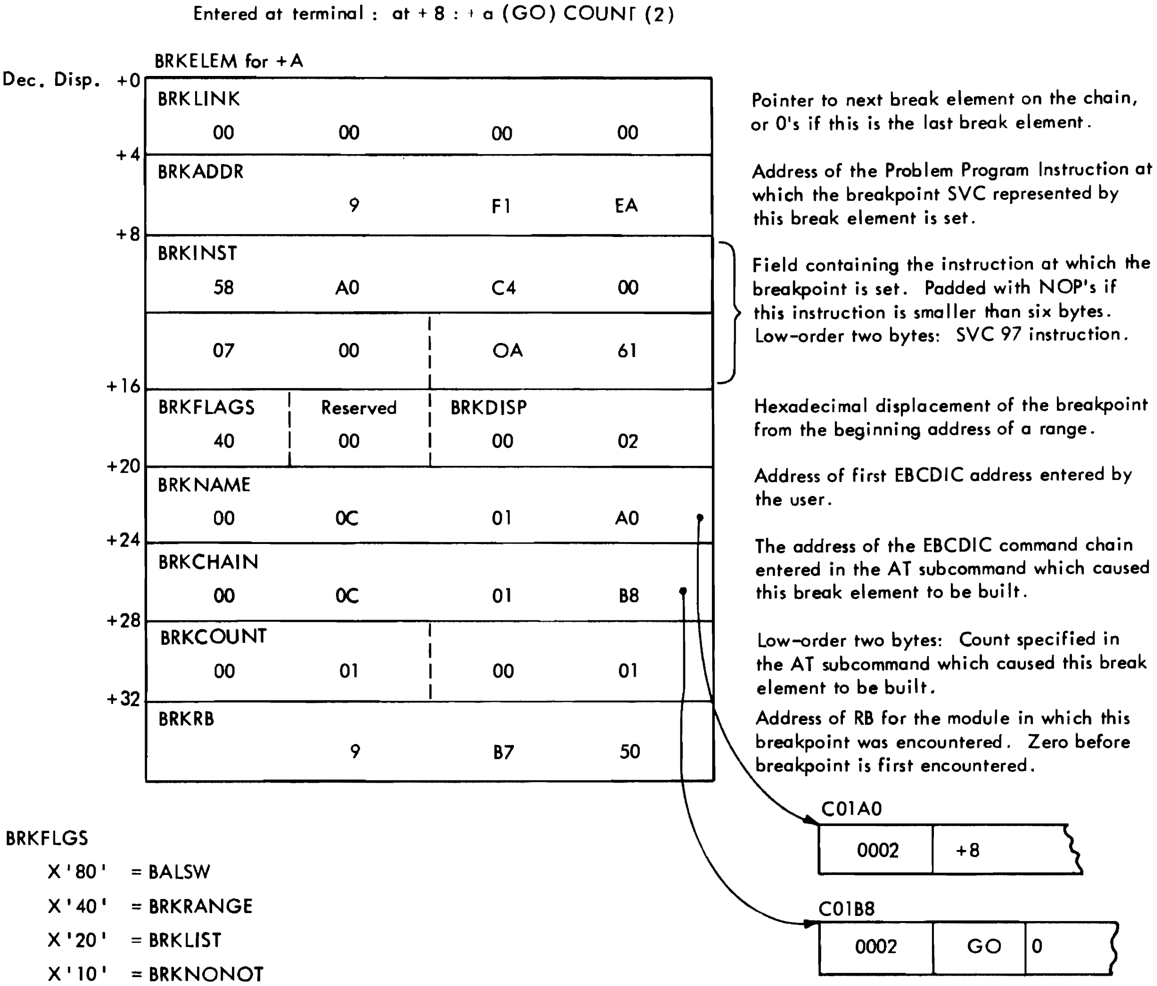

BREAKTAB (In TCOMTAB)
Address of Break Element Queue
BRKLINK
BRKADDR
BRKINSTR
BRKFLGS
BRKDISP
BRKNAME
BRKCHAIN
BRKCOUNT
BRKRB
Address of Next Break Element.
Address of Original Instruction in Tested Program.
Original Instruction in (Padded with NO OPs to six bytes)
SVC 97 Instr.
Flags
Reserved
Displacement from First Address of Range
Address of First Address Entered by User.
Address of Subcommand Chain
Count
Count
Address of Tested Program RB or 0
0
4
8
14
16
17
18
20
24
28
30
32
36
Length of Address
First Address Entered by User.
0
2
Length of Subcommand Chain
Subcommand Chain.
00
0
2
Address of Next Break Element.
Address of Original Instruction in Tested Program.
Original Instruction in Tested Program. (Padded with NO OPs to six bytes)
SVC 97 Instr.
Flags
Reserved
Displacement from First Address of Range
Address of First Address Entered by User.
Address of Subcommand Chain
Count
Count
Address of Tested Program RB or 0
0
4
8
14
16
17
18
20
24
28
30
32
36
Legend:
Pointer

Figure 14. Break Element Queue

Figure 15 shows an example of a break element.



Entered at terminal : at + 8 : + a (GO) COUNT (2)

BRKELEM for +A

Dec. Disp.	Field					
+0	BRKLINK	00	00	00	00	Pointer to next break element on the chain, or 0's if this is the last break element.
+4	BRKADDR		9	F1	EA	Address of the Problem Program Instruction at which the breakpoint SVC represented by this break element is set.
+8	BRKINST	58	A0	C4	00	Field containing the instruction at which the breakpoint is set. Padded with NOP's if this instruction is smaller than six bytes. Low-order two bytes: SVC 97 instruction.
		07	00	0A	61	
+16	BRKFLAGS	40	Reserved 00	BRKDISP 00	02	Hexadecimal displacement of the breakpoint from the beginning address of a range.
+20	BRKNAME	00	0C	01	A0	Address of first EBCDIC address entered by the user.
+24	BRKCHAIN	00	0C	01	B8	The address of the EBCDIC command chain entered in the AT subcommand which caused this break element to be built.
+28	BRKCOUNT	00	01	00	01	Low-order two bytes: Count specified in the AT subcommand which caused this break element to be built.
+32	BRKRB		9	B7	50	Address of RB for the module in which this breakpoint was encountered. Zero before breakpoint is first encountered.

C01A0

0002	+8

C01B8

0002	GO	0

BRKFLGS

X'80'	= BALSW
X'40'	= BRKRANGE
X'20'	= BRKLIST
X'10'	= BRKNONOT

Contents of main storage before and after breakpoint processing:

Before	9F1E8	+8	05C0	After	9F1E8	+8	0A61
	9F1EA	+A	58A0C400		9F1EA	+A	0A61C400

Figure 15. Break Element Example

Break Element

The macro name of the element is BRKELEM.

Dec. Disp.	Hex. Disp.	Field Name	Field Size	Contents	Set by	Referenced by	Cleared by
0	0	BRKLINK	4	Link field. Pointer to next break element on chain (or is zero if it is the last break element).	IKJEGAT	SVC 97 IKJEGGO IKJEGMNL IKJEGLSA IKJEGLST IKJEGASN IKJEGOFF IKJEGQFY IKJEGCPY	
4	4	BRKADDR	4	The absolute address in IKJEGAT, the problem program at which the breakpoint (SVC 97) represented by this break element is set.	IKJEGAT	IKJEGMNL SVC 97 IKJEGOFF IKJEGLST IKJEGPCH IKJEGGO IKJEGCPY IKJEGQFY IKJEGLSA IKJEGASN	
8	4	BRKINST	8	Field containing the instruction at whose SVC is set. The last two bytes of the field contain a breakpoint SVC instruction. Any remaining space between this SVC and the saved instruction at the beginning of this field contains NOPs.	IKJEGAT	SVC 97 IKJEGLST IKJEGOFF IKJEGMNL IKJEGQFY	
16	10	BRKFLGS	1	The following are possible values for BRKFLGS:			
		BALSW	1.......	Indicates that the saved instruction is a BAL or a BALR.	IKJEGAT	SVC 97	
		BRKRANGE	.1......	Indicates that this break element is one of a range of addresses.	IKJEGAT	IKJEGOFF IKJEGMNL	
		BRKLIST	..1.....	Indicates that the break element is one of a list of addresses.	IKJEGAT	IKJEGOFF	
		BRKNONOT	...1....	Indicates that no notification to the terminal is to be written when the breakpoint represented by this element is encountered.	IKJEGAT	IKJEGMNL	
			xxxx	Reserved.			
17	11		1	Reserved space.			
18	12	BRKDISP	2	Binary (hexadecimal) displacement of the breakpoint from the beginning address of a range of addresses.	IKJEGAT	IKJEGMNL	

Dec. Disp.	Hex. Disp.	Field Name	Field Size	Contents	Set by	Referenced by	Cleared by
20	14	BRKNAME	4	Points to the first EBCDIC address (or only one, if not a range) entered by the user when the AT subcommand caused this break element to be built.	IKJEGAT	IKJEGOFF IKJEGMNL	
24	28	BRKCHAIN	4	The address of the EBCDIC subcommand chain entered in the AT subcommand that caused this break element to be built.	IKJEGAT	IKJEGOFF IKJEGMNL IKJEGIO	
28	1C	BRKCOUNT	4	The low order two bytes of this field contain the count (minus one) specified in the AT subcommand that caused this break element to be built. Both the low order two bytes and the high order two bytes are set to this count value by IKJEGAT. This high order two bytes are decremented by the SVC 97 routine until the value reaches zero, at which time the SVC 97 routine resets the high order two bytes to the value of the low order two bytes. A breakpoint is actually taken only when the high order two bytes equal zero.	IKJEGAT SVC 97	SVC 97	
32	20	BRKRB	4	Address of RB for the SVC 97 module in which the breakpoint represented by this break element was encountered.	SVC 97	IKJEGOFF	

Defer Element Queue

Figure 16 shows a defer element queue. The defer element queue consists of one or more defer module elements and associated defer break elements. The elements are built by module IKJEGATD and are queued from the DEFERTAB field of TCOMTAB, when the user issues an AT DEFER subcommand. The elements point to information that describes one or more breakpoints in a module of the problem program. When the module is loaded, IKJEGMNL links to module IKJEGAT to build break elements and set breakpoints in the specified module.

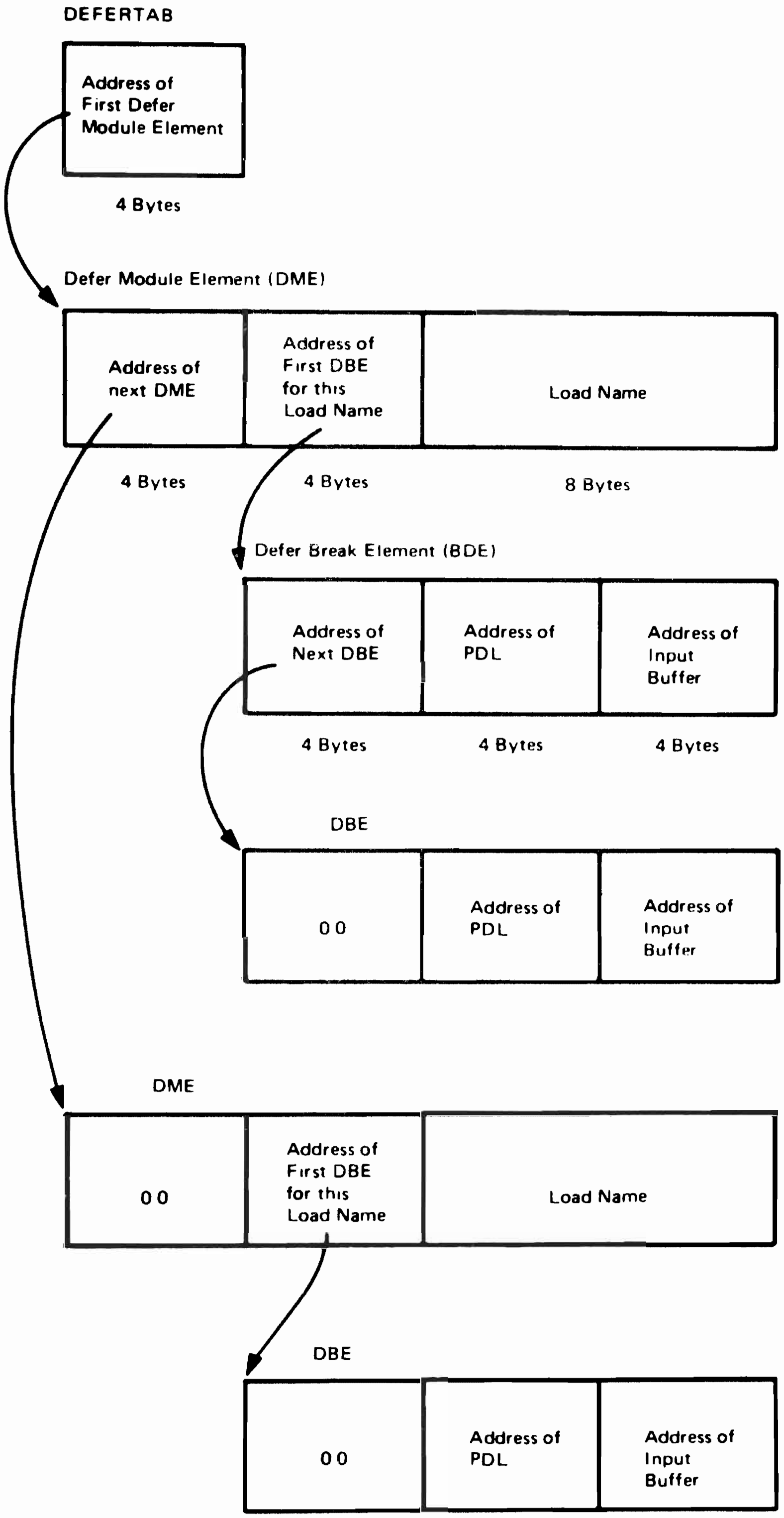


Figure 16. Defer Element Queue

Defer Module Element (DME)

Defer module elements and defer break elements are removed from the queue by module IKJEGOFF. This occurs when the user eneters an OFF subcommand, or when IKJEGMNL prepares to return to the TMP.

Dec. Disp.	Hex. Disp.	Field Name	Field Size	Contents	Set by	Referenced by	Cleared by
0	0	DMEDME	0	Pointer to the next DME or contains zeros, if no more defer module elements exist.	IKJEGATD	IKJEGAT IKJEGOFF	IKJEGOFF
4	4	DMEDBE	4	Pointer to the first defer break element.	IKJEGATD	IKJEGAT IKJEGOFF	IKJEGOFF
8	8	DMELOAD	8	Name of load module in which breakpoints are to be set.	IKJEGATD	IKJEGAT IKJEGOFF	IKJEGOFF

Defer Break Element (DBE)

Dec. Disp.	Hex. Disp.	Field Name	Field Size	Contents	Set by	Referenced by	Cleared by
0	0	DBEDBE	4	Pointer to the next defer break element, or contains zeros, if no more break elements exist.	IKJEGATD	IKJEGAT IKJEGOFF	IKJEGOFF
4	4	DBEPDL	4	Pointer to the parameter descriptor list that describes the operand of the AT DEFER subcommand.	IKJEGATD	IKJEGAT IKJEGOFF	IKJEGOFF
8	8	DBEINBUF	4	Pointer to the input buffer that contains the AT DEFER subcommand.	IKJEGATD	IKJEGAT IKJEGOFF	IKJEGOFF

IKJPARMA

The IKJPARS address PDE described by this control block is dependent upon the PDE definition made by IKJPARS.

Dec. Disp.	Hex. Disp.	Field Name	Field Size	Contents	Set by	Referenced by	Cleared by
0	0	PDELDNAM	4	Pointer to load name.			
4	4	PDELEN1	2	Length of load name.			
6	6	PDEFLG1 LDNAMFLG	1 1....... .xxxxxxx	Flags for load name. Load name present. Reserved.			
7	7	PDERSV1	1	Reserved			
8	8	PDECTNAM	4	Pointer to entry name.			
12	C	PDELEN2	2	Length of entry name.			
14	E	PDEFLG2 CTNAMFLG	1 1....... .xxxxxxx	Flag for entry name. Entry name present. Reserved bits.			
15	F	PDERSV2	1	Reserved.			
16	10	PDEADRPT	4	Pointer to address string.			
20	14	PDELEN3	2	Length of address string.			
22	16	PDEFLG3 AFLG	1 1....... .xxxxxx	Flag for address string. Address present. Reserved.			
23	17	PDERSV3	1	Reserved.			
24	18	PDEFLG4 ADSADDR SYMADDR RELADDR GENR LFPR SFPR CTONLY	1 0....... 1....... .1...... ..1..... ...1.... ....1.. .....1.. xx	Flag for address type. Absolute address. Symbolic address. Relative address. General register. Double precision floating point register. Single precision floating point register. Entry name is not followed by symbolic or relative address. Reserved.			
25	19	PDESIGN	1	Expression sign.			
26	1A	PDEINDCT	2	Indirect count.			
28	1C	PDEEXPTR	4	Pointer to first expression PDE.			

Dec. Disp.	Hex. Disp.	Field Name	Field Size	Contents	Set by	Referenced by	Cleared by
32	20	PDEFLG5	1	Flag for PDEUSER field.			
		PDESIB	1.......	PDEUSER points to SIB.			
		PDECONVD	.1......	Address is converted.			
		PDEADLST	..1.....	PDE is one of a list.			
		PDEYECH	...1....	PDE is useless			
		PDEWRITE	....1...	Address is writeable.			
			xxx	Reserved.			
33	21	PDEUSER	3	Resolved address or pointer to SIB.			
36	24	PDECHAIN		Chain pointer filed for address list. This is the overlay for the address range.			
36	24	PDE2LDNA	4	Pointer to load name.			
40	28	PDE2LEN1	2	Length of load name.			
42	2A	PDE2FLG1	1	Flags for load name.			
		LDNAMFLG	1.......	Load name present.			
			.xxxxxx	Reserved.			
43	2B	PDE2RSV1	1	Reserved.			
44	2C	PDE2CTNA	4	Pointer to entry name.			
48	30	PDE2LEN2	2	Length of entry name.			
50	32	PDE2FLG2	1	Flags for entry name.			
		CTNAMFLG	1.......	Entry name present.			
			.xxxxxxx	Reserved.			
51	33	PDE2RSV2	1	Reserved.			
52	34	PDE2ADRP	4	Pointer to address string.			
56	38	PDE2LEN3	2	Length of address string.			
58	3A	PDE2FLG3	1	Flag for address string.			
		AFLG	1.......	Address string present.			
			.xxxxxxx	Reserved.			
59	3B	PDE2RSV3	1	Reserved.			

Dec. Disp.	Hex. Disp.	Field Name	Field Size	Contents	Set by	Referenced by	Cleared by
60	3C	PDE2FLG4	1	Flags for address type.			
		ABSADDR	1.......	Absolute address.			
		SYMADDR	0.......	Symbolic address.			
		RELADDR	.1......	Relative address.			
		GENR	..1.....	General register.			
		LFPR	...1....	Double precision floating-point register.			
		SFPR	....1...	Single precision floating-point register.			
		CTONLY	.....1..	Entry name is not followed by symbolic or relative address.			
			xx	Reserved.			
61	3D	PDE2SIGN	1	Expression sign.			
62	3E	PDE2INDC	2	Indirect count.			
64	40	PDE2EXPT	4	Pointer to first expression PDE.			
68	44	PDE2FLG5 PDESIB PDECONVD PDEADLST PDEYECH PDEWRITE	1 1....... 1....... ..1..... .1...... ....1... ..1..... ...1.... ....1... xxx	Flag for PDE2USER field. PDE2USER points to SIB. Address is converted. PDE is one of a list. PDE is useless. Address is writable. Reserved.			
69	45	PDE2USER	3	Resolved address of pointer to SIB.			
72	48	PDE2CHAI	4	Chain pointer field for address list.			

Entry for Symbol Table Built in Storage

This table contains elements that are built and chained by the EQUATE subcommand. These elements are unchained and freed by the DROP subcommand. This table is used by IKJEGSYM in the first search in an attempt to resolve a symbolic address.

The maximum symbol length at hexadecimal displacement 10 is eight characters.

Dec. Disp.	Hex. Disp.	Field Name	Field Size	Contents	Set by	Referenced by	Cleared by
0	0	CHAIN	4	Points to next symbol table in storage.			
4	4	ADDRSYM	4	Pointer to symbol location.			
8	8	ATTRTYPE	1	Internal hexadecimal code related to the type of character entered on the EQUATE subcommand. 00 Character format. 04 Hexadecimal format. 08 Binary format. 0C Instruction format. 10 Fixed point, half-word format. 14 Fixed point, half-word format. 18 Floating point, full-word format. 1C Floating point, double-word format. 20 Address constant, A or Q format. 24 Address constant, Y format. 28 Address is base displacement format. 30 Packed decimal format. 34 Zoned decimal format.			
9	9	ATTRMULT	3	Multiplicity factor.			
12	C	ATTRLNTH	2	Length attribute.			
14	E	SYMLNGTH	2	Symbol length.			
16	10	SYMBOL	8	Symbol in EBCDIC as entered in EQUATE subcommand.			

Subcommand Name Table (IKJEGSCD)

The subcommand name table contains one entry for each subcommand, except the assignment function. The table is used by both IKJEGMNL and IKJEGSTA. IKJEGMNL searches the table for a matching name when the user enters a subcommand. It uses the associated ID as an index to an entry in the SCPPAD table. The SCPPAD table entry points to a compressed BLDL entry in the BLDL table. IKJEGMNL expands the BLDL entry, then specifies the entry in a LINK macro that it issues.

TEST's STAE Exit routine uses the subcommand name table to obtain the name of the failing subcommand. It places the name of the subcommand in a first-level diagnostic message that it issues to the terminal. Figure 17 shows the subcommand name table format.

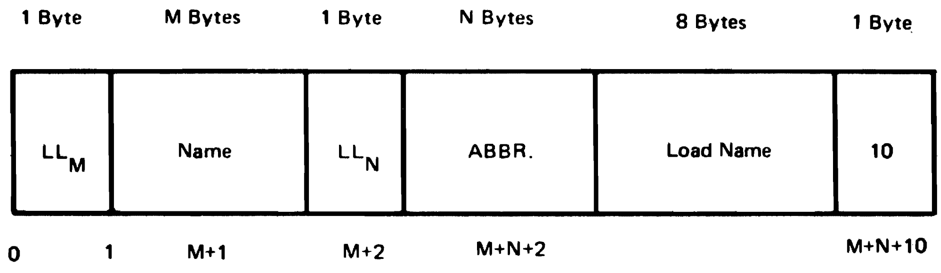


Figure 17. Subcommand Name Table Format

SVC Information Block

This block is built by the SVC 61 routine (IGC0006A). The SVB is pointed to by the appropriate SVQ for the TCB under which the load module was fetched. SVC information blocks are searched by IKJEGSYM in an attempt to resolve a symbol, entry name, or offset belonging to a load module of the problem program.

Dec. Disp.	Hex. Disp.	Field Name	Field Size	Contents	Set by	Referenced by	Cleared by
0	0	SVBLDNAM	8	EBCDIC load name of the fetched module.			
8	8	SVCEP	4	Address at which module is fetched.			
12	C	SVBTTR	4	Beginning of TTR.			
16	10	SVBATTR1	1	Module attributes byte 1.			
		SVBRENT	1.......	Can be entered again.			
		SVBREUS	.1......	Reusable.			
		SVBOVLY	..1.....	Overlay.			
		SVBTEST	...1....	Module to be tested.			
		SVBOL	....1...	Only loadable.			
		SVBSCTR	.....1..	Scatter load format.			
		SVBEXEC	......1.	Executable.			
		SVB1BLK	1	Module contains a block of text without RLD items.			
			0	Module contains multiple records with at least one block of text.			

Dec. Disp.	Hex. Disp.	Field Name	Field Size	Contents	Set by	Referenced by	Cleared by
17	17	SVBATTR2	1	Module attributes byte 2.			
		SVBLKEDF	1.......	Module can be processed by Linkage Editor F only.			
			0.......	Module can be processed by all levels of the Linkage Editor.			
		SVBTEXTO	.1......	Linkage Editor assigned origin or first text block is zero.			
			.0......	Linkage Editor assigned origin of first text block is not zero.			
		SVBEPO	..1.....	Entry point assigned by the Linkage Editor is zero.			
		SVBNORLD	...1...	Module contains no RLD items.			
		SVBNOLE	....1...	Module cannot be reprocessed by the Linkage Editor.			
		SVBSYM	.....1..	Module contains symbol records.			
		SVBLEVF	......1.	Module created by Linkage Editor F.			
		SVBREFR	1	Refreshable.			
18	12	SVBFLGS1	1	Flag byte.			
		SVBDDNME	1.......	DDname is present.			
		SVBLNKLB	.1......	Data set is LINKLIB.			
			..xxxxxx	Reserved.			
19	13	SVBCNCAT	...1	Concatenation number.			
20	14	SVBDDNAM	8	DD Name of data set from which the module is fetched.			
28	1C	SVBTCBPT	4	TCB address for fetched module.			
32	20	SVBLNKPT	4	Address of next SVC information block, or contains zeros if this is the last block.			

SVC Information Block Queue Element

This block is built by the SVC 61 routine (IGC0006A). The SVC 61 routine is invoked from Contents Supervision when a load module has been fetched. If TEST has been invoked, SVC information block queue elements are queued from the TSTTRN field in TCOMTAB. If TEST has not been invoked, these elements are queued from the invoking task TCBTRN field.

Dec. Disp.	Hex. Disp.	Field Name	Field Size	Contents	Set by	Referenced by	Cleared by
0	0	SVQLNKPT	4	Address of the next SVC queue block or zero.			
4	4	SVQTCBPT	4	Address of TCB for which this SVC queue block exists.			
8	8	SVQBLKPT	4	Address of the queue of SVC information blocks for this TCB.			

Symbol Information Block

A symbol information block (SIB) is created only by IKJEGSYM. SIBs are chained from the SICHAIN field in TCOMTAB as they are built. When a subcommand processor returns to IKJEGMNL, it frees all SIBs and they are not available to the next subcommand processor.

Dec. Disp.	Hex. Disp.	Field Name	Field Size	Contents	Set by	Referenced by	Cleared by
0	0	LOADEDAD	4	Pointer to symbol location.			
4	4	TYPE	1	Assembler-created attribute code.			
5	5	MULTIPLT	3	Assembler-created multiplicity factor.			
8	8	LENGTH	2	Assembler-created length attribute.			
10	A	RESERVD1	2	Reserved.			
12	C	NEXTANSB	4	Pointer to SIB or contains zeros if this is the last SIB.			

TEST Communication Table (TCOMTAB)

This table is built by IKJEGINT, is used by many TEST modules, and is freed by IKJEGMNL.

Dec. Disp.	Hex. Disp.	Field Name	Field Size	Contents	Set by	Referenced by	Cleared by
0	0	ECBPP	4	ECB waited on by SVC 97 and SVC 61.			
4	4	ECBLIST	0	Beginning of ECB list for wait.			
4	4	ECBTST	4	Pointer to ECB posted by SVC 97 and SVC 61.			
8	8	ECBTERM	4	Pointer to ECB posted by completion of oldest subtask of TEST.			
12	C	ECBTMPS	4	Pointer to posted by TMP STAI Exit Routine at ABEND of any subtask of TEST.			
16	10	ECBTMPA	4	Pointer to ECB posted by TMP Attention Exit Routine or TEST's Attention Exit Routine.			
20	14	ECBLOG	4	Pointer to ECB posted by STOP/MODIFY TS operator command.			
24	18	TSTTCB	0	Pointer to TCB for TMP/TEST task.			
24	18	TSTTCBR	1.......	Reserved.			
25	19	TSTTCPB	.3......	Pointer to TCB for TMP/TEST task.			
28	1C	PPTCB	0	Pointer to TCB for problem program task currently qualified.			
28	1C	PPTCBR	1.......	Reserved.			
29	1D	PPTCBP	.3......	Pointer to TCB for problem program task currently qualified.			
32	20		8	Reserved.			
40	28	OUTBUF	4	Pointer to small, general purpose work area used as an output buffer for messages or data.			

Dec. Disp.	Hex. Disp.	Field Name	Field Size	Contents	Set by	Referenced by	Cleared by
44	2C	BLDLARA	0	Address of build area used by IKJEGINT and IKJEGLDR.			
44	2C	CONAREA	4	Pointer to small work area in which IKJEGCVT returns converted values.			
48	30	WORKAREA	4	Pointer to a general work area with general and reserved sections (see TSTCWORK).			
52	34	REGSAVE1	4	Pointer to register save area for IKJEGMNL.			
56	38	REGSAVE2	4	Pointer to register save area for TEST subcommands.			
60	3C	REGSAVE3	4	Pointer to register save area reserved for subcommand validity check exit routines.			
64	40	REGSAVE4	4	Pointer to register save area reserved for IKJEGCVT.			
68	44	REGSAVE5	4	Pointer to register save area reserved for IKJEGIO.			
72	48	REGSAVE6	4	Pointer to register save area reserved for IKJEGSRH.			
76	4C		4	Reserved.			
80	50	TPLPTR	4	Pointer to TPL.			
84	54	TMPLL	2.......	Logical line size of the user's terminal.			
86	56	TSTMTASK	..1.....	Full byte queuing switch to force POSTs from SVC 61 or SVC 97 into FIFO order.			
87	57	TSTRSVD1	...1....	Reserved.			
88	58	TSTWHR	4	Communication word used for information passed from IKJEGSYM to IKJEGWHR.			

Dec. Disp.	Hex. Disp.	Field Name	Field Size	Contents	Set by	Referenced by	Cleared by
92	5C	PARMLIST	0	Standard section of TMP Service Routine parameter list.			
92	5C	TSTUPT	4	Pointer to UPT.			
96	60	TSTECT	4	Pointer to ECT.			
100	64	TSTCPECB	4	Pointer to ECB (save ECB pointed to by ECBTMPA).			
104	68	TSTANSPL	4	Answer field used by all modules calling IKJPARS.			
108	6C	ABENTAB	4	Pointer to list that contains addresses of instructions in resident TEST modules where an ABEND is possible.			
112	70	ABENTAB1	4	Pointer to list that contains addresses of instructions in transient TEST modules where an ABEND is possible.			
116	74	TSTRETRY	0	STAE retry communication word.			
116	74	TSTRTYCD	1.......	Subcommand ID.			
117	75	TSTRTYPT	.3......	Pointer to a STAE retry routine within any TEST module currently in control that has issued a STAE macro instruction.			
120	78	INBUF	4	Pointer to storage containing a subcommand ready for processing.			
124	7C	TSTIODSN	4	Pointer to queue of DSNAMEs used as parameters for PRINT keyword with LIST TYPE subcommands.			
128	80	TGETADDR	4	Pointer to GETLINE subroutine in IKJEGIO.			
132	84	TPUTADDR	4	Pointer to PUTLINE subroutine in IKJEGIO.			

Dec. Disp.	Hex. Disp.	Field Name	Field Size	Contents	Set by	Referenced by	Cleared by
136	88	TPUTGET	4	Pointer to PUTGET subroutine in IKJEGIO.			
140	8C	TSTCONVT	4	Pointer to IKJEGCVT.			
144	90	TSTADDR	4	Pointer to IKJEGBLD, a subroutine whose function is to assembly an EBCDIC address string from the components pointed to by an address-type PDE.			
148	94	TSTSTAE	4	Pointer to IKJEGSTA.			
152	98	TSTFLGS1	1.......	Switches, byte 1.			
		1.......		PCHLSTVL - IKJEGPCH is processing a list of values.			
		..1.....		TSTPRINT - output from IKJEGIO is to a data set.			
		...1....		TSTFIRST - general purpose switch for use by any subcommand or subroutine. Not to be passed.			
		....1...		RANGESW - indicates a "range" address parameter.			
		.....1..		TSTBUILD - indicates that a new module is being loaded for the problem program.			
		......1.		ENDSW - indicates END processing has begun. TMP will force DETACH.			
		1		RUNSW - indicates RUN. TMP will not force DETACH.			
		.x......		Reserved.			

Dec. Disp.	Hex. Disp.	Field Name	Field Size	Contents	Set by	Referenced by	Cleared by
153	99	TSTFLGS2	.1......	Switches, byte 2.			
		1.......		TSTLDF - indicates IKJEGLDF is running under subtask TCB.			
		.1......		TSTXCTL - indicates that subcommand processor has issued an XCTL macro instruction.			
		..1.....		TOFFDEF - internal switch used by IKJEGWHR.			
		...1....		TTESTTCB - not used.			
		....1...		TADDROUT - internal switch used by IKJEGWHR.			
		.....1..		TWHRLOAD - internal switch used by IKJEGOFF.			
		1		TYMIOMST - internal switch used by IKJEGIO.			
		......x,		Reserved.			
154	9A	TSTFLGS3	..1.....	Switches, byte 3.			
		1.......		TSTGOSW - indicates a pseudo breakpoint set by IKJEGLDR at entry point of the problem program.			
		.1......		TSTSTAI - indicates that a task of the problem program has begun to ABEND.			
		..1.....		SYMMESG - indicates that IKJEGSYM is to write no diagnostic messages.			
		...1....		TCSECTCK - internal switch used by IKJEGOFF.			
		....1...		TDUPNAME - internal switch used by IKJEGOFF.			
		.....1..		TSTLINK - internal switch used by IKJEGMNL.			
		......1.		TSTHELP - indicates that the HELP command processor has been ATTACHed.			
		1		NOPARMS - indicates subcommand entered without operands.			

Dec. Disp.	Hex. Disp.	Field Name	Field Size	Contents	Set by	Referenced by	Cleared by
155	9B	TSTFLGS4 ...1....		Switches, byte 4.			
		1.......		TSTA - indicates input from the terminal.			
		.1......		TSTB - indicates a terminal element has been added to the input stack.			
		..1.....		TSTFLUSH - indicates to IKJEGMNL that the TCLEARQ macro instruction should be issued to purge input buffers.			
		...xxxxx		Reserved.			
156	9C	BREAKTAB	4	Origin or active break element queue.			
160	A0	DEFERTAB	4	Origin of defer element queue.			
164	A4	PPLOAD	4	Base address for relative addresses. Contains the loaded address of the currently testable module.			
168	A8	PPTEMP	4	Temporary base for relative addresses.			
172	AC	SUBCHAIN	4	Pointer to in-core subcommand which is part of a chain associated with a breakpoint element.			
176	B0	TSTGO	0	Area used for handling pseudo breakpoints.			
176	B0	TSTGOPSW	4	A 4-byte save area used for pseudo breakpoints. It contains either the first two bytes of the problem program's entry point instruction when TSTGOSW contains "1"B or the problem program's saved restart address (RBOPSW +4) when TSTGOSW contains '0'B.			
180	B4	TSTGOWCF	1.......	Wait count from RBWCF for a module of the problem program when a pseudo breakpoint has been set.			

Dec. Disp.	Hex. Disp.	Field Name	Field Size	Contents	Set by	Referenced by	Cleared by
181	B5	TSTRSVD2	.1......	Reserved.			
182	B6	TSTSVC	..2.....	Contains an SVC 97 instruction (X'0A61' that is executed to cause a pseudo breakpoint to be taken.			
184	B8	PPRB	4	Pointer to PRB for problem program module that is currently qualified.			
188	BC	TSTIODCB	4	Pointer to DCB opened to write to a data set specified in a PRINT keyword parameter.			
192	C0	CALLPARM	4	Head of chain of CALL parameter lists built by IKJEGGO when it is entered at entry point IKJEGCAL.			
196	C4		4	Reserved.			
200	C8	TSTCURLD	8	EBCDIC name of currently qualified load module.			
208	D0	TSTCURCT	8	EBCDIC name of currently qualified CSECT.			
216	D8	TSTSYMBA	4	Base address for symbolic addresses.			
220	DC	TSTTTRN	4	Pointer to chain of control words built by IKJEGINT from information supplied by the SVC 61 routine (see IKJEGSVQ).			
224	E0	SICHAIN	4	Pointer to symbol information block (SIB) queue built by IKJEGSYM.			
228	E4	TSTSYMWK	4	Pointer to work area acquired for symbol processing.			
232	E8	SYMTABLE	4	Pointer to queue of table elements built by IKJEGEQU.			
236	EC	PPEXIT	0	Contains SVC instructions.			
	EC	PPEXIT1	2.......	Contains an SVC 97 instruction (x'0A61').			
238	EE	PPEXIT2	..2.....	Contains an SVC 3 instruction (x'0A03').			

Dec. Disp.	Hex. Disp.	Field Name	Field Size	Contents	Set by	Referenced by	Cleared by
240	F0	TSTDCB	4	Pointer to queue of DCBs maintained in overlay mode.			
244	F4	OPCODTAB	4	Pointer to table of valid operation codes.			
248	F8	TSTOPCD2	4	Pointer to table of valid S/370 two-byte operation codes.			
252	FC	TSTSVCQ	4	Pointer to queue of ECBs which order breakpoints in FIFO sequence in a multi-tasking environment.			
256	100	TSTDDM	4	Pointer to common subroutine to handle error conditions returned by IKJDIAR.			
260	104	TSTHTCB	4	Pointer to TCB for HELP subcommand.			
264	108	TSTAQUAL	4	Automatic qualification information.			
264	108	TSTAQLDM	8	EBCDIC load module name of the load module that IKJEGMNL last automatically qualified.			
272	110	TSTAQEP	4	Pointer to entry point of the load module that IKJEGMNL last automatically qualified.			
276	114	TSTRSTRT	4	Pointer to retry routine that is executed after a STAE exit routine intercepted an ABEND in the TEST program.			
280	118	TSTSRHR	4	Address of resident breakpoint Search routine.			
284	11C	TSTSTAX	20	STAX parameter list.			
304	130	TSTDSECB	4	TEST dispatchability ECB.			
308	134	TSTMNLWK	40	IKJEGMNL work area.			
348	15C	TSTSTAE	16	STAE parameter list for IKJEGMNL.			
364	170		64	Reserved.			

TEST Parameter List (TPL)

The TEST Parameter List is constructed by module IKJEFT01 in the TMP. It is referenced by IKJEGINT directly, and by other TEST modules indirectly through TPL information placed in the TCOMTAB by TEST initialization.

Dec. Disp.	Hex. Disp.	Field Name	Field Size	Contents	Set by	Referenced by	Cleared by
0	2	TPLCBUF	4	Address of command buffer.			
4	4	TPLUPT	4	Address of user profile table (UPT).			
8	8	TPLPSCB	4	Address of protected step control block (PSCB).			
12	C	TPLECT	4	Address of environment control table (ECT).			
16	10	TPLTBUF	4	Address of TEST buffer.			
20	14	TPLCTCB	4	Address of attached command processor's TCB.			
24	18	TPLSTAI	4	Address of TMP STAI Exit routine.			
28	1C	TPLSPLS	4	Address of STAI parameter list.			
32	20	TPLNECB	4	Address of ECB for a terminating command processor. The TMP STAI Exit routine (IKJEGT04) waits on the ECB.			
36	24	TPLNTCB	4	Address of TCB for a terminating command processor or problem program.			

Dec. Disp.	Hex. Disp.	Field Name	Field Size	Contents	Set by	Referenced by	Cleared by
40	28	TPLMECB	4	Address of STOP/MODIFY ECB. Equivalent of ECBLOG in the TCOMTAB.			
44	2C	TPLCECB	4	Address of attached command processor's ECB. Posted by supervisor's EOT routine when command processor (or problem program) completes normally. Equivalent of ECBTERM in TCOMTAB.			
48	30	TPLIECB	4	Address of TMP STAI ECB. Posted by the TMP STAI Exit routine. Equivalent of ECBTMPA in TCOMTAB.			
52	34	TPLAECB	4	Address of TMP's attention ECB. Equivalent of ECBTMP in TCOMTAB.			
56	38	TPLLDCB	4	Address of command library DCB.			

TEST's Work Area (TSTCWORK)

TSTCWORK, the permanent work area, is acquired at initialization by IKJEGINT. Partition control of the various sections is done by the macro TSTCWORK.

Dec. Disp.	Hex. Disp.	Field Name	Field Size	Contents	Set by	Referenced by	Cleared by
0	0	CWORKCVT	64	Work area for use only by IKJEGCVT.	IKJEGCVT IKJEGMNL	IKJEGCVT IKJEGINT	
64	40	CWORKCM	168	Work area for use only by processors.		Command processors IKJEGMNL, IKJEGINT	
232	E8	CWORKIO	32	Work area for use only by IKJEGIO.		IKJEGIO	
264	108	CWORKAT	164	Work area for use only by IKJEGATN.		IKJEGATN	
428	1AC	CWORKSTA	32	Work area for use only by IKJEGSTA.		IKJEGSTA	
460	1CC	CWORKSY	16	Work area for use only by IKJEGSYM.		IKJEGSYM	

Note: IKJEGINT and IKJEGLDR may use TSTCWORK, because other modules that use it are not yet in main storage.

Section 6: Diagnostic Aids

This section contains information on how IKJEGMNL handles STAE-intercepted abnormal terminations that occur in the TEST program.

Passing control to TEST

Figure 18 shows why IKJEGMNL received control, who from, and the resultant processing.

Event and Routine that passed Control	Pointer to the ECB in TCOMTAB	Resultant Processing by IKJEGMNL
A. (1) SVC 97 routine IBC0009G has handled a user breakpoint or a pseudo breakpoint. IGC0009G issues a POS/VS2T macro instruction to set IKJEGMNL's ECB.	ECBTST	1. IKJEGMNL automatically qualifies the problem program to the currently executable load module, if a new RB or TCB is active. 2. It then determines the type of breakpoint that the program executed. 3. If a user breakpoint was encountered, IKJEGMNL issues an AT message, updates the SUBCHAIN pointer in TCOMTAB from the current break element, and branches to location SCREQ to get a new subcommand 4. If, however, a pseudo breakpoint was encountered, IKJEGMNL restores the saved program restart address and registers, and branches to location SCREQ to get a new subcommand. Two special pseudo breakpoints receive unique treatment. One is set by IKJEGLDF during the processing of a LOAD, DELETE, GETMAIN, or FREEMAIN subcommand. The other special pseudo breakpoint is set by IKJEGINT to handle the exit from the problem program when the program completes (see event A(1)).
A. (2) The problem program has completed its execution normally. A special pseudo breakpoint (set by IKJEGINT) is encountered. IGC0009G issues a POS/VS2T macro instruction to set IKJEGMNL's ECB.	ECBTST	IKJEGMNL sets the problem program's restart address to that of an SVC 3 instruction to enable the program to exit via the supervisor's exit routine. IKJEGMNL issues the message, "PROGRAM UNDER TEST HAS TERMINATED NORMALLY" It then branches to location SCREQ to determine what subcommand the user wants to enter If the user enters GO, CALL, or RUN without operands, the problem program exits. However, if the user enters GO, CALL, or RUN with an operand, the restart address is altered by IKJEGGO, and the problem program continues to execute.
B. IGC0006A has received control from Contents Supervision or the overlay supervisor during the fetch of a load module of the problem program. IGC0006A issues a POST macro instruction to set IKJEGMNL's ECB.	ECBTST	IKJEGMNL determines if there are any deferred breakpoints that have to be activated in the newly fetched module by testing the defer element queue origin (DEFERTAB) in TCOMTAB. If there are any deferred breakpoints, IKJEGMNL links to the AT subcommand processor to set the breakpoints. It then issues a POST macro instruction to set IGC0006A to restart the problem program. Otherwise, IKJEGMNL merely posts IGC0006A and does not link to the AT subcommand processor

Figure 18. (Part 1 of 2) Passing Control to TEST

Event and Routine that passed Control	Pointer to the ECB in TCOMTAB	Resultant Processing by IKJEGMNL
C. The problem program has begun to terminate abnormally. The TMP's STAI Exit routine, IKJEFT04, has been entered from the ABEND/STAE Interface of the supervisor. IKJEFT04 issues a POST macro instruction to allow TEST to retry.	ECBTMPS	IKJEGMNL saves the problem program's registers and restart address. It then issues a POST macro instruction to set an ECB on which the TMP STAI Exit waits, specifying a retry code (X'7F') and passing the retry address. The retry address is TSTGO +6, the location of a pseudo breakpoint instruction. When the ABEND/STAE Interface regains control from the STAI Exit, it obtains linkage to the pseudo breakpoint (an SVC 97 instruction). When the SVC 97 routine has handled the breakpoint (event A), IKJEGMNL regains control to get a new subcommand from the terminal.
D. The problem program has completed normal termination. The supervisor EOT routine issues a POST macro instruction to set the termination's ECB.	ECBTERM	IKJEGMNL issues the message, "PROGRAM UNDER TEST HAS TERMINATED." It then branches to location ENDIT1 to clean up TEST and return to the TMP. This processing can occur only when there are no more RBs queued from TESTs subtask TCB.
E. Operator has issued a STOP TS or MODIFY TS command to set the LOGON/LOGOFF Scheduler's ECB using a POST macro instruction. IKJEGMNL tests this ECB as part of its ECB list.	ECBLOG	IKJEGMNL branches to location ENDIT1 to clean up TEST and return to the TMP. As a result, the LOGON/LOGOFF Scheduler logs off the terminal user.
F. The user enters or simulates two successive attention interruptions, while the problem program is running. He enters an input line other than "?" or a carrier return. The TMP Attention Exit routine, IKJEFT03, issues a POST macro instruction to set the attention ECB.	ECBTMPA	IKJEGMNL assumes that the IKJEFT04 issued a POST after an ABEND in the problem program. IKJEGMNL branches to IKJEGIO to issue a termination message. IKJEGIO tests the attention ECB, detects the attention interruption, does not issue the message, and returns to IKJEGMNL with a return code of 16, indicating that an attention interuption has been detected. IKJEGMNL tests the post code in the attention ECB and determines that the TMP, not TEST, was posted. IKJEGMNL then branches to location ENDIT1 to clean up TEST and return to the TMP. The TMP will invoke the new command processor that the user requested when he entered the attention interruption.

Figure 18. (Part 2 of 2) Passing Control to TEST

ABEND Handling in TEST

When IKJEGMNL encounters an ABEND in TEST, RETRY (the retry routine) frees the 104-byte STAE work area, if it exists. IKJEGMNL issues a STAE macro instruction to set up and queue a STAE control block (SCB). These actions establish TEST's STAE Exit routine.

If location TSTRSTRT in TCOMTAB contains a nonzero value, IKJEGMNL branches to the address contained at TSTRSTRT. When TSTRSTRT contains zeros, IKJEGMNL determines if END processing has begun. If END has begun, IKJEGMNL branches to location TMPRTURN, cleans up the TEST command processor, and returns to the TMP. Otherwise, IKJEGMNL's retry routine completes retry processing according to the type of error that caused the ABEND, as follows:

- If ATTACH of the Help command processor failed, IKJEGMNL issues the "HELP FAILED" message and branches to location SCREQ1 to get a new subcommand.
- If an error occurred in IKJEGMNL itself, it branches to location ENDIT1, cleans up the TEST command processor and returns to the TMP.
- If a second LINK to the same subcommand processor failed, IKJEGMNL issues the "LINK TO SUBCOMMAND FAILED" message, branches to ENDIT1, cleans up the TEST command processor, and returns to the TMP.
- If a subcommand processor failed transferring control to another subcommand module, IKJEGMNL branches to location SCREQ1 to get a new subcommand.
- If a link to a subcommand processor failed because of an I/O error during its fetch and a module of the problem program was being fetched concurrently, IKJEGMNL issues the "LINK TO SUBCOMMAND FAILED" message. It then branches to location IKJEGCTL which posts control back to the SVC 61 routine. The SVC routine continues the processing of the fetched module.
- If a link to a subcommand processor failed because of an I/O error during its fetch, and a module of the problem program was not being fetched concurrently, IKJEGMNL issues the " LINK TO SUBCOMMAND FAILED" message and branches to location SCREQ1 to get a new subcommand.

Special Error Termination Processing

For an error condition that requires a return to the TMP, TEST modules return to IKJEGMNL to conditionally flush the input stack and clear input buffers. IKJEGMNL conditionally removes non-terminal elements from the input stack (via the Stack service routine) and clears TCAM and time-sharing control buffers (via the TCLEARQ macro instruction). IKJEGMNL causes this cleanup before it returns to the TMP. (See the TSTFLUSH bit in the TSTFLGS4 field in TCOMTAB.)

Setting Pseudo Breakpoints

TEST Module That Sets the Pseudo Breakpoint	Special Condition That Exists when Breakpoint is Set	Processing by TEST Module
1. IKJEGLDR	Problem program has been loaded by the OS/VS2 Loader.	IKJEGLDR saves the first two bytes of the problem program's entry point instruction at location TSTGO in TCOMTAB. It places an SVC 97 instruction at the problem program's entry point. IKJEGLDR turns on the TSTGOSW flag in the TSTFLG3 field of TCOMTAB, and then issues an XCTL macro instruction to the program's entry point to execute the breakpoint. IKJEGINT will restore the first two bytes of the program's entry point instruction when control is passed to it from IGC0009G after the breakpoint is taken.
2. IKJEGLDR	IKJEGLDR is about to invoke a load module of the problem program. The module is already in main storage.	Processing is the same in number 1. If the desired load module is in system-protected storage, a program check occurs when IKJEGLDR issues the XCTL macro instruction.
3. IKJEGLDR	IKJEGLDR is about to invoke a load module of the problem program. The module is not in storage and will be fetched via an XCTL macro instruction.	IKJEGLDR turns on the TSTGOSW flag in the TSTFLG3 field of TCOMTAB, which indicates a pseudo breakpoint is set at the problem program's entry point. (The actual breakpoint is set by IKJEGINT as in number 1 when IKJEGINT temporarily gets control during the fetch process.)
4. IKJEGINT	TEST has been entered because a subtask of the TMP has started to terminate abnormally.	IKJEGINT moves the problem program's restart address in the TSTGO field of TCOMTAB. The saved address is the RBOPSW+4 field in the current PRB or IRB or the terminating task. IKJEGINT passes to the TMP STAI Exit routine a retry address (TSTGO+6) that contains an SVC 97 instruction.

TEST Module That Sets the Pseudo Breakpoint	Special Condition That Exists when Breakpoint is Set	Processing by TEST Module
5. IKJEGINT	TEST has been entered because a subtask of the TMP has been interrupted by an attention interruption.	IKJEGINT saves the problem program's restart address in the TSTGO field of TCOMTAB. The saved address is the RBOPSW+4 field in the current PRB or IRB of the oldest program task. IKJEGINT changes the RBOPSW+4 field to point to TSTGO+6, that contains an SVC 97 instruction.
6. IKJEGMNL	An ABEND has occurred during execution of the problem program or the HELP command processor. IKJEGMNL received control from the TMP STAI Exit routine.	IKJEGMNL sets a pseudo breakpoint as in number 4.
7. IKJEGATN	The user has entered or simulated an attention interruption during the TEST session.	An Attention Exit routine sets a pseudo breakpoint as in number 5.
8. IKJEGLDF	IKJEGLDF needs to force a task switch from the subtask to the TEST task.	IKJEGLDF saves the problem program's restart address in the TSTGO field of TCOMTAB. It then points the problem program's RBOPSW+4 field to a part of IKJEGLDF that is to be executed under the subtask. When this part of the module completes, IKJEGLDF branches to location TSTGO+6 to execute the pseudo breakpoint.
9. IKJEGQFY		IKJEGQFY sets a pseudo breakpoint as in number 5.
10. IKJEGINT		IKJEGINT sets a special pseudo breakpoint to permit the SVC 97 routine and IKJEGMNL to detect the problem program's completion. IKJEGINT points the saved problem program register 14 to an SVC 97 instruction at location PPEXIT in TCOMTAB. When executed, this pseudo breakpoint is recognized by both the SVC 97 routine and IKJEGMNL because of its unique location in TCOMTAB.

Index

Indexes to program logic manuals are consolidated in OS/VS Master Index of Logic, GY28-0603. For additional information about any subject listed below, refer to other publications listed for the same subject in the Master Index. The entries in this index are shown as they appear in the text of the book, that is, capitalized words are capitalized in text, lowercase words are lowercase in text, etc. When more than one reference is given for an entry, the primary reference is listed first.