File No. S360-36
Order No. GY28-6776-0

Program Logic

IBM System/360 Operating System:

Time Sharing Option

Command Processor Program Logic Manual

Volume 6	Program Number:
PROFILE	360S-CI-555
PROTECT	360S-UT-506
RENAME	360S-UT-506
RUN	360S-CI-555
SEND	360S-CI-555
SUBMIT	360S-CI-555
TERMINAL	360S-CI-555
TIME	360S-CI-555
WHEN/END	360S-CI-555

This publication describes the internal logic of the programs that handle the PROFILE, PROTECT, RENAME, RUN, SEND, SUBMIT, TERMINAL, TIME, and WHEN/END commands

The command processors are described with method of operation diagrams, flowcharts, and supporting text. Major data areas are shown, and directories for routines and principal data areas are provided.

This manual is intended for persons involved in program maintenance, or systems programmers who are altering the program design; it is not intended for normal use or operation of the programs described.

Information in this publication for TSO is for planning purposes until that item is available.

First Edition (March, 1971)

This edition applies to release 20.1, of IBM System/360 Operating System, 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 SRL Newsletter, Order No. GN20-0360, for the editions that are applicable and current.

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

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Programming Systems Publications, Department D58, PO Box 390, Poughkeepsie, N. Y. 12602

Preface

This publication is designed to supplement the program listings and make the information in the listing easier to access.

New techniques are used to help you understand the program and to enable you to refer quickly to flow charts or listings for details on the implementation. The publication relies on method of operation diagrams that show how the command processor routines operate and that provide functional "maps" (via cross-reference through the listings and flowcharts.

This command processor volume contains an introduction to the volume, followed by the documentation of the command processors. Each command processor is treated as a separate PLM. The General Information section contains information common to all the enclosed PLMs. The PLMs are:

- PROFILE Command Processor
- PROTECT Command Processor
- RENAME Command Processor
- RUN Command Processor
- SEND Command Processor
- SUBMIT Command Processor
- TERMINAL Command Processor
- TIME Command Processor
- WHEN/END Command Processor

Each of these PLMs is organized in six sections:

SECTION 1. INTRODUCTION: Relates the PLM to the users of its services, provides the operational considerations and physical characteristics of the Command Processor, and describes the environment in which the Command Processor routines operate.

SECTION 2. METHOD OF OPERATION: Describes the functions performed by the command processor. One of the new techniques used in this publication is the focal point of this section - the method of operation diagram. These diagrams have been designed to present the internal logic of a command processor, without relying on long stretches of text. The diagrams provide four kinds of information:

- Basic function (provided in the picture area).
- Modular interfaces (shown in Input to and Output from the module that is processing).
- Implementation supporting the function (provided as accompanying description).

- Pointers into the listings of flow charts (cross-references provided with the text description to lead you to the applicable routine writeup, flow chart, or label in the listings).

The diagrams are packaged at the rear of each PLM so that you can refer to them while using some other section of the same PLM.

SECTION 3. PROGRAM ORGANIZATION: Contains hierarchy drawings that show all the possible calling sequences of the routines that make up a command processor. Individual routine descriptions and flowcharts (which show the structure of the code) are provided.

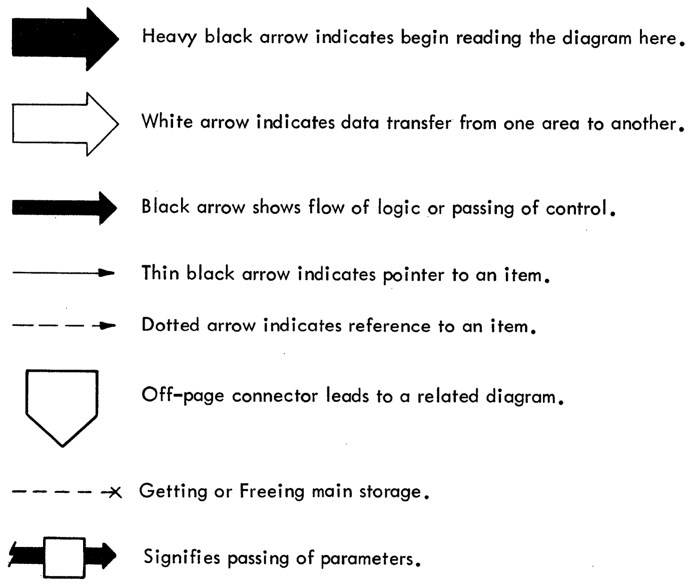
SECTION 4. DIRECTORY: Contains routine and major data area directories. Cross references from a routine or a data area, to the appropriate diagram or flowchart enhance the value of this section.

SECTION 5. DATA AREAS: Contains descriptions (or mappings) of the major data areas and tells which routines create the areas, which routines use and update the areas, and which routines refer to the areas without changing them.

SECTION 6. DIAGNOSTIC AIDS: Contains a register summary and a list of the messages that the command processor can display to the TSO user.

Symbols

The following symbols are used on the method of operation diagrams:



Heavy black arrow indicates begin reading the diagram here.

White arrow indicates data transfer from one area to another.

Black arrow shows flow of logic or passing of control.

Thin black arrow indicates pointer to an item.

Dotted arrow indicates reference to an item.

Off-page connector leads to a related diagram.

Getting or Freeing main storage.

Signifies passing of parameters.

Cross Reference

Each PLM is cross-referenced from one section to another. PLM-to-listing cross references are made in the method of operation section. Standard references in this volume are as follows:

MO -- Method of operation diagram.

FC -- Flowchart.

Prerequisite Publications

To use this volume you should be familiar with the publication, IBM System/360 Operating System: Time Sharing Option Command Language Reference, Order Number GC28-6732.

Related Publications

The TSO Command Processor PLMs are packaged in seven volumes. The contents of each volume is listed below.

Volume 1 GY28-6771

ACCOUNT
ACCOUNT ADD
ACCOUNT CHANGE
ACCOUNT DELETE
ACCOUNT LIST
ACCOUNT BROADCAST
ACCOUNT SUBROUTINES

Volume 2 GY28-6772

ALLOCATE
CALL
CANCEL/STATUS
DELETE

Volume 3 GY28-6773

EDIT

RELATED PUBLICATIONS (Cont.)

Volume 4 GY28-6774

EXEC
FREE
HELP
LINK
LISTALC
LISTBC

Volume 5 GY28-6775

LISTCAT
LISTDS
LOADGO
OPERATOR
OUTPUT

Volume 6 GY28-6776

PROFILE
PROTECT
RENAME
RUN
SEND
SUBMIT
TERMINAL
TIME
WHEN/END

Volume 7 GY28-6777

TEST

Contents — Volume 6

General Information

This section contains information on processing that is common to all the enclosed TSO Command Processor PLMs. Individual introductions are provided with each separate PLM that is included in this volume.

Terminal Monitor Program

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

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

- Constructs and initializes the data areas it requires.
- Loads the TIME command processor.
- Sets up STAE and STAI exits.
- Sets up Attention exits.
- Initializes the input stack with a terminal element.
- Issues the EXTRACT macro instruction to obtain pointers to both the STOP/MODIFY ECB and to the Protected Step Control Block (PSCB) that is built by the LOGON/LOGOFF scheduler.
- Informs the terminal it is "READY" for a command.
- Obtains the command buffer and validates the command.

The TMP is then able to attach a command processor as a subtask.

NOTE: The TEST Command Processor is entered from the TMP via a LINK macro instruction allowing it to execute at the same level as the TMP. The TIME Command Processor is branched to directly.

When a command processor completes its processing, control is returned to the TMP. For more information on the TMP, please refer to the publication, IBM System/360 Operating System: Time Sharing Option Terminal Monitor Program and Service Routines Program Logic Manual, Form No. GY28-6770.

Service Routines

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

- GETLINE, which obtains a line of input from an area defined as its source of input. Normally, this area contains input from the terminal.

- PUTLINE, which sends a line of output to the terminal.

- PUTGET, which sends a line of output to the terminal and waits for a line of input as a response.

- STACK, which establishes the source of input as a terminal; or (if not from a terminal) which places lines of input into areas from which GETLINE or PUTGET can obtain data.

- Command Scan, which checks the syntax of designated data to see if it is syntactically valid.

- PARSE (IKJPARS) (not used by the TMP), which checks the syntax of parameters of TSO commands. In certain cases, PARSE is directed to take exits to validity checking routines (provided by the processors). The validity checking routines are designed to dynamically assist the parse operation in providing valid input to the command or subcommand processor.

- Dynamic Allocation Interface Routine (IKJDAIR) (not used by the TMP), which provides information to the MVT dynamic allocation routines. In turn, these routines allocate, free, and concatenate data sets that relate to a TSO session.

These service routines are documented in full in the publication, IBM System/360 Operating System: Time Sharing Option Terminal Monitor Program and Service Routines PLM, Order Number GY28-6770.

Attention Interruptions

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

TMP ATTENTION EXIT ROUTINE

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

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

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

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

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

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

TEST command
: Without operands
: : the user wants to contine processing, under the control of the TEST command processor. Test receives control, to allow the user to enter a TEST subcommand(s). For instance; if GO is entered as a subcommand (without operands), processing is continued from the point of interruption in the command processor that has experienced the Attention interrupt.

: With operands
: : the command processor that has experienced the Attention interrupt is deleted via the DETACH macro, and a new environment is set up by the TEST command processor.

Command Processor Attention Exit Routine

None of the command processors in this volume have an Attention Exit routine.

ABEND Processing

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

When a command processor experiences an ABEND, the TMP's STAI Exit routine gets control to ensure the following:

- The user is notified that his command processor has experienced an ABEND

- The READY message is issued

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

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

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

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

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

TEST command
 Without operands
  the user wants to continue processing, under the control of the TEST command processor. TEST receives control, to allow the user to enter a TEST subcommand(s). For instance; if GO is entered as a subcommand (without operands), processing is continued from the point of interruption in the command processor that has experienced the ABEND.

TEST command
With operands
the command processor that has experienced the ABEND is deleted via the DETACH macro, and a new environment is set up by the TEST command processor.

Error Termination Procedure

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

Message Handling

Each TSO Command Processor has a message CSECT. The address of a particular message is provided (by the command processor) to the PUTLINE service routine -- which writes the message to the terminal.

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

Part 1: PROFILE Command Processor

1

Contents

Illustrations

Charts

Diagrams

Figures

Section 1. Introduction

The PROFILE command processor affords a terminal user the opportunity to define, or redefine at will, the personal set of options he wishes to use at a terminal. Once defined, the options specified are transferred to the User Profile Table assigned to the particular user. A user profile will remain in effect until its terminal user redefines it. The 'Operational Considerations' paragraph in this section lists the options available to the terminal user. The options available by default are shown in Figure 2-2 at the end of Section 2.

The PROFILE command is entered at a terminal to begin the process of defining a user's characteristics (or profile) to the system. A permanent record of a particular user's profile is kept in a User Profile Table (UPT). Once a user profile has been entered successfully, via a PROFILE command with at least one operand, it will remain in effect until another PROFILE command from the same user overrides it by specifying new operand(s). Any operands not specified on an overriding command entry will remain unchanged. A user who enters the PROFILE command unaccompanied by any operands will receive the following error message (hereafter referred to as the 'COMMAND IGNORED' message):

```
'IKJ56670I  NO OPERANDS, COMMAND IGNORED'
```

Functions

The PROFILE command processor records a terminal user's options by setting bits in the User Profile Table.

The processing sequence is as follows:

1. Establishes addressability by receiving the parameters passed as input. (See Diagram 2-1.)

2. If operand(s) are present in the command buffer, IKJEFT82 issues the LINK macro to cause control to be passed to PARSE (IKJPARS).

3. IKJPARS validates the operand(s) in the command buffer, builds a Parameter Control List (PCL) in dynamically acquired storage, records the user's options in this PCL, sets up a return code and returns to IKJEFT82. If the return code is not zero, 'COMMAND SYSTEM ERROR+' is displayed at the terminal via the PUTLINE service routine. Upon request, the second-level message, PARSE ERROR CODE xxx', is also displayed. Control is returned to the TMP.

4. IKJEFT82 is now able to transfer the user's profile information from its temporary home in the PCL to a permanent location in the User Profile Table (UPT). Dynamic storage used for the PCL is now released. When successful, PROFILE (IKJEFT82) sends a return code of 0 to the TMP.

Environment

The PROFILE command processor is processed in a TSO user's region under the region's protection key. The processor itself is loaded into about 2K of storage. Additional storage, for the TSO service routines used by PROFILE, is dynamically acquired by the PARSE routine and later freed when PROFILE issues the IKJRLSA macro.

Physical Characteristics

The PROFILE command processor is a load module named PROFILE that is a member of SYS1.CMDLIB. It has an alias name of 'PROF'. PROFILE is about 2K bytes long.

IKJEFT82 is the entry point name, the control section (CSECT) name, and the module name of the main processing program - the PROFILE command processor.

The Terminal Monitor Program (TMP-IKJEFT02) issues the ATTACH macro to enable IKJEFT82 to receive control.

Operational Considerations

PROFILE uses the TSO PUTLINE (IKJPUTL) service routine to send messages to a terminal. The PUTLINE macro generates a LINK macro which enables module IKJPUTL to receive control.

The Putline Parameter Block (IKJPTPB) is pointed to by the parameter list that is passed to PUTLINE.

The four parameters that comprise the input to this command processor are known as the Command Processor Parameter List (CPPL). When PROFILE receives control, register 1 contains the address of a CPPL that is structured as follows:

- Word 1 (CPPLCBUF) Points to the command buffer.

- Word 2 (CPPLUPT) Points to the User Profile Table (UPT).

- Word 3 (CPPLPSCB) Points to the Protected Step Control Block (PSCB).

- Word 4 (CPPLECT) Points to the Environmental Control Table (ECT).

The PROFILE command consists of the following:

PROFILE	Commandname
CHAR(character)/ CHAR(BS)/ NOCHAR	Specifies a character delete control character. Specifies the backspace as a character delete control character. Specifies that a character delete control character is not defined for this user.
LINE(character)/ LINE(ATTN)/ NOLINE	Specifies a line delete control character. Specifies the ATTN key as a line delete control character. Specifies that a line delete control character is not defined for this user.
PROMPT/ NOPROMPT	Specifies the user's desire to be prompted whenever possible. Specifies that the user should not be prompted.
INTERCOM/ NOINTERCOM	Specifies the user's desire to receive messages sent to him by other terminal users. Specifies that the user does not wish to receive messages from other users.
PAUSE/ NOPAUSE	When the user receives a message that has additional message levels available, he is to have the option of either requesting the next level message (by entering a "?"), or of ignoring any further message levels (by entering a nullline). Each additional message level can be accessed by a separate request ("?"). When the last level has been reached, the user will be notified by an appropriate message. Specifies that the user does not wish to be notified (by prompting) that the message he has just received has additional messages available with it.
MSGID/ NOMSGID	Specifies that messages sent to this user are to include message identifiers. Specifies that messages sent to this user are to exclude message identifiers.

NOTE: From each group of operands shown above, only one operand may be specified each time a user profile is entered.

Section 2. Method of Operation

This section describes the program logic of the PROFILE command processor. The complete program logic of the TMP and the TSO service routines used by PROFILE can be found in the Terminal Monitor Program and Service Routines Program Logic Manual, GY28-6770.

PROFILE Command Processing Summary

The information contained in the Command Processor Parameter List (CPPL) is used to initialize pointers which will be used by PROFILE-and by its service routines-to access data in storage. (See Diagram 2-1).

The ECTNOPD bit switch in the Environmental Control Table (ECT) is tested to determine if the command buffer field has any operands. If the bit is zero, there are no operands; in this case, the 'COMMAND IGNORED' message is sent to the user at his terminal via the facilities of the PUTLINE macro, and control is returned to IKJEFT02, the calling routine.

If operands are present in the command buffer, PROFILE builds the Parse Parameter List (PPL), puts the address of the PPL in register 1, and issues the LINK macro to pass control to IKJPARS(PARSE).

PARSE issues a Getmain macro for storage in which it builds a Parameter Descriptor List(PDL). The User options, specified by the particular operands in the command buffer, are validated and then recorded by setting switches in the PDL. A pointer to this PDL can then be passed back to PROFILE. PARSE then sets up a return code in register 15 and returns control to PROFILE.

If the return code from PARSE is not zero, PROFILE sets up the INVPARM message, and issues a PUTLINE macro to deliver it to the user at the terminal, before returning control to IKJEFT02. If the return code is not zero, PROFILE then checks the pointer to the PDL. If this PDL pointer equals zero, PROFILE sets up the 'COMMAND IGNORED' message, issues the PUTLINE macro to display it to the user, and returns control to IKJEFT02.

PROFILE can now examine the Parameter Descriptor List. PROFILE uses the PDEP mapping structure, which has been declared as an overlay DSECT, to refer to the PDL core received from PARSE. Each user specified option will be indicated by a value of one(1) or two(2) in its respective field in the PDEP overlay of the storage acquired by the PARSE routine. (Zero in the field indicates the absence of the option). This temporary record of user options can now be made permanent by setting the appropriate switches in the User Profile Table (UPT).

NOTE: The UPT can be modified by Command Processors and by Service Routines, but its storage cannot be freed by them.

The PDL storage which was acquired dynamically by the PARSE routine can now be freed. The IKJRLSA macro, whose expansion generates a Freemain macro, is issued by PROFILE and the storage is released. Control is now returned to IKJEFT02, the routine that called PROFILE, along with a normal return of zero in register 15.

PARSE Service Routine Processing Summary

Summary of PARSE processing when used by the PROFILE command processor (See Diagram 2-2).

IKJPARS(PARSE) receives control from IKJEFT82 (the main processing program). PARSE issues a GETMAIN macro for storage in which to build a PDL. PARSE now concentrates on the Parameter Control List (PCL) which consists of a number of Parameter Control Entries (PCE's). There is one PCE for each IKJNAME macro specified by PROFILE. (This actually amounts to a list of all the keyword operands that may be used with the PROFILE command). PARSE accomplishes the syntax check of the operands appearing in the command buffer approximately as follows (refer to the TMP Service Routines PLM for a detailed explanation):

1. Simultaneously scans the PCL and the command buffer looking for matches.

2. If an operand in the command buffer also appears in the PCL, the operand is a valid one.

3. Each valid operand generates an entry in the PDL in the order in which the valid operands appear in the PCE's. Thus, the order of the PDL reflects the order of the PCL, which in turn reflects the order in which the Parameter Control Entries were specified by the PARSE macros.

   NOTE: PARSE is a table driven routine according to the type of Parameter Control Entry (PCE) found in the Parameter Control List (PCL). Each type of PCL is further processed by a separate, lower-level routine. These secondary routines: check for missing or default parameters; check syntax of parameters; check for validity check exits; prompt the user in error situations, if required.

4. PARSE determines if either of the two Validity Checking Routines, that have been link edited into the PROFILE load module (as in-line subroutines), will be used:

A. CHARCHEK ROUTINE - This routine is entered by PARSE whenever the user has entered the 'CHAR' keyword along with a specified character(s). A maximum of two characters may be used. Any single character is valid except the new line (NL) character. The characters, 'BS', specify the backspace character as the character delete control character.

B. LINECHEK ROUTINE - This routine is entered by PARSE whenever the user has entered the 'LINE' keyword along with a specified character or character string. Any single character is valid except the new line (NL) character. The character string 'ATTN', specifies that ATTENTION is to be used as the Line Delete control character.

5. The PDEP data area is an overlay of the PDL whose pointer is returned to PROFILE by IKJEFP01(PARSE) via an answer place provided by Profile in the PPL.

6. PARSE uses register 15 to send PROFILE a return code:

```
    0 = SUCCESSFUL
NON 0 = UNSUCCESSFUL
```

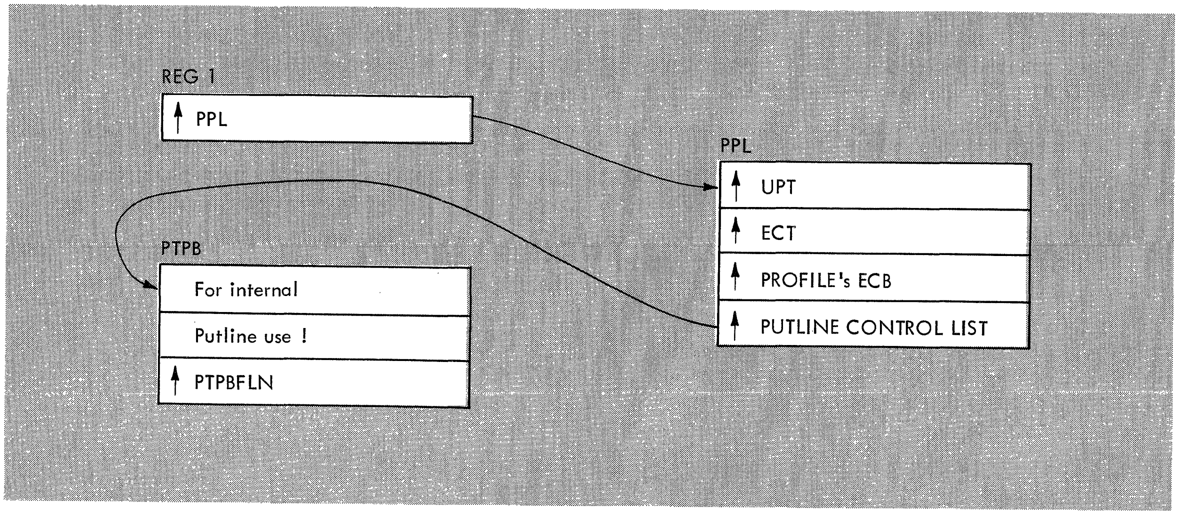
Method of Operation Diagrams

The diagrams illustrate the method of operation-or processing-of the PROFILE command processor. The cross reference table in the diagram may help find the corresponding blocks on a flowchart, or the respective sections of code in the program listings.

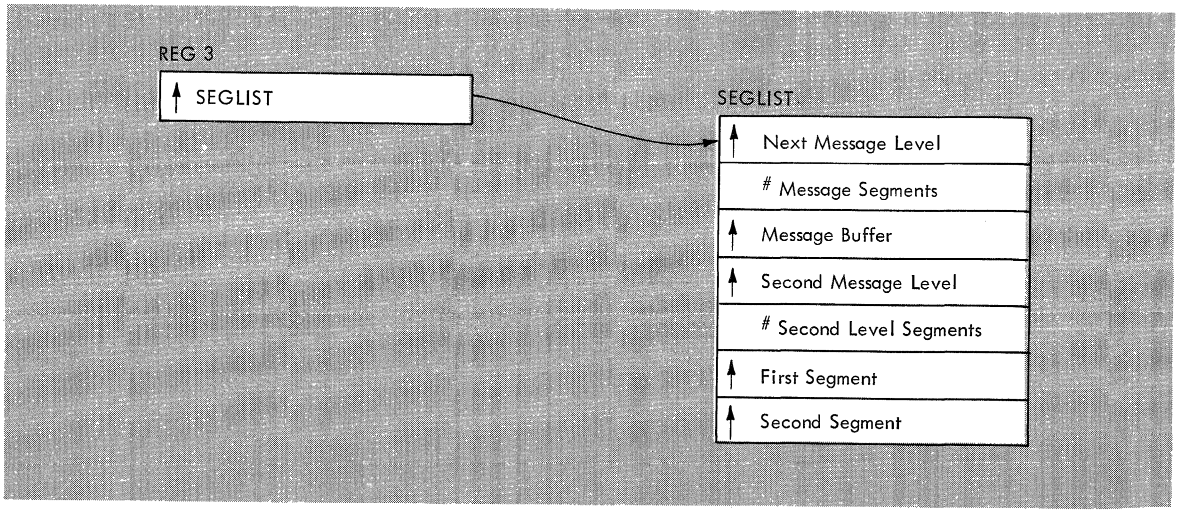
Note: The method of operation diagrams appear at the rear of this PLM.

ADDITIONAL FIGURES

The following figures are referenced on the method of operation diagrams:

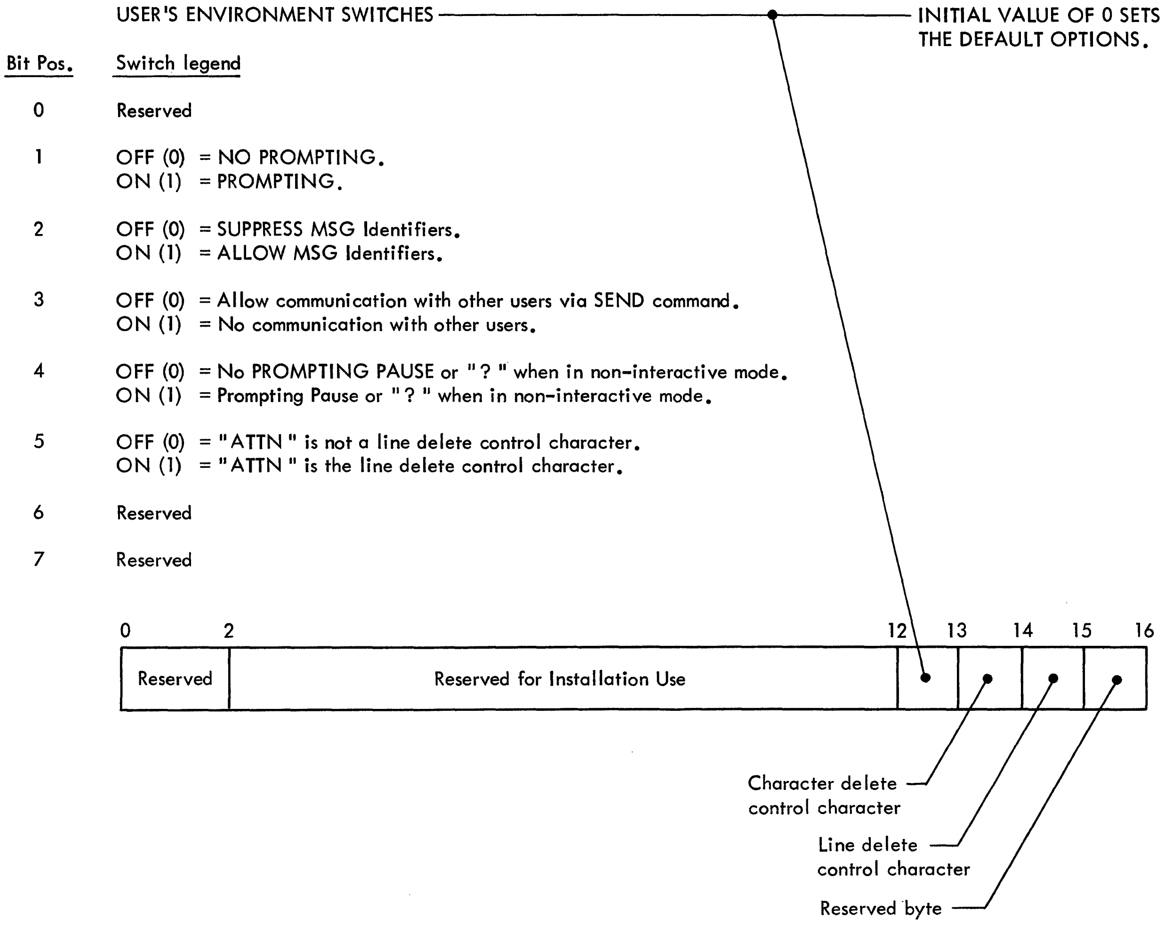


Input to PUTMSGS from IKJEFT82.



Input to IKJPUTL from PUTMSGS.

Figure 2-1. Total Input to IKJPUTL



USER'S ENVIRONMENT SWITCHES
INITIAL VALUE OF 0 SETS THE DEFAULT OPTIONS.
0
2
12
13
14
15
16
Reserved
Reserved for Installation Use
Character delete control character
Line delete control character
Reserved byte

Bit Pos.	Switch legend
0	Reserved
1	OFF (0) = NO PROMPTING. ON (1) = PROMPTING.
2	OFF (0) = SUPPRESS MSG Identifiers. ON (1) = ALLOW MSG Identifiers.
3	OFF (0) = Allow communication with other users via SEND command. ON (1) = No communication with other users.
4	OFF (0) = No PROMPTING PAUSE or "? " when in non-interactive mode. ON (1) = Prompting Pause or "? " when in non-interactive mode.
5	OFF (0) = "ATTN " is not a line delete control character. ON (1) = "ATTN " is the line delete control character.
6	Reserved
7	Reserved

Figure 2-2. User Profile Table (16 bytes long)

Section 3. Program Organization

This section presents the PROFILE command processor, its associated routines, and the relationships among them.

The hierarchy of the routines is shown in Figure 3-1.

Hierarchy

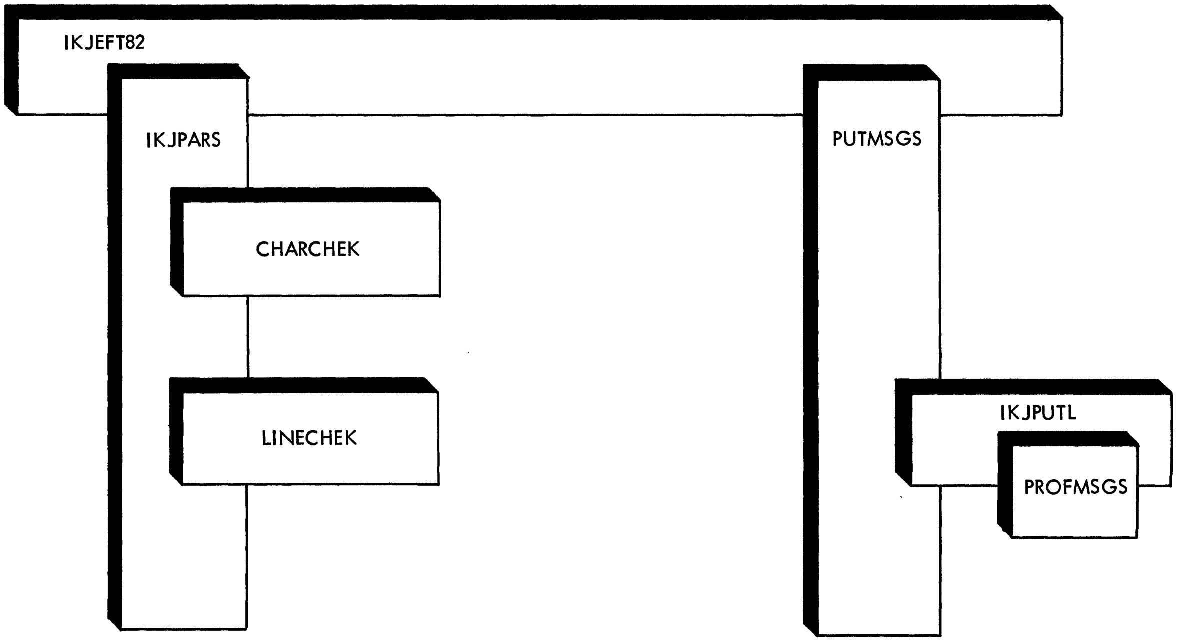


Figure 3-1. Hierarchy of the PROFILE Command Processor

IKJEFT82—PROFILE Control Routine

<table>
<tr><td>Entry Points</td><td>IKJEFT82</td></tr>
<tr><td>Processing Operations</td><td>Specify subpool 1 as recipient of all dynamically acquired storage.
Map PUTLINE's PCL in subpool 1.
Pick up parameters for use by service routines.
Test ECT bit for presence of operands in command buffer.
Use PUTLINE to send error messages.
Use IKJPARS to validate operands.</td></tr>
<tr><td>Data Areas</td><td><table>
<tr><th>Data Area</th><th>Defined</th><th>Used</th><th>Changed</th></tr>
<tr><td>BSATTN</td><td>X</td><td></td><td></td></tr>
<tr><td>CHARDEL</td><td>X</td><td></td><td></td></tr>
<tr><td>CHARPDE</td><td></td><td>X</td><td></td></tr>
<tr><td>CHARTDEL</td><td>X</td><td></td><td></td></tr>
<tr><td>CODE</td><td>X</td><td></td><td>X</td></tr>
<tr><td>CPECB</td><td>X</td><td></td><td></td></tr>
<tr><td>CPPL</td><td></td><td>X</td><td></td></tr>
<tr><td>DELETCAL</td><td>X</td><td></td><td></td></tr>
<tr><td>ECT</td><td></td><td></td><td>X</td></tr>
<tr><td>ERRCODE</td><td>X</td><td></td><td></td></tr>
<tr><td>IGMSG</td><td>X</td><td></td><td></td></tr>
<tr><td>INVPARM</td><td>X</td><td></td><td></td></tr>
<tr><td>MSGS</td><td>X</td><td></td><td></td></tr>
<tr><td>PDEP</td><td></td><td>X</td><td></td></tr>
<tr><td>PDEPTR</td><td>X</td><td></td><td></td></tr>
<tr><td>PPL</td><td>X</td><td></td><td>X</td></tr>
<tr><td>PTPB</td><td></td><td>X</td><td>X</td></tr>
<tr><td>PTPBPTR</td><td>X</td><td></td><td></td></tr>
<tr><td>PUTLIST</td><td>X</td><td></td><td></td></tr>
<tr><td>PUTLNG</td><td>X</td><td></td><td></td></tr>
<tr><td>PUTPARM</td><td>X</td><td></td><td></td></tr>
<tr><td>RCODE</td><td>X</td><td></td><td></td></tr>
<tr><td>SEGLIST</td><td>X</td><td></td><td></td></tr>
<tr><td>UPT</td><td></td><td>X</td><td>X</td></tr>
</table></td></tr>
<tr><td>Routines Used</td><td>IKJPARS, IKJPUTL</td></tr>
<tr><td>Exits</td><td>LINK to IKJPARS, LINK to IKJPUTL, RET to TMP(IKJEFT02)</td></tr>
</table>

IKJEFT82 -- PROFILE Control Routine (Cont.)

Registers on Entry and Exits	On entry, register 1 contains the addr of a CPPL. On exit to PARSE, register 1 contains the addr of the PPL. On exit to PUTLINE: register 3 contains the addr of SEGLIST (the message list); while register 1 contains the addr of PPL (the first four words are also used as the Putline parameter list). On return to TMP, register 15 contains a return code.

Chart AA. IKJEFT82. PROFILE Command Processor (1 of 2)

Cross Reference: Diagram 2-1, 2-2.

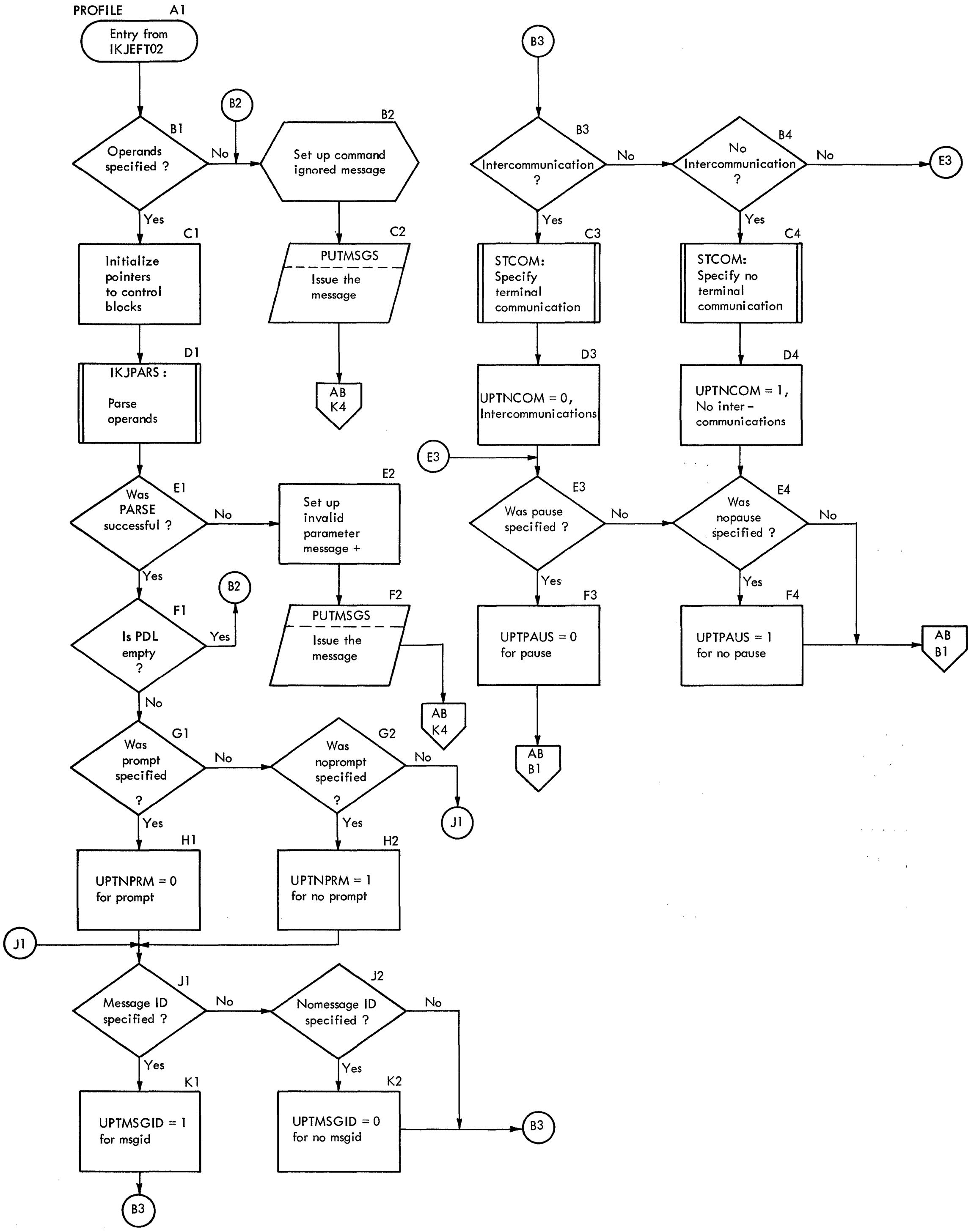
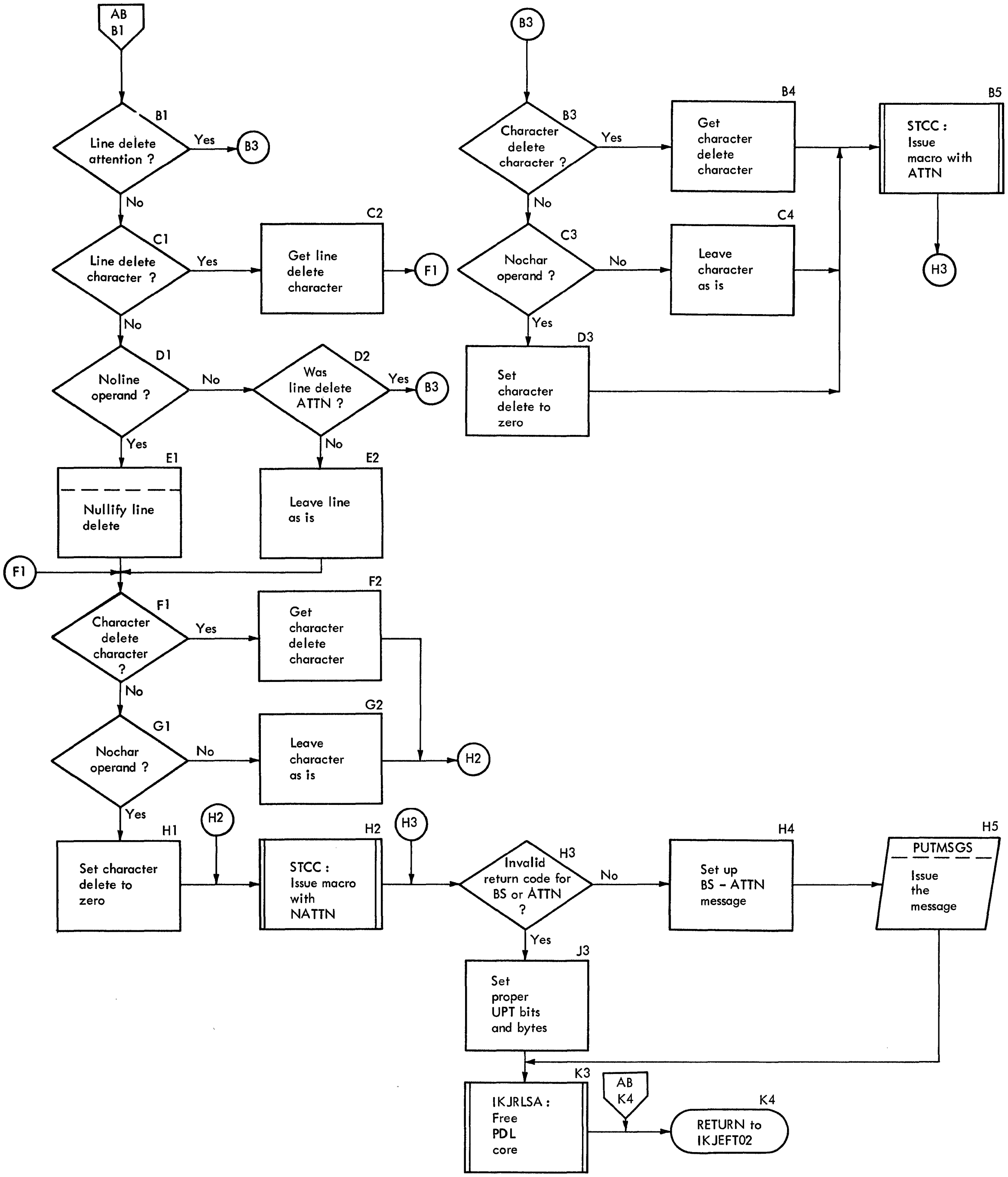


Chart AB. IKJEFT82. PROFILE Command Processor (2 of 2)

Cross Reference: Diagram 2-3.


AB B1
B1 Line delete attention ?
Yes
B3
No
C1 Line delete character ?
Yes
C2 Get line delete character
F1
No
D1 Noline operand ?
No
D2 Was line delete ATTN ?
Yes
B3
No
Yes
E1 Nullify line delete
E2 Leave line as is
F1
F1 Character delete character ?
Yes
F2 Get character delete character
No
G1 Nochar operand ?
No
G2 Leave character as is
H2
Yes
H1 Set character delete to zero
H2
H2 STCC: Issue macro with NATTN
H3
H3 Invalid return code for BS or ATTN ?
No
H4 Set up BS - ATTN message
H5 PUTMSGS Issue the message
Yes
J3 Set proper UPT bits and bytes
K3 IKJRLSA: Free PDL core
AB K4
K4 RETURN to IKJEFT02
B3
B3 Character delete character ?
Yes
B4 Get character delete character
B5 STCC: Issue macro with ATTN
H3
No
C3 Nochar operand ?
No
C4 Leave character as is
Yes
D3 Set character delete to zero

Section 4. Directory

This section consists of two directories, each organized alphabetically. The first, entitled "Routine Directory", contains all entry point and routine names referenced or used by the PROFILE command processor. The second, entitled "Data Area Directory", lists all the data areas used by PROFILE.

Routine Directory

Entry or	Load	Control	Cross reference	
Routine Name	Module	Section	MO	FC
CHARCHEK	IKJEFT82	IKJEFT82	2-2	AB
IKJEFT02	IKJEFT02	IKJEFT82	2-1	AA
IKJPARS	IKJPARS	IKJEFP01	2-1	AA
IKJPUTL	IKJPTGT	IKJEFT40	2-1	AA
LINECHEK	IKJEFT82	IKJEFT82	2-2	AB
PROFMSGS	IKJEFT82	IKJEFT82	2-1	AA
PUTMSGS	IKJEFT82	IKJEFT82	2-1	AA

Data Area Directory

Data Area	MO	FC
BSATTN	6-1	AB
CHARDEL	2-2	AB
CHARPDE	2-3	AA
CHARTDEL	N/A	AB
CODEM	2-4	AA
CPECB	N/A	AA
DELETCAL	N/A	(See listings)
ECT	Mapping Macro (See listings)	AA
ERRCODE	6-1	AA
INVPARM	6-1	AA
MSGS	6-1	AA
PDEP	2-2	AA
PDEPTR	2-2	AA
PPL	2-2	AA
PTPB	2-4	AA
PTPBPTR	2-1	AA
PUTLIST	N/A	(See listings)
PUTLNG	N/A	(See listings)
PUTPARM		
RCODE		
SEGLIST	2-4	AA
UPT	2-5	AB

Section 5. Data Areas

This section contains the major data areas used by the PROFILE command processor. Each major data area is represented in table format; subfields within the major data area have separate entries in the table. The size and use of each subfield is included in the table. For each major data area, both the creating and the using routines are given.

CHARPDE—PDE from PARSE

Created by: IKJPARS (PARSE)

Used by: Used when IKJPARS(PARSE) enters a validity checking routine in reponse to the appearance of either the 'LINE' or the 'CHAR' keyword, and a character(s).

Contents:

Disp	Field	Size and Use
0 (0)	CHARPTR	4 bytes -- Points to character(s) specified as either the Line Keyword or the CHARACTER Keyword.
4 (4)	CHARLEN	2 bytes -- Length of character(s) specified in CHARPTR.
6 (6)	FLGRESV	2 bytes -- Unused.

ECT—Environmental Control Table

Created by: The Environmental Control Table (ECT) is built by the TMP and stored in a nonshared subpool.

Used by: Its fields can be modified by a command processor, or by a service routine, but it cannot be freed by a command processor.

Contents:

Disp	Field	Size and Use
0 (0)	ECTRCDF	1 byte -- High-order bit on indicates a CP Abend.
1 (1)	ECTRTCD	3 bytes -- Return code from previous CP or ABEND code (if ECTRCDF is ON).
4 (4)	ECTIOWA	4 bytes -- Address of I/O work area.
8 (8)	ECTMSGF	1 byte -- High-order bit on indicates delete second level message.
9 (9)	ECTSMSG	3 bytes -- Address of second level message chain.
12 (C)	ECTPCMD	8 bytes -- Primary command name.
20 (14)	ECTSCMD	8 bytes -- Secondary command name.
28 (1C)	ECTSWS	1 byte -- Switches.
	ECTNOPD	bit 0 -- ON=No operands exist in command buffer.
	*	bit 1 -- Reserved.
	ECTATRM	bit 2 -- ON=CP terminated by TMP detach with STAE EQU.
	ECTLOGF	bit 3 -- ON=LOGON/OFF requested TMP to LOGOFF user bits.
	ECTNMAL	bit 4 -- ON= No user messages to be received at LOGON.
	ECTNNOT	bit 5 -- ON= No broadcast notices to be received at LOGON.
	*	bit 6 -- Reserved.
	*	bit 7 -- Reserved.
29 (1D)	*	3 bytes -- Reserved.

PDEP—DSECT of PDL

Created by: This mapping of the PDL is created by Parse module, IKJEFP01.

Used by: IKJEFT82 -- PROFILE COMMAND PROCESSOR.

Contents:

Disp	Field	Size and Use
0 (0)	STORPTR	4 bytes -- Ptr to Parameter Descriptor List (PDL).
4 (4)	STORB	4 bytes -- Subpool number and PDL size.
8 (8)	PRMT	2 bytes -- Prompt/No prompt Keyword.
10 (A)	MSGID	2 bytes -- Message ID/No MSGID keyword.
12 (C)	INTERCOM	2 bytes -- Intercom/No Intercom keyword.
14 (E)	PAUSE	2 bytes -- Pause/No Pause keyword.
16 (10)	CHART	2 bytes -- Character Delete/No Character Delete Keyword.
18 (12)	LINE	2 bytes -- Line Delete/No Line Delete Keyword.
20 (14)	CHARID	4 bytes -- Ptr to character delete character.
24 (18)	CARLEN	2 bytes -- Length of character delete character.
26 (1A)	*	2 bytes -- Unused.
28 (1C)	LINEID	4 bytes -- Ptr to line delete character.
32 (20)	LINLEN	2 bytes -- Length of line delete character.
34 (22)	*	2 bytes -- Unused.

PPL—Parse Parameter List (Dual Purpose)

Created by: IKJEFT82(PROFILE).

Used by: The first four fields are used by PUTLINE. Parse module IKJEFP01 uses the complete data area.

Contents:

Disp	Field	Size and Use
0 (0)	UPTPTR	4 bytes -- Ptr to User Profile Table (UPT).
4 (4)	ECTPTR	4 bytes -- Ptr to Environmental Control Table (ECT).
8 (8)	ECBPTR	4 bytes -- Ptr to Command Processor's (CP) Event Control Block (ECB).
12 (C)	PUTPARMP	4 bytes -- Ptr to Putline Control List.
12 (C)	PCLPTR	4 bytes -- Ptr to Parameter Control List. (Occupies same field as PUTPARMP).
16 (10)	ANSPTR	4 bytes -- Ptr to Parameter Descriptor List's (PDL) Pointer.
20 (14)	CBUFAD	4 bytes -- Ptr to Command Buffer.
24 (18)	WORKAP	4 bytes -- Ptr to PROFILE's save area.

PTPB—Putline Parameter Block

Created by: IKJEFT82(PROFILE).

Used by: Used by Putline to control its own functions as well as to return data to the calling routine.

Contents:

Disp	Field	Size and Use
0 (0)	*	8 bytes -- Internal Putline usage.
8 (8)	PTPBLFN	4 bytes -- Ptr to formatted line returned because 'OUTPUT=(ADDR,FORMAT)' was specified.

Note: The PTPB is illustrated in Figure 2-1.

SEGLIST—Message Segment List

Created by: IKJEFT82(PROFILE).

Used by: IKJPUTL(PUTLINE) uses SEGLIST to send messages to the terminal.

Contents:

Disp	Field	Size and Use
0 (0)	LEVEL2P	4 bytes -- Contains address of second level message list, or X'FF00000' if none.
4 (4)	NUMSEG	4 bytes -- Number of message segments.
8 (8)	SEGMENT	4 bytes -- Contains address of First Level message.
12 (C)	LEVELNP	4 bytes -- Pointer to the Next Level of message segments.
16 (10)	NUMSEG2	4 bytes -- Number of message segments at the Second level.
20 (14)	SEGMENT2	4 bytes -- Contains address of EFirst message segment.
24 (18)	SEGMENT3	4 bytes -- Contains address of Second message segment.

Note: SEGMENT2 and SEGMENT3 make up the second level message.

UPT—User Profile Table

Created by: UPT is built by the LOGON/LOGOFF Scheduler from data stored in the UADS.

Used by: Shared by the Terminal Monitor Program (TMP) and LOGON/LOGOFF. It can be modified by command processors and by service routines, but cannot be freed by them.

Contents:

Disp	Field	Size and Use
0 (0)	*	2 bytes -- Reserved on a word boundary.
2 (2)	UPTUSER	10 bytes -- Reserved for installation use.
12 (C)	UPTSWS	1 byte -- User's environment switches.
	*	bit 0 -- Reserved.
	UPTNPRM	bit 1 -- OFF= No prompting.
	UPTMID	bit 2 -- OFF= Suppress message identifiers.
	UPTNCOM	bit 3 -- OFF= User communication allowed via SEND command.
	UPTPAUS	bit 4 -- OFF= No prompting pause or '?' when in noninteractive mode.
	UPTALD	bit 5 -- OFF= ATTN is not the line delete character. ON= ATTN is the line delete character.
	*	bit 6 -- Reserved.
	*	bit 7 -- Reserved.
13 (D)	UPTCDEL	1 byte -- Character delete character.
14 (E)	UPTLDEL	1 byte -- Line delete character.
15 (F)	*	1 byte -- Reserved.

NOTE: The UPT is illustrated in Figure 2-2.

Section 6. Diagnostic Aids

This section contains a register summary and a list of PROFILE messages.

Register Summary

The PROFILE command processor, and its associated routines, all use the following registers in the conventional way:

Register	Contents/Use
1	Ptr to Parameter list.
11	Base Register.
12	Base Register for DSECT Addressability of Reentrant code.
13	Ptr to current routine's save area.
14	Return address.
15	Return code.

Additional register usage is broken down by routines:

IKJEFT82(PROFILE)

Register	Contents/Use
3	Pointer to RCODE if PARSE is unsuccessful. Character delete character.
6	Line delete character.

CHARCHEK

Register	Contents/Use
3	Pointer to PDE.
11&12	Restored from PARSE's save area to establish addressability.

LINECHEK

Register	Contents/Use
3	Pointer to PDE.
11&12	Restored from PARSE's save area to establish addressability.

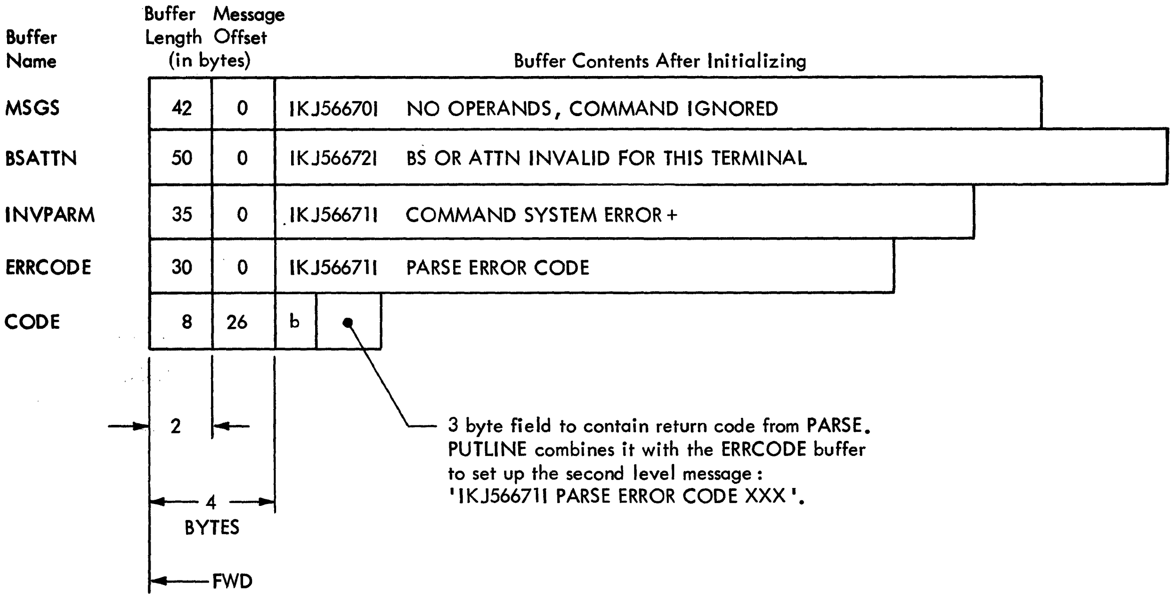
PUTMSGS

Register	Contents/Use
3	Pointer to the message segment list.
4	Register 14 (Return address) is saved here when PUTMSGS is entered.

Messages

IKJEFT82(PROFILE) uses an overlay CSECT to address the message list residing permanently in the PROFILE load module. The message list consists of 4 buffers which reside contiguously in the main storage assigned to PROFILE at loading time. The overlay CSECT, 'PROFMSGS', is used to initialize the four message buffers to the values shown in Figure 6-1. The fifth buffer, 'CODE', will receive a copy of any error return code that PARSE may send to PROFILE. Any such error code (of three digits) can then be added to the ERRCODE buffer by PUTLINE, before sending the "PARSE ERROR CODE XXX" message to a terminal.

NOTE: The ERRCODE buffer contains a second level message to the message in the INVPARM buffer.



Buffer Name	Buffer Length (in bytes)	Message Offset (in bytes)	Buffer Contents After Initializing
MSGS	42	0	IKJ56670I NO OPERANDS, COMMAND IGNORED
BSATTN	50	0	IKJ56672I BS OR ATTN INVALID FOR THIS TERMINAL
INVPARM	35	0	IKJ56671I COMMAND SYSTEM ERROR +
ERRCODE	30	0	IKJ56671I PARSE ERROR CODE
CODE	8	26	b ●

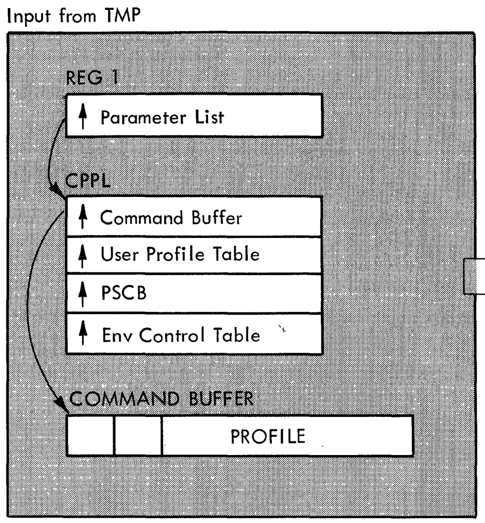
3 byte field to contain return code from PARSE. PUTLINE combines it with the ERRCODE buffer to set up the second level message : 'IKJ56671I PARSE ERROR CODE XXX'.

Figure 6-1. PROFILE Message CSECT(PROFMSGS)

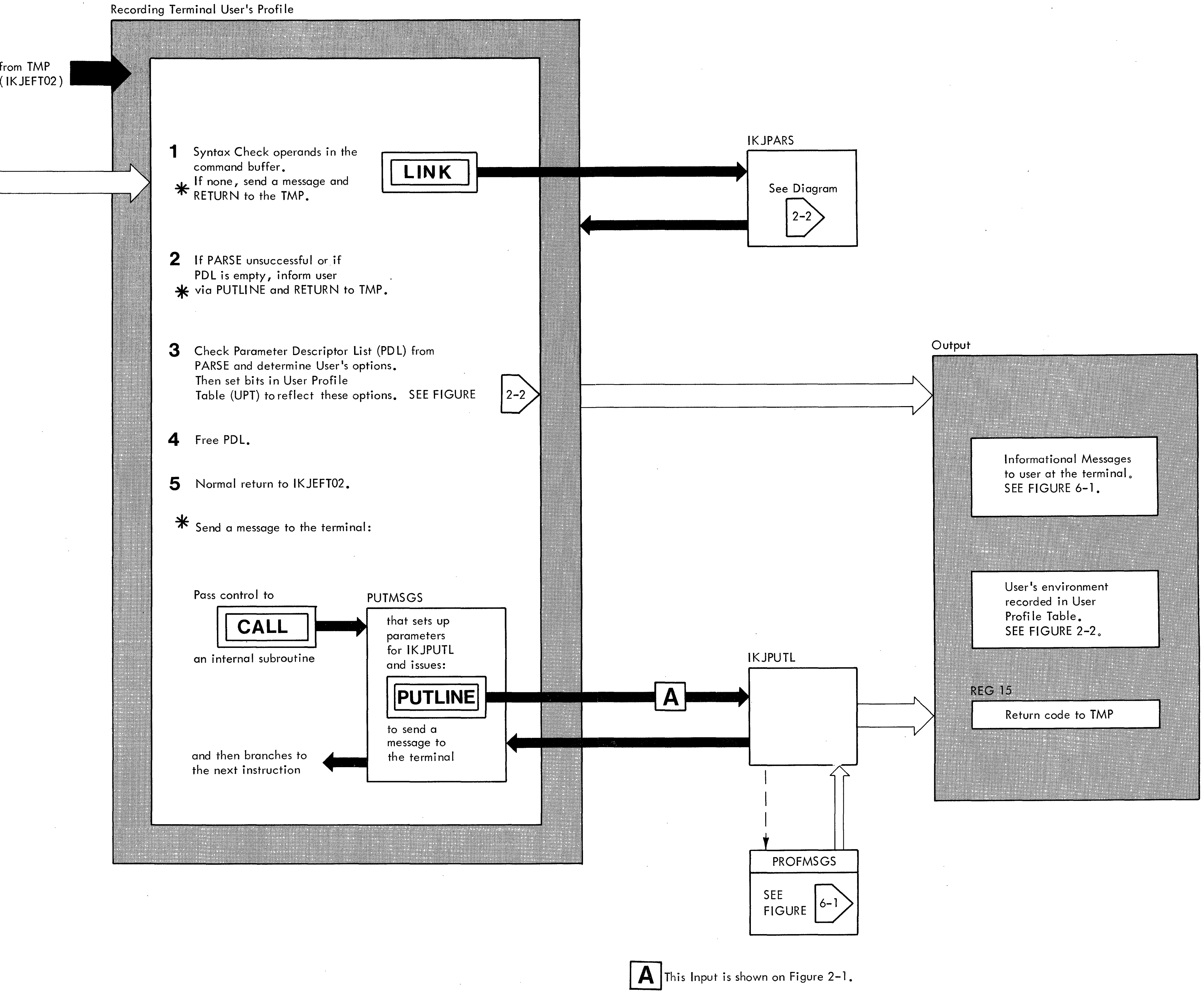
Index

Indexes to program logic manuals are consolidated in the publication <u>IBM System/360 Operating System: Program Logic Manual Master Index</u>, Order No. GY28-6717. For additional information about any subject listed below, refer to other publications listed for the same subject in the Master Index.

Input from TMP



Recording Terminal User's Profile



1 Syntax Check operands in the command buffer.
* If none, send a message and RETURN to the TMP.

2 If PARSE unsuccessful or if PDL is empty, inform user
* via PUTLINE and RETURN to TMP.

3 Check Parameter Descriptor List (PDL) from PARSE and determine User's options. Then set bits in User Profile Table (UPT) to reflect these options. SEE FIGURE 2-2

4 Free PDL.

5 Normal return to IKJEFT02.

* Send a message to the terminal:

PUTMSGS that sets up parameters for IKJPUTL and issues: PUTLINE to send a message to the terminal

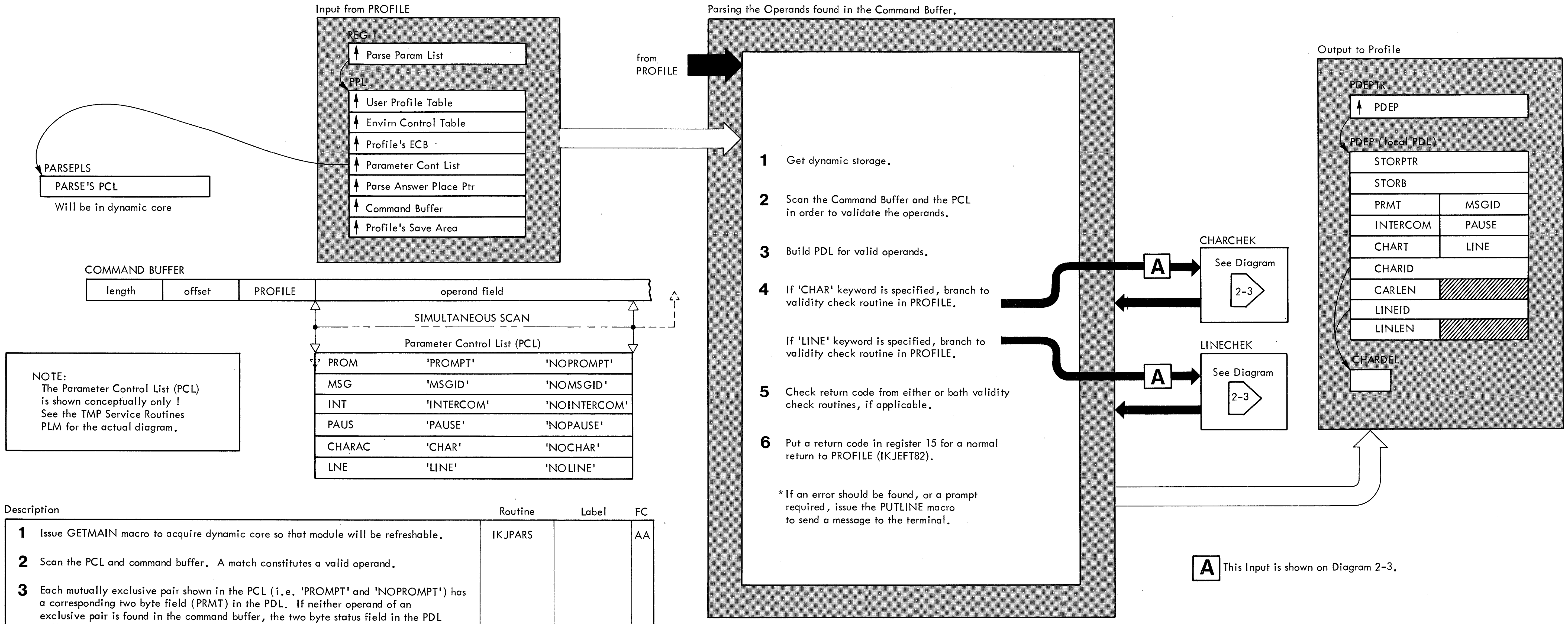
Informational Messages to user at the terminal. SEE FIGURE 6-1.

User's environment recorded in User Profile Table. SEE FIGURE 2-2.

A This Input is shown on Figure 2-1.

Description	Routine	Label	FC
1 Issue list form of PUTLINE macro to map its PCL in subpool 1. Check ECT bitswitch (ECTNOPD) for presence of operands in command buffer. If ON (1): No operands are in command buffer. "Ignored" message is sent to user via PUTLINE. Return code of 0 is sent via RETURN to IKJEFT02. If OFF (0): Operands are in command buffer. Ptrs are initialized (using CPPL parameter values). Pass control to IKJPARS (PARSE) via LINK macro.	IKJEFT82	PUTPARM IGNORMSG	AA AA
2 When control is returned, check return code from PARSE. If reg 15 ≠ 0: PARSE was not successful; "PARSE Error" message is sent to user via PUTLINE. Return to TMP with return code = 0. Check pointer to Parameter Descriptor List built by PARSE. If PDEPTR = 0: PDL is empty; "Ignored" message is sent to user via PUTLINE. Return to TMP with return code = 0.		PUTMSGS IGNORMSG	AA AA
3 The STCC macro is issued to set or change either a line-delete, or a character delete, control character(s) in the Time Sharing Block. If this macro returns an '8' return code, the 'BSATTN' message will be sent to the terminal. Processing them continues from step 4.		CHKMSGID	AB
4 Issue IKJRLSA macro to free the main storage found at the location pointed to by PDEPTR.		FREEUP1	AB
5 Normal return to IKJEFT02. Register 15 has a 0 as the return code.		RET	AB
*Set up the message address in the SEGMENT data field, along with the next message level indicator field (LEVEL2P), before issuing the CALL macro.			

Diagram 2-1. PROFILE Processing



Input from PROFILE

COMMAND BUFFER

length	offset	PROFILE	operand field

Parameter Control List (PCL)

PROM	'PROMPT'	'NOPROMPT'
MSG	'MSGID'	'NOMSGID'
INT	'INTERCOM'	'NOINTERCOM'
PAUS	'PAUSE'	'NOPAUSE'
CHARAC	'CHAR'	'NOCHAR'
LNE	'LINE'	'NOLINE'

NOTE:
The Parameter Control List (PCL) is shown conceptually only ! See the TMP Service Routines PLM for the actual diagram.

Parsing the Operands found in the Command Buffer.

1 Get dynamic storage.

2 Scan the Command Buffer and the PCL in order to validate the operands.

3 Build PDL for valid operands.

4 If 'CHAR' keyword is specified, branch to validity check routine in PROFILE.

If 'LINE' keyword is specified, branch to validity check routine in PROFILE.

5 Check return code from either or both validity check routines, if applicable.

6 Put a return code in register 15 for a normal return to PROFILE (IKJEFT82).

*If an error should be found, or a prompt required, issue the PUTLINE macro to send a message to the terminal.

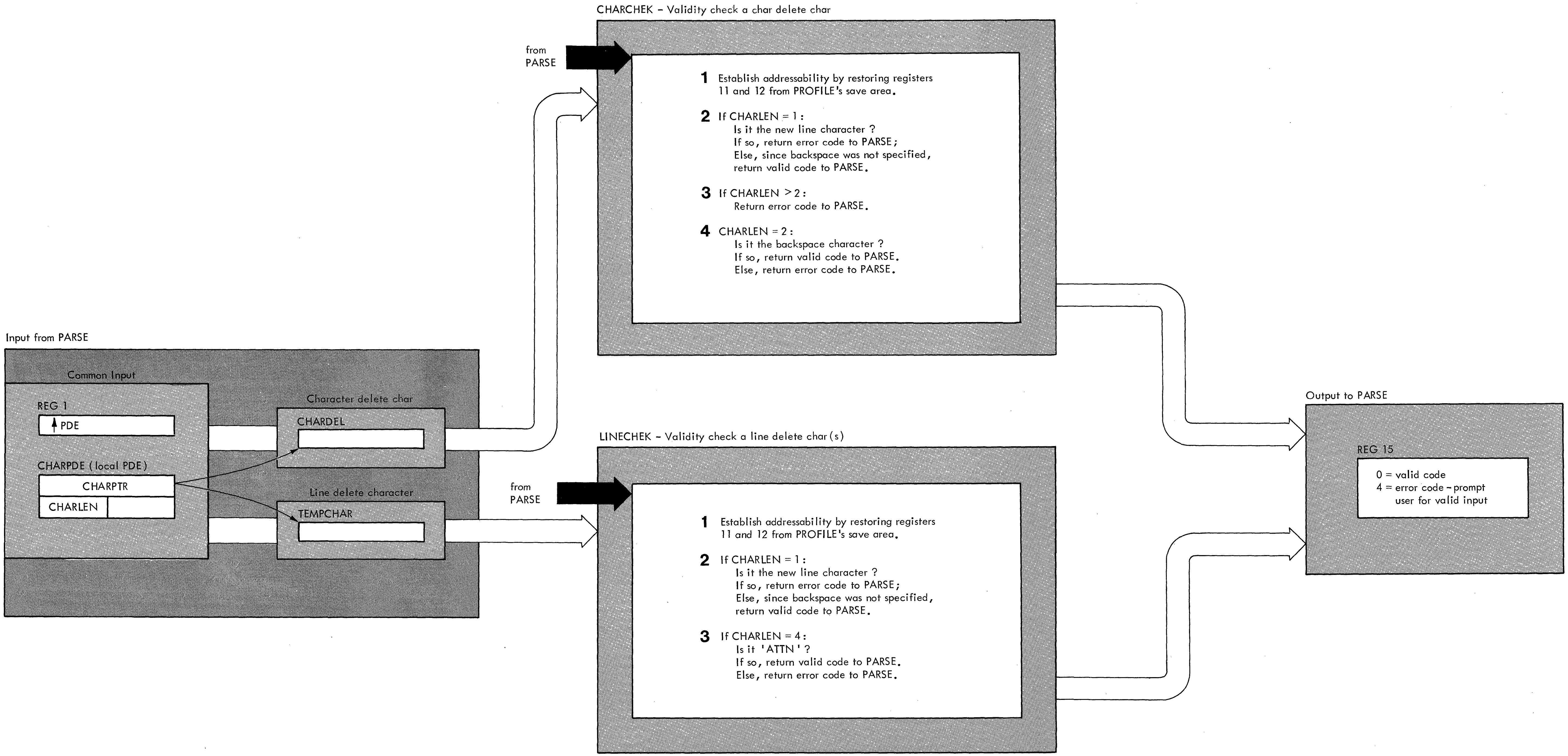
Output to Profile

STORPTR	
STORB	
PRMT	MSGID
INTERCOM	PAUSE
CHART	LINE
CHARID	
CARLEN	▨
LINEID	
LINLEN	▨

A This Input is shown on Diagram 2-3.

Description	Routine	Label	FC
1 Issue GETMAIN macro to acquire dynamic core so that module will be refreshable.	IKJPARS		AA
2 Scan the PCL and command buffer. A match constitutes a valid operand.			
3 Each mutually exclusive pair shown in the PCL (i.e. 'PROMPT' and 'NOPROMPT') has a corresponding two byte field (PRMT) in the PDL. If neither operand of an exclusive pair is found in the command buffer, the two byte status field in the PDL will contain '0'. If the positive operand of a pair is found (e.g. PAUSE), the status field will contain '1'. If the negative one (e.g. NOPAUSE) is found, the status field will contain '2'.			
4 If 'CHAR' along with one or two characters is found, PARSE executes a BALR instruction to branch to CHARCHEK (an in-line subroutine of PROFILE).	CHARCHEK		AB
If 'LINE' along with a character or character string is found, PARSE executes a BALR to branch to LINECHEK (another in-line validity checking routine).	LINECHEK		AB
5 Both CHARCHEK and LINECHEK use the same return code: 0 = VALID CONTROL CHARACTER(S). 4 = INVALID (tells PARSE to issue the prompt message followed by 'REENTER', and then wait for a response).			
6 Use register 15 to send a return code to PROFILE: 0 = SUCCESSFUL parse of OPERANDS NON 0 = UNSUCCESSFUL RETURN to PROFILE.			

Diagram 2-2. PARSE Processing When Used by PROFILE



CHARCHEK - Validity check a char delete char

1 Establish addressability by restoring registers 11 and 12 from PROFILE's save area.

2 If CHARLEN = 1:
Is it the new line character ?
If so, return error code to PARSE;
Else, since backspace was not specified, return valid code to PARSE.

3 If CHARLEN > 2:
Return error code to PARSE.

4 CHARLEN = 2:
Is it the backspace character ?
If so, return valid code to PARSE.
Else, return error code to PARSE.

LINECHEK - Validity check a line delete char(s)

1 Establish addressability by restoring registers 11 and 12 from PROFILE's save area.

2 If CHARLEN = 1:
Is it the new line character ?
If so, return error code to PARSE;
Else, since backspace was not specified, return valid code to PARSE.

3 If CHARLEN = 4:
Is it 'ATTN' ?
If so, return valid code to PARSE.
Else, return error code to PARSE.

Diagram 2-3. PROFILE's Validity Check Routines When Used by PARSE

Part 2: PROTECT Command Processor

2

Contents

Illustrations

CHARTS

DIAGRAMS

FIGURES

Section 1. Introduction

The PROTECT Command Processor is used to establish or change access information concerning the user's data sets. Passwords necessary for access are specifed. It may also be used to obtain security information about the user's data sets.

Functions

The PROTECT Command Processor performs the following functions:

- Obtains the PROTECT command string from the command buffer.
- Scans and syntax checks the command using IKJPARS.
- Processes data set information.
- Uses Dynamic Allocation Interface Routine (IKJDAIR) to unallocate the data set.
- Builds SVC parameter list for the requested option.
- Issues SVC 98 to perform requested operations.
- Issues appropriate messages.

Environment

The PROTECT Command Processor operates in a TSO user's region under the protection key assigned to that region. The processor's static data and instructions require about 5K of storage. Additional storage is needed for the TSO service routines used by this processor. For detailed information on storage for TSO and the Storage Estimates SRL, GC28-6551.

Physical Characteristics

The PROTECT Command Processor consists of one load module, IKJEHPRO. The main processing program, IKJEHPRO, handles all the major functions of the processor. The messages control section, IKJEHMSG, is in this program. IKJEHPCD and IKJEHPWP are used for the IKJPARS macros.

Operational Considerations

PROTECT communicates with a terminal through the PUTLINE service routine (IKJPUTL).

The processor uses the System/360 Operating System service routine, SVC 98, to maintain the PASSWORD Data Set (PWDS) and to update the data set control block (DSCB).

The PARSE service routine (IKJPARS) scans and syntax checks the command string. IKJPARS also prompts for missing or incorrect parameters. When a control password is required to authorize acces to an entry in the PWDS, IKJPARS prompts for this password.

Data set names, if not fully qualified, are fully qualified by the DEFAULT service routine, IKJEHDEF.

Input to the processor is the Command Processor Parameter List (CPPL). Upon receiving control, register 1 points to the CPPL.

The command buffer, addressed by the CPPLCBUF in the CPPL, contains the command string. The command string may have the following:

- The command name, PROTECT.
- The required positional operand, data set name.
- The optional keyword parameters (See Command Language Reference, GC28-6732).

Messages are printed at the terminal. These may be informational messages, diagnostic messages, or prompting messages.

Section 2. Method of Operation

This section discusses the major programming operations of the PROTECT Command Processor.

General

The PROTECT command is serviced by the PROTECT Command Processor and by the action of the PROTECT SVC, SVC 98.

The purpose of the PROTECT Command Processor is to set up for and issue SVC 98, and then to analyze the return code from the SVC 98 function and provide appropriate messages or information to the user.

The SVC 98 function maintains the PWDS and updates the DSCB.

The Protect Command Processor

In preparing to issue an SVC 98, the PROTECT Command Processor first scans the input for the PROTECT command provided through the TMP. Then, according to the type of user request (ADD, REPLACE, DELETE, or LIST), the information is formatted and placed in a parameter list. The parameter list may vary, depending on the function requested.

When the SVC 98 function returns control, it provides an appropriate return code that describes the action that was taken.

PROTECT Command Processor operation is illustrated in Diagram 2-1.

SVC 98 Operation

The SVC function first determines the type of parameter list passed by the command processor. Then, according to the request type, appropriate action is taken. The possible actions are illustrated in Diagram 2-2.

Note: The SVC 98 code is not actually part of the PROTECT Command Processor. As such, the cross-references into the implementation do not appear on Diagram 2-2. Refer to the publication IBM System/360 Operating System Direct Access Device Space Management (DADSM) PLM, Form GY28-6607, for more detailed information about SVC 98.

Introduction to the Diagrams

The remainder of this section consists of the method of operation diagrams of the PROTECT Command Processor.

The diagrams are designed to serve you in different ways:

- Initially, they quickly provide you with an understanding of the programming functions of the command processor.

- They show how the code supports the functions to be performed.

- They provide "pointers" to the actual implementations (by cross references to the flowcharts or program listings).

You can use them to become familiar with the system, or later, to find your way to specific implementation points in the program listings or other areas of the PLM.

Organization of Diagrams

Diagram 2-1 shows the overview of the command processor. Diagram 2-1 also acts as a directory to the other diagrams in this PLM.

The processing block lists the various functions performed to produce the final output.

The descriptions provided at the left of the diagram list the major implementation steps performed in support of the processing functions.

The routine, label and flowchart cross-referencing provided with the descriptions are "key" points designed to show you where to locate the implementations in the program listings or flowcharts.

Symbols

This PLM uses the following symbols:

➡ Heavy black arrow indicates begin reading the diagram here.

⇨ White arrow indicates data transfer from one area to another.

➔ Black arrow shows flow of logic or passing of control.

→ Thin black arrow indicates pointer to an item.

⇢ Dotted arrow indicates reference to an item.

⬟ Off-page connector leads to a related diagram.

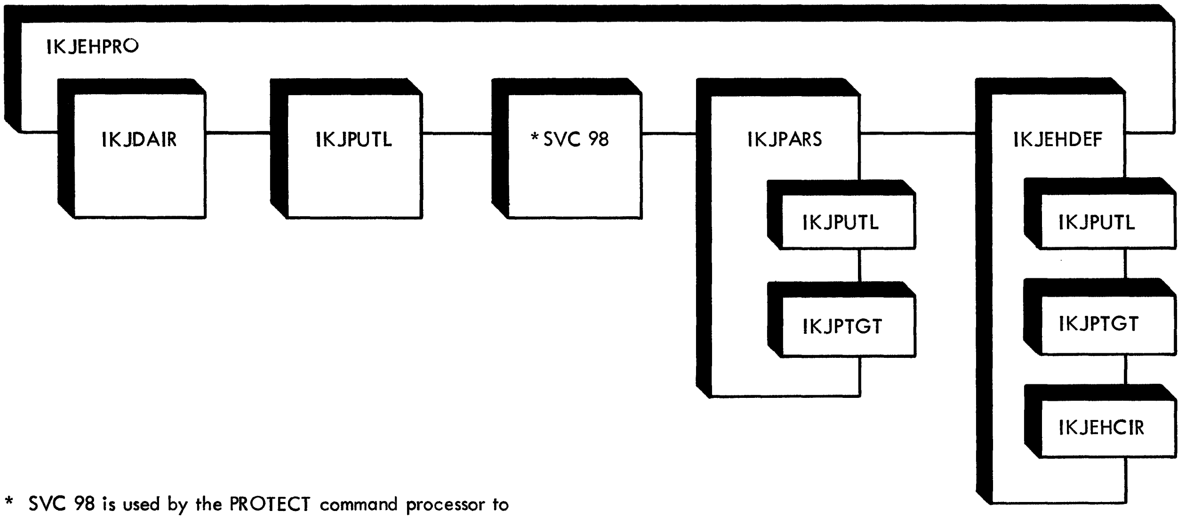
– – – – -× Getting or Freeing main storage.

➡□➡ Signifies passing of parameters.

Section 3. Program Organization

This section contains the PROTECT Command Processor routines. It illustrates how they relate to each other. Flowcharts are included following the discussion.

Hierarchy information is shown in Figure 3-1.



* SVC 98 is used by the PROTECT command processor to maintain the PWDS and update the DSCB. It is not documented in this PLM. See the Direct Access Device Space Management (DADSM) PLM, GY28-6607.

Figure 3-1. Control Module Hierarchy.

IKJEHPRO

Entry point	IKJEHPRO.
Operation	Syntax checks command. Prompts for missing or incorrect operands. Qualifies data set name if necessary. Unallocates data set. Issues SVC 98. Formats requested information when LIST is specified. Issues appropriate messages.
Data Areas used	CPPL, PPL, DAPL, DAPB18, IOPL, PDL,SVCPARMS, DFPL, DFPB, PSCB.
Routines called	LINK to IKJPARS LINK to IKJPUTL LINK to IKJDFLT (IKJEHDEF) Issue SVC 98 LINK to IKJDAIR
Exits	Normal return to caller.
Registers	Upon entry, register 1 points to the CPPL. Upon exit to IKJPARS, register 1 points to PPL. Upon exit to IKJPUTL, register 1 points to IOPL. Upon exit to IKJEHDEF, register 1 points to DFPL. Upon exit to TMP, register 15 has a return code. Upon exit to SVC 98, register 1 points to SVCPARMS. Upon exit to IKJDAIR, register 1 points to DAPL.

Cross reference	FC*	MO**
	AA	2-1
	AB	

FC* indicates flowcharts
MO** indicates method of operation diagram

IKJEHMSG

Entry point	IKJEHMSG, IKJEHSMG.
Operation	This control section contains the messages used by IKJEHPRO (first and second level) and two branch tables, one for each level of message.
Data Areas used	Messages (See Section 6).
Routines called	None.
Registers	N/A.
Exits	None.

Chart AA. IKJEHPRO. PROTECT Command Processing (1 of 2)

Cross Reference: Diagram 2-1.

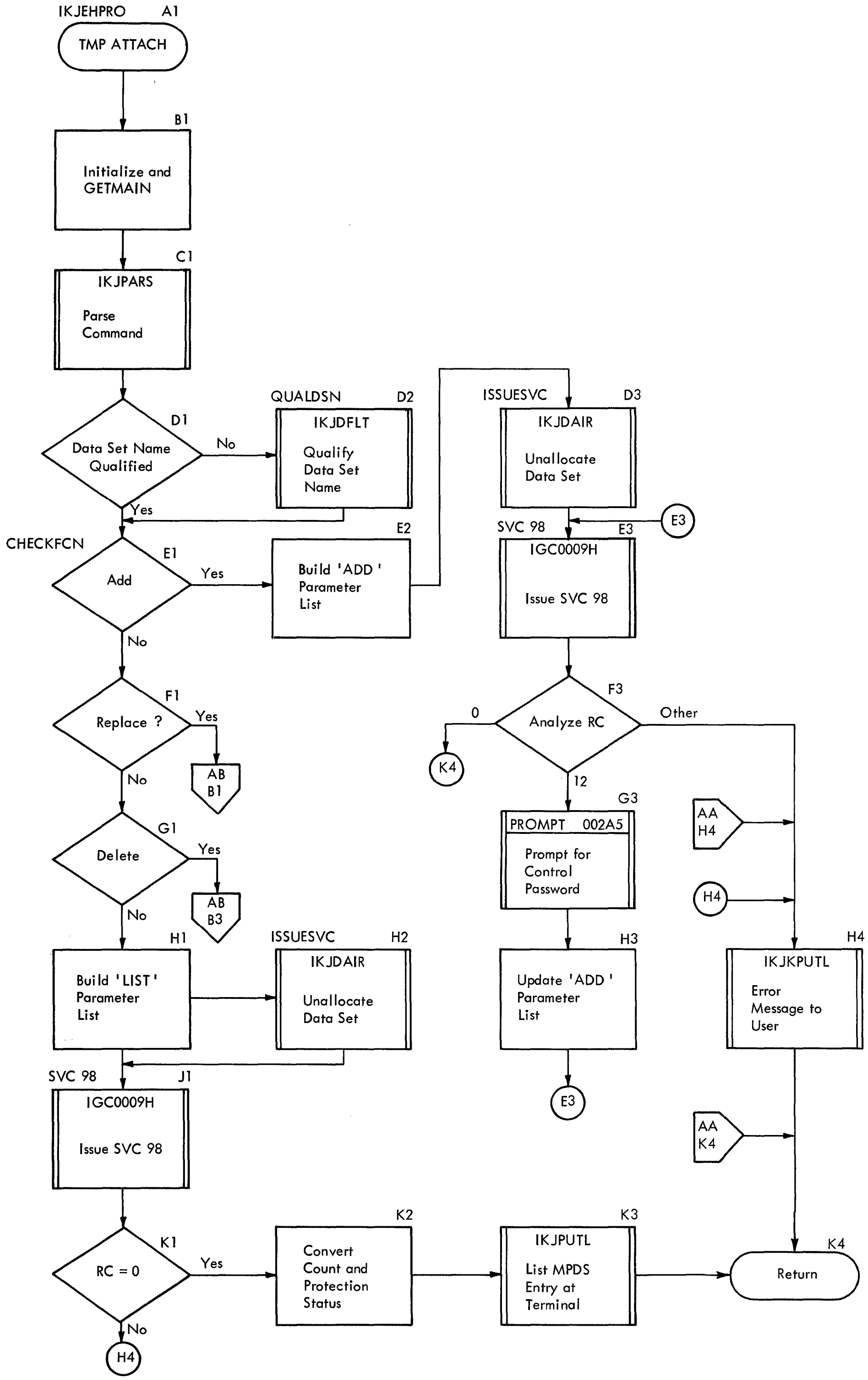
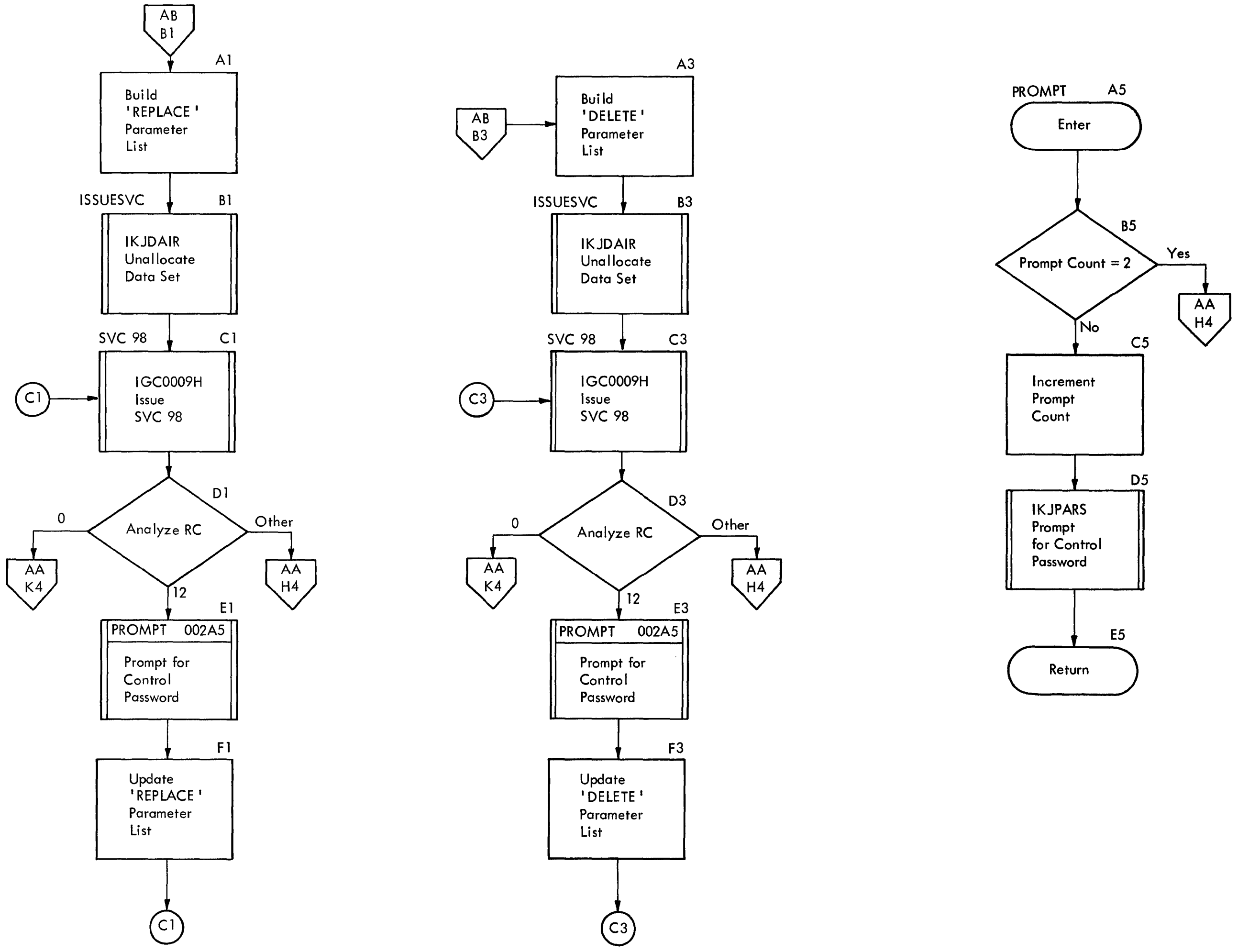


Chart AB. IKJEHPRO. PROTECT Command Processing (2 of 2)

Cross Reference: Diagram 2-1.



Section 4. Directory

This section contains the routine directory organized alphabetically by entry point or routine name. A data area directory is provided also and is arranged alphabetically.

Routine Directory

Entry point or Routine name	Load Module	Control Section	Cross-reference	
			MO	FC
IKJEHPCD	IKJEHPRO	IKJEHPCD	N/A	AA
IKJEHPRO	IKJEHPRO	IKJEHPRO	2-1	AA-AB
IKJEHPWP	IKJEHPRO	IKJEHPWP	N/A	AA
IKJEHSMG	IKJEHPRO	IKJEHSMG	N/A	AA

Data Area Directory

Data Area or Subfields	MO	FC
CPPL	2-1	AA
DAPB18	N/A	AA,AB
DAPL	N/A	AA
DFPB	N/A	AA
DFPL	N/A	AA
IKJPARMD	2-1	AA,AB
IKJPWPRM	N/A	AA,AB
IOPL	N/A	AA
PPL	N/A	AA
PSCB	N/A	AA
PWDS	N/A	AA
SVCPARMS	2-1	AA,AB

Section 5. Data Areas

This section contains the major data areas in the PROTECT Command Processor. Pertinent information, such as which routine constructed the data area, which routine uses the data area, and what the data area contains, is included.

CPPL—Command Processor Parameter List

Constructed by: TMP.

Used by: IKJEHPRO.

Contents: The CPPL is a list of pointers to tables, control blocks, and the command buffer. Upon entry it is pointed to by register 1.

Disp Dec (Hex)	Field	Size	Use
0 (0)	CPPLCBUF	4 bytes	Points to the command buffer.
4 (4)	CPPLUPT	4 bytes	Points to the User Profile Table (UPT).
8 (8)	CPPLPSCB	4 bytes	Points to the Protected Step Control Block (PSCB).
12 (C)	CPPLECT	4 bytes	Points to the Environment Control Table (ECT).

Cross reference FC MO

AA 2-1

DAPB18—A Parameter List Used When the Requested dsname or ddname is to be Unallocated

Constructed by: IKJEHPRO.

Used by: IKJDAIR.

Contents: Data set information found by adding the contents of register 13 and the displacement value of DAIR18 in the cross reference table.

Disp Dec(Hex)	Field	Subfield	Size	Use
0 (0)	DA18CD		2 bytes	IKJDAIR entry code X'0018".
2 (2)	DA18FLG		2 bytes	Functions to be performed when return code = 0.
		DA18FPE	1 byte	X'80' indicates function performed but an error is indicated by the return code.
4 (4)	DA18DARC		2 bytes	Dynamic allocation return code.
6 (6)	DA18CTRC		2 bytes	CATALOG routine return code.
8 (8)	DA18PDSN		4 bytes	Points to dsname to be searched in DSE.
12 (C)	DA18DDN		8 bytes	Ddname to be searched for in DSE.
20 (14)	DA18MNM		8 bytes	Member name.
28 (1C)	DA18CLS		2 bytes	SYSOUT class desired when unallocating a SYSOUT data set.

DAPB18 -- A Parameter List Used When the Requested Dsname or Ddname is to be Unallocated (Cont.)

Disp Dec(Hex)	Field	Subfield	Size	Use
30 (1E)	DA18DPS2	 DA18KEEP DA18DEL DA18CAT DA18UCAT	1 byte 1 byte 1 byte 1 byte 1 byte	disposition. X'08' change disp to KEEP. X'04' change disp to DELETE. X'02' change disp to CATALOG. X'01' change disp to UNCATALOG.
31 (1F)	DA18CTL		1 byte	Flags for special IKJDAIR processing.
		DA18UID		X'20' userid is to be prefixed to dsname.
32 (20)	DA18JBNM		8 bytes	Jobname.

Cross reference FC MO

AA N/A
AB

DAPL—Dynamic Allocation Parameter List

Constructed by: IKJEHPRO.

Used by: IKJDAIR.

Contents: DAPL is a parameter list passed to IKJDAIR. This is based on the address of DAIRPARM. DAIRPARM is found by adding the contents of register 13 to the displacement value of DAIRPARM in the cross reference table.

Disp Dec (Hex)	Field	Size	Use/Contents
0 (0)	DAPLUPT	4 bytes	Points to the UPT.
4 (4)	DAPLECT	4 bytes	Points to the ECT.
8 (8)	DAPLECB	4 bytes	Points to the Event Control Block (ECB).
12 (C)	DAPLPSCB	4 bytes	Points to the PSCB.
14 (10)	DAPLDAPL	4 bytes	Points to the IKJDAIR parameter block.

Cross reference	FC	MO
	AA	N/A

DFPB—IKJEHDEF Parameter Block

Constructed by: IKJEHPRO.

Used by: IKJEHDEF.

Contents: DFPB is a parameter block that controls operation of the IKJEHDEF service routine. It is pointed to by DFPLDFPB in the DFPL. It can also be found by adding the contents of register 13 the displacement value of DFLTPBLK in the cross reference table.

Disp Dec(Hex)	Field	Subfield	Size	Use/Contents
0 (0)	DFPBCODE		1 byte	IKJEHDEF entry code data set name.
		DFPB0C DFPB08 DFPB04 DFPB00		X'0C' entry "0C'. X'08' entry code '08'. X'04' entry code '04'. X'00' entry code '00'.
1 (1)	DFPBDSN		3 bytes	Points to data set name length and
4 (4)	DFPBCNTL		1 byte	IKJEHDEF control byte.
		DFPBUID DFPBRET DFPBADD DFPBMSG		Userid prefix indicator X'20'. Return added qualifier X'04'. Add specified qualifier X'02'. Issue message X'01'.
5 (5)	DFPBPSCB		3 bytes	Points to PSCB.
8 (8)	DFPBLORC		1 byte	LOCATE return code returned here.
9 (9)	DFPBQUAL		3 bytes	Points to IKJEHDEF qualifier.

Cross reference FC MO

AA N/A

DFPL—IKJEHDEF Service Routine Parameter List

Constructed by: IKJEHPRO.

Used by: IKJEHDEF.

Contents: The DFPL is a list of addresses passed to IKJEHDEF by the command processor. It is found by adding the contents of register 13 to the displacement value of DFLTPARM in the cross reference table.

Disp Dec(Hex)	Field	Size	Use/Contents
0 (0)	DFPLUPT	4 bytes	Points to the UPT.
4 (4)	DFPLECT	4 bytes	Points to the ECT.
8 (8)	DFPLECB	4 bytes	Points to the ECB.
12 (C)	DFPLDFPB	4 bytes	Points to IKJEHDEF parameter block (DFPB).

Cross reference	FC	MO
	AA	N/A

IKJPARMD—IKJPARS Descriptor List

Constructed by: IKJPARS.

Used by: IKJEHPRO.

Contents: IKJPARMD contains data set information returned by IKJPARS. It is pointed to by the contents of register 9 upon return from IKJPARS.

Disp Dec(Hex)	Field	Size	Use/Contents
0 (0)		8 bytes	Storage chains for IKJPARS.
8 (8)	PRODSN	4 bytes	Points to data set name.
12 (C)		2 bytes	Length of data set name.
14 (E)		1 byte	flags for data set information.
15 (F)		1 byte	Not used (reserved).
16 (10)		4 bytes	Points to member name.
20 (14)		2 bytes	Length of member name.
22 (16)		1 byte	Flags for member name information.
23 (17)		1 byte	Not used (reserved).
24 (18)		4 bytes	Points to password.
28 (1C)		2 bytes	Length of password.
30 (1E)		1 byte	Flags for password information.
31 (1F)		1 byte	Not used (reserved).
32 (20)	PROTYPE	2 bytes	Indicates function. 1 indicates 'ADD' specified or defaulted. 2 indicates 'REPLACE' specified. 3 indicates 'DELETE' specified. 4 indicates 'LIST' specified.

IKJPARMD -- IKJPARS Descriptor List (Cont.)

Disp Dec(Hex)	Field	Size	Use/Contents
34 (22)	RDTYPE	2 bytes	Indicates READ specification. 0 indicates no READ keyword specified. 1 indicates 'PWREAD' specified. 2 indicates 'NOPWREAD' specified.
36 (24)	WRTYPE	2 bytes	Indicates WRITE specification. 0 indicates no WRITE keyword specified. 1 indicates 'PWWRITE' specified. 2 indicates 'NOWRITE' specified.
38 (26)	DATATYPE	2 bytes	Indicates DATA specification. 0 indicates no DATA keyword specified. 1 indicates 'DATA' specified.
			IKJPARS provides an additional doubleword for each password or data field specified in the command. These fields are referenced using the labels ADDNEWPW, REPOLDPW, REPNEWPW, DELOLDPW, LSTOLDPW, STRDATA. The format of each doubleword is as follows:
		4 bytes 2 bytes 1 byte 1 byte	Points to the character string. Length of the character string. Flags. Not used (reserved).

Cross reference	FC	MO
	AA	2-1
	AB	

IKJPWPRM—IKJPARS Descriptor List

Constructed by: IKJPARS.

Used by: IKJEHPRO.

Contents: IKJPWRM contains information about the control password. It is pointed to by register 9 upon return from IKJPARS.

Disp Dec (Hex)	Field	Size	Use
0 (0)		8 bytes	Storage chains for IKJPARS.
8 (8)	CNTRLPW	4 bytes	Points to character string.
12 (C)		2 bytes	Length of character string.
14 (E)		1 byte	Flags.
15 (F)		1 byte	Unused (reserved).

Cross reference	FC	MO
	AA AB	N/A

IOPL—Input/Output Parameter List Passed to IKJPUTL

Constructed by: IKJEHPRO.

Used by: IKJPUTL.

Contents: IOPL contains a list of pointers to tables and blocks used by the I/O service routines. It is found by adding the contents of register 13 and the displacement value of IOPARM in the cross reference table.

Disp Dec(Hex)	Field	Size	Use/Contents
0 (0)	IOPLUPT	4 bytes	Points to UPT.
4 (4)	IOPLECT	4 bytes	Points to ECT.
8 (8)	IOPLECB	4 bytes	Points to the ECB.
12 (C)	IOPLIOPB	4 bytes	Points to the I/O service routine parameter block.

Cross reference	FC	MO
	AA	N/A

PPL—IKJPARS Parameter List

Constructed by: IKJEHPRO.

Used by: IKJPARS.

Contents: The PPL is a list of addresses from the command processor. It is found by adding the contents of register 13 and the displacement value of PARSPARM in the cross reference table.

Disp Dec(Hex)	Field	Size	Use/Contents
0 (0)	PPLUPT	4 bytes	Points to the UPT.
4 (4)	PPLECT	4 bytes	Points to the ECT.
8 (8)	PPLECB	4 bytes	Points to the ECB.
12 (C)	PPLPCL	4 bytes	Points to the IKJPARS control list.
16 (10)	PPLANS	4 bytes	Points to the answer place.
20 (14)	PPLCBUF	4 bytes	Points to the command buffer.
24 (18)	PPLUWA	4 bytes	Points to the user work area for validity check routines.

Cross reference	FC	MO
	AA	N/A

PSCB—Protected Step Control Block

Constructed by: LOGON.

Used by: IKJEHPRO.

Contents: The PSCB contains user attributes and accounting data on a userid basis. It is pointed to by CPPLPSCB in the CPPL.

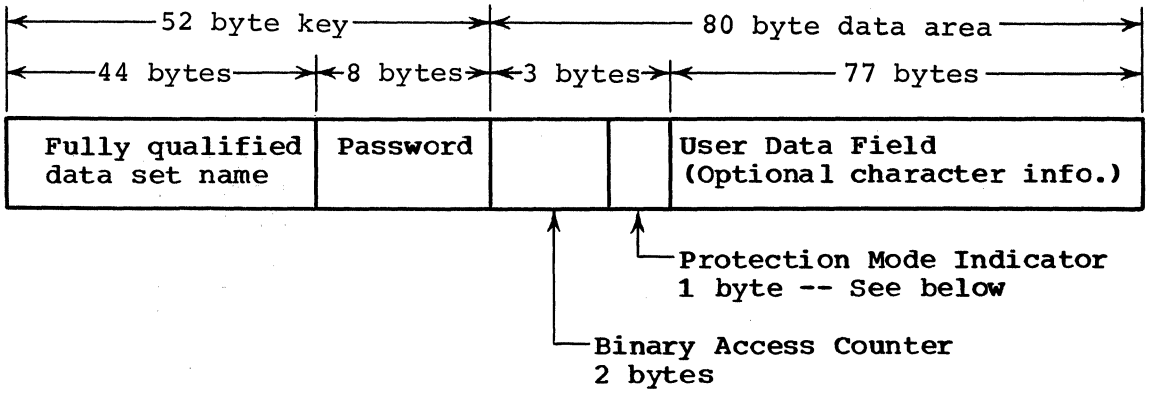
Disp Dec (Hex)	Field	Subfield	Size	Use
0 (0)	PSCBUSER		7 bytes	Userid.
7 (7)	PSCBUSRL		1 byte	Length of userid.
8 (8)	PSCBGPNM		8 bytes	Used by dynamic allocation when unit name is not specified.
16 (10)	PSCBATR1		1 byte	Bit string of user attributes.
		PSCBCTRL PSCBACCT PSCBJCL		X'80' OPERATOR command user. X'40' ACCOUNT command user. X'20' SUBMIT command user. The remaining bits, 3-15, are reserved for IBM use.
17 (11)	Unnamed		1 byte	Not used (reserved).
18 (12)	PSCBATR2		1 byte	Reserved for installation use.
19 (13)	Unnamed		1 byte	Not used (reserved).
20 (14)	PSCBCPU		4 bytes	Cumulative CPU time used during session.
24 (18)	PSCBSWP		4 bytes	Cumulative time resident in the region.
28 (1C)	PSCBLTIM		4 bytes	Actual Logon time of day.

PSCB -- Protected Step Control Block (Cont.)

Disp Dec (Hex)	Field	Subfield	Size	Use/Contents
32 (20)	PSCBTCPU		4 bytes	Total CPU time used in this accounting period.
36 (24)	PSCBTSWP		4 bytes	Total time user is resident in the region during this accounting period.
40 (28)	PSCBTCON		4 bytes	Total time user is connected during this accounting period.
44 (2C)	PSCBTCO1		4 bytes	
48 (30)	PSCBLGB		4 bytes	Points to RELOGON buffer.
52 (34)	PSCBUPT		4 bytes	Points to user profile table.
56 (38)	PSCBUPTL		2 bytes	Length of UPT.
58 (34)	Unnamed		6 bytes	Reserved for IBM.
64 (3C)	PSCBU		8 bytes	Reserved for installation use.

Cross reference	FC	MO
	AA	N/A

PWDS—PASSWORD Data Set Record Format


52 byte key
80 byte data area
44 bytes
8 bytes
3 bytes
77 bytes
Fully qualified data set name
Password
User Data Field (Optional character info.)
Protection Mode Indicator 1 byte -- See below
Binary Access Counter 2 bytes

Protection Mode Indicator:

bit 00 On indicates control password.
Off indicates secondary password.

01
02
03
04 Not used (reserved).
05
06

07 On indicates READ/WRITE.
Off indicates READ ONLY.

Cross reference	FC	MO
	AA	N/A

SVCPARMS—SVC Parameter List

Constructed by: IKJEHPRO.

Used by: SVC 98.

Contents: SVCPARMS is a list of information concerning function that is passed to SVC 98. It is found by adding the contents of register 13 and the displacement value of SVCPARMS in the cross reference table.

Disp Dec(Hex)	Field	Size	Use
0 (0)	SVCOPT	1 byte	SVC 98 entry code.
1 (1)	Unnamed	3 bytes	Not used (reserved).
4 (4)	SVCDSL	1 byte	SVC 98 data set name length.
5 (5)	SVCDSN	3 bytes	Points to the data set name.
8 (8)	SVCCURPW	4 bytes	Points to current password.
12 (C)	SVCCONPW	4 bytes	Points to control password.
16 (10)	SVCVOLS	1 byte	Number of volumes.
17 (11)	SVCVLIST	3 bytes	Points to volume list.
20 (14)	SVCPCODE	1 byte	Protection code.
21 (15)	SVCNEWPW	3 bytes	Points to new password.
24 (18)	SVCSTLTH	1 byte	String length.
25 (19)	SVCSTRNG	3 bytes	Points to string.

Cross reference	FC	MO
	AA	2-1
	AB	

Section 6. Diagnostic Aids

This section contains a register summary and the messages dealing with the PROTECT command processor.

Register Summary

Register	Use
0	Used as a work register.
1	Used as a parameter register and as a work register.
2	Used as a work register.
3	Used in 'EXEC MVC' as a pointer to 'TO AREA'. Also used as a work register.
4	Used in 'EXEC MVC' as a pointer to 'FROM AREA'. Also used as a work register.
5	Used as a work register.
6	Used as a work register.
7	Not used.
8	Used as a work register.
9	Used as base register for IKJPARS dsect.
10	Used as a work register.
11	Used as a work register.
12	Used as base register for this program.
13	Used as save area register, and as base register for GETMAIN area.
14	Used as the link register.
15	Used as the branch and return code register.

Messages

Issued by	Message ID	Message Content
PROTECT PROTECT PROTECT PROTECT	IKJ58101I *IKJ58101I *IKJ58101I *IKJ58101I	UNABLE TO PROTECT DATA SET dsname+ PASSWORD DATA SET NOT FOUND PASSWORD DATA SET FULL I/O ERROR IN PASSWORD DATA SET
PROTECT	IKJ58102I	NOT ENOUGH MAIN STORAGE TO EXECUTE COMMAND
PROTECT	IKJ58103I	DATA SET dsname NOT IN CATALOG
PROTECT PROTECT PROTECT PROTECT PROTECT PROTECT PROTECT	IKJ58104I *IKJ58104I *IKJ58104I *IKJ58104I *IKJ58104I *IKJ58104I *IKJ58104I	UNABLE TO MODIFY PROTECTION FLAGS OF DATA SET dsname+ DATA SET NOT IN CATALOG DATA SET NOT ON VOLUME REQUIRED VOLUME NOT MOUNTED TAPE DATA SET CANNOT BE PROTECTED BY PROTECT I/O ERROR WHILE UPDATING SECURITY FLAGS DATA SET IN USE
PROTECT	IKJ58105I	INCORRECT PASSWORD ENTERED TWICE
PROTECT	IKJ58106I	PASSWORD password ALREADY EXISTS
PROTECT	IKJ58107I	PASSWORD password DOES NOT EXIST
PROTECT	IKJ58108I	PROTECTION IS FOR DATA SETS, NOT MEMBERS
PROTECT	IKJ58109I	USER DATA FIELD BLANK
PROTECT	IKJ58110I	INCORRECT CONTROL PASSWORD password
PROTECT	IKJ58111I	INVALID DATA SET NAME, 'dsname' EXCEEDS 44 CHARACTERS
PROTECT PROTECT PROTECT PROTECT	IKJ58112I *IKJ58112I *IKJ58112I *IKJ58112I	COMMAND SYSTEM ERROR+ PARSE ERROR CODE xxxx DEFAULT ERROR CODE xxxx SVC 98 RETURN CODE xxxx
IKJPARS IKJPARS IKJPARS IKJPARS IKJPARS	PROMPT PROMPT PROMPT PROMPT PROMPT	ENTER DATA SET NAME- ENTER NEW PASSWORD- ENTER CURRENT PASSWORD- ENTER DATA- ENTER CONTROL PASSWORD-

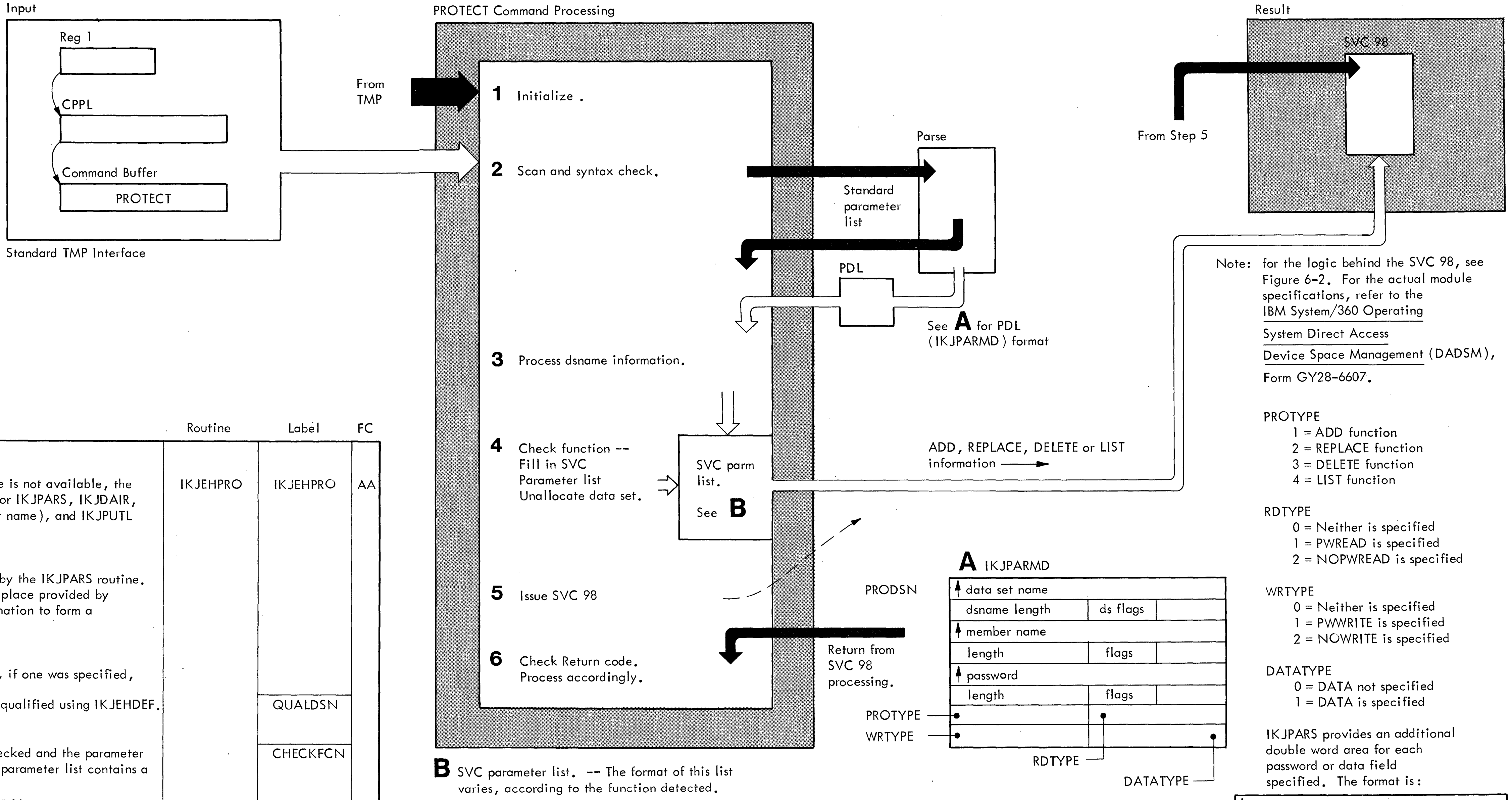
Messages (Cont.)

Issued by	Message ID	Message Content
IKJPARS	HELP	ENTER NAME OF DATA SET TO BE PROTECTED-
IKJPARS	HELP	ENTER PASSWORD OF ENTRY TO BE ADDED-
IKJPARS	HELP	ENTER PASSWORD OF ENTRY TO BE REPLACED-
IKJPARS	HELP	ENTER PASSWORD OF ENTRY TO BE DELETED-
IKJPARS	HELP	ENTER PASSWORD OF ENTRY TO BE LISTED-
IKJPARS	HELP	ENTER UP TO 77 BYTES OF CHARACTER INFORMATION IN QUOTES-
IKJPARS	HELP	ENTER PASSWORD OF CONTROL (FIRST) ENTRY FOR THIS DATA SET-
SVC 98	IEC021I	NO SPACE IN THE PASSWORD DATA SET (Issued to operator console)

* indicates second level message for documentation only.

Index

Indexes to program logic manuals are consolidated in the publication IBM System/360 Operating System: Program Logic Manual Master Index, Order No. GY28-6717. For additional information about any subject listed below, refer to other publications listed for the same subject in the Master Index.


Input
Reg 1
CPPL
Command Buffer
PROTECT
Standard TMP Interface
PROTECT Command Processing
From TMP
1 Initialize .
2 Scan and syntax check.
3 Process dsname information.
4 Check function -- Fill in SVC Parameter list Unallocate data set.
SVC parm list.
See B
5 Issue SVC 98
6 Check Return code. Process accordingly.
Parse
Standard parameter list
PDL
See A for PDL (IKJPARMD) format
ADD, REPLACE, DELETE or LIST information →
Return from SVC 98 processing.
Result
SVC 98
From Step 5

Note: for the logic behind the SVC 98, see Figure 6-2. For the actual module specifications, refer to the IBM System/360 Operating System Direct Access Device Space Management (DADSM), Form GY28-6607.

A IKJPARMD

PRODSN	↑ data set name		
	dsname length	ds flags	
	↑ member name		
	length	flags	
	↑ password		
	length	flags	
PROTYPE	●	● RDTYPE	
WRTYPE	●		● DATATYPE

PROTYPE
1 = ADD function
2 = REPLACE function
3 = DELETE function
4 = LIST function

RDTYPE
0 = Neither is specified
1 = PWREAD is specified
2 = NOPWREAD is specified

WRTYPE
0 = Neither is specified
1 = PWWRITE is specified
2 = NOWRITE is specified

DATATYPE
0 = DATA not specified
1 = DATA is specified

IKJPARS provides an additional double word area for each password or data field specified. The format is:

↑ password string		
string length	flags	

Applicable names are:

ADDNEWPW
REPOLDPW
REPNEWPW
DELOLDPW
LSTOLDPW
STRDATA

Description

Description	Routine	Label	FC
1 A GETMAIN is issued to request storage. If sufficient storage is not available, the program is abnormally terminated. Then general parameters for IKJPARS, IKJDAIR, IKJEHDEF (used if it is necessary to fully qualify the data set name), and IKJPUTL are initialized.	IKJEHPRO	IKJEHPRO	AA
2 Then the command is scanned and checked for correct syntax by the IKJPARS routine. IKJPARS places the address of the PDL (see A) in the answer place provided by IKJEHPRO. The PROTECT command processor uses this information to form a parameter list for the SVC 98 (PROTECT SVC) modules.			
3 After checking the IKJPARS return code the control password, if one was specified, is moved to the buffer. Then, if the data set name was not fully qualified, it is fully qualified using IKJEHDEF.		QUALDSN	
4 Next the function to be performed, found in PROTYPE, is checked and the parameter list for SVC 98 is filled in accordingly. The first byte of the parameter list contains a hexadecimal value indicating the function, as follows: X'01' ADD an entry to the password data set (PWDS). X'02' REPLACE an entry in the password data set. X'03' DELETE an entry from the PWDS. X'04' LIST protection, security counter, and optional data information of a protected data set. (The last 80 bytes of the PWDS entry for this data set password is placed in the 80 byte buffer pointed to by the SVC parameter list.)		CHECKFCN	
Unallocates the data set.		ISSUESVC	AA
5 Issue SVC 98 (see Diagram 2-2.)		SVC98	AB
	IGC0009H module		
6 Upon return from SVC 98, control is passed to the appropriate message processing and/or clean up locations, according to the return code provided by SVC 98.	IKJEHPRO	SVCRCTAB	

B SVC parameter list. -- The format of this list varies, according to the function detected.

X'01'

0	01	00 00 00
4	dslength	↑ dsname
8	Prot code	↑ New password
12	00	↑ Control password
16	String length	↑ String
20	Num of Vols	↑ Volume list
	1 byte	3 bytes

X'02'

0	02	00 00 00
4	dslength	↑ dsname
8	00	↑ Current password
12	Prot code	↑ New password
16	00	↑ Control password
20	String length	↑ String
24	Num of Vols	↑ Volume list

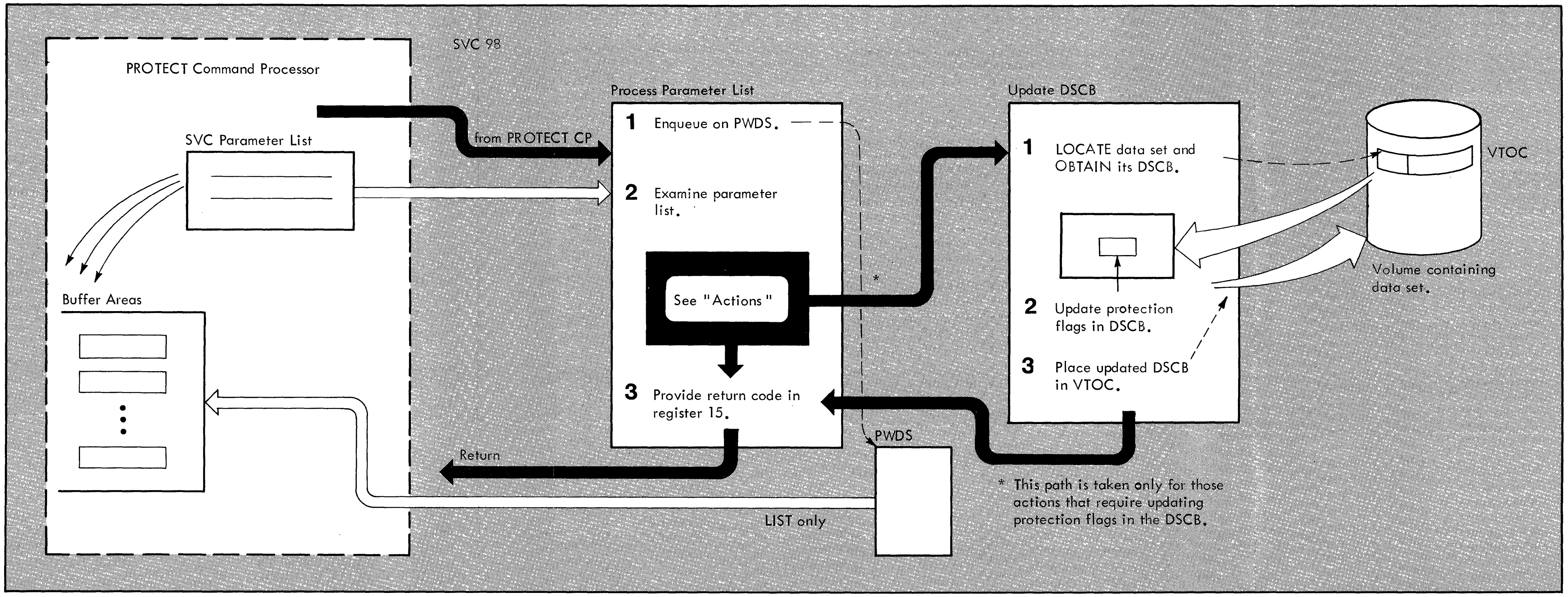
X'03'

0	03	00 00 00
4	dslength	↑ dsname
8	00	↑ Current password
12	00	↑ Control password
16	Num of Vols	↑ Volume list

X'04'

0	04	↑ Buffer (80 byte)
4	dslength	↑ dsname
8	00	↑ Current password

Diagram 2-1. PROTECT Command Processing


SVC 98
PROTECT Command Processor
SVC Parameter List
Buffer Areas
from PROTECT CP
Process Parameter List
1 Enqueue on PWDS.
2 Examine parameter list.
See "Actions"
3 Provide return code in register 15.
Return
*
PWDS
LIST only
Update DSCB
1 LOCATE data set and OBTAIN its DSCB.
2 Update protection flags in DSCB.
3 Place updated DSCB in VTOC.
VTOC
Volume containing data set.
* This path is taken only for those actions that require updating protection flags in the DSCB.

ACTIONS

The SVC 98 modules compare specified password information with password data set (PWDS) entries, maintain the PWDS, provide PWDS information to the PROTECT Command processor (for LIST only), and/or update the protection flags in the DSCB of a data set whose protection features have been modified.

Processing depends on the function requested by the user.

LIST Data Set Password Entry

- Validity Check buffer address
- Search PWDS for the password
- Place last 80 bytes of PWDS record into buffer area.

DELETE Data Set Password Entry

- Search PWDS for entry.
  - If entry is a control entry, delete all entries from the PWDS for this data set. Then dequeue the PWDS and update the DSCB protection flags.
  - If entry is not a control entry, search the PWDS for the control entry. Then, if the control password is correctly specified, delete the applicable entry and dequeue the PWDS.
  - If the entry cannot be found, or if the control password was specified incorrectly, a return code is generated.

ADD a Data Set Password Entry

- Search PWDS for entry. If one is found (or if the control password is specified incorrectly or the PWDS is full), an applicable return code is set. Otherwise :
  - If no corresponding entry is found, the specified password is the first for this data set, and space is available in the PWDS,
    - The PWDS is dequeued.
    - The DSCB protection flags are updated.
  - If no corresponding entry is found, but this password is not the first specified for the data set,
    - The PWDS is searched for the control entry.
    - If the control password is specified correctly and space is available in the PWDS, the requested addition is made. Otherwise an applicable return code is set.

REPLACE Data Set Password Entry

- Search PWDS for entry.
  - If the entry found is not the control entry, the PWDS is searched for the control entry. Then, if the control password was specified correctly, the entry to be replaced is replaced with the new password information.
  - If the entry is the control entry it is replaced. Then, if the protection type was changed, the PWDS is dequeued and the DSCB protection flags are updated.
  - If no corresponding entry is found in the PWDS, or if the control password is specified incorrectly, an applicable return code is set.

Diagram 2-2. PROTECT (SVC 98) Processing

Part 3: RENAME Command Processor

3

Contents

Illustrations

Charts

Diagrams

Figures

Section 1. Introduction

RENAME is a TSO Command Processor designed to:

- Change the name of a sequential data set residing on a direct access device.
- Change the name of a partitioned data set.
- Change the name of a member of a partitioned data set.
- Create an alias for a member of a partitioned data set.

Functions

RENAME communicates with a terminal through the PUTLINE service routine (IKJPUTL). Access to the VTOCs, PDS directory entries and the system catalog are all accomplished through System/360 Operating System (S/360 OS) routines -- OPEN, BLDL, STOW, CLOSE, CATALOG, LOCATE, INDEX, and RENAME.

The Dynamic Allocation Interface Routine (IKJDAIR) allocates data sets. The PARSE Routine (IKJPARS) scans and syntax checks the command string. The Catalog Information Routine (IKJEHCIR) searches the catalog for index levels and simple data names.

The RENAME Command Processor performs the following functions:

- Obtains the RENAME command string from the command buffer.
- Checks for a valid asterisk qualifier in the data set name. Checks for the same index level on both data set names.
- Builds new data set names.
- Qualifies dsname if necessary.
- Allocates data set (using OS enqueuing facility).
- If RENAME is for a member of a partitioned data set:

  Verifies, opens, and closes data set.

  Creates alias (if specified) or replaces member name.

- If RENAME is for a sequential data set:

  Renames data set.

  Catalogs new data set and uncatalogs old.

  Builds new index levels as necessary.

  Deletes old index levels as necessary.

  Renaming process repeats for next data set (only applicable if * processing).

Environment

The RENAME Command Processor operates in a TSO user's region under the protection key assigned to that region. The processor's data and instructions require about 6K bytes of storage. Additional storage is used for the service routines. Definitive figures for using TSO and the RENAME Command Processor can be found in the _Storage Estimates SRL_, GC28-6551.

Physical Characteristics

The processor consists of one load module containing the main program control sections (IKJEHREN and IKJEHRN2) and the messages control section (IKJEHMSG). The IKJEHRN2 control section handles all the major functions of the processor and returns control at the end of processing to the Terminal Monitor Program (TMP). IKJEHPDL contains the IKJPARS macros.

Operational Considerations

Input is the standard Command Processor Parameter List (CPPL). The TMP passes the address of the CPPL by register 1 to the command processor. The command buffer, pointed to by the CPPL, contains the RENAME command and its optional operands.

The RENAME command consists of the following:

- The command name RENAME.
- Positional operands specifying the old data set name and the new data set name. (See the Command Language Reference,GC28-6732).
- The optional operand, ALIAS, indicating the member name specified is to become an alias for the member identified in the old data set name.

Messages are printed at the terminal. These include diagnostic messages, informational messages, and prompting messages.

Section 2. Method of Operation

This section describes the logic of the RENAME Command Processor. It emphasizes the flow of data and control information through buffers and tables, and contains detailed functional descriptions of the RENAME operation.

Program logic information for the TMP and the TSO service routines used by the RENAME Command Processor is contained in the IBM System/360 Operating System Time Sharing Option Terminal Monitor Program and Service Routines Program Logic Manual, GY28-6770.

RENAME Command Processing Summary

RENAME changes the name of a data set, or a partitioned data set (PDS) member. It optionally creates an alias for an existing PDS member.

RENAME command processing begins when control is received from the Terminal Monitor Program (TMP). The IKJPARS parameter list is initialized and control is given to IKJPARS to check the syntax of the RENAME command. Upon successful completion, control returns. The data set name is scanned for valid asterisk qualifiers. If valid qualifiers are found, IKJEHCIR gets the data set names or the index names, as requested. Data sets are allocated and unallocated using IKJDAIR. Complete data set names are built. The data set or member is renamed.

When working with members of a PDS, the name change or alias creation is accomplished by updating the directory of the specified PDS.

For non-member name changes, in addition to modifying the VTOC, the RENAME Command Processor removes the old data set reference from the system catalog and inserts the new data set reference.

Data set names are fully qualified when necessary, using the TSO Default service routine (IKJEHDEF).

Introduction to the Diagrams

The remainder of this section consists of the method of operation diagrams for the RENAME Command Processor.

The diagrams are designed to serve you in different ways:

- Initially, they quickly provide you with an understanding of the programming functions of the command processor.
- They show how the code supports the functions to be performed.
- They provide "pointers" to the actual implementations (by cross references to the flowcharts or program listings).

You can use them to become familiar with the system, or later, to find your way to specific implementation points in the program listings or other areas of the PLM.

Organization of the Diagrams

Diagram 2-1 shows the functions of the controlling module. The other diagram included in this PLM is diagram 2-2. It shows specific data areas.

The processing block lists the functions performed in producing final output.

The descriptions provided at the left of the diagram list the major implementation steps performed in support of the processing functions.

The routine, label and flowchart cross-referencing provided with the descriptions are "key" points designed to show you where to locate the implementations in the program listings or flowcharts.

Symbols

This PLM uses the following symbols:

➡ Heavy black arrow indicates begin reading the diagram here.

⇨ White arrow indicates data transfer from one area to another.

➔ Black arrow shows flow of logic or passing of control.

⟶ Thin black arrow indicates pointer to an item.

⇢ Dotted arrow indicates reference to an item.

⬠ Off-page connector leads to a related diagram.

- - - -× Getting or Freeing main storage.

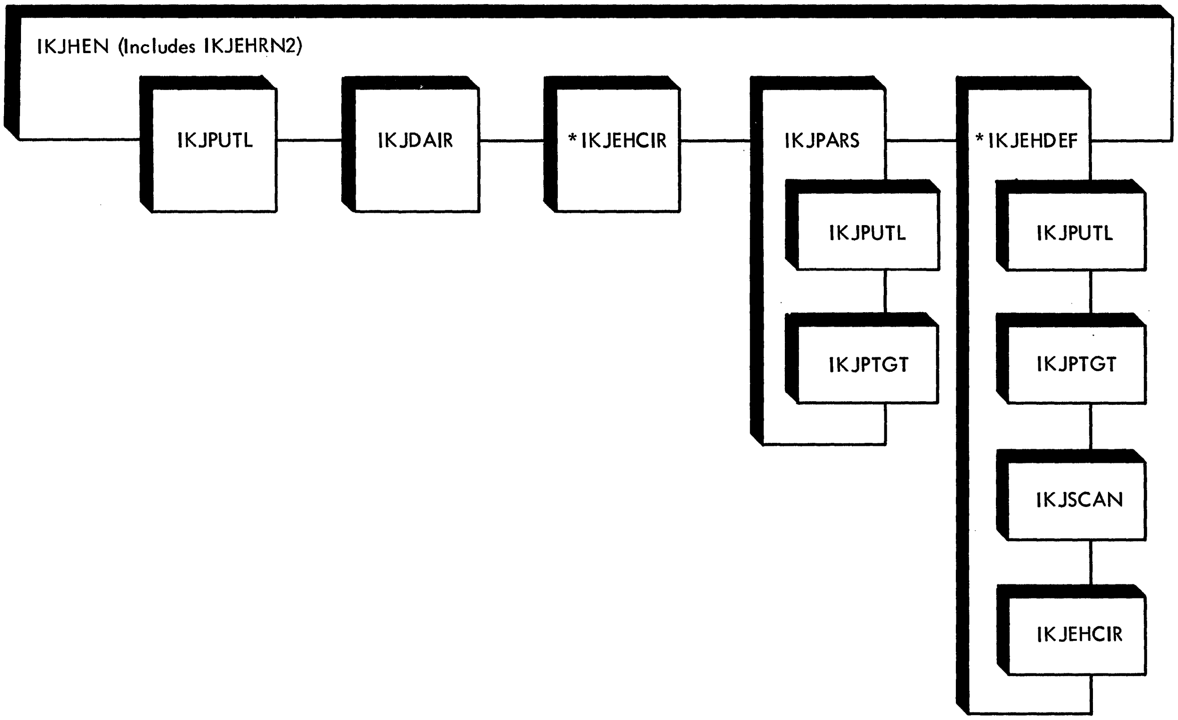
➡□➡ Signifies passing of parameters.

Section 3. Program Organization

This section describes the RENAME Command Processor routines and how they relate to each other. Flowcharts are included at the end of the discussion.

Figure 3-1 shows the control module hierarchy.

Hierarchy



*This external service routine is common to several command processors dealing with data set structures residing on a direct access device. It is discussed briefly in this PLM. A more thorough presentation can be found in the _Terminal Monitor Program and Service Routines Program Logic Manual_, GY28-6770.

Figure 3-1. Control Module Hierarchy

IKJEHCIR—Catalog Information Routine

Entry point	IKJEHCIR.
Operation	Provides an interface between processing programs and the system catalog. Returns a list of various entries identified by a one byte code preceding the information. Format of the entry varies depending on the option requested. X'01' returns all DS names at a particular index. X'02' returns index names. Two other request codes are used by other routines.
Data areas used by IKJEHCIR	WRK1AREA.
Routines called	NONE.
Registers	Upon entry, register 1 points to CIRPARM. Standard linkage conventions. Upon exit, register 15 has a return code.
Exits	Normal return to caller.

Cross reference	FC*	MO**
	AB	2-1,2-2

Note: FC* indicates Flowchart
MO**indicates Method of
Operation Diagram.

IKJEHDEF—Default Service Routine

Entry point	IKJDFLT.
Operation	Prefixes the user ID. Gets list of data set names from IKJEHCIR. Prompts users for choice of names from list. Checks users choice for validity. Adds name to make data set fully qualified. Returns fully qualified name to caller.
Data areas used by IKJDFLT	WRKAREA1.
Routines called	LINK to IKJPUTL LOAD and CALL to IKJEHCIR LINK to IKJPTGT LINK to IKJSCAN.
Registers	Upon entry, register 1 points to DFPL. Standard linkage conventions.
Exits	Normal return to caller.

Cross reference	FC	MO
	AA	2-1

IKJEHMSG— Message Control Section

Entry points	IKJEHMSG, IKJEHSMG.
Operation	Contains first and second level messages for the RENAME Command Processor.
Data Areas used	Table of messages (See Section 6).
Routines called	NONE.
Exits	Normal return to caller.
Registers	Register usage remains the same as in the calling routine.

Cross reference	FC	MO
	AA AC	2-1

IKJEHREN— Initialization Module

Entry Point	IKJEHREN.
Operation	Sets up initialization for processing.
Data Areas used	N/A.
Routines called	IKJEHRN2.
Exits	None.
Registers	Upon entry, register 1 points to the CPPL. Standard linkage conventions. Upon exit to the TMP, register 15 has a return code.

IKJEHRN2—Control Module

Entry point	IKJEHRN2.
Operation	Checks for valid asterisk. Determines level of asterisk. Qualifies data set names. Sets options for IKJEHCIR. Gets data set names. (Additional calls may be necessary to obtain all data set names under an index structure. If so, a TTR and a volume serial number are returned for use in the next call.) Builds data set name using input and IKJEHCIR returned information. Creates alias or replaces member name of a PDS. Renames data set, catalogs new and uncatalogs old data set.
Data areas used by IKJEHRN2	CIRPARM, DAIRPARM (DAPL), DAIR08 (DAPB08), DAIR18 (DAPB18), WRK1AREA, DFLTPBLK (DFPB), PARSPARM (PPL), DFLTPARM (DFPL), CPPL, IOPL.
Routines called	IKJPUTL, IKJEHCIR, IKJPARS, IKJDAIR, IKJEHDEF.
Exits	RETURN to TMP LINK to IKJPARS LINK to IKJPUTL LOAD and CALL to IKJDAIR LOAD and CALL to IKJEHCIR LINK to IKJDFLT (IKJEHDEF)
Registers	Upon entry, register 1 points to the standard (CPPL) parameter list. Standard linkage conventions. Upon exit to the TMP, register 15 has a return code.

Cross reference	FC	MO
	AA	2-1
	AB	2-2
	AC	

Chart AA. IKJEHREN. RENAME Command Processing (1 of 3)

Cross Reference : M.O. Diagram 2-1

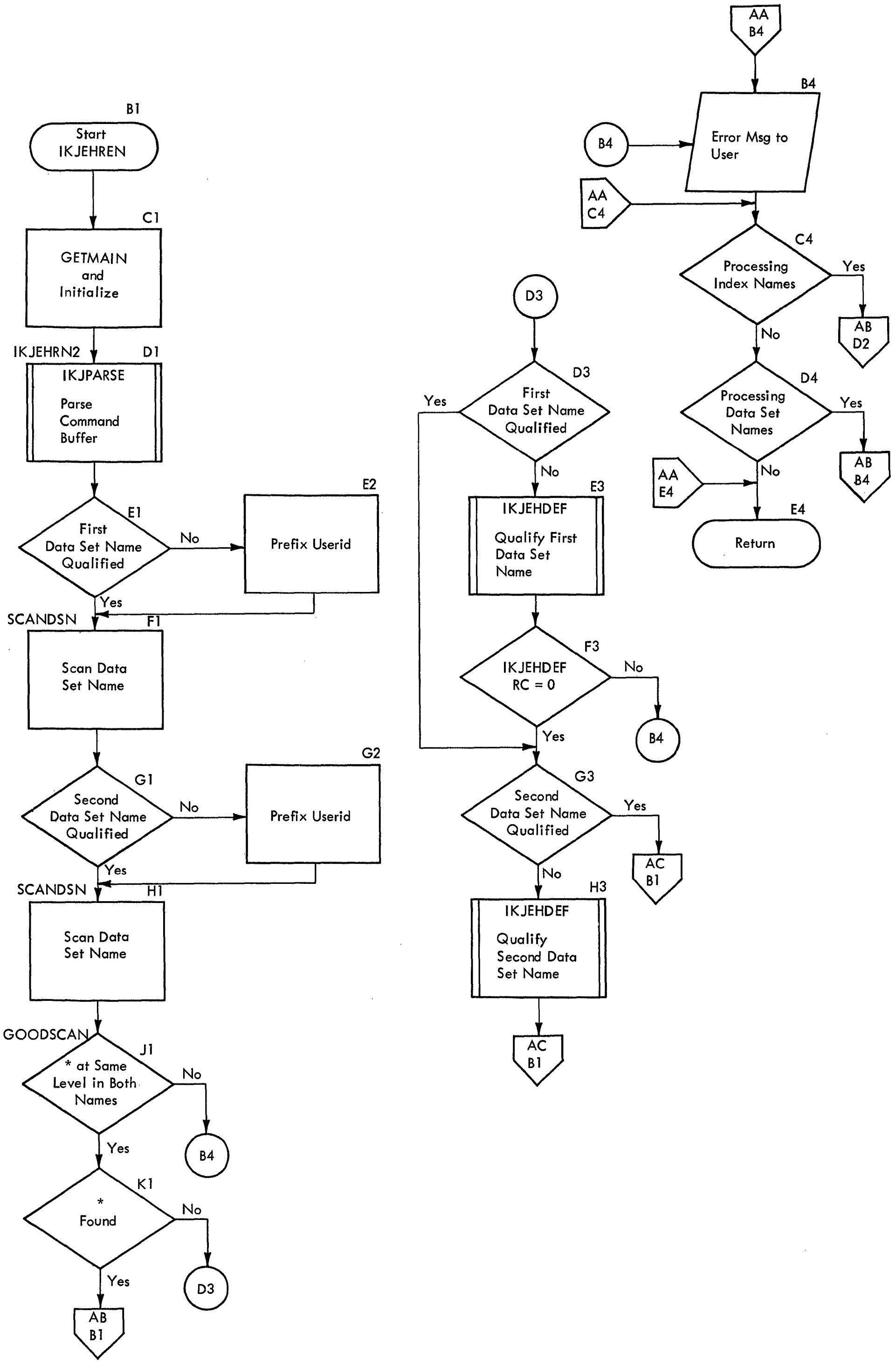
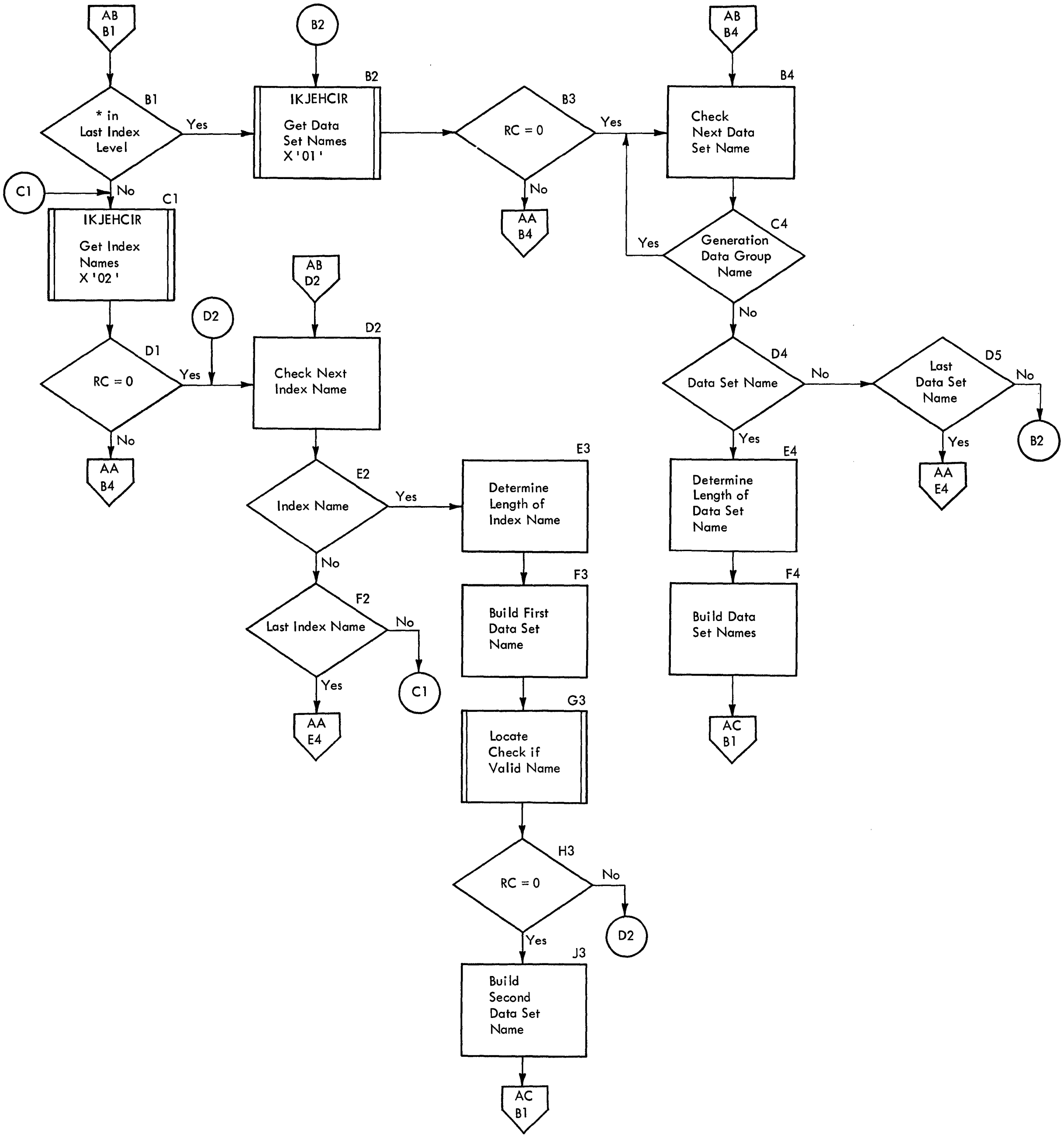
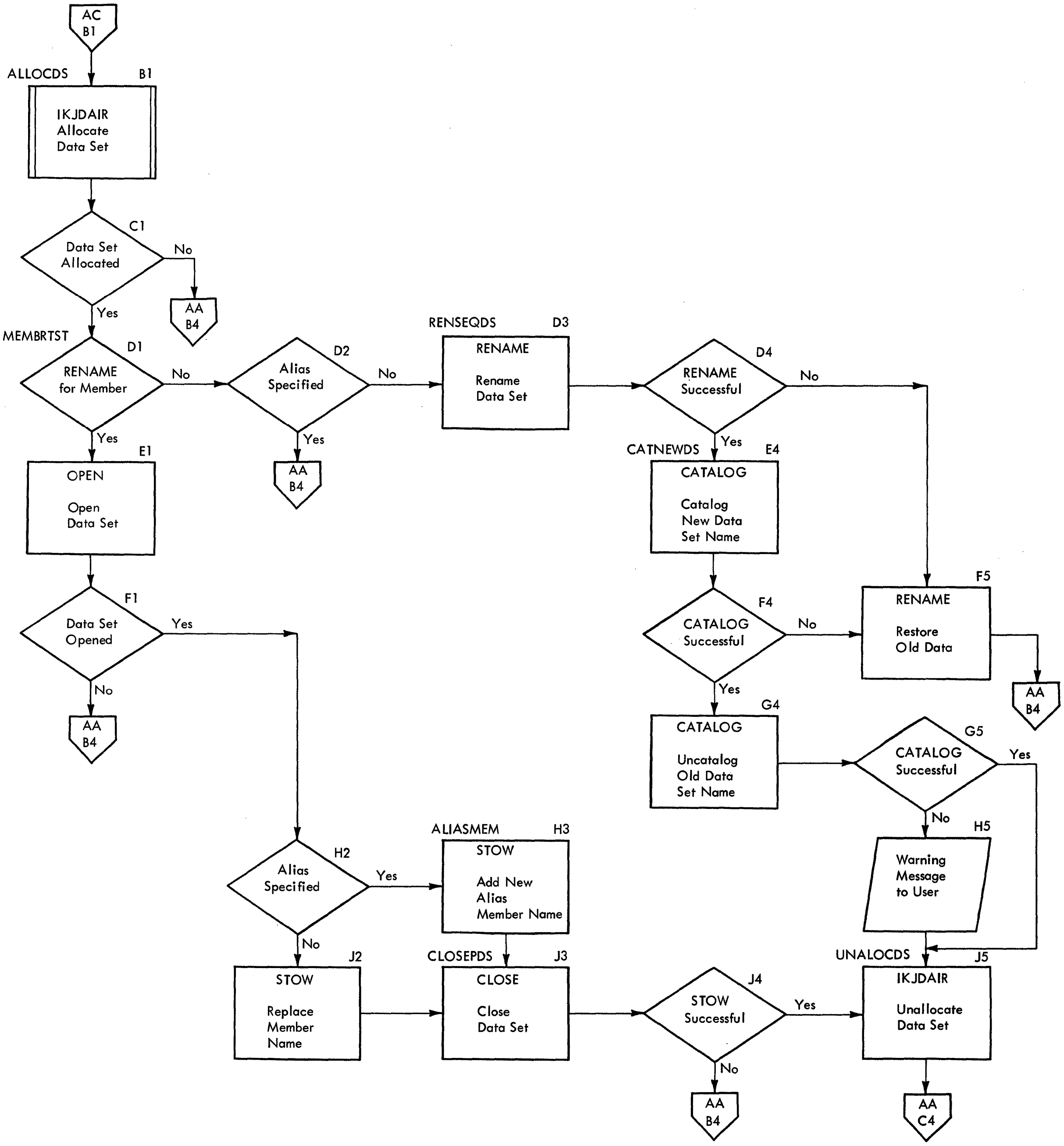


Chart AB. IKJEHREN. RENAME Command Processing (2 of 3)

Cross Reference : M.O. Diagram 2-1.



Cross Reference: M.O. Diagram 2-1.



Section 4. Directory

This section contains a routine directory organized alphabetically. Also included is a data area directory, arranged alphabetically. Flowchart and method of operation diagram references are provided, when applicable.

Routine Directory

Entry point or Routine name	Load Module	Control Section	Cross Reference	
			MO	FC
IKJEHCIR	IKJEHCIR	IKJEHCIR	2-1	AB
IKJDFLT	IKJEHDEF	IKJEHDEF	2-1	AA
IKJEHREN	IKJEHREN	IKJEHREN	2-1	AA
IKJEHRN2	IKJEHREN	IKJEHRN2	2-1	AA,AB, AC
IKJEHMSG	IKJEHREN	IKJEHMSG	N/A	AA,AC
IKJEHPDL	IKJEHREN	IKJEHPDL	N/A	AA.AC
IKJEHSMG	IKJEHREN	IKJEHMSG	N/A	N/A

Data Area Directory

Data area and Subfields	MO	FC
CIRPARM	2-1	AA,AB
Command Buffer	2-1	AA
CPPL	2-1	AA
DAPB08	2-1 2-2	AA AC
DAPB18	2-1 2-2	AA AC
DAPL	2-1	AA,AC
DFPB	2-2	AA
DFPL	2-2	AA
IKJEHCIR List	2-2	AB
IKJPARMD	2-2	AA
IOPL	N/A	AA,AC
PPL	2-1 2-2	AA
PSCB	N/A	AA

Section 5. Data Areas

This section contains the major data areas in the RENAME Command Processor. Pertinent information, such as which routine created the data area, which routine uses or updates the data area, and what the data area contains, is included. Where applicable, the data area is broken down into its component parts.

CIRPARM—A List Including Work Areas for IKJEHCIR

Constructed by: IKJEHREN.	Cross reference	FC	MO
Used by: IKJEHREN, IKJEHCIR.		AA AB	2-2

Contents: CIRPARM describes a general work area. It is found by adding the contents of register 11 and the displacement value of CIRPARM in the cross reference table.

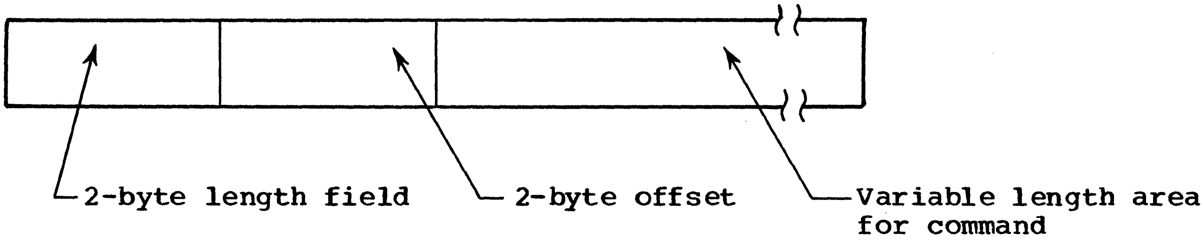
Displacement Dec. Hex.	Field	Size	Use
0 (0)	CIRPARM	1 byte	Data set name/index name options on calling IKJEHCIR.
1 (1)	Unnamed	3 bytes	Not used.
4 (4)	NAMEPTR	4 bytes	Points to index name or ttr.
8 (8)	VOLIDPTR	4 bytes	Points to volume identification.
12 (C)	WORK1PTR	4 bytes	Points to WRK1AREA.
16 (10)	WORK2PTR	4 bytes	Points to WRK2AREA.

Command Buffer

Constructed by: TMP.

Used by: IKJEHREN.

Contents: Command buffer information is painted to by the CPPLCBUF in the CPPL.



Cross reference	FC	MO
	AA	2-2

CPPL—Command Processor Parameter List Passed to IKJEHREN By the TMP

Constructed by: TMP.

Used by: IKJEHREN.

Cross reference	FC	MO
	AA	2-1

Contents: CPPL is a parameter list. It is pointed to by register 1 upon entry.

Displacement Dec.	Hex.	Field	Size	Use
0	(0)	CPPLCBUF	4 bytes	Points to the command buffer.
4	(4)	CPPLUPT	4 bytes	Points to the User Profile Table (UPT).
8	(8)	CPPLPSCB	4 bytes	Points to the Protected Step Control Block (PSCB).
12	(C)	CPPLECT	4 bytes	Points to the Environment Control Table (ECT).

DAPB08—Used to Allocate A Data Set

Cross reference	FC	MO
	AA	2-1
	AC	2-2

Constructed by: IKJEHREN.

Used by: IKJDAIR.

Contents: DAPB08 contains data set information found by adding the contents of register 11 and the displacement value of DAIR08 in the cross reference table.

Displacement Dec.	 Hex.	Field	Subfield	Size	Use
0	(0)	DA08CD		2 bytes	IKJDAIR entry code X'0008'.
2	(2)	DA08FLG		2 bytes	Functions performed when return code=0.
			DA08FPE	1 byte	Indicates function performed but an error is indicated in the return code.
4	(4)	DA08DARC		2 bytes	Dynamic allocation return code.
6	(6)	DA08CTRC		2 bytes	CATALOG routine return code.
8	(8)	DA08PDSN		4 bytes	Points to dsname to be searched in DSE.
12	(C)	DA08DDN		8 bytes	Ddname to be searched in DSE.
20	(14)	DA08UNIT		8 bytes	Unit name requested.
28	(1C)	DA08SER		8 bytes	Volume serial number.
36	(24)	DA08BLK		4 bytes	Data set average record length.
40	(28)	DA08PQTY		4 bytes	Primary space quantity.
44	(2C)	DA08SQTY		4 bytes	Secondary space quantity.

DAPB08 -- Used to allocate a data set (Cont.)

Displacement		Field	Subfield	Size	Use
Dec	Hex.				
48	(30)	DA08DQTY		4 bytes	Directory block quantity.
52	(34)	DA08MNM		8 bytes	Member name.
60	(3C)	DA08PSWD		8 bytes	Password.
68	(44)	DA08DSP1		1 byte	Data set status flags.
			DA08SHR		X'08' data set is SHR.
			DA08NEW		X'04' data set is NEW.
			DA08MOD		X'02' data set is MOD.
			DA08OLD		X'01' data set is OLD.
69	(45)	DA08DPS2		1 byte	Data set disposition.
			DA08KEEP		X'08' disp is KEEP.
			DA08DEL		X'04' disp is DELETE.
			DA08CAT		X'02' disp is CATALOG.
			DA08UCAT		X'01' disp is UNCATALOG.
70	(46)	DA08DPS3		1 byte	Data set conditional disposition.
			DA08KEP		X'08' disp is KEEP.
			DA08DELE		X'04' disp is DELETE.
			DA08CATL		X'02' disp is CATALOG.
			DA08UNCT		X'01' disp is UNCATALOG.
71	(47)	DA08CTL		1 byte	Flags to control actions of IKJDAIR.
			DA08TRKS		X'80' Units are trks.
			DA08ABKL		X'40' Space quantity in average rec. len.
			DA08UID		X'20' Userid is to be prefixed.
			DA08RLSE		X'10' Release requested.
			DA08PERM		X'08' data set must be specifically unallocatted.
			DA08DMMY		X'04' Dummy data set desired.
72	(48)	unnamed		3 bytes	Reserved.
75	(4B)	DA08DSO		1 byte	Data set organization.

DAPB18—A Parameter List Used When the Requested dsname or ddname is to be Unallocated

Constructed by: IKJEHREN.

Used by: IKJDAIR.

Contents: DAPB18 contains data set information found by adding the contents of register 11 and the displacement value of DAIR18 in the cross reference table.

Displacement Dec.	Hex.	Field	Subfield	Size	Use
0	(0)	DA18CD		2 bytes	IKJDAIR entry code X'0018'.
2	(2)	DA18FLG		2 bytes	Functions to be performed when return code=0.
			DA18FPE	1 byte	X'80' indicates function performed but an error is indicated by the return code.
4	(4)	DA18DARC		2 bytes	Dynamic allocation return code.
6	(6)	DA18CTRC		2 bytes	CATALOG routine return code.
8	(8)	DA18PDSN		4 bytes	Points to dsname to be searched in DSE.
12	(C)	DA18DDN		8 bytes	Ddname to be searched in DSE.
20	(14)	DA18MNM		8 bytes	Member name.
28	(1C)	DA18SCLS		2 bytes	SYSOUT class desired when unallocating a SYSOUT data set.
30	(1E)	DA18DPS2		1 byte	Data set disposition.
			DA18KEEP DA18DEL DA18CAT DA18UCAT		X'08' Change disp. to KEEP. X'04' Change disp. to DELETE. X'02' Change disp. to CATALOG. X'01' Change disp. to UNCATALOG.

DAPB18 (Cont.)

Displacement Dec. Hex.	Field	Subfield	Size	Use
31 (1F)	DA18CTL		1 byte	Flags for special IKJDAIR processing.
		DA18UID		X'20' Userid is to be prefixed to dsname.
32 (20)	DA18JBNM		8 bytes	Jobname.

Cross reference	FC	MO
	AA	2-1
	AC	2-2

DAPL—Dynamic Allocation Parameter List

Cross reference	FC	MO
	AB,AC	2-2

Constructed by: IKJEHREN.

Used by: IKJDAIR.

Contents: DAPL is a parameter List passed to IKJDAIR. This is a dsect map based on the address of DAIRPARM.

Displacement Dec.	 Hex.	Field	Size	Use
0	(0)	DAPLUPT	4 bytes	Points to the UPT.
4	(4)	DAPLECT	4 bytes	Points to the ECT.
8	(8)	DAPLECB	4 bytes	Points to the ECB.
12	(C)	DAPLPSCB	4 bytes	Points to the PSCB.
16	(10)	DAPLDAPB	4 bytes	Points to the IKJDAIR parameter block.

DFPB—Default (IKJEHDEF) Parameter Block

Constructed by: IKJEHREN.	Cross reference	FC	MO
Used by: IKJEHDEF.		AA	2-2

Contents: DFPB is a parameter list. It is pointed to by DFPLDFPB in DFPL.

Displacement Dec.	 Hex.	Field	Subfield	Size	Use
0	(0)	DFPBCODE		1 byte	IKJEHDEF entry code.
			DFPBOC		X'0C' entry code '0C'.
			DFPB08		X'08' entry code '08'.
			DFPB04		X'04' entry code '04'.
			DFPB00		X'00' entry code '00'.
1	(1)	DFPBDSN		3 bytes	Points to data set length and data set name.
4	(4)	DFPBCNTL		1 byte	IKJEHDEF control byte.
			DFPBUID		Userid prefix indicator X'20'.
			DFPBRET		Return added qualifier X'04'.
			DFPADD		Add specified qualifier X'02'.
			DFPBMSG		Issue msg X'01'.
5	(5)	DFPBPSCB		3 bytes	Points to PSCB.
8	(8)	DFPBLORC		1 byte	LOCATE return code returned here.
9	(9)	DFPBQUAL		3 bytes	Points to IKJEHDEF qualifier.

DFPL—A Parameter List Passed to IKJEHDEF

Constructed by: IKJEHREN.　　Cross reference　FC　MO

Used by: IKJEHDEF.　　AA　2-2

Contents: DFPL is a parameter list. This is a dsect map based on the address of DFLTPARM.

Displacement Dec.	Hex.	Field	Size	Use
0	(0)	DFPLUPT	4 bytes	Points to the UPT.
4	(4)	DFPLECT	4 bytes	Points to the ECT.
8	(8)	DFPLECB	4 bytes	Points to the ECB.
12	(C)	DFPLDFPB	4 bytes	Points to the IKJEHDEF parameter block.

IKJEHCIR—List of Information Returned to Caller

Constructed by: IKJEHCIR.

Used by: IKJEHREN.

Contents: This list of information obtained from the system catalog is pointed to by WRK1AREA if asterisks are being processed. If asterisks are not being processed, this information is not available here.

The list passed back by this module consists of various entries, each identified by a one-byte code. An entry is included in the list if its corresponding option has been requested. Options include the X'01' and X'02'.

<table>
<tr><th>Option</th><th>Size and Use of Entry</th></tr>
<tr><td>X'01'</td><td>Two types of entries may be included in the returned list.
<table>
<tr><td>07</td><td>dsname</td></tr>
<tr><td>1</td><td>8</td></tr>
</table>
This nine-byte entry represents a regular (simple) dsname entry found in the current block of the specified level of index of the catalog.</td></tr>
<tr><td></td><td>
<table>
<tr><td>02</td><td>gdgname</td><td>ttr</td><td>data</td></tr>
<tr><td>1</td><td>8</td><td>3</td><td>4</td></tr>
</table>
This sixteen-byte entry represents a generation data group (gdg) entry found in the specified level of index of the catalog. The ttr points to an index block that contains the simple dsname entries for the group. The generation number will be in the two's complement form as found in the dsname entries and must be converted back before using. The data field contains four bytes of information describing the characteristics of the group.</td></tr>
</table>

IKJEHCIR -- List of Information Returned to Caller. (Cont.)

<table>
<tr><th>Option</th><th>Size and Use of Entry</th></tr>
<tr><td>X'02'</td><td><table><tr><td>00</td><td>indexname</td><td>ttr</td></tr><tr><td>1</td><td>8</td><td>3</td></tr></table>This is a twelve-byte entry that represents a simple qualifier (index name) found at the specified level of index in the catalog. The ttr points to the first block of the named index.</td></tr>
<tr><td>Link</td><td><table><tr><td>FF</td><td>ttr</td><td>volser</td></tr><tr><td>1</td><td>3</td><td>6</td></tr></table>Link is a ten-byte entry terminating every list regardless of options specified. If the ttr is nonzero, it points to the next index block, and the volser is the CVOL it is on.</td></tr>
</table>

Cross reference FC MO
AB 2-2

IKJPARMD—A Parameter List Returned from IKJPARS (Local IKJPARS Descriptor List)

Constructed by: IKJPARS, IKJEHREN (format determined by RENAME macros).

Used by: IKJEHREN.

Contents: IKJPARMD contains data set information. It is pointed to by ANSPLACE. ANSPLACE is pointed to by PPLANS in the PPL.

Displacement Dec.	 Hex.	Field	Size	Use
0	(0)	DSNAME1	4 bytes	Points to dsname.
4	(4)		2 bytes	Length of dsname.
6	(6)		1 byte	Flags for dsname information.
7	(7)		1 byte	Not used.
8	(8)		4 bytes	Points to member name.
12	(C)		2 bytes	Length of member name.
14	(E)		1 byte	Flags for member name information.
15	(F)		1 byte	Not used.
16	(10)		4 bytes	Points to password.
20	(14)		2 bytes	Length of password.
22	(16)		1 byte	Flags for password information.
23	(17)		1 byte	Not used.
24	(18)	DSNAME2	4 bytes	Points to second dsname.
28	(1C)		2 bytes	Length of second dsname.
30	(1E)		1 byte	Flags for second dsname information.
31	(1F)		1 byte	Not used.
32	(20)		4 bytes	Points to member name.

Displacement Dec. Hex.	Field	Size	Use
36 (24)		2 bytes	Length of member name.
38 (26)		1 byte	Flags for member name information.
39 (27)		1 byte	Not used.
40 (28)		4 bytes	Points to password.
44 (2C)		2 bytes	Length of password.
46 (2E)		1 byte	Flags for password information.
47 (2F)		1 byte	Not used.
48 (30)	RENTYPE	2 bytes	Used for keyword specifications. 0 indicates no keyword specified. 1 indicates 'ALIAS' keyword specified.

Note: DSNAME1, DSNAME2, and RENTYPE are referenced by labels, not displacements.

Cross reference	FC	MO
	AA	2-2

IOPL—Input/Output Parameter List

Constructed by: IKJEHREN.

Contents: IOPL is a parameter list passed to IKJPUTL. It is a dsect based on address of IOPARM.

Displacement Dec.	 Hex.	Field	Size	Use
0	(0)	IOPLUPT	4 bytes	Points to the UPT.
4	(4)	IOPLECT	4 bytes	Points to the ECT.
8	(8)	IOPLECB	4 bytes	Points to the ECB.
12	(C)	IOPLIOPB	4 bytes	Points to the I/O parameter block.

Cross reference	FC	MO
	AA,AC	N/A

PPL—A Parameter List Passed to IKJPARS

Constructed by: IKJEHREN.

Used by: IKJEHREN, IKJPARS.

Cross reference	FC	MO
	AA	2-1 2-2

Contents: PPL is a parameter list found by adding the contents of register 11 and the displacement value for PARSPARM in the cross reference table.

Displacement Dec. Hex.	Field	Size	Use
0 (0)	PPLUPT	4 bytes	Points to the UPT.
4 (4)	PPLECT	4 bytes	Points to the ECT.
8 (8)	PPLECB	4 bytes	Points to the ECB.
12 (C)	PPLPCL	4 bytes	Points to the PCL.
16 (10)	PPLANS	4 bytes	Points to the answer place.
20 (14)	PPLCBUF	4 bytes	Points to the command buffer.
24 (18)	PPLUWA	4 bytes	Points to the user work area.

PSCB—Protected Step Control Block

Constructed by: LOGON.

Used by: IKJEHREN.

Cross reference	FC	MO
	AA	N/A

Contents: PSCB contains user attribute and accounting data. It is pointed to by the CPPLPSCB in the CPPL.

Displacement Dec. Hex.	Field	Subfield	Size	Use
0 (0)	PSCBUSER		7 bytes	Userid.
7 (7)	PSCBUSRL		1 byte	Length of userid.
8 (8)	PSCBGPNM		8 bytes	Esoteric group name.
16 (10)	PSCBATR1		1 byte	User attributes
		PSCBCTRL		X'80' indicates OPERATOR command user.
		PSCBACCT		X'40' indicates ACCOUNT command user.
		PSCBJCL		X'20' indicates SUBMIT command user.
				Bits 3-15 are reserved for IBM use.
17 (11)	unnamed		1 byte	Reserved.
18 (12)	PSCBATR2		1 byte	User attributes (for installation use).
19 (13)	unnamed		1 byte	Reserved.
20 (14)	PSCBCPU		4 bytes	Cumulative CPU time of session.
24 (18)	PSCBSWP		4 bytes	Cumulative time resident in region.

PSCB -- Protected Step Control Block (Cont.)

Displacement Dec. Hex.	Field	Subfield	Size	Use
28 (1C)	PSCBLTIM		4 bytes	Actual LOGON time of day.
32 (20)	PSCBTCPU		4 bytes	Total CPU time (accounting period).
36 (24)	PSCBTSWP		4 bytes	Total time resident (accounting period).
40 (28)	PSCBTCON		4 bytes	Total time connected (accounting period).
44 (2C)	PSCBTCO1		4 bytes	Additional field for accounting period.
48 (30)	PSCBRLGB		4 bytes	Points to relogon buffer.
52 (34)	PSCBUPT		4 bytes	Points to UPT.
56 (38)	PSCBUPTL		2 bytes	Length of UPT.
58 (3A)	unnamed		6 bytes	Reserved for IBM.
64 (40)	PSCBU		8 bytes	Reserved for installation use.

Section 6. Diagnostic Aids

This section contains a register summary for the control sections IKJEHREN and IKJEHREN2 and the messages dealing with RENAME command processing.

Register Summary

Register	Usage
0	Used as a work register.
1	Used as a parameter register, and as a work register.
2	Used initially to save the parameter list; then as a work register.
3	Used in 'EXEC MVC' as a pointer to 'TO AREA' and as work register.
4	Used in 'EXEC MVC' as a pointer to 'FROM AREA' and as work register.
5,6	Used as a work register.
7	Not used.
8	Used as an index register for IKJEHCIR workarea.
9	Used as a base register for IKJPARS DSECT.
10	Used as a work register.
11	Used as a base register for GETMAIN dsect common.
12	Used as a base register for IKJEHREN.
13	Used as a save area register.
14	Used as a link register.
15	Used as a branch register, return code register.

Messages

Issued by	Message ID	Message Text
IKJEHREN	IKJ58201I	DATA SET dsname NOT IN CATALOG
IKJEHREN	IKJ58202I	NOT ENOUGH MAIN STORAGE TO EXECUTE COMMAND
IKJEHREN IKJEHREN	IKJ58203I #IKJ58203I	DATA SET dsname NOT RENAMED, EXPIRATION DATE HAS NOT PASSED+ SYSTEM OPERATOR DID NOT AUTHORIZE ACCESS TO DATA SET
IKJEHREN IKJEHREN	IKJ58204I #IKJ58204I	DATA SET dsname NOT USABLE+ OPEN ERROR CODE xxxx
IKJEHREN IKJEHREN	IKJ58205I #IKJ58205I	INVALID USE OF ALIAS KEYWORD+ ALIAS VALID ONLY FOR MEMBERS OF PARTITIONED DATA SETS
IKJEHREN IKJEHREN IKJEHREN IKJEHREN IKJEHREN	IKJ58206I #IKJ58206I #IKJ58206I #IKJ58206I #IKJ58206I	INVALID USE OF * IN dsname+ MORE THAN ONE * IN DATA SET NAME * NOT VALID AS FIRST INDEX LEVEL DATA SET NAMES USED * AT DIFFERENT LEVELS * CANNOT BE USED WITH MEMBER
IKJEHREN IKJEHREN IKJEHREN IKJEHREN IKJEHREN	IKJ58207I #IKJ58207I #IKJ58207I #IKJ58207I #IKJ58207I	UNABLE TO RENAME DATA SET dsname+ RENAME ERROR CODE xxxx BLDL ERROR CODE xxxx STOW ERROR CODE xxxx DATA SET RESIDES ON MULTIPLE VOLUMES
IKJEHREN IKJEHREN	IKJ58208I #IKJ58208I	INVALID USE OF COMMAND+ BOTH MEMBERS MUST BE FROM THE SAME PARTITIONED DATA SET
IKJEHREN	#IKJ58208I	A DATA SET CANNOT BE RENAMED TO A MEMBER NAME
IKJEHREN	#IKJ58208I	A MEMBER NAME CANNOT BE RENAMED TO A DATA SET
IKJEHREN	IKJ58209I	DATA SET NAME IS REQUIRED WHEN MEMBER member IS SPECIFIED
IKJEHREN IKJEHREN	IKJ58210I #IKJ58210I	DATA SET RENAMED BUT dsname STILL CATALOGED+ CATALOG ERROR CODE xxxx
IKJEHREN IKJEHREN	IKJ58211I #IKJ58211I	DATA SET dsname NOT ALLOCATED, TOO MANY DATA SETS+ USE FREE COMMAND TO FREE UNUSED DATA SETS

MESSAGES (Cont.)

Issued by	Message ID	Message Text
IKJEHREN	IKJ58212I	DATA SET dsname NOT ALLOCATED, DATA SET NOT ON VOLUME+
IKJEHREN	#IKJ58212I	CATALOG INFORMATION INCORRECT
IKJEHREN	IKJ58213I	DATA SET dsname NOT ALLOCATED, REQUIRED VOLUME NOT MOUNTED+
IKJEHREN	#IKJ58213I	VOLUME OR CVOL NOT ON SYSTEM AND CANNOT BE ACCESSED
IKJEHREN	IKJ58214I	DATA SET dsname NOT ALLOCATED, SYSTEM OR INSTALLATION ERROR+
IKJEHREN	#IKJ58214I	DYNAMIC ALLOCATION ERROR CODE xxxx
IKJEHREN	#IKJ58214I	CATALOG I/O ERROR
IKJEHREN	IKJ58215I	DATA SET dsname ALREADY IN USE, TRY LATER+
IKJEHREN	#IKJ58215I	DATA SET IS ALLOCATED TO ANOTHER JOB OR USER
IKJEHREN	IKJ58216I	PROPER PASSWORD NOT SUPPLIED FOR dsname
IKJEHREN	IKJ58217I	SPECIFIED MEMBER NOT IN DATA SET dsname (member)
IKJEHREN	IKJ58218I	INVALID DATA SET NAME, 'dsname' EXCEEDS 44 CHARACTERS
IKJEHREN	IKJ58219I	NO DATA SETS FOUND TO MATCH * IN dsname
IKJEHREN	IKJ58220I	DATA SET dsname NOT ON A DIRECT ACCESS DEVICE, NOT SUPPORTED
IKJEHREN	IKJ58221I	DATA SET NAME dsname RESIDES ON MULTIPLE VOLUMES, NOT SUPPORTED
IKJEHREN	#IKJ58222I	DATA SET dsname ALREADY EXISTS
IKJEHREN	IKJ58223I	MEMBER member ALREADY EXISTS
IKJEHREN	IKJ58224I	MEMBER SPECIFIED BUT dsname NOT A PARTITIONED DATA SET

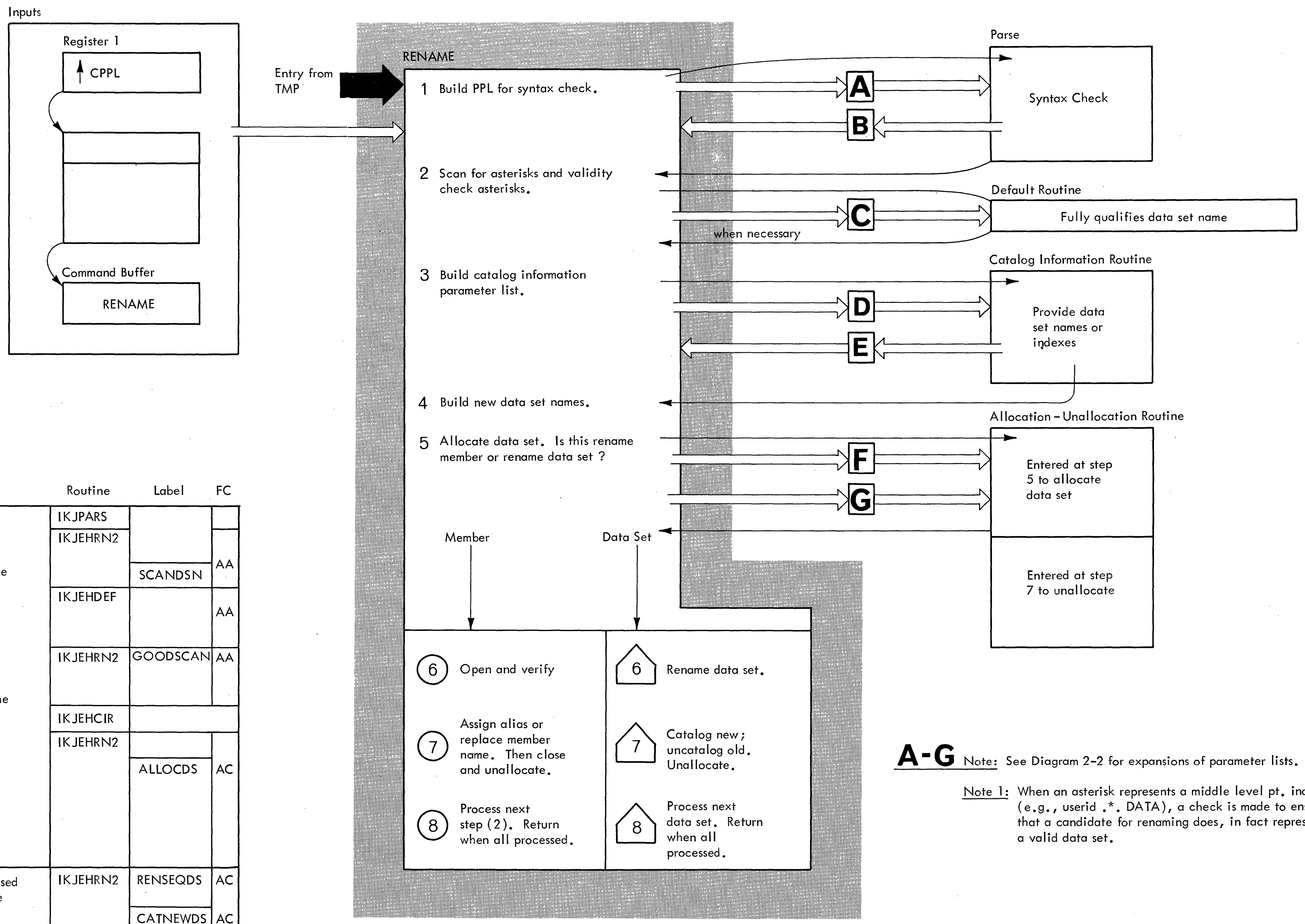
MESSAGES (Cont.)

Issued by	Message ID	Message Text
IKJEHREN IKJEHREN IKJEHREN IKJEHREN	IKJ58225I #IKJ58225I #IKJ58225I #IKJ58225I	COMMAND SYSTEM ERROR+ PARSE ERROR CODE xxxx DAIR ERROR CODE xxxx DEFAULT ERROR CODE xxxx
IKJEHREN	IKJ58226I	NO SPACE IN DIRECTORY FOR ALIAS
IKJEHREN IKJEHREN	IKJ58227I #IKJ58227I	DATA SET NAME dsname WILL CREATE INVALID CATALOG STRUCTURE+ A QUALIFIER CANNOT BE BOTH AN INDEX AND THE LAST QUALIFIER OF A DATA SET NAME
IKJEHREN IKJEHREN	IKJ58228I #IKJ58228I	UNABLE TO CATALOG dsname+ CATALOG ERROR CODE xxxx
IKJEHREN IKJEHREN IKJEHREN	IKJ58229I #IKJ58229I #IKJ58229I	DATA SET dsname NOT ALLOCATED+ INVALID UNIT IN USER ATTRIBUTE DATA SET NO UNIT AVAILABLE
IKJPARS IKJPARS IKJPARS IKJPARS	IKJ56700A IKJ56700A IKJ56700A IKJ56700A	ENTER OLD DATA SET NAME- ENTER NEW DATA SET NAME- ENTER NAME OF DATA SET TO BE RENAMED- ENTER NEW NAME TO BE GIVEN THE DATA SET-

# used to indicate second level messages for documentation only.

Index

Indexes to program logic manuals are consolidated in the publication IBM System/360 Operating System: Program Logic Manual Master Index, Order No. GY28-6717. For additional information about any subject listed below, refer to other publications listed for the same subject in the Master Index.


Inputs
Register 1
CPPL
Command Buffer
RENAME
Entry from TMP
RENAME
1 Build PPL for syntax check.
2 Scan for asterisks and validity check asterisks.
3 Build catalog information parameter list.
4 Build new data set names.
5 Allocate data set. Is this rename member or rename data set ?
Member
Data Set
6 Open and verify
7 Assign alias or replace member name. Then close and unallocate.
8 Process next step (2). Return when all processed.
6 Rename data set.
7 Catalog new; uncatalog old. Unallocate.
8 Process next data set. Return when all processed.
A
B
C
D
E
F
G
when necessary
Parse
Syntax Check
Default Routine
Fully qualifies data set name
Catalog Information Routine
Provide data set names or indexes
Allocation - Unallocation Routine
Entered at step 5 to allocate data set
Entered at step 7 to unallocate

A-G Note: See Diagram 2-2 for expansions of parameter lists.

Note 1: When an asterisk represents a middle level pt. index (e.g., userid .*. DATA), a check is made to ensure that a candidate for renaming does, in fact represent a valid data set.

Descriptions	Routine	Label	FC
1 The PARSE subroutine syntax checks the command. Storage is obtained for work areas.	IKJPARS		
2 A user id is prefixed if necessary. The data set name is scanned for asterisks. If none, prompting is done for any necessary qualification of data set names by the default routine. (See note 1) **C**. Then operation continues from step 4.	IKJEHRN2	SCANDSN	AA
If asterisks are found, they are checked to ensure that they occur in the same relative position within the fully qualified data set names.	IKJEHDEF		AA
3 If asterisks were found, the catalog information routine is used to look up candidates for renaming. 4 The new data set names are built in preparation for the renaming operation.	IKJEHRN2	GOODSCAN	AA
5 Allocation is done to make use of the OS enqueuing facility which ensures that the data set is not renamed while some other user is using it. (Also, this enables the OPEN and CLOSE operation for partitioned members.)	IKJEHCIR		
	IKJEHRN2	ALLOCDS	AC

Member	Routine	Label	FC	Data Set	Routine	Label	FC
6 OPEN and BLDL are used to open the PDS.	IKJEHRN2	MEMBRTST	AC	6 RENAME is used to rename the data set.	IKJEHRN2	RENSEQDS CATNEWDS	AC AC
7 STOW is used to assign the alias or new member name.		CLOSEPDS	AC	7 CATALOG is used to catalog new and uncatalog old.		UNALLOCDS	AC
8 The data set is closed and unallocated.				8 Repeat from step 4, if applicable.			

Diagram 2-1. RENAME Command Processing

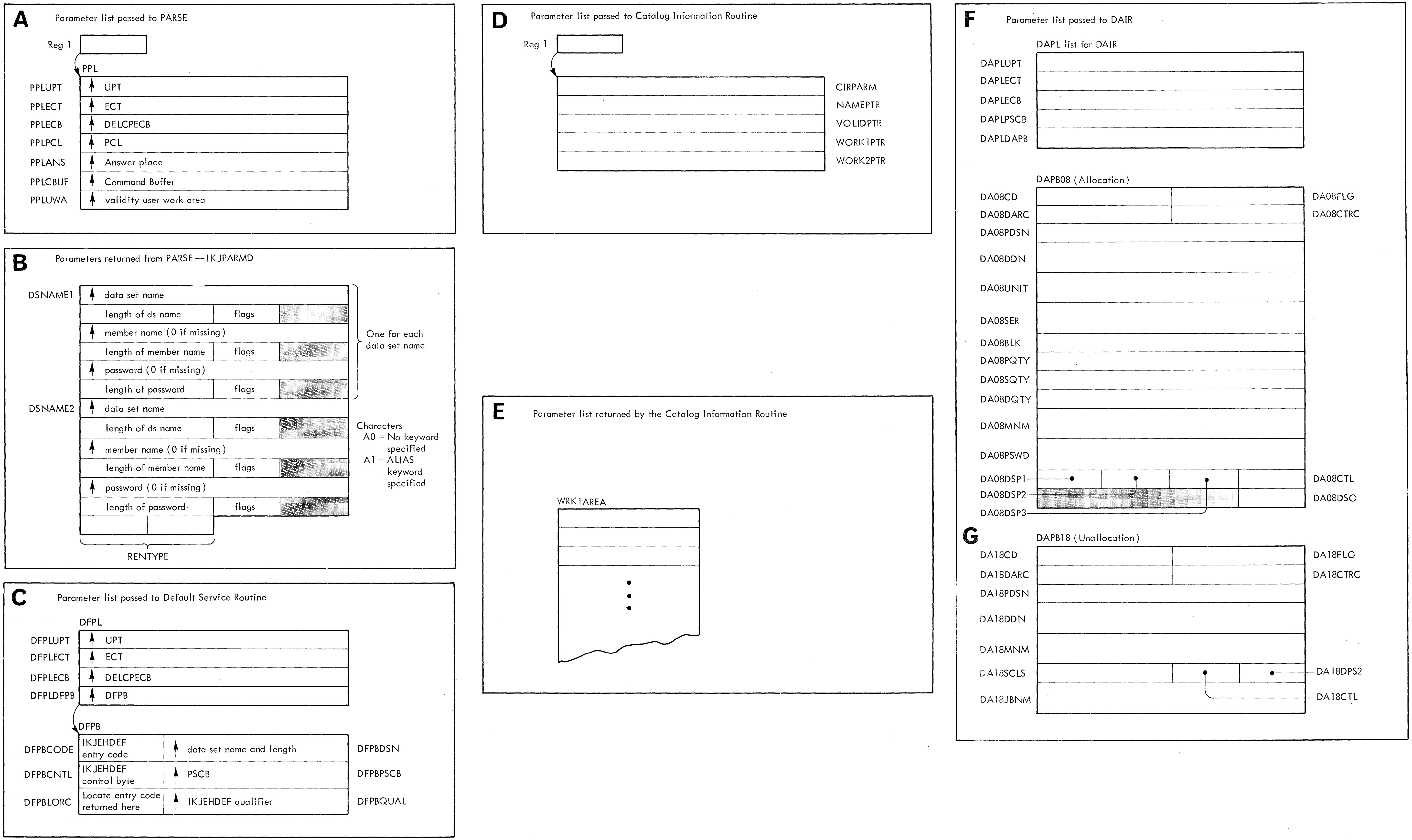


Diagram 2-2. RENAME Parameter Lists

Part 4: RUN Command Processor

4

Contents

Illustrations

Charts

Diagrams

Figures

Section 1. Introduction

The RUN Command Processor is used to build an in-storage list of commands to compile, load and execute a specified source program. This list of commands is placed on the input stack that describes the currently active sources of input. Upon return to the Terminal Monitor Program (TMP), the next GETLINE (IKJGETL) issued by the TMP results in obtaining the first command in the list.

Functions

The RUN Command Processor performs the following functions:

- Obtains the command name from the command buffer.
- Scans and syntax checks the command.
- Determines data set name and verifies it.
- Determines data set type.
- Builds in-storage list.
- Builds Input Source Descriptor (table describing the in-storage list).
- Makes the list available for input.
- Issues appropriate messages.

Environment

The RUN Command Processor operates in a TSO user's region under the protection key assigned to that region. The processor's static instructions and data requires about 5K of storage. Additional storage is needed for the TSO service routines used by this processor. Definitive information on space needed for TSO and the RUN Command Processor can be obtained in the Storage Estimates SRL, GC28-6551.

Physical Characteristics

The RUN Command Processor consists of one load module IKJEFR00. It has an alias of RUN.

IKJEFR00 is the main processing program and handles all the major functions of RUN.

Operational Considerations

RUN communicates with a terminal through the TSO PUTLINE service routine (IKJPUTL).

The TSO routines IKJPARS and IKJDAIR also are required. IKJPARS scans and syntax checks the command. It checks the user's data set type. IKJDAIR is used to allocate a data set and verify the specified data set name.

Input is the standard Command Processor Parameter List (CPPL). Upon receiving control, register 1 points to the CPPL.

The command buffer contains the RUN command and its operands.

The RUN command consists of the following:

- The command name RUN.
- The optional keyword parameters (See the Command Language Reference, GC28-6732).

Messages are printed at the terminal. These include diagnostic messages, informational messages, and prompting messages.

Section 2. Method of Operation

This section describes the logic of the RUN command processor. It emphasizes the flow of data and control information through buffers and tables and contains detailed functional descriptions of the RUN operation.

Program logic information for the TMP and the TSO service routines used by the RUN command processor is contained in the Terminal Monitor Program and Service Routines Program Logic Manual, GY28-6770.

RUN Command Processing Summary

When the RUN command processor receives control from the TMP the command is scanned and syntax checked. The data set name and data set type are processed. The specified data set name is verified. An in-storage command list and a table describing the List Source Descriptor is built. The command list is made available for input.

Introduction to the Diagrams

The remainder of the section consists of the method of operation diagrams for the RUN command processor. Some of the diagrams contain a cross-reference table to help you locate the corresponding flowcharts and applicable sections of the program listings.

The method of operation diagrams are designed to serve you in different ways:

- Initially, they quickly provide you with an understanding of the programming functions of the command processor.

- They show how the code supports the functions to be performed.

- They provide "pointers" to the actual implementations (via cross references to the flowcharts or program listings).

You can use them to become familiar with the system, or later, to find your way to specific implementation points in the program listings or other areas of the PLM.

Organization of the Diagrams

Diageam 2-1 shows the overview of the Command Processor. Diagram 2-2 shows how to build a RUN list.

The processing block lists the functions performed in producing final output.

The descriptions provided at the left of the diagram list the major implementation steps performed in support of the processing functions.

The routine label and flowchart cross-referencing provided with the descriptions are "key" points designed to show you where to pick up the implementations in the program listings or flowcharts.

Symbols

This PLM uses the following symbols:

➡	Heavy black arrow indicates begin reading the diagram here.
⇨	White arrow indicates data transfer from one area to another.
➞	Black arrow shows flow of logic or passing of control.
——►	Thin black arrow indicates pointer to an item.
– – –►	Dotted arrow indicates reference to an item.
⬠	Off-page connector leads to a related diagram.
– – – –×	Getting or Freeing main storage.
■□➡	Signifies passing of parameters.

Section 3. Program Organization

This section describes the RUN command processor routines and how they relate to each other. Flowcharts are included following the discussion.

Hierarchy information is shown in Figure 3-1.

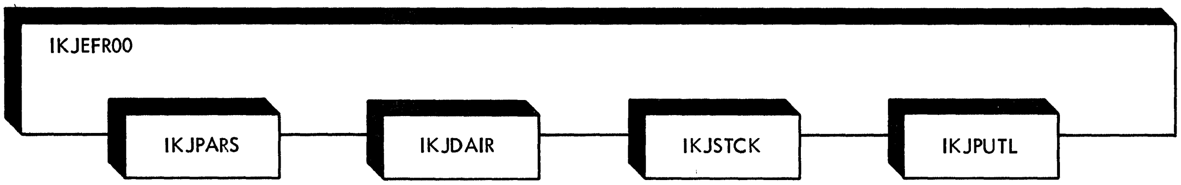


Figure 3-1. Control Module Hierarchy.

IKJEFROO

Entry Point	IKJEFR00.
Operation	Scans and syntax checks the command. Processes the data set name and data set type. Verifies the data set name. Builds the command list. Makes the command list available for input. Process messages.
Data Areas Used	CPPL, COMMDBUF, DAPB08, DSNPDE, INSRCDES, IOPL, PPL, RUNPDL, WORKAREA.
Routines Called	LINK to IKJPARS LOAD and branch to IKJPUTL LINK to IKJDAIR LOAD and branch to IKJSTCK
Exits	Normal return to caller.
Registers	Upon entry, register 1 points to CPPL. Upon exit to TMP, register 15 contains a return code. Return codes are 0 and 12, indicating no error or error in processing.

Cross reference	FC*	MO**
	AA-AF	2-1,2-2

* indicates flowchart
** indicates method of operation diagram

Chart AA. IKJEFR00. RUN Command Processing (Overview) (1 of 6)

Cross Reference: Diagram 2-1.

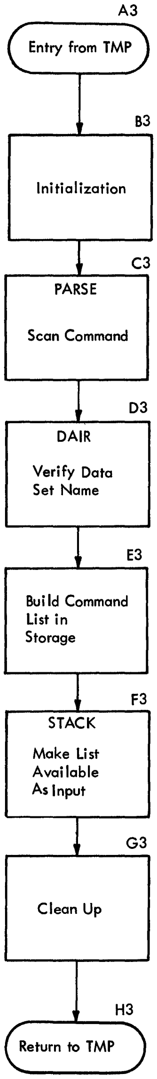


Chart AB. IKJEFR00. RUN Command Processing (2 of 6)

Cross Reference: Diagram 2-1.

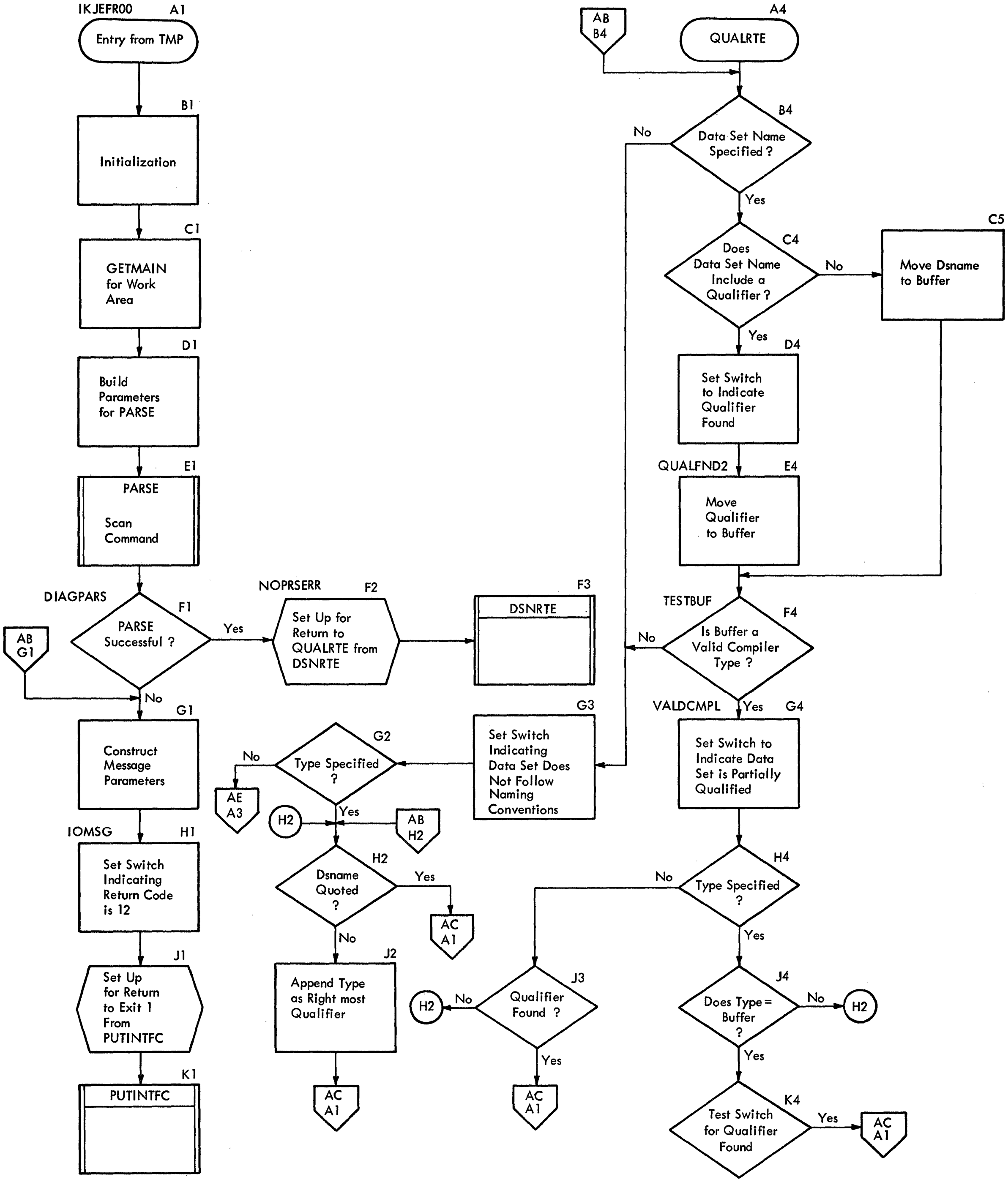
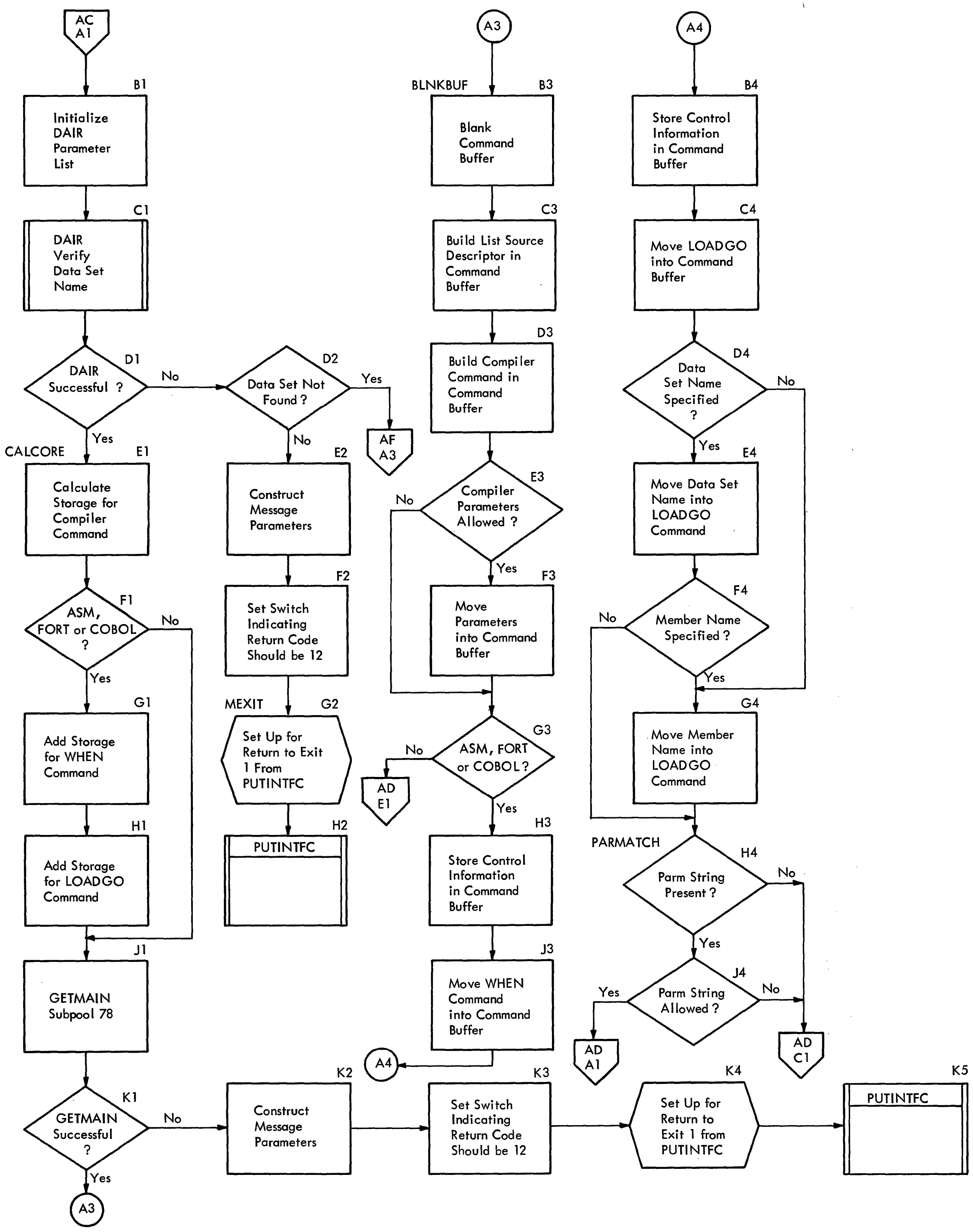

IKJEFR00 A1
Entry from TMP
B1
Initialization
C1
GETMAIN for Work Area
D1
Build Parameters for PARSE
E1
PARSE
Scan Command
DIAGPARS
F1
PARSE Successful ?
Yes
No
AB G1
G1
Construct Message Parameters
IOMSG H1
Set Switch Indicating Return Code is 12
J1
Set Up for Return to Exit 1 From PUTINTFC
K1
PUTINTFC
NOPRSERR F2
Set Up for Return to QUALRTE from DSNRTE
F3
DSNRTE
G2
Type Specified ?
No
AE A3
Yes
H2
AB H2
H2
Dsname Quoted ?
Yes
AC A1
No
J2
Append Type as Right most Qualifier
AC A1
G3
Set Switch Indicating Data Set Does Not Follow Naming Conventions
AB B4
A4
QUALRTE
B4
Data Set Name Specified ?
No
Yes
C4
Does Data Set Name Include a Qualifier ?
No
C5
Move Dsname to Buffer
Yes
D4
Set Switch to Indicate Qualifier Found
QUALFND2 E4
Move Qualifier to Buffer
TESTBUF
F4
Is Buffer a Valid Compiler Type ?
No
VALDCMPL Yes G4
Set Switch to Indicate Data Set is Partially Qualified
H4
Type Specified ?
No
Yes
J3
Qualifier Found ?
H2 No
Yes
AC A1
J4
Does Type = Buffer ?
No H2
Yes
K4
Test Switch for Qualifier Found
Yes
AC A1

Chart AC. IKJEFR00. RUN Command Processing (3 of 6)

Cross Reference: Diagrams 2-1, 2-2.


AC A1
B1
Initialize DAIR Parameter List
C1
DAIR Verify Data Set Name
D1
DAIR Successful ?
No
Yes
CALCORE
E1
Calculate Storage for Compiler Command
F1
ASM, FORT or COBOL ?
No
Yes
G1
Add Storage for WHEN Command
H1
Add Storage for LOADGO Command
J1
GETMAIN Subpool 78
K1
GETMAIN Successful ?
No
Yes
A3
D2
Data Set Not Found ?
Yes
AF A3
No
E2
Construct Message Parameters
F2
Set Switch Indicating Return Code Should be 12
MEXIT
G2
Set Up for Return to Exit 1 From PUTINTFC
H2
PUTINTFC
K2
Construct Message Parameters
A3
BLNKBUF
B3
Blank Command Buffer
C3
Build List Source Descriptor in Command Buffer
D3
Build Compiler Command in Command Buffer
E3
Compiler Parameters Allowed ?
No
Yes
F3
Move Parameters into Command Buffer
G3
ASM, FORT or COBOL?
No
AD E1
Yes
H3
Store Control Information in Command Buffer
J3
Move WHEN Command into Command Buffer
A4
K3
Set Switch Indicating Return Code Should be 12
A4
B4
Store Control Information in Command Buffer
C4
Move LOADGO into Command Buffer
D4
Data Set Name Specified ?
No
Yes
E4
Move Data Set Name into LOADGO Command
F4
Member Name Specified ?
No
Yes
G4
Move Member Name into LOADGO Command
PARMATCH
H4
Parm String Present ?
No
Yes
J4
Parm String Allowed ?
Yes
No
AD A1
AD C1
K4
Set Up for Return to Exit 1 from PUTINTFC
K5
PUTINTFC

Cross Reference: Diagram 2-1.

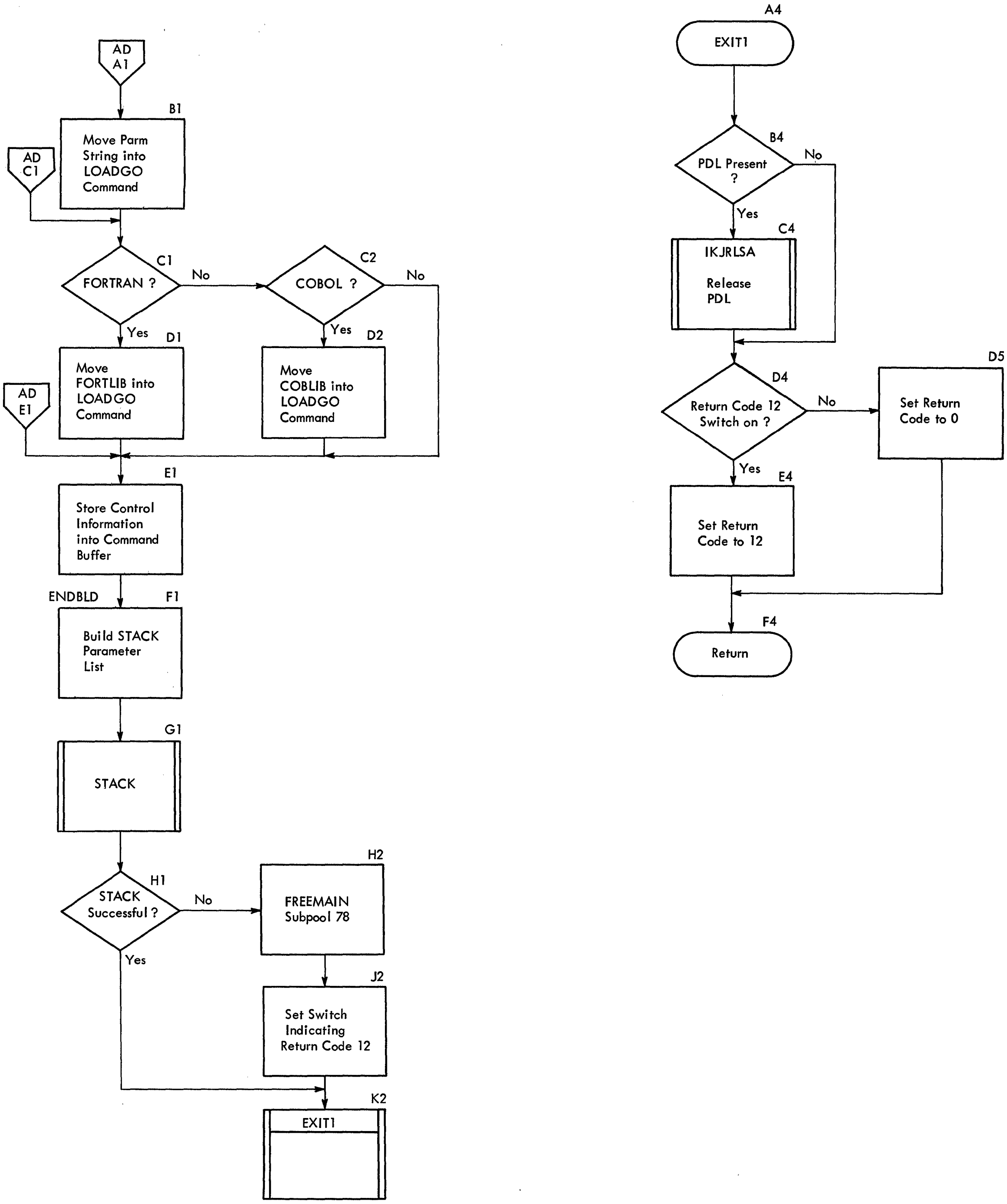

AD A1
B1
Move Parm String into LOADGO Command
AD C1
C1
FORTRAN ?
No
C2
COBOL ?
No
Yes
D1
Move FORTLIB into LOADGO Command
Yes
D2
Move COBLIB into LOADGO Command
AD E1
E1
Store Control Information into Command Buffer
ENDBLD
F1
Build STACK Parameter List
G1
STACK
H1
STACK Successful ?
No
H2
FREEMAIN Subpool 78
Yes
J2
Set Switch Indicating Return Code 12
K2
EXIT1
A4
EXIT1
B4
PDL Present ?
No
Yes
C4
IKJRLSA
Release PDL
D4
Return Code 12 Switch on ?
No
D5
Set Return Code to 0
Yes
E4
Set Return Code to 12
F4
Return

Chart AE. IKJEFR00. RUN Command Processing (5 of 6)

Cross Reference: Diagram 2-1.

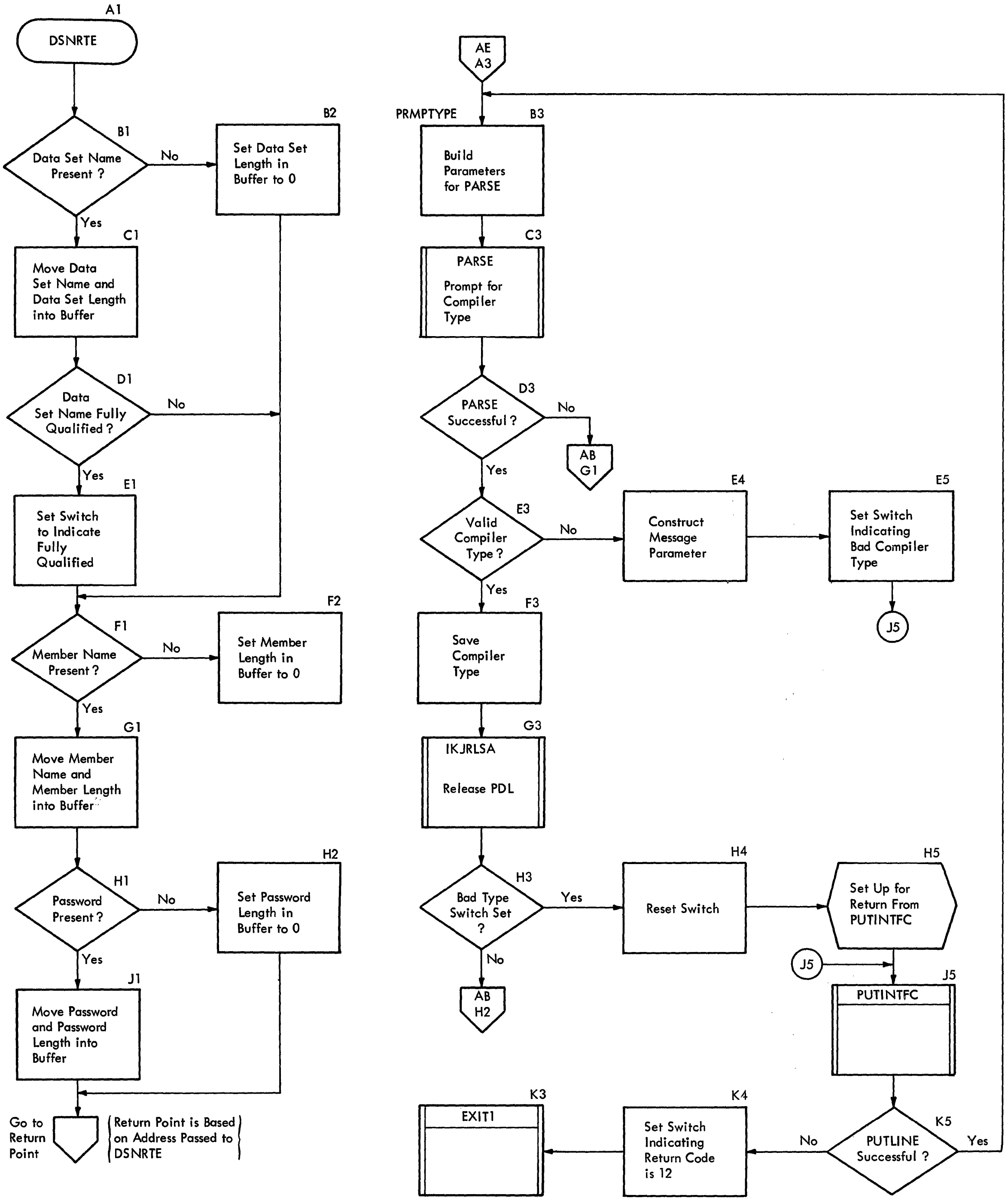
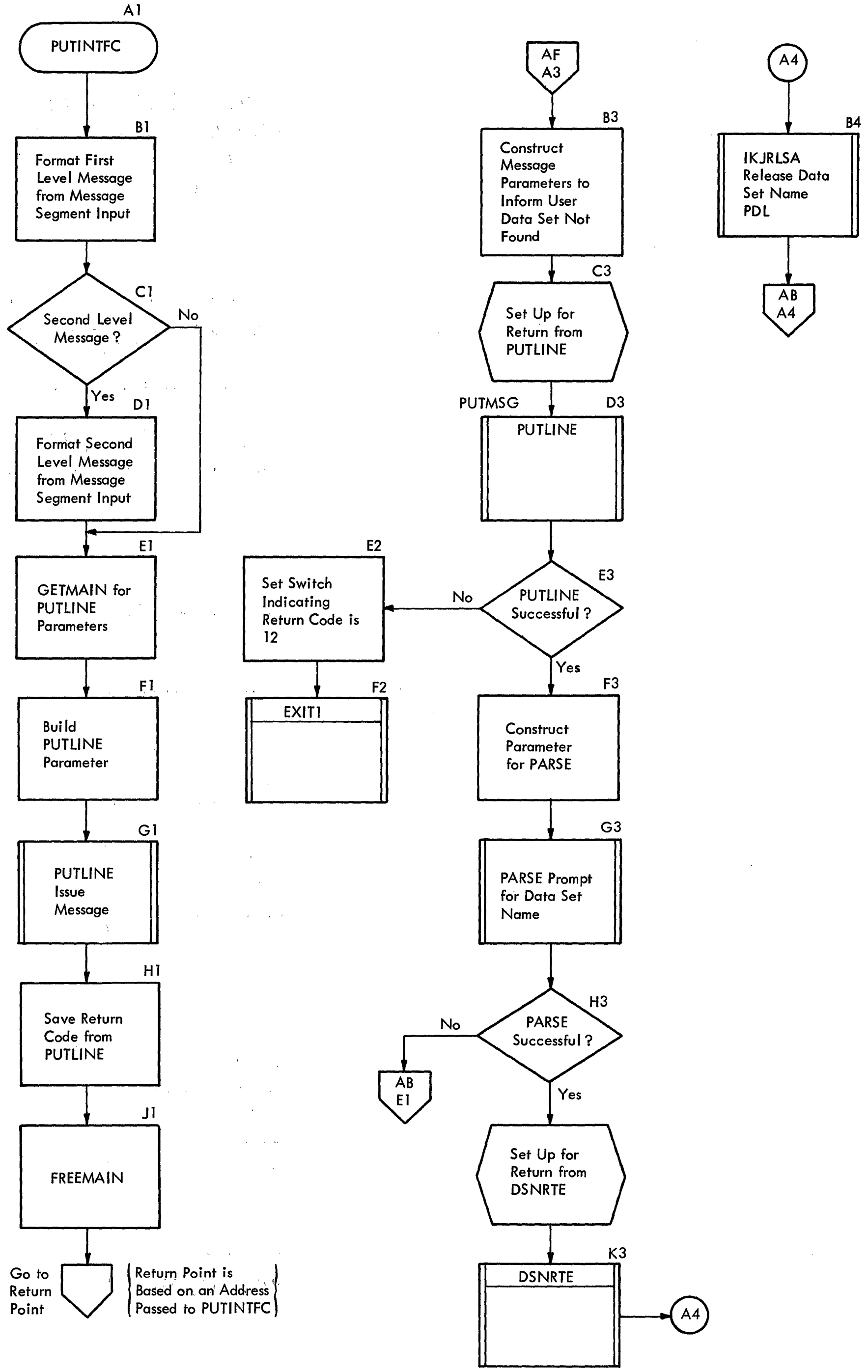

A1
DSNRTE
B1
Data Set Name Present ?
No
B2
Set Data Set Length in Buffer to 0
Yes
C1
Move Data Set Name and Data Set Length into Buffer
D1
Data Set Name Fully Qualified ?
No
Yes
E1
Set Switch to Indicate Fully Qualified
F1
Member Name Present ?
No
F2
Set Member Length in Buffer to 0
Yes
G1
Move Member Name and Member Length into Buffer
H1
Password Present ?
No
H2
Set Password Length in Buffer to 0
Yes
J1
Move Password and Password Length into Buffer
Go to Return Point
Return Point is Based on Address Passed to DSNRTE
AE A3
PRMPTYPE
B3
Build Parameters for PARSE
C3
PARSE
Prompt for Compiler Type
D3
PARSE Successful ?
No
AB G1
Yes
E3
Valid Compiler Type ?
No
E4
Construct Message Parameter
E5
Set Switch Indicating Bad Compiler Type
J5
Yes
F3
Save Compiler Type
G3
IKJRLSA
Release PDL
H3
Bad Type Switch Set ?
Yes
H4
Reset Switch
H5
Set Up for Return From PUTINTFC
No
AB H2
J5
J5
PUTINTFC
K5
PUTLINE Successful ?
Yes
No
K4
Set Switch Indicating Return Code is 12
K3
EXIT1

Chart AF. IKJEFR00. RUN Command Processing (6 of 6)

Cross Reference: Diagram 2-1.


A1
PUTINTFC
B1
Format First Level Message from Message Segment Input
C1
Second Level Message ?
No
Yes
D1
Format Second Level Message from Message Segment Input
E1
GETMAIN for PUTLINE Parameters
F1
Build PUTLINE Parameter
G1
PUTLINE Issue Message
H1
Save Return Code from PUTLINE
J1
FREEMAIN
Go to Return Point
Return Point is Based on an Address Passed to PUTINTFC
E2
Set Switch Indicating Return Code is 12
F2
EXIT1
AF A3
B3
Construct Message Parameters to Inform User Data Set Not Found
C3
Set Up for Return from PUTLINE
PUTMSG
D3
PUTLINE
E3
PUTLINE Successful ?
No
Yes
F3
Construct Parameter for PARSE
G3
PARSE Prompt for Data Set Name
H3
PARSE Successful ?
No
AB E1
Yes
Set Up for Return from DSNRTE
K3
DSNRTE
A4
A4
B4
IKJRLSA Release Data Set Name PDL
AB A4

Section 4. Directory

This section contains the routine directory organized alphabetically by entry point or routine name. A data area directory also is provided and is arranged alphabetically. Cross references to flow charts and method of operation diagrams are provided when applicable.

Routine Directory

Entry point, Routine name	Load Module	Control Section	Cross reference	
			MO	FC
IKJEFR00	IKJEFR00	IKJEFR00	2-1,2-2	AA,AF

Data Area Directory

Data Area, Subfields	MO	FC
COMMDBUF	2-1	AC
CPPL	2-1	AA
DAIRSTOR	N/A	AB
DAPB04	N/A	AC
DAPL	N/A	N/A
DSNPDE	2-1	AD
INSRCDES	N/A	N/A
IOPL	N/A	AE
PPL	N/A	AA
RUNPDL	N/A	AA
WORKAREA	2-1	AA,AC,AD

Section 5. Data Areas

This section contains the major data areas in the RUN Command Pocessor. Pertinent information, such as which routine created the data area, which routine uses or updates the data area, and what the data area contains, is included.

COMMDBUF

Created by: TMP.

Used by: IKJEFR00.

Contents: COMMDBUF is the command buffer. It is addressed in the parameter list (CPPL) passed in register 1 upon entry.

2 byte length	2 byte offset	Variable length for command

CPPL

Created by: TMP.

Used by: IKJEFR00.

Contents: CPPL is the parameter list pointed to by register 1 upon entry.

Disp Dec Hex	Field	Size	Use
0 (0)	CPPLCBUF	4 bytes	Points to the command buffer.
4 (4)	CPPLUPT	4 bytes	Points to the UPT.
8 (8)	CPPLPSCB	4 bytes	Points to the PSCB.
12 (C)	CPPLECT	4 bytes	Points to the ECT.

Cross reference	FC	MO
	AA	2-1

DAIRSTOR

Created by: IKJEFR00.

Used by: IKJDAIR.

Contents: DAIRSTOR is the IKJDAIR parameter list. It is found in WORKAREA.

Displacement Dec.	 Hex.	Field	Subfield	Size	Use
0	(0)	DAIRPARM		16 bytes	IKJDAIR parameter list.
16	(10)	DSBUFFER		46 bytes	Buffer.
			DSLENG	2 bytes	Length of dsname.
			DSNBUF	44 bytes	Data set name.
62	(3E)	Unnamed		2 bytes	Alignment.

DAPB04

Constructed by: IKJEFR00.

Used by: IKJDAIR.

Contents: DAPB04 is used to search the DSE for a specified data set name. If not found, the system catalog is also searched. It is found by adding the contents of register 12 to the displacement value of DAPB04 in the cross reference table in the program listing.

Displacement Dec. Hex.	Field	Subfield	Size	Use
0 (0)	DA04CD		2 bytes	IKJDAIR entry code.
2 (2)	DA04FLG		2 bytes	Reserved for flags.
		DA04CAT	1 bit	Indicates data set name found in catalog.
		DA04DSE	1 bit	Indicates data set name found in DSE.
4 (4)	Unnamed		2 bytes	Reserved.
6 (6)	DA04CTRC		2 bytes	Catalog return.
8 (8)	DA04DSN		4 bytes	Points to dsname to be searched.
12 (C)	DA04CTL		1 byte	Control flags for special processing.
		DA04UID		Userid to be prefixed to data set name.
13 (D)	Unnamed		2 bytes	Reserved.
15 (F)	DA04DSO		1 byte	Indicates data set organization.

Cross reference	FC	MO
	AC	N/A

DAPL

Constructed by: IKJEFR00.

Used by: IKJDAIR, IKJEFR00.

Contents: DAPL is a parameter list. It is found by adding the contents of register 12 to the displacement value of DAPL in the cross reference table in the program listing.

Displacement Dec. Hex.	Field	Size	Use
0 (0)	DAPLUPT	4 bytes	Points to UPT.
4 (4)	DAPLECT	4 bytes	Points to ECT.
8 (8)	DAPLECB	4 bytes	Points to ECB.
12 (C)	DAPLPSCB	4 bytes	Points to PSCB.
16 (10)	DAPLDAPB	4 bytes	Points to IKJDAIR parameter block.

DSNPDE

Created by: IKJPARS.

Used by: IKJEFR00.

Contents: DSNPDE is an IKJPARS descriptor element. It is found in the RUNPDL.

Disp Dec Hex	Field	Subfield	Size	Use
0 (0)	DSNPTR		4 bytes	Points to data set name.
4 (4)	DSLEN		2 bytes	Length of data set name.
6 (6)	DSNFLAG		1 byte	Flag byte.
		DSNPRES		1 bit when on indicates dsname present.
		DSNQUOT		1 bit when on indicates dsname quoted.
7 (7)	unnamed		1 byte	Reserved.
8 (8)	MBRPTR		4 bytes	Points to member name.
12 (C)	MBRLEN		2 bytes	Length of member name.
14 (E)	MBRFLAG		1 byte	Flag byte.
		MBRPRES		1 bit when on indicates member name present.
15 (F)	unnamed		1 byte	Reserved.
16 (10)	PASSPTR		4 bytes	Points to password.
20 (14)	PASSLEN		2 bytes	Length of password.
22 (16)	PASSFLAG		1 byte	Flag byte.
		PASSPRES		1 bit; when on, indicates password present.
23 (17)	unnamed		1 byte	Reserved.

INSRCDES

Created by: IKJEFR00.

Used by: IKJSTCK.

Contents: List Source Descriptor. It is pointed to by ADSTOR.

Disp Dec Hex	Field	Size	Use
0 (0)	DATAAREA	4 bytes	Points to command list.
4 (4)	RECLEN	2 bytes	Record lengths. Zero here indicates variable length records.
6 (6)	LISTLEN	2 bytes	Length of command list.
8 (8)	NEXTBYTE	4 bytes	Points to the next byte.
12 (C)	unnamed	4 bytes	Reserved.

IOPL

Created by: IKJEFR00.

Used by: IKJPUTL.

Contents: IOPL is a parameter list for IKJPUTL. It is found by adding the contents of register 12 to the displacement value of IOPL in the cross reference table in the program listings.

Disp Dec Hex	Field	Size	Use
0 (0)	IOPLUPT	4 bytes	Points to the UPT.
4 (4)	IOPLECT	4 bytes	Points to the ECT.
8 (8)	IOPLECB	4 bytes	Points to the ECB.
12 (C)	IOPLIOPB	4 bytes	Points to the I/O parameter block.

Cross reference	FC	MO
	AE	N/A

PPL

Constructed by: IKJEFR00.

Used by: IKJPARS.

Contents: PPL is a parameter list passed to IKJPARS. It is found by adding the contents of register 12 to the displacement value of PPL in the cross reference table in the program listings.

Disp Dec Hex	Field	Size	Use
0 (0)	PPLUPT	4 bytes	Points to the UPT.
4 (4)	PPLECT	4 bytes	Points to the ECT.
8 (8)	PPLECB	4 bytes	Points to the ECB.
12 (C)	PPLPCL	4 bytes	Points to the parameter control list.
16 (10)	PPLANS	4 bytes	Points to the answer place.
20 (14)	PPLCBUF	4 bytes	Points to the user work area.

Cross reference	FC	MO
	AA	N/A

RUNPDL

Created by: IKJPARS.

Used by: IKJEFR00.

Contents: RUNPDL is a parameter descriptor list. It is pointed to by PDLADDR.

Disp Dec Hex	Field	Subfield	Size	Use
0 (0)	unnamed		8 bytes	Storage chains used by IKJPARS.
8 (8)	DSNPDE		24 bytes	See DSNPDE described earlier in this section.
32 (20)	PARMPDE		8 bytes	PDE for 'PARM' parameter.
		PARMPTR		4-byte that point to parm string.
		PARMLEN		2-byte length field. Bit 0 set to on indicates parameters present. It is called PARMPRES. The remaining bits are not used.
		PARMFLAG		1-byte for flags.
		unnamed		1-byte reserved.
40 (28)	TYPEPDE		2 bytes	Data set type PDE. 0 -- not specified. 1 -- ASM. 2 -- FORT. 3 -- COBOL. 4 -- BASIC. 5 -- IPLI. 6 -- GOFORT. 7 -- PLI. The remaining bits are reserved.
42 (2A)	MSGPDE		2 bytes	PDE for LMSG/SMSG. When the bit is on: 0 -- not specified. 1 -- LMSG. 2 -- SMSG. The remaining bits are reserved.

RUNPDL (Cont.)

Disp Dec Hex	Field	Subfield	Size	Use
44 (2C)	PRECPDE		2 bytes	PDE for LPREC/SPREC. When the bit is on: 0 -- not specified. 1 -- LPREC. 2 -- SPREC. The remaining bits are reserved.
46 (2E)	TESTPDE		2 bytes	PDE for TEST/NOTEST. When the bit is on: 0 -- not specified. 1 -- TEST. 2 -- NOTEST. The remaining bits are reserved.
48 (30)	FORMPDE		2 bytes	PDE for FIXED/FREE. When the bit is on: 0 -- not specified. 1 -- FIXED. 2 -- FREE. The remaining bits are reserved.
50 (32)	OPTPDE		2 bytes	PDE for CHECK/OPT. When the bit is on: 0 -- not specified. 1 -- check. 2 -- Opt. The remaining bits are reserved.

Cross reference	FC	MO
	AA	N/A

WORKAREA

Created by: IKJEFR00.

Used by: IKJEFR00.

Contents: It is a general workarea. It is pointed to by register 12.

Disp Dec(Hex)	Field	Subfield	Size	Use
0 (0)	PARSPARM		8 bytes	PARSE parm. list.
		PDLADDR		4 bytes for answer place pointer.
		PDLADDR2		4 bytes for pointer to PDL.
8 (8)	STACKPRM		8 bytes	STACK parameter list.
		unnamed		4 bytes for setting proper value by list form of STACK.
		ISDPTR		4 bytes to point to List Source Descriptor.
16 (10)	DAIRSTOR		64 bytes	See DAIRSTOR description.
80 (50)	COMMECB		4 bytes	Communications ECB.
84 (54)	SERTEPRM		28 bytes	Service routine parameter list.
		ADUPT		Address of UPT (4 bytes).
		ADECT		Addres of ECT (4 bytes).
		ADECB		Address of ECB (4 bytes).
		unnamed		Pointers to service routines (16 bytes).
112 (70)	MBRBUF		10 bytes	Subfield insertions.
		MBRLENG		2-byte length field.
		MBRNAME		8-byte buffer for member name.

WORKAREA (Cont.)

Disp Dec Hex	Field	Subfield	Size	Use
122 (7A)	PASSBUF		10 bytes	Subfield insertions.
		PASSLENG		2-byte length field.
		PASSNAME		8-byte buffer for password.
132 (84)	QUALBUF		8 bytes	Buffer to hold qualifier of data set name.

Cross reference	FC	MO
	AA,AC,AD	2-2

Section 6. Diagnostic Aids

This section contains a register summary and the messages from the RUN command processor.

Register Summary

Register	Use
1	Points to CPPL. Also used as work register.
2	Work register.
3	Points to IKJPARS PDL.
4	Work register.
5	Work register.
6	Work register.
7	Work register.
8	Work register.
9	Base register.
10	Not used.
11	Base register.
12	Base register for WORKAREA.
13	Save area register.
14	Branch register.
15	Return code register.

Messages

Message ID	Message Text
IKJ56051I	DATA SET dsname NOT IN CATALOG
IKJ56052I	COMMAND SYSTEM ERROR+
*IKJ56052I	service rtn ERROR CODE xxxx
IKJ56053I	PARAMETER FIELD TRUNCATED TO 100 CHARACTERS
IKJ56054I	type INVALID DATA TYPE
IKJ56055I	parameters IGNORED FOR type PROCESSING

The following IKJPARS messages are also issued if applicable:

INVALID DATA SET TYPE-

REENTER DATA SET TYPE-

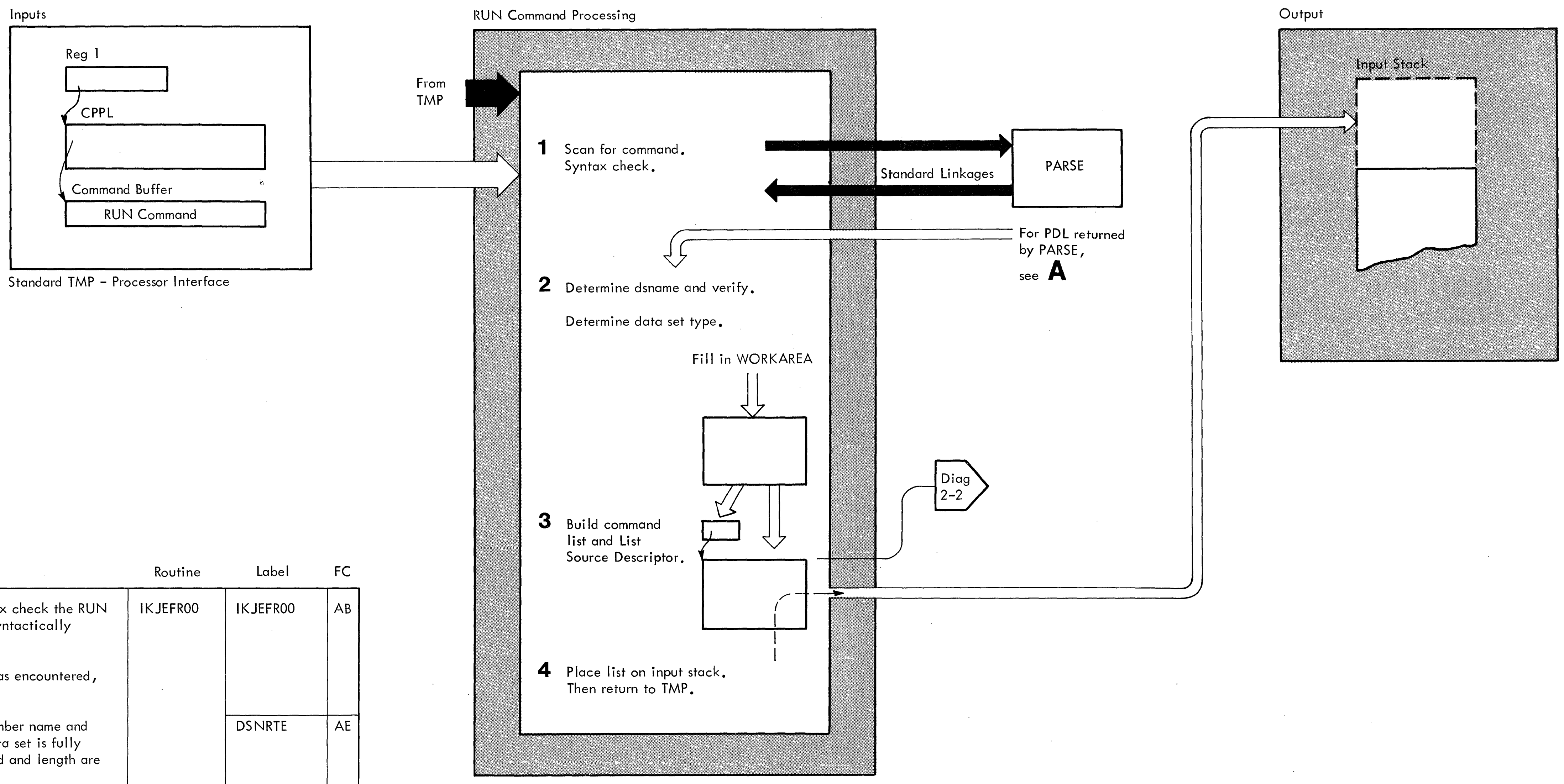
ENTER DATA SET NAME-

ENTER DATA SET TYPE-

*indicates a second level message for documentation only.

Index

Indexes to program logic manuals are consolidated in the publication IBM System/360 Operating System: Program Logic Manual Master Index, Order No. GY28-6717. For additional information about any subject listed below, refer to other publications listed for the same subject.



Inputs
Reg 1
CPPL
Command Buffer
RUN Command
Standard TMP - Processor Interface
From TMP
RUN Command Processing
1 Scan for command. Syntax check.
Standard Linkages
PARSE
For PDL returned by PARSE, see A
2 Determine dsname and verify.
Determine data set type.
Fill in WORKAREA
3 Build command list and List Source Descriptor.
Diag 2-2
4 Place list on input stack. Then return to TMP.
Output
Input Stack

Description	Routine	Label	FC
1 IKJEFR00 uses the PARSE service routine, IKJPARS, to scan and syntax check the RUN command. Prompting occurs if required parameters are missing or if syntactically incorrect parameters are present.	IKJEFR00	IKJEFR00	AB
2 Upon return from IKJPARS, the return code is checked. If an error was encountered, a message is issued to the user; otherwise, processing continues.			
Control passes to a routine that examines the specified data set or member name and places applicable information into a buffer in WORKAREA. If the data set is fully qualified, an indicator is set. If a password is specified, the password and length are placed in WORKAREA.		DSNRTE	AE
Then the data set type (ASM, etc) is determined and placed in the data set type buffer of WORKAREA. IKJPARS is again used, if necessary, to prompt for the data set type.		QUALRTE	AB
The DAIR service routine is then given control to search for a data set having the specified name. First the user's DSE is searched; then if necessary, the system catalog. If the data set is found, processing continues; otherwise, the user is prompted for a respecification, and another search is made.		DAIRINIT	
3 When a data set is verified as existing, storage is obtained in shared subpool 78 for an in-storage command list and a table (the List Source Descriptor) describing the list. See Diagram 2-2 for details of this operation.		CALCORE	
4 After the in-storage command list and List Source Descriptor are built, the address of the List Source Descriptor is placed in the STACK parameter list and control is passed to the STACK Service routine. This routine places the command list on the input stack. Then control returns to the TMP. The TMP will select the next command from the top of the input stack.		ENDBLD	AD

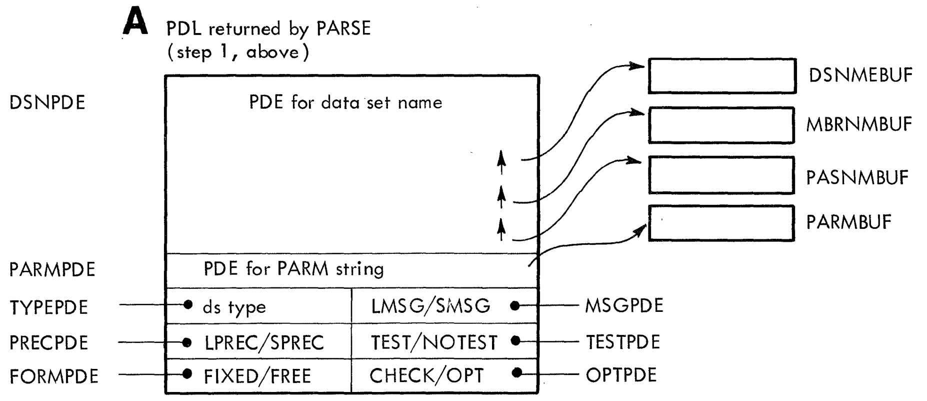
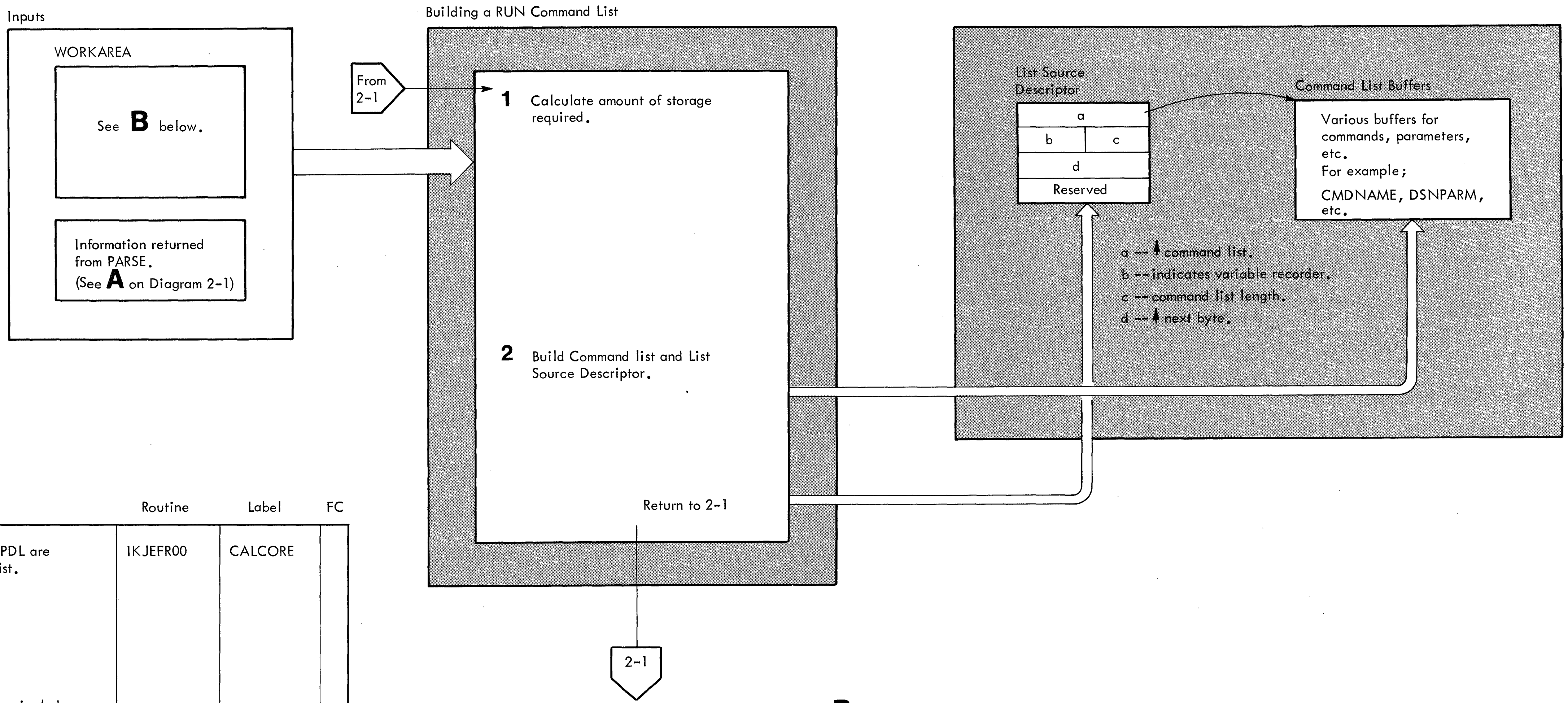


Diagram 2-1. RUN Command Processing Overview


Inputs
WORKAREA
See B below.
Information returned from PARSE.
(See A on Diagram 2-1)
Building a RUN Command List
From 2-1
1 Calculate amount of storage required.
2 Build Command list and List Source Descriptor.
Return to 2-1
2-1
List Source Descriptor
a
b
c
d
Reserved
Command List Buffers
Various buffers for commands, parameters, etc.
For example;
CMDNAME, DSNPARM, etc.
a -- ↑ command list.
b -- indicates variable recorder.
c -- command list length.
d -- ↑ next byte.

Description	Routine	Label	FC
1 WORKAREA fields and parse information previously located through the PDL are examined to calculate the amount of storage required for the command list. Included in the calculation are : • The length of the List Source Descriptor (16 bytes) • The size of the compiler command. • The length of the data set name. • Compiler parameters, if any. • LOADGO command size (for ASM, FORT, or COBOL). This size includes control information length; LOADGO length; LOADGO data set name length; The length of the WHEN/END command, which is used to prevent execution of the program in the event the compiler does not complete successfully; parameter information, if any; COBLIB and FORTLIB length (for COBOL and FORT data sets.)	IKJEFR00	CALCORE	
2 The command list and List Source Descriptor are built. The List source descriptor is filled in as the command list is constructed.		BLNKBUF	AC
First, the compiler command (type) is built. Control information consists of a two-byte length field followed by two bytes containing 0. The compiler command is moved to the appropriate buffer (CMDNAME).		INSRCBLD	
Then the data set name is moved to the command list buffer (DSNPARM). If a compiler parameter is specified (for BASIC, or GOFORT), it is placed in the buffer, along with the parameter length. If the compiler is ASM, FORT, or COBOL, the WHEN command is built and placed in the list. Then the LOADGO command is created. This consists of placing in the buffer the proper data set name, applicable parameters, and for the FORT and COBOL data sets, FORTLIB or COBLIB, respectively. The length of the LOADGO command is placed in the control field.		MOVEDSN	
After the command list is complete, it is placed on the input stack (See Step 4 of Diagram 2-1).		ENDBLD	AD

Note: Shaded areas were previously filled in by DSNRTE (if applicable) from information found through the PDL. (See Diagram 2-1).

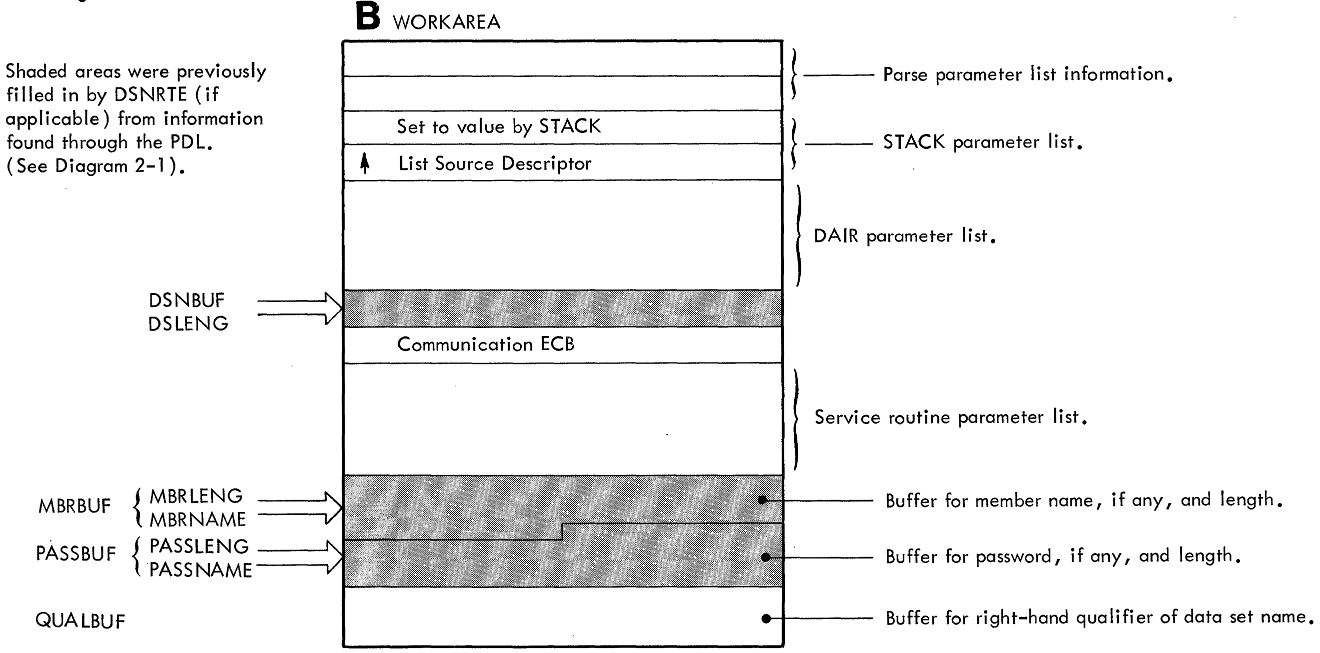

B WORKAREA
Parse parameter list information.
Set to value by STACK
↑ List Source Descriptor
STACK parameter list.
DAIR parameter list.
DSNBUF
DSLENG
Communication ECB
Service routine parameter list.
MBRBUF { MBRLENG MBRNAME
Buffer for member name, if any, and length.
PASSBUF { PASSLENG PASSNAME
Buffer for password, if any, and length.
QUALBUF
Buffer for right-hand qualifier of data set name.

Diagram 2-2. Building a RUN Command List

Part 5: SEND Command Processor

5

Contents

Illustrations

Charts

Diagrams

Figures

Section 1. Introduction

The Send Command Processor enables the terminal user to send a message to any system operator console or to any authorized TSO user.

Function

When the message is for an operator the number of the operator console is a part of the command. The processor uses the WTO macro to send the message to the console.

When the message is for a user, his USERID is given in the command. If the user is logged on, his USERID will be in the time sharing job block (TJB) associated with the terminal he is using. The processor locates the TJB and sends the message to the terminal via the TPUT macro.

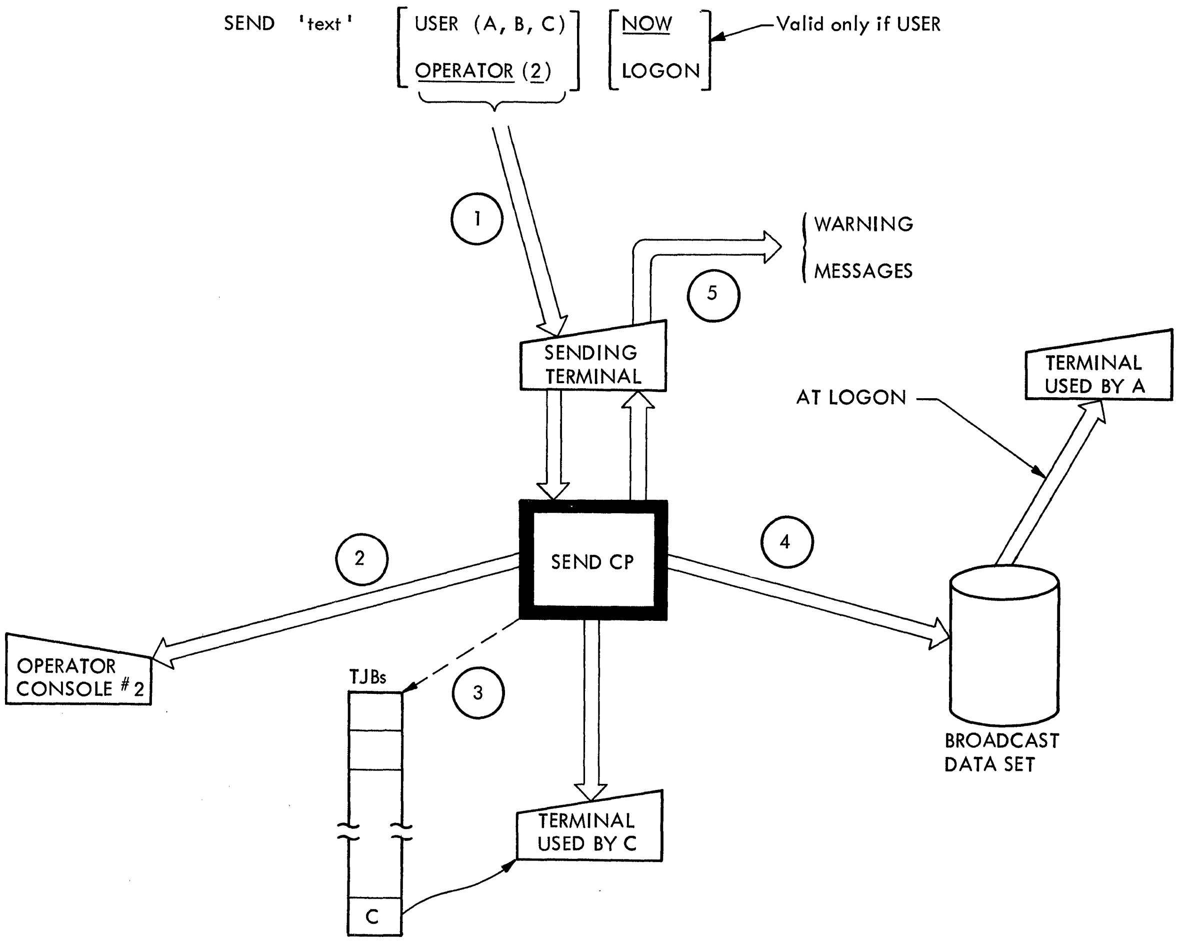
Environment

If the user is not logged on, the message can be written in the Broadcast Data Set, a BDAM data set which contains the USERID of every authorized user and space for messages. The message is chained to the USERID of its intended receiver. One function of Logon is the listing of such messages at the user's terminal. The user may also receive such messages by using the LISTBC Command.

In addition to sending the message to an operator console, to a receiving terminal, or to the Broadcast Data Set, the processor also sends warning messages to the sending terminal. These warning messages tell the sender of failure or other unexpected conditions encountered during processing.

The physical structure of the program (load modules and control sections) is given in the Routine Directory in Section 3.

Figure 1-1 shows the format of the command and summarizes the message flow.



1 The command is entered.

IF OPERATOR IS SPECIFIED

2 The message is sent to the given console number.

IF USER IS SPECIFIED

3 The TJB array is searched on USERID. The TJB in which a user's ID is found points to the terminal being presently used by him. The message is sent to that terminal.

4 If a user is not logged on his ID will not be found in a TJB. The message is sent to the Broadcast Data Set where it is chained from his USERID. When he starts to use any terminal in the system, the message is sent to him as a part of the function of LOGON.

5 Warning messages are sent to the sending terminal. For instance, if USERID B is not found in the Broadcast Data Set, the "unauthorized-user" message is sent.

Figure 1-1. SEND COMMAND MESSAGE FLOW

Section 2. Method of Operation

This section describes the logic of the USER SEND command processor. It emphasizes the flow of data and control information through buffers and tables, and contains detailed functional descriptions of the operation.

Program logic information for the TMP and the TSO service routines used is contained in the Terminal Monitor Program and Service Routines Program Logic Manual, GY28-6770.

Summary of SEND Processing

The Send processor is a message delivery program. It is driven by the parameters used on the command, and by the operating status of the receiving user.

Depending upon command parameters, messages are delivered to a system operator or a terminal user. Depending upon the operating status of a receiving terminal user, messages may be: delivered immediately, stored for future delivery, or not delivered. In the latter case a warning message is sent to the sending terminal.

A further functional summary and a legend of the symbols used on the diagrams are given on Diagram 2-1.

Diagrams 2-2 through 2-5 give information on the implementation of the functions.

Method of Operation Diagrams

NOTE: The method of operation diagrams are in the back of this PLM.

Section 3. Program Organization

This section describes the routines of the Send processor and the relationships among them. Figure 3-1 gives hierarchy information. The remainder of this section consists of module descriptions and flowcharts.

Hierarchy

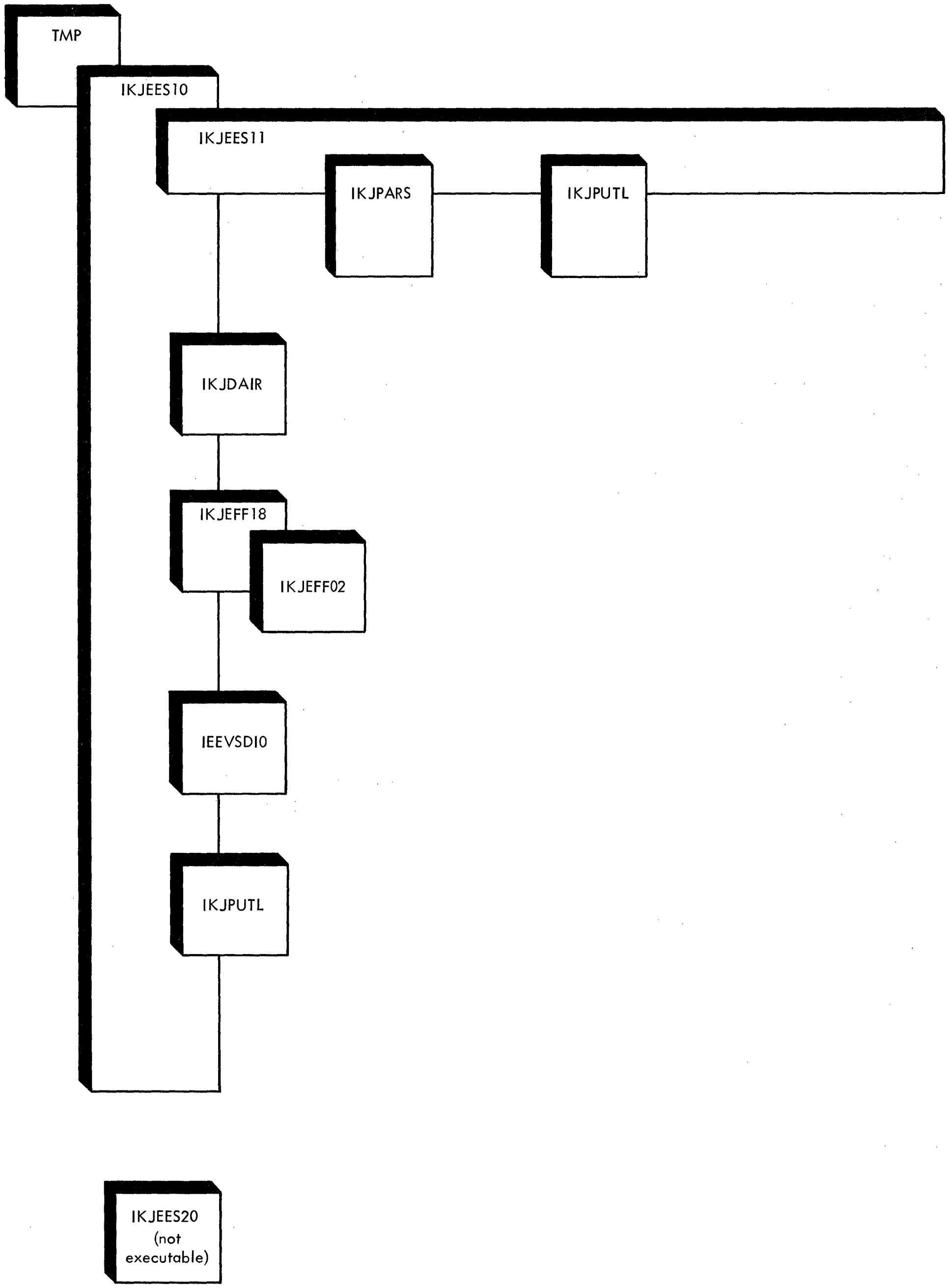


Figure 3-1. Hierarchy of the SEND Command Processor

IEEVSDIO—I/O Module

Entry Point	IEEVSDIO -- called by IKJEES10 Upon entry: • Reg 1 points to SDIOPARM, a 32 byte area containing information necessary for I/O operations. • Reg 14 contains the return address in IKJEES10. • Reg 15 contains the entry point address.
Operation	IEEVSDIO performs five functions for the Send command processor. The function requested is indicated by the first two bytes of the parameter list (PARM1 and PARM2) as follows: • PARM1=X'80' -- open broadcast data set. • PARM1=X'40' -- read the record at the relative block address (RBA) pointed to by PARMRBA in the parameter list. • PARM2=X'80' -- write the record from the address in PARMIOAD to the RBA pointed to by PARMRBA. • PARM2=X'40' -- write the record from the address in PARMIOAD to the first free space (KEY=FF) in the broadcast data set. • PARM2=X'20' -- close the broadcast data set. Before Send calls this module it allocates the broadcast data set (using IKJDAIR) and puts the DD name in PARMDDNM in the parameter list. This module performs other functions when used by Acct/Broadcast Data Set Interface (IKJEES40).
Data Areas Used	• DCB -- for broadcast data set • DECB -- data event control block • ECB -- to test exception code bit settings after I/O operation
Macros Issued	Getmain Freemain Read Write Check Open Close

IEEVSDIO -- I/O Module (Cont.)

Exits	Error -- None, this routine is covered by the TMP STAE/ STAI environment. Normal -- Branch on register 14 to caller
Registers	• R3 -- address of 129 byte read/write area • R4 -- address of RBA (relative block address) of block to be written or read • R6 -- loop control • R7 -- address of IOPARML (parameters passed by IKJEES10 to this routine) • R10 -- base register for DCB DSECT • R11 -- base register for CSECT • R12 -- base register for Data Area DSECT • R13 -- address of this routine save area • R14 -- return address of calling routine • R15 -- return code

IKJEES10—Initialization, Control, Cleanup

Entry Point	IKJEES10 -- attached by TMP. At entry Register 1 points to the CPPL which contains: • Command Buffer address • User Profile Table address • Protected Step Control Block address • Environment Control Table address Register 13 points to the register save area Register 14 contains the return address Register 15 points to entry point -- IKJEES10
Operation	The Send command processor is used by a sending terminal to deliver messages to three places: • Operator consoles • Terminals -- for logged on users • Broadcast data set -- for users not logged on In addition it sends warning messages to the sending terminal. This routine builds a common data area, then passes control to IKJEES11, which sends messages to operator consoles and terminals. Control is returned to this routine which writes messages to the Broadcast data set using IKJDAIR and IEEVSDIO, and sends warning messages to the sending terminal using the IKJPUTL service routine. The order of operation is: 1. Receive control from TMP. 2. Set up common data area for IKJEES11. 3. Call IKJEES11 to syntax check the send command (via Parse) and send messages to operator consoles and user terminals. 4. On return from IKJEES11, if 'LOGON' is specified: A. Call DAIR to allocate broadcast data set. B. Call IEEVSDIO to open broadcast data set and read user directory. C. Add user mail message record(s) to data set. D. Close the broadcast data set. 5. Issue warning message to sender for any abnormal condition encountered in IKJEES10 or IKJEES11. 6. Return to TMP.

IKJEES10 -- Initialization, Control, Cleanup (Cont.)

Data Areas Used	• Command Buffer -- command parameters from TMP. • COMMON -- common data area for this routine and IKJEES11. Built and partly filled by this routine. • CPPL -- parameter list from TMP. • DAPL -- parameter list for IKJDAIR. • DAPB08 -- DAIR parameter block, pointed to by DAPL, for allocation (code '08') of broadcast data set. • PSCB -- protected step control block, contains user ID of sender. • IOPL -- parameter list for PUTLINE. • IKJEES20 -- warning message module.
Routines Called	IKJDAIR -- via LINK, to interface with dynamic allocation and obtain the ddname for the broadcast data set. IKJPUTL -- via PUTLINE macro, to issue the TPUT necessary to send warning messages to the sending terminal. IEEVSDIO -- via CALL, to do I/O operations involving the Broadcast data set. IKJEES11 -- via CALL, to send messages to operator consoles and user terminals. IKJEFF18 -- via LOAD and CALL, DAIR error analysis.
Exits	Error -- none, this processor is covered by the TMP STAE/STAI environment. Normal -- Control returns to TMP.
Registers	R1 -- pointer parameter lists R4 -- loop control R13 -- address of this routines save area R14 -- return address R15 -- return code: 0 is successful completion

IKJEES11—Operator Console and Terminal Messages

Entry Points	IKJEES11 -- called by IKJEES10, at entry: Register 1 points to COMMON Register 13 points to the register save area Register 14 contains the return address in IKJEES10 Register 15 points to IKJEES11, the entry point
Operation	This routine is called by IKJEES10 to send the message to the operator console or to terminals. The order of operation is: 1. Link to PARSE to syntax check the command parameters and build the PDL (parameter descriptor list) which specifies the operation called for by the command parameters. 2. If operator is specified, send the message to the specified operator console via the WTO macro. 3. If user is specified, send the message to logged on terminal users specified by the user ID list. The TPUT macro is used. 4. Return to IKJEES10 to complete processing and to send any required warning messages.
Data Areas	• Command Buffer -- has command name and parameters. • Common -- interface between IKJEES10 and this routine • CVT -- contains pointer to Time Sharing CVT • TSCVT -- Time Sharing CVT, contains pointer to TJB. • TJB -- Time Sharing Job Block. One is allocated to each logged on user; it contains his user ID and information required to send a message to his terminal. • PSCB -- Protected Step Control Block contains the senders user ID.
Routines Called	IKJPARS -- via LINK, to syntax check command parameter and build the PDL parameter descriptor list. TPUT macro -- to send the message to the terminal. WTO macro -- to send the message to the operator console.
Exits	Error -- None, this processor is covered by the TMP STAE/STAI environment. Normal -- control returns to the caller.
Registers	R1 -- address of passed parameters R4 -- index into USERID table R6 -- address of PDL (Parameter Descriptor List) R14 -- return address

IKJEFF18—Common DAIR Failure Message Routine

Entry Point	IKJEFF18 -- at entry register 1 points to a list of pointers to: • The input parameter list used when the call to DAIR was made (the DAPL), including the DAIR request parameter block. (DAPB08) • A full word containing the DAIR return code from register 15. • A full word containing zeroes. • A half word containing the caller number (1 for SEND).
Operation	IKJEFF18 diagnoses DAIR allocation and unallocation failures and puts out appropriate messages to the sending terminal. Input to this routine consists essentially of the input to and the output from DAIR. Thus if the return code from DAIR is non-zero, IKJEFF18 may be called to diagnose the error. The sequence of operation is: 1. The caller's situation is identified (what operation was being performed). 2. A table relating return codes to their meanings for the caller's situation is used. From this table a message ID for an appropriate message is obtained. 3. The message ID is passed to IKJEFF02 which send the message to the terminal via PUTLINE.
Data Areas Used	CPPL DAPL DAPB08
Routines Called	IKJEFF02 -- via LOAD and CALL, to send messages to a terminal.
Exits	Both normal and error return to caller with register 15 set to zero.
Registers	Usage of Registers 1, 13, 14 and 15 is standard.

IKJEES20—Message Module

Entry Point	None.
	This module contains the four warning messages issued by Send. The module is not executable, but it must be included in the load module IKJEES10.

Chart AA. IKJEES10. I/O Control and Warning Messages (1 of 4)

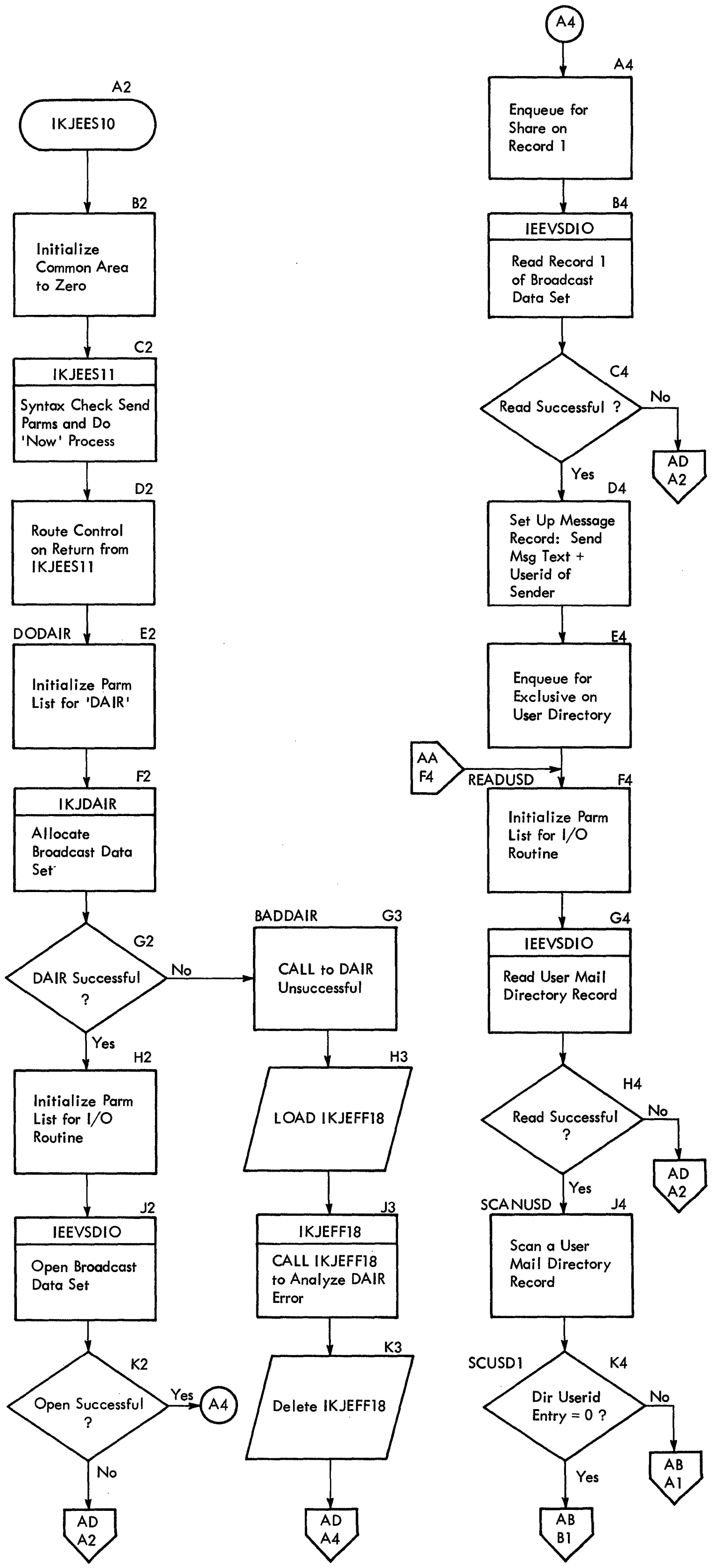

A2
IKJEES10
B2
Initialize Common Area to Zero
C2
IKJEES11
Syntax Check Send Parms and Do 'Now' Process
D2
Route Control on Return from IKJEES11
DODAIR
E2
Initialize Parm List for 'DAIR'
F2
IKJDAIR
Allocate Broadcast Data Set
G2
DAIR Successful ?
No
Yes
BADDAIR
G3
CALL to DAIR Unsuccessful
H2
Initialize Parm List for I/O Routine
H3
LOAD IKJEFF18
J2
IEEVSDIO
Open Broadcast Data Set
J3
IKJEFF18
CALL IKJEFF18 to Analyze DAIR Error
K2
Open Successful ?
Yes
A4
No
AD A2
K3
Delete IKJEFF18
AD A4
A4
A4
Enqueue for Share on Record 1
B4
IEEVSDIO
Read Record 1 of Broadcast Data Set
C4
Read Successful ?
No
AD A2
Yes
D4
Set Up Message Record: Send Msg Text + Userid of Sender
E4
Enqueue for Exclusive on User Directory
AA F4
READUSD
F4
Initialize Parm List for I/O Routine
G4
IEEVSDIO
Read User Mail Directory Record
H4
Read Successful ?
No
AD A2
Yes
SCANUSD
J4
Scan a User Mail Directory Record
SCUSD1
K4
Dir Userid Entry = 0 ?
No
AB A1
Yes
AB B1

Chart AB. IKJEES10. I/O Control and Warning Messages (2 of 4)

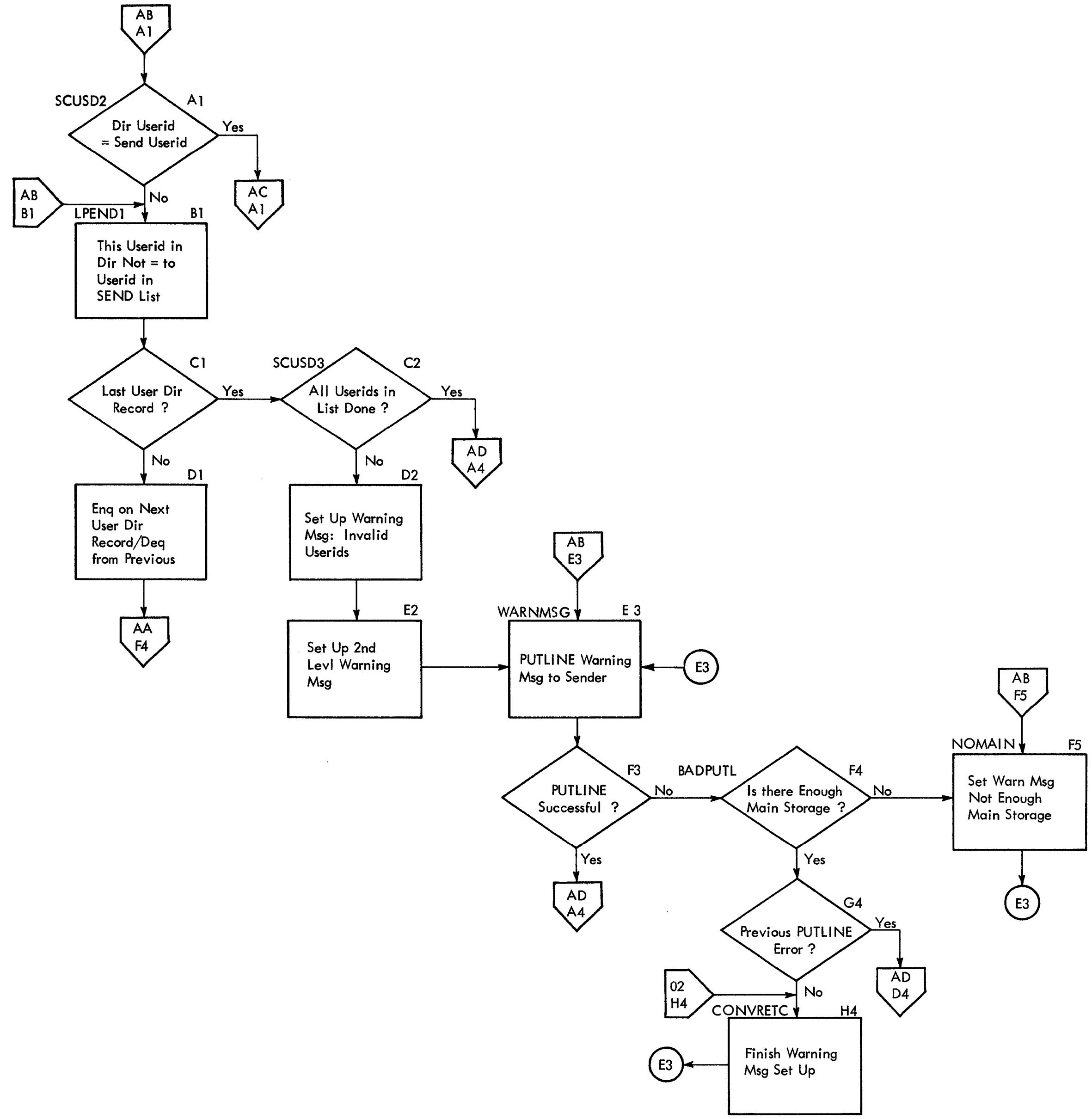

AB
A1
SCUSD2
A1
Dir Userid = Send Userid
Yes
AC
A1
No
AB
B1
LPEND1
B1
This Userid in Dir Not = to Userid in SEND List
C1
Last User Dir Record ?
Yes
SCUSD3
C2
All Userids in List Done ?
Yes
AD
A4
No
D1
Enq on Next User Dir Record/Deq from Previous
AA
F4
No
D2
Set Up Warning Msg: Invalid Userids
E2
Set Up 2nd Levl Warning Msg
AB
E3
WARNMSG
E 3
PUTLINE Warning Msg to Sender
E3
AB
F5
NOMAIN
F5
Set Warn Msg Not Enough Main Storage
E3
F3
PUTLINE Successful ?
No
BADPUTL
F4
Is there Enough Main Storage ?
No
Yes
AD
A4
Yes
G4
Previous PUTLINE Error ?
Yes
AD
D4
02
H4
No
CONVRETC
H4
Finish Warning Msg Set Up
E3

Chart AC. IKJEES10. I/O Control and Warning Messages (3 of 4)

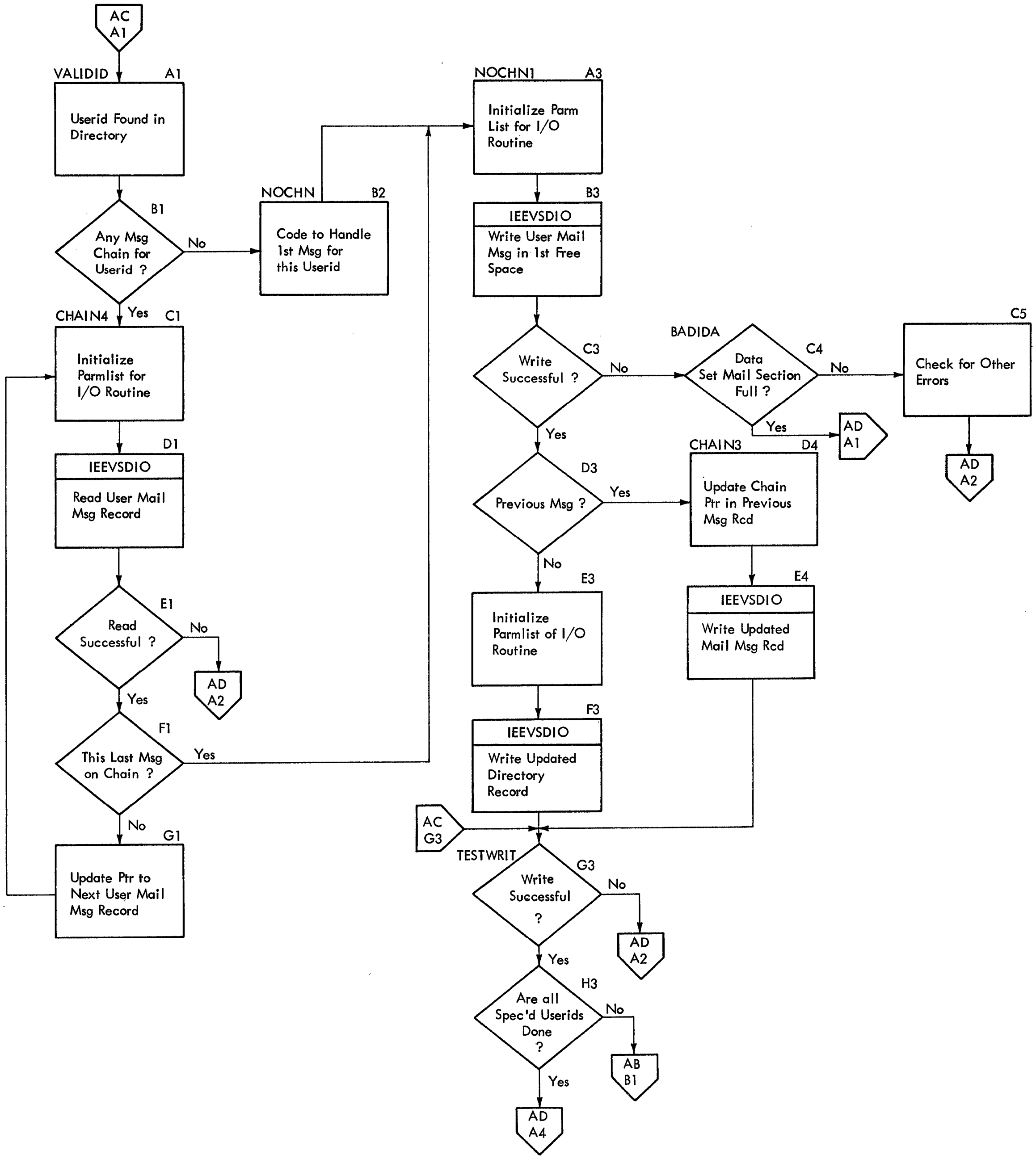

AC A1
VALIDID A1
Userid Found in Directory
B1
Any Msg Chain for Userid ?
No
NOCHN B2
Code to Handle 1st Msg for this Userid
CHAIN4 Yes C1
Initialize Parmlist for I/O Routine
D1
IEEVSDIO
Read User Mail Msg Record
E1
Read Successful ?
No
AD A2
Yes
F1
This Last Msg on Chain ?
Yes
No
G1
Update Ptr to Next User Mail Msg Record
NOCHN1 A3
Initialize Parm List for I/O Routine
B3
IEEVSDIO
Write User Mail Msg in 1st Free Space
C3
Write Successful ?
No
BADIDA
C4
Data Set Mail Section Full ?
No
C5
Check for Other Errors
AD A2
Yes
AD A1
Yes
D3
Previous Msg ?
Yes
CHAIN3 D4
Update Chain Ptr in Previous Msg Rcd
No
E3
Initialize Parmlist of I/O Routine
E4
IEEVSDIO
Write Updated Mail Msg Rcd
F3
IEEVSDIO
Write Updated Directory Record
AC G3
TESTWRIT G3
Write Successful ?
No
AD A2
Yes
H3
Are all Spec'd Userids Done ?
No
AB B1
Yes
AD A4

Chart AD. IKJEES10. I/O Control and Warning Messages (4 of 4)

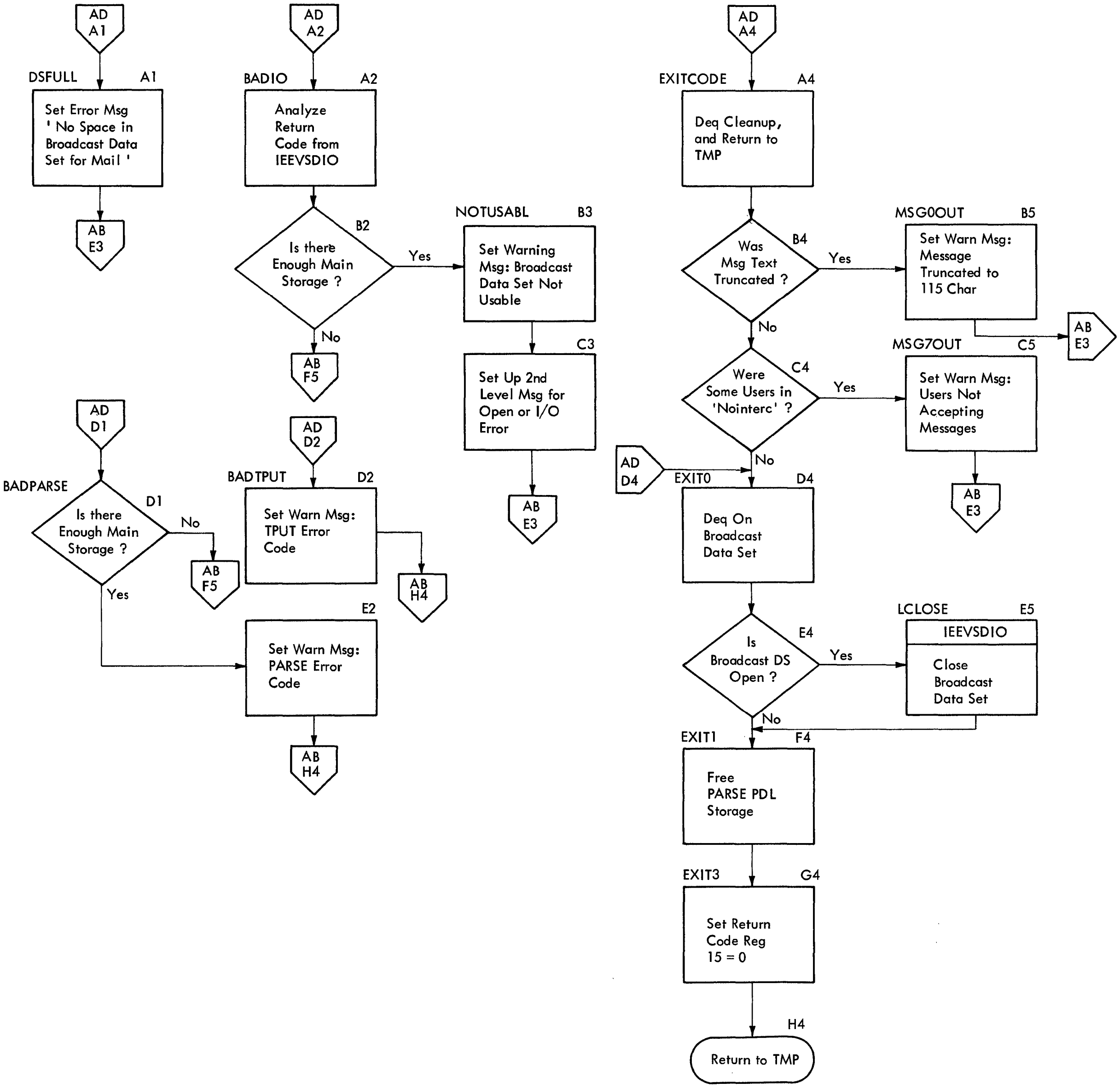

AD A1
DSFULL A1
Set Error Msg ' No Space in Broadcast Data Set for Mail '
AB E3
AD A2
BADIO A2
Analyze Return Code from IEEVSDIO
B2
Is there Enough Main Storage ?
Yes
No
AB F5
NOTUSABL B3
Set Warning Msg: Broadcast Data Set Not Usable
C3
Set Up 2nd Level Msg for Open or I/O Error
AB E3
AD D1
BADPARSE D1
Is there Enough Main Storage ?
No
AB F5
Yes
AD D2
BADTPUT D2
Set Warn Msg: TPUT Error Code
AB H4
E2
Set Warn Msg: PARSE Error Code
AB H4
AD A4
EXITCODE A4
Deq Cleanup, and Return to TMP
B4
Was Msg Text Truncated ?
Yes
MSGOOUT B5
Set Warn Msg: Message Truncated to 115 Char
AB E3
No
C4
Were Some Users in 'Nointerc' ?
Yes
MSG7OUT C5
Set Warn Msg: Users Not Accepting Messages
AB E3
No
AD D4
EXIT0 D4
Deq On Broadcast Data Set
E4
Is Broadcast DS Open ?
Yes
LCLOSE E5
IEEVSDIO
Close Broadcast Data Set
No
EXIT1 F4
Free PARSE PDL Storage
EXIT3 G4
Set Return Code Reg 15 = 0
H4
Return to TMP

Chart BA. IKJEES11. Operator and User Messages (1 of 3)

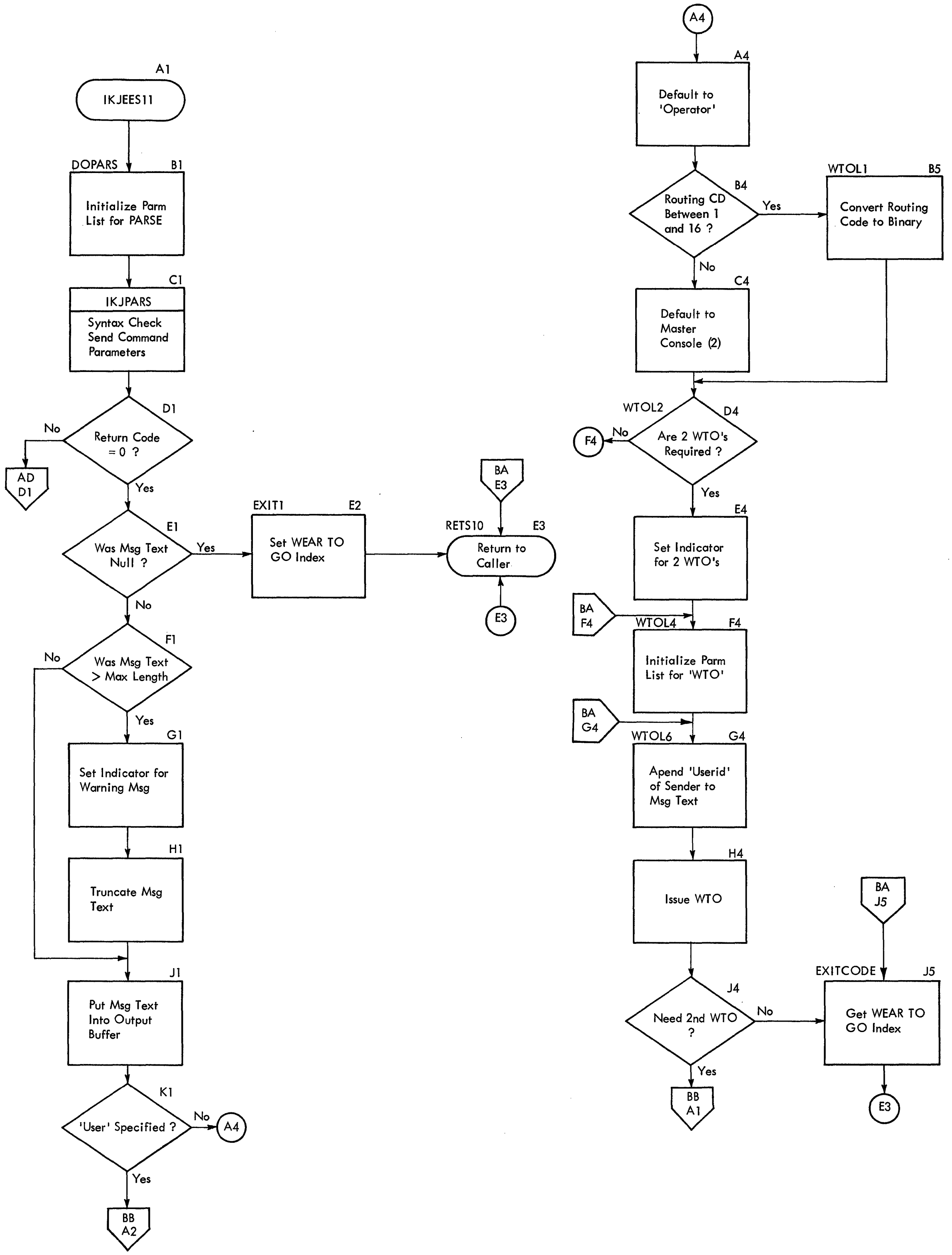
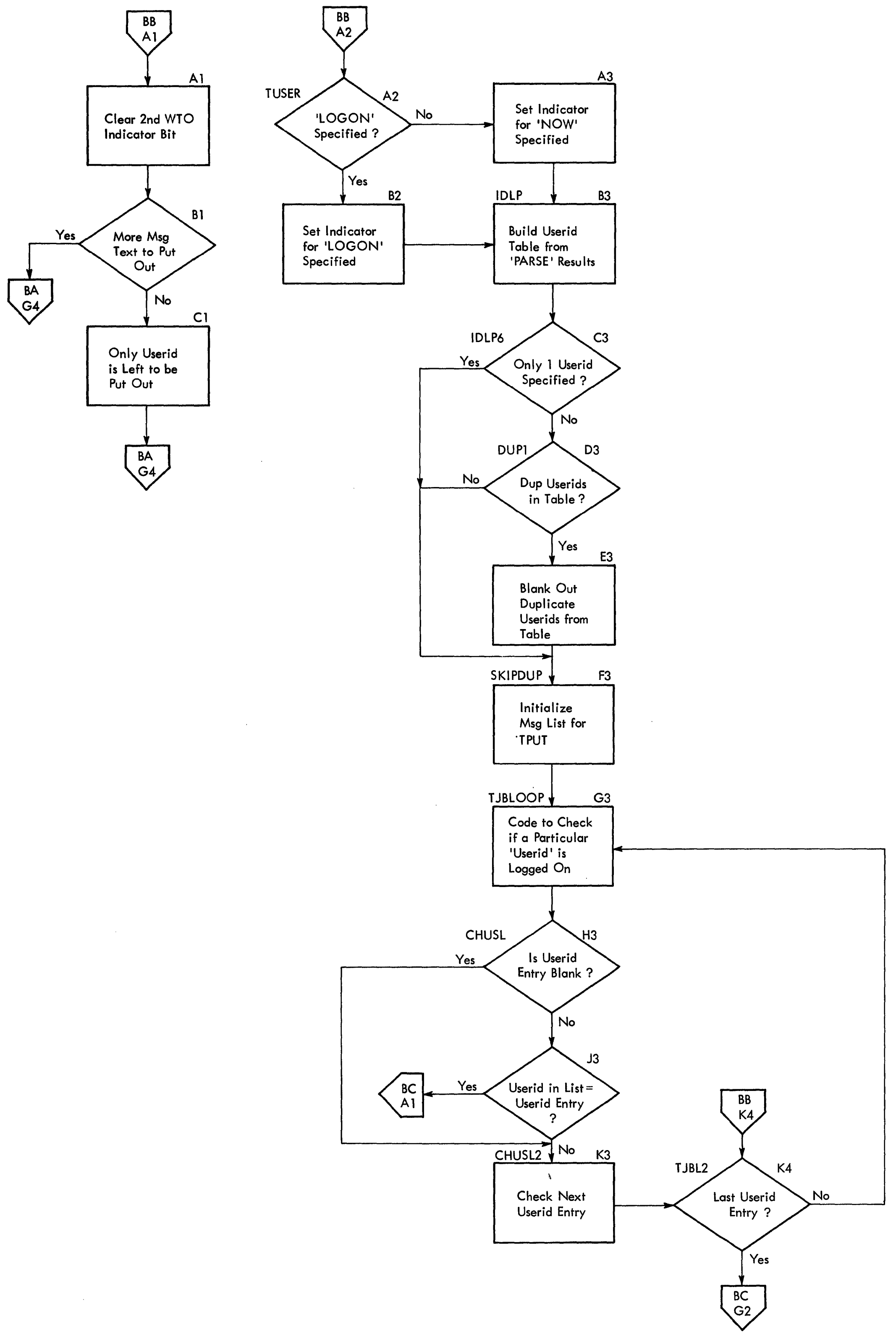


Chart BB. IKJEES11. Operator and User Messages (2 of 3)



BB A1
A1
Clear 2nd WTO Indicator Bit
B1
More Msg Text to Put Out
Yes
BA G4
No
C1
Only Userid is Left to be Put Out
BA G4
BB A2
TUSER
A2
'LOGON' Specified ?
No
Yes
A3
Set Indicator for 'NOW' Specified
B2
Set Indicator for 'LOGON' Specified
IDLP
B3
Build Userid Table from 'PARSE' Results
IDLP6
C3
Only 1 Userid Specified ?
Yes
No
DUP1
D3
Dup Userids in Table ?
No
Yes
E3
Blank Out Duplicate Userids from Table
SKIPDUP
F3
Initialize Msg List for TPUT
TJBLOOP
G3
Code to Check if a Particular 'Userid' is Logged On
CHUSL
H3
Is Userid Entry Blank ?
Yes
No
J3
Userid in List = Userid Entry ?
Yes
BC A1
No
CHUSL2
K3
Check Next Userid Entry
BB K4
TJBL2
K4
Last Userid Entry ?
No
Yes
BC G2

Chart BC. IKJEES11. Operator and User Messages (3 of 3)

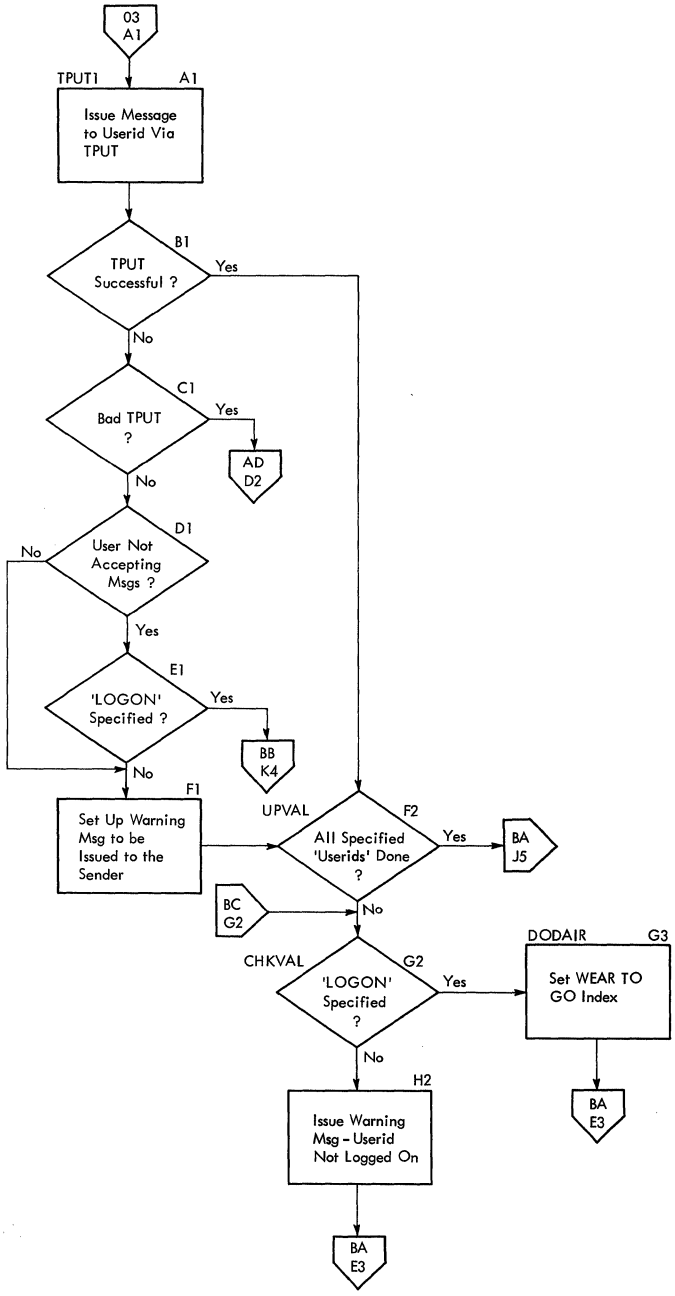


Chart CA. IEEVSDIO. SEND I/O Module (1 of 4)

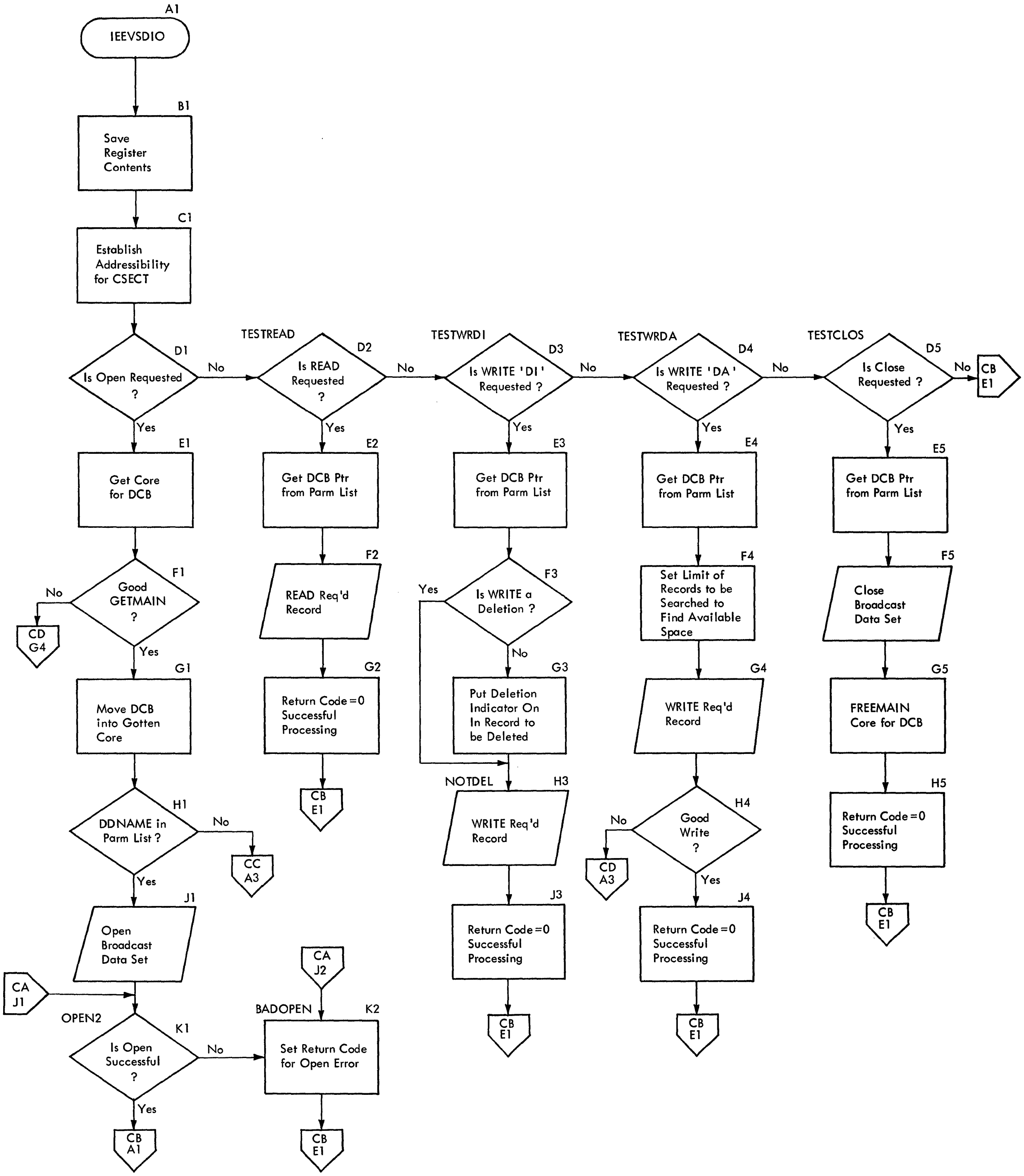


Chart CB. IEEVSDIO. SEND I/O Module (2 of 4)

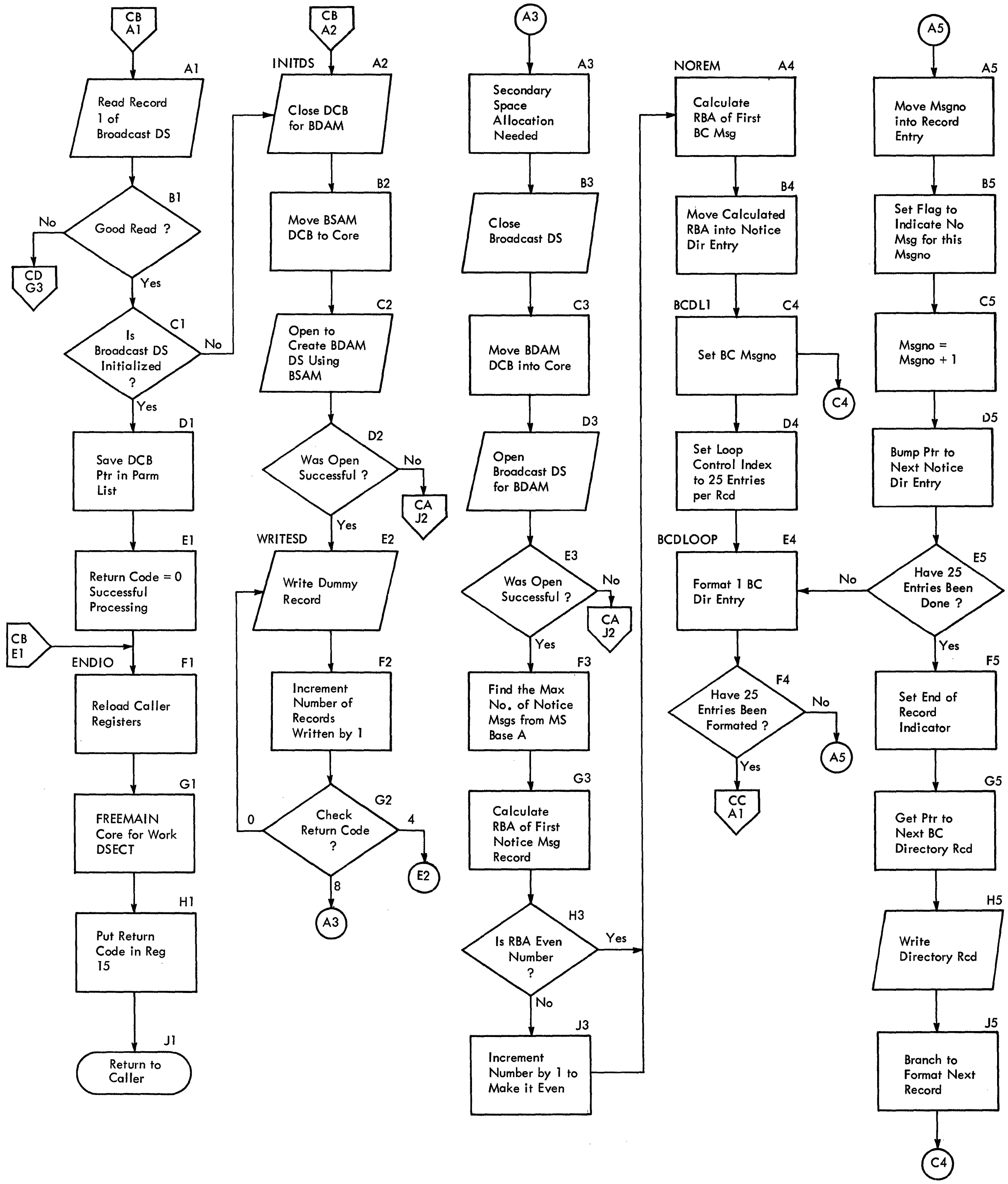

CB A1
A1
Read Record 1 of Broadcast DS
B1
Good Read ?
No
CD G3
Yes
C1
Is Broadcast DS Initialized ?
No
Yes
D1
Save DCB Ptr in Parm List
E1
Return Code = 0 Successful Processing
CB E1
ENDIO
F1
Reload Caller Registers
G1
FREEMAIN Core for Work DSECT
H1
Put Return Code in Reg 15
J1
Return to Caller
CB A2
INITDS
A2
Close DCB for BDAM
B2
Move BSAM DCB to Core
C2
Open to Create BDAM DS Using BSAM
D2
Was Open Successful ?
No
CA J2
Yes
WRITESD
E2
Write Dummy Record
F2
Increment Number of Records Written by 1
G2
Check Return Code ?
0
4
E2
8
A3
A3
A3
Secondary Space Allocation Needed
B3
Close Broadcast DS
C3
Move BDAM DCB into Core
D3
Open Broadcast DS for BDAM
E3
Was Open Successful ?
No
CA J2
Yes
F3
Find the Max No. of Notice Msgs from MS Base A
G3
Calculate RBA of First Notice Msg Record
H3
Is RBA Even Number ?
Yes
No
J3
Increment Number by 1 to Make it Even
NOREM
A4
Calculate RBA of First BC Msg
B4
Move Calculated RBA into Notice Dir Entry
BCDL1
C4
Set BC Msgno
C4
D4
Set Loop Control Index to 25 Entries per Rcd
BCDLOOP
E4
Format 1 BC Dir Entry
F4
Have 25 Entries Been Formated ?
No
A5
Yes
CC A1
A5
A5
Move Msgno into Record Entry
B5
Set Flag to Indicate No Msg for this Msgno
C5
Msgno = Msgno + 1
D5
Bump Ptr to Next Notice Dir Entry
E5
Have 25 Entries Been Done ?
No
Yes
F5
Set End of Record Indicator
G5
Get Ptr to Next BC Directory Rcd
H5
Write Directory Rcd
J5
Branch to Format Next Record
C4

Chart CC. IEEVSDIO. SEND I/O Module (3 of 4)

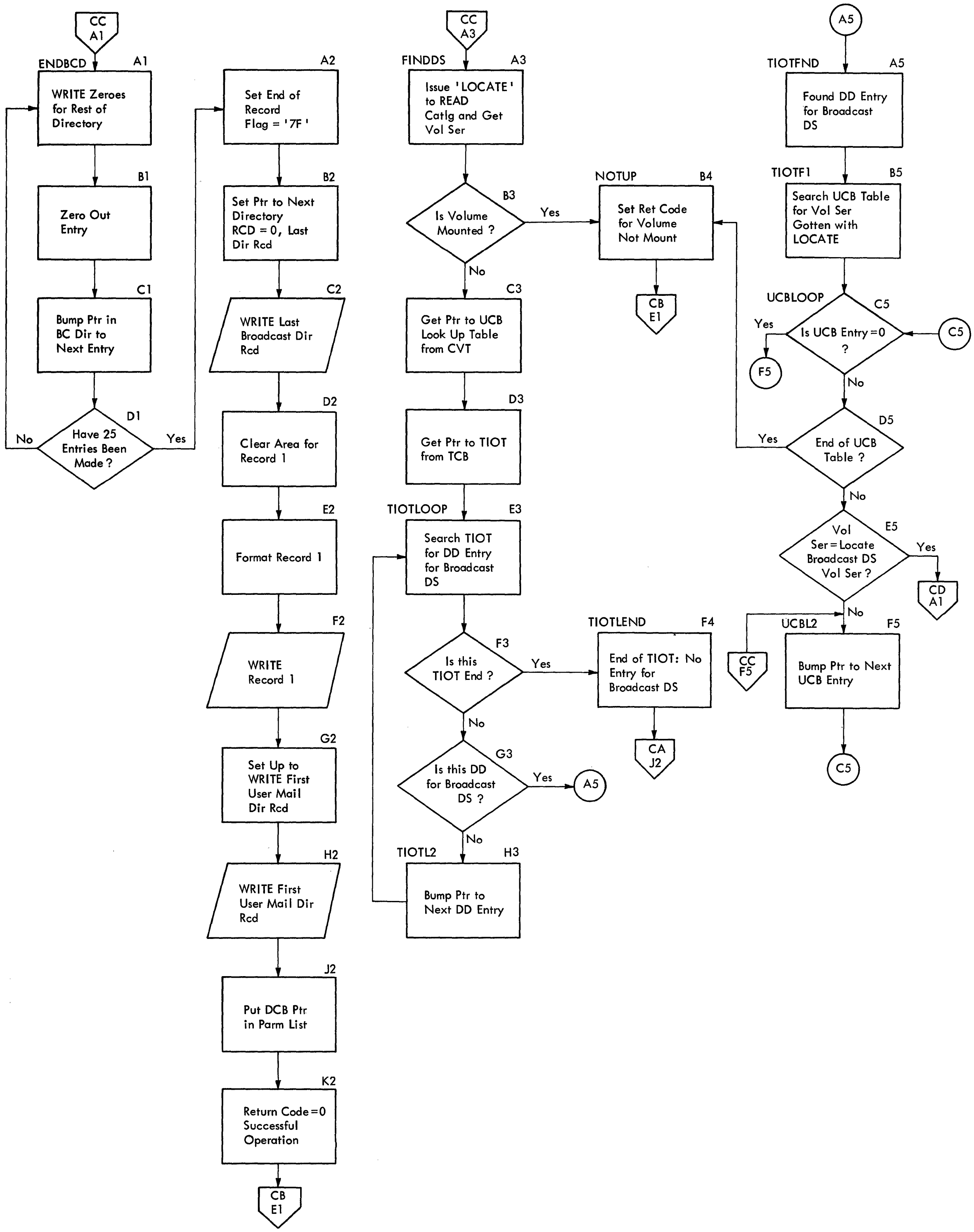
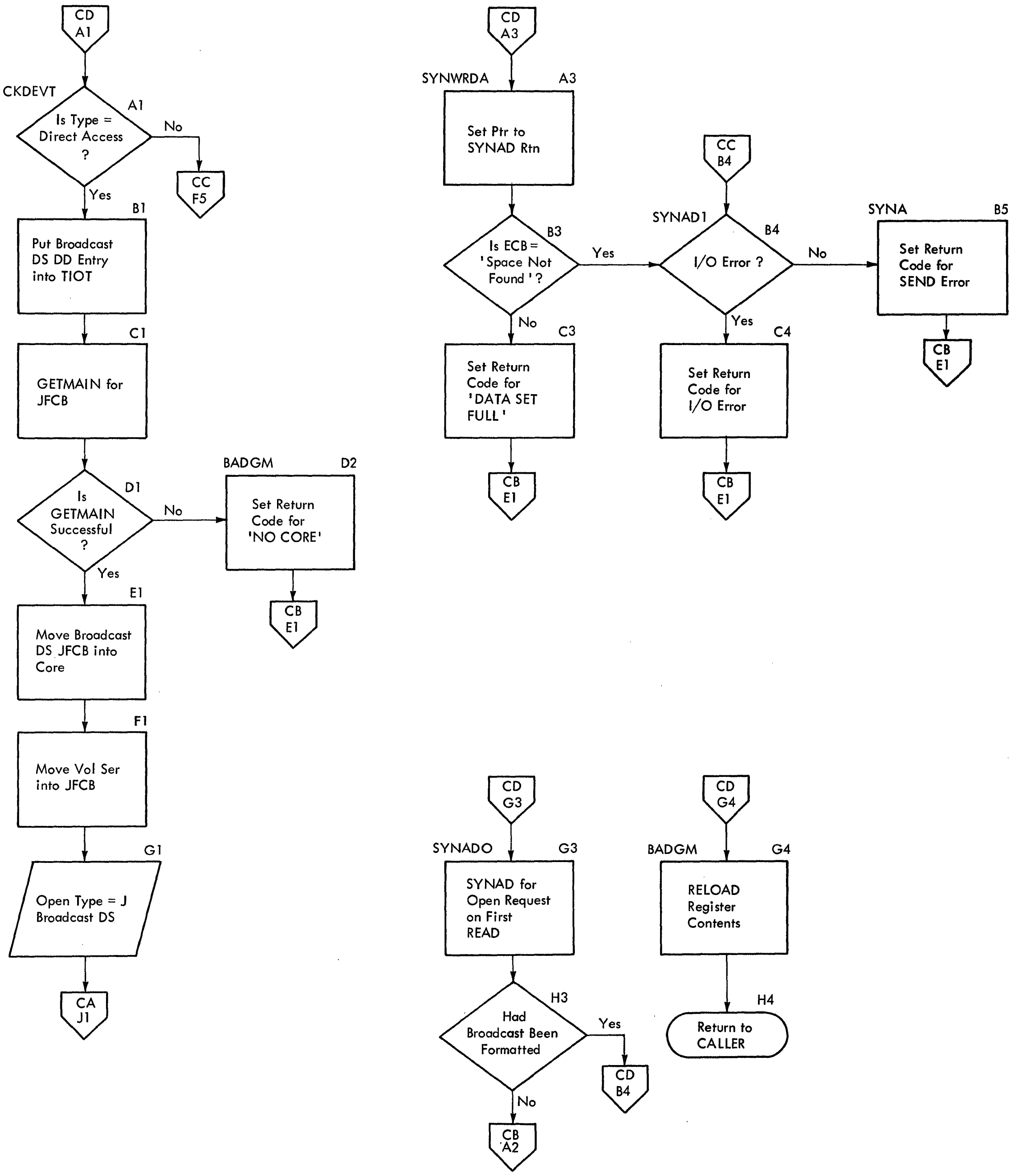

CC A1
ENDBCD
A1
WRITE Zeroes for Rest of Directory
B1
Zero Out Entry
C1
Bump Ptr in BC Dir to Next Entry
D1
Have 25 Entries Been Made ?
No
Yes
A2
Set End of Record Flag = '7F'
B2
Set Ptr to Next Directory RCD = 0, Last Dir Rcd
C2
WRITE Last Broadcast Dir Rcd
D2
Clear Area for Record 1
E2
Format Record 1
F2
WRITE Record 1
G2
Set Up to WRITE First User Mail Dir Rcd
H2
WRITE First User Mail Dir Rcd
J2
Put DCB Ptr in Parm List
K2
Return Code=0 Successful Operation
CB E1
CC A3
FINDDS
A3
Issue 'LOCATE' to READ Catlg and Get Vol Ser
B3
Is Volume Mounted ?
Yes
No
C3
Get Ptr to UCB Look Up Table from CVT
D3
Get Ptr to TIOT from TCB
TIOTLOOP
E3
Search TIOT for DD Entry for Broadcast DS
F3
Is this TIOT End ?
Yes
No
G3
Is this DD for Broadcast DS ?
Yes
A5
No
TIOTL2
H3
Bump Ptr to Next DD Entry
NOTUP
B4
Set Ret Code for Volume Not Mount
CB E1
TIOTLEND
F4
End of TIOT: No Entry for Broadcast DS
CA J2
A5
TIOTFND
A5
Found DD Entry for Broadcast DS
TIOTF1
B5
Search UCB Table for Vol Ser Gotten with LOCATE
UCBLOOP
C5
Is UCB Entry=0 ?
Yes
F5
C5
No
D5
End of UCB Table ?
Yes
No
E5
Vol Ser=Locate Broadcast DS Vol Ser ?
Yes
CD A1
No
CC F5
UCBL2
F5
Bump Ptr to Next UCB Entry
C5

Chart CD. IEEVSDIO. SEND I/O Module (4 of 4)


CD A1
CKDEVT A1
Is Type = Direct Access ?
No
CC F5
Yes
B1
Put Broadcast DS DD Entry into TIOT
C1
GETMAIN for JFCB
D1
Is GETMAIN Successful ?
No
BADGM D2
Set Return Code for 'NO CORE'
CB E1
Yes
E1
Move Broadcast DS JFCB into Core
F1
Move Vol Ser into JFCB
G1
Open Type = J Broadcast DS
CA J1
CD A3
SYNWRDA A3
Set Ptr to SYNAD Rtn
B3
Is ECB = 'Space Not Found'?
Yes
No
C3
Set Return Code for 'DATA SET FULL'
CB E1
CC B4
SYNAD1 B4
I/O Error ?
No
Yes
C4
Set Return Code for I/O Error
CB E1
SYNA B5
Set Return Code for SEND Error
CB E1
CD G3
SYNADO G3
SYNAD for Open Request on First READ
H3
Had Broadcast Been Formatted
Yes
CD B4
No
CB A2
CD G4
BADGM G4
RELOAD Register Contents
H4
Return to CALLER

Section 4. Directory

This section contains a routine directory and a data area directory both organized alphabetically.

Routine Directory

Entry name,	Load	Control	Cross reference	
Routine Name	Module	Section	MO	FC
IEEVSDIO	IKJEES10	IEEVSDIO	2-4	CA,CB,CC,CD
IKJEES10	IKJEES10	IKJEES10	2-2	AA,AB,AC,AD
IKJEES11	IKJEES10	IKJEFF11	2-2,2-3	BA,BB,BC
IKJEES20	IKJEES10		2-5	AB
IKJEFF02	IKJEFF02	IKJEFF02	2-5	AB
IKJEFF18	IKJEFF18	IKJEFF18	2-5	AA

Data Area Directory

Data Area	MO	FC
COMBUF	2-2	AA
COMMON	2-2 thru 2-5	AA
CPPL	2-2 thru 2-5	AA
DAPB08	2-4	AA
DAPL	2-4	AA
IOPL	2-5	AB
PPL	2-2	BA
PSCB	2-2,2-4	AA
SDIOPARM	2-4	AA
SENDPDL	2-2	BA
TJB	2-2	BB
TSCVT	2-3	BB
WEARTOGO		AA
WTOLIST	2-2	BA

Section 4. Data Areas

This section contains descriptions of the major data areas in the Send command processor.

Unnamed (Command Buffer)

Created by: TMP

Used by: IKJEES10

Contents: The TSO Command.

Disp Dec (Hex)	Field	Size (Bytes)	Use/Contents
0 (0)	CBLNG	2	Length of entire buffer.
2 (2)	CBOFF	2	Offset at which IKJSCAN stopped.
4 (4)	CBCOM	100	Command name and parameter.

COMMON

Created by: IKJEES10

Used by: IKJEES10 and IKJEES11

Use: Common data area for IKJEES10 and IKJEES11

Disp	Field	Subfield	Size	Use/Contents
Dec (Hex)			(Bytes)	
0 (0)	FLAGS		3	If bit=1 meaning is:
		FLOGON	1bit	LOGON specified.
		FOPEN	1bit	Broadcast Data Set is open.
		FPARS	1bit	Parse PDL space must be freed.
		FMSG7	1bit	"Not accepting" message must be sent.
		FWT02		Second write to operator is needed.
		*	1bit	Reserved.
		FNOW	1bit	'NOW' specifid or by default.
		FMSGO	1bit	'Message text truncated' message must be sent.
		FENQ1	1bit	Enqueued on record 1.
			1bit	Reserved.
		FENQUSD	1bit	Enqueued on user mail directory.
		FCHAIN	1bit	Message chain for this user ID.
		FBADPUTL	1bit	PUTLINE error.
3 (3)	COMROUT		1	Index into where to go table on return from IKJEES11.
4 (4)	VALCNT		1	Number of valid user IDs processed. (VALCNT=IDCNT, all are done)
8 (8)	COMCPPL		4	Pointer to CPPL (parameters from TMP)
12 (C)	CPDLMSGT		4	Pointer to PDE for send message text (returned by PARSE).
16 (10)	PDLPTR		4	Pointer to PDL supplied by parse, passed to IKJEES10 for IKJRLSA.

COMMON (Cont.)

Disp Dec (Hex)	Field	Size (Bytes)	Use/Contents
20 (14)	IDCNT	1	Number of user IDs specified in send command, equal to number of entries in IDUSRID.
21 (15)	IDTBL	3	Pointer to table of user IDs in IKJEESIO (IDUSRID).
24 (18)	M7IND	1	Index to MSG7 user ID insert.
25 (19)	COMMSG7	3	Pointer to MSG7 user ID insert (MSG7INS, in IKJEESI O)
28 (1C)	COMRETC	2	Non-zero return code from PARSE or TPUT.
30 (1E)	*	2	Reserved.

DAPL

Created by: IKJEES10

Used by: IKJDAIR

Contents: Pointers for SEND - DAIR Interface

Disp Dec (Hex)	Field	Size (Bytes)	Use/Contents
0 (0)	DAPLUPT	4	Pointer to UPT.
4 (4)	DAPLECT	4	Pointer to ECT.
8 (8)	DAPLECB	4	Pointer to Command Processor's ECB.
12 (C)	DAPLPSCB	4	Pointer to PSCB.
16 (10)	DAPLDAPB	4	Pointer to DAIR parameter block.

IOPARML

Created by: IKJEES10

Used by: IEEVSDIO, IKJEES10

Contents: Parmlist for call to IEEVSDIO.

Disp Dec (Hex)	Field	Size (Bytes)	Use/Contents
0 (0)	SDIOPARM	20	Area cleared for each call, for details see "SDIOPARM".
20 (14)	PARMLMCT	4	Limit count for DCB with extended search option.
24 (18)	PARMDCB	4	Address of DCB for Broadcast data set.
28 (1C)	PARMJFCB	4	Address of JFCB for Broadcast data set.

IOPL

Created by: IKJEES10

Used by: IKJPUTL

Contents: Pointers for SEND - PUTLINE Interface.

Disp Dec (Hex)	Field	Size (Bytes)	Use/Contents
0 (0)	IOPLUPT	4	Pointer to User Profile Table.
4 (4)	IOPLECT	4	Pointer to Environment Control Table.
8 (8)	IOPLECB	4	Pointer to Send Event Control Block.
12 (C)	IOPLIOPB	4	Pointer to Putline parameter block.

PPL

Created by: IKJEES11

Used by: IKJPARS

Contents: Pointers for SEND - PARSE Interface.

Disp Dec (Hex)	Field	Size (Bytes)	Use/Contents
0 (0)	PPLUPT	4	Pointer to User Profile Table.
4 (4)	PPLECT	4	Pointer to Environment Control Table.
8 (8)	PPLECB	4	Pointer to Command Processor ECB.
12 (C)	PPLPCL	4	Pointer to parse control list in IKJEES11.
16 (10)	PPLANS	4	Pointer to the field, in which PARSE returns a pointer to the Parameter Description List.
20 (14)	PPLCBUF	4	Pointer to Command Buffer.
24 (18)	PPLUWA	4	Pointers to Work Area for validity check routine.

SDIOPARM

Created by: IKJEES10

Used by: IEEVSDIO, IKJEES10

Contents: Interface for I/O operations or Broadcast data set.

Disp Dec (Hex)	Field	Subfield	Size (Bytes)	Use/Contents
0 (0)	*		1	Operation indicator flags.
		P1OPEN	1bit	Open data set.
		P1READ	1bit	Read a record.
	*		1	More operation indicator flags.
		P2WRDI	1bit	Write by block ID.
		P2WRDAF	1bit	Write type 'DAF' in free space.
		P2CLOSE	1bit	Close data set.
		P2DEL	1bit	This write deletes a record, put PARMFDBK into first byte of data and set key = X'FF'.
		P2JFCB	1bit	Free core for JFCB, not used by User Send Command Processor.
2 (2)	PARMKEY		1	Key for record to be written.
3 (3)	PARMFDBK		1	Feedback from read (position of record on track). Put into first byte of data for delete.
4 (4)	PARMRBA		4	Address of RBA for read/write.
8 (8)	PARMIOAD		4	Address of area for I/O record.
12 (C)	PARMDDNM		8	DDname for broadcast data set.
20 (14)	PARMLMCT		4	Limit count for DCB with extended search option.
24 (18)	PARMDCB		4	Address of DCB for broadcast data set (used by Send command processor).
28 (1C)	PARMJFCB		4	Address of JFCB (not used by SEND CP).

SENDPDL (Parameter Descriptor List)

Created by: IKJPARS

Used by: IKJEES11, IKJEES10

Contents: Driving list for Send Command Processor

Disp	Field	Subfield	Size	Use/Contents
Dec (Hex)			(Bytes)	
0 (0)	*		8	Storage chain information used by IKJPARS.
8 (8)	PDLMSGT		8	
		PDLMPTR	4	Pointer to message text.
12 (C)		PDLMLL	2	Length of message.
14 (E)		PDLMFL	1	Flags.
15 (F)		PDLMRES	1	Reserved.
16 (10)	PDLSDEE		2	Receiver of message 1 = operator 2 = user
18 (12)	PDLWHEN		2	Now/logon parameter 1 = now 2 = logon
20 (14)	PDLLIST		12	USER ID PDE.
		USLUSPTR	4	Pointer to user ID.
24 (18)		USLLL	2	Length of user ID.
26 (1A)		USLFL	1	Flags.
27 (1B)		USLRES	1	Reserved.
28 (1C)		USLCHN	4	Pointer to next USERID PDE X'FF000000' if this is last.

WEARTOGO

Created by: IKJEES10

Used by: IKJEFS10, IKJEES11

Contents: An index into this table set by S11 controls its effective return point in S10.

Disp Dec (Hex)	Field	Size (Bytes)	Use/Contents
0 (0)	EXITCODE	4	Normal exit.
4 (4)	EXIT1	4	No message text, terminate.
8 (8)	EXIT3	4	Not used.
12 (C)	MSG2OUT	4	Send 'users not logged on' message.
16 (10)	BADPARSE	4	Send 'badparse' message.
20 (14)	BADTPUT	4	Send bad TPUT message.
24 (18)	DODAIR	4	TEXT must go to Broadcast data set.

WTOLIST

Created by: IKJEES11

Used by: IKJEES11

Contents: Parameter list for write to operator macro.

Disp Dec (Hex)	Field	Size (Bytes)	Use/Contents
0 (0)	WTOMSGL	2	Length of message text + 4.
2 (2)	WTOMCSF	2	MCS flags.
4 (4)	WTOTEXT	137	Message text.

Section 6. Diagnostic Aids

This section contains a register usage summary and the messages issued by the Send command processor.

Register Summary

All routines in this processor use the following four registers in the conventional ways:

R1 -- parameter list pointer

R13 -- register save area pointer

R14 -- return address

R15 -- return code

Other significant register usages are:

Routine	Reg	Usage
IEEVSDIO	0	Used by Getmain and Freemain.
	3	Address of 129-byte Read/Write area.
	4	Address of relative block address (RBA) of block to be read.
	6	Counter for loop control
	7	Address of parameter list.
	10	Base register for DCB DSECT.
	11	Base register for CSECT.
	12	Base register for data area DSECT.
	13	Address of this routines save area.

Routine	Reg	Usage
IEEVSDIO (Cont.)	14	Return address of calling routine.
	15	Return code.

Routine	Reg	Usage
IKJEES10	1	Parameter list address.
	4	Counter for loop control.
	13	Address of this routine's save area.
	14	Return address.
	15	Return code.
IKJEES11	1	Parameter list address.
	4	Index to IDUSRID (Table of USERIDs to receive message).
	14	Return address.
	15	Return code.

Messages

Dair Failure Messages

These messages are issued by IKJEFF18, the Common Dair Failure Message Routine. Where parenthesis (n) and a number appear in the message, the following are inserted:

(1) SYS1.BROADCAST
(2) ALLOCATED
(3) The name of the failing routine

Message ID	Message Text
IKJ56220I	(1) NOT ALLOCATED, TOO MANY DATA SETS+ USE FREE COMMAND TO FREE UNUSED DATA SETS
IKJ56221I	(1) NOT ALLOCATED, DATA SET NOT ON VOLUME+ CATALOG INFORMATION IINCORRECT
IKJ56222I	(1) NOT ALLOCATED, REQUIRED VOLUME NOT MOUNTED+ VOLUME NOT ON SYSTEM AND CANNOT BE ACCESSED
IKJ56223I	(1) NOT (2), SYSTEM ERROR+ (3) ERROR CODE xxxxx
IKJ56225I	DATA SET -dsname- ALREADY IN USE, TRY LATER+ DATA SET IS ALLOCATED TO ANOTHER JOB OR USED
IKJ56227I	NOT ENOUGH MAIN STORAGE TO EXECUTE COMMAND
IKJ56228I	DATA SET -dsname- NOT IN CATALOG

Send Termination and Warning Messages

These messages are issued by IKJEES10.

Message ID	Message Text
IKJ55071I (warning)	INVALID USERID(S) -userid(s)-+ USERID NOT AUTHORIZED FOR TSO
IKJ55072I (warning)	USER(S) -userid(s)- NOT LOGGED ON, MESSAGE CANCELED
IKJ55074I (termination)	NO SPACE IN BROADCAST DATA SET FOR MAIL
IKJ55075I (termination)	DATA SET SYS1.BROADCAST NOT USABLE+ I/O SYNAD ERROR or CANNOT OPEN DATA SET
IKJ55077I (warning)	USER(S) -userid(s)- NOT ACCEPTING MESSAGES, MESSAGE CANCELED
IKJ55080I (warning)	MESSAGE TRUNCATED TO 115 CHARACTERS
IKJ55081I (termination)	COMMAND SYSTEM ERROR+ PUTLINE ERROR CODE -code- or TPUT ERROR CODE -code- or terminal PARSE ERROR CODE -code-
IKJ55082I (termination)	NOT ENOUGH MAIN STORAGE TO EXECUTE COMMAND

Index

Indexes to program logic manuals are consolidated in the publication IBM System/360 Operating System: Program Logic Manual Master Index, Order No. GY28-6717. For additional information about any subject listed below, refer to other publications listed for the same subject in the Master Index.

User Send delivers messages to various places depending upon command parameters and the operating status of the intended receiver.

1. If OPERATOR is specified the WTO macro is used to send the message to the specified console.

2. If USER is specified the processor sends the message to all specified and logged on users.

3. If LOGON is specified the message is sent to the Broadcast data set to be stored as mail for the specified and not logged on users. When a user logs on, his mail is delivered to him.

4. Warning messages are sent to the sending terminal telling of failures or other unexpected conditions. For example, if NOW is specified and a receiving user is not logged on, the message "USER(s) - userids - NOT LOGGED ON" is sent.

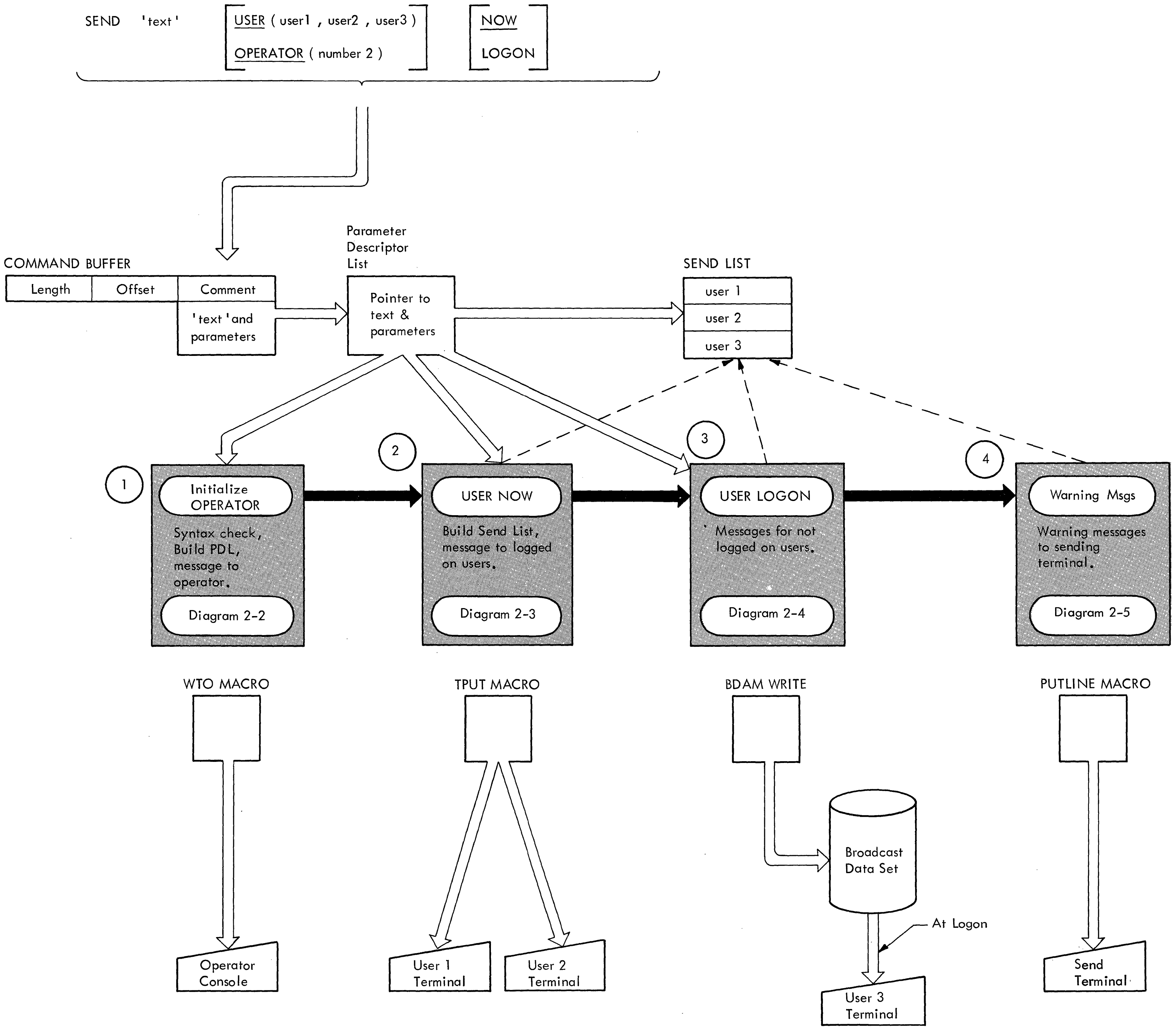
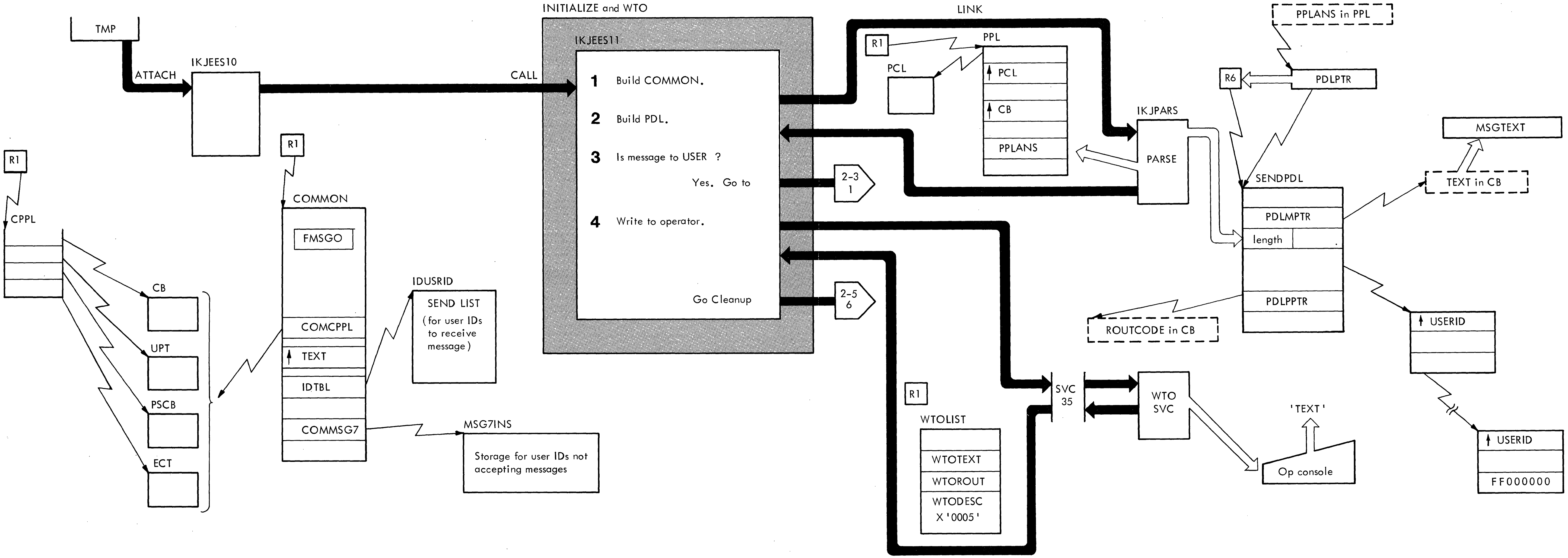
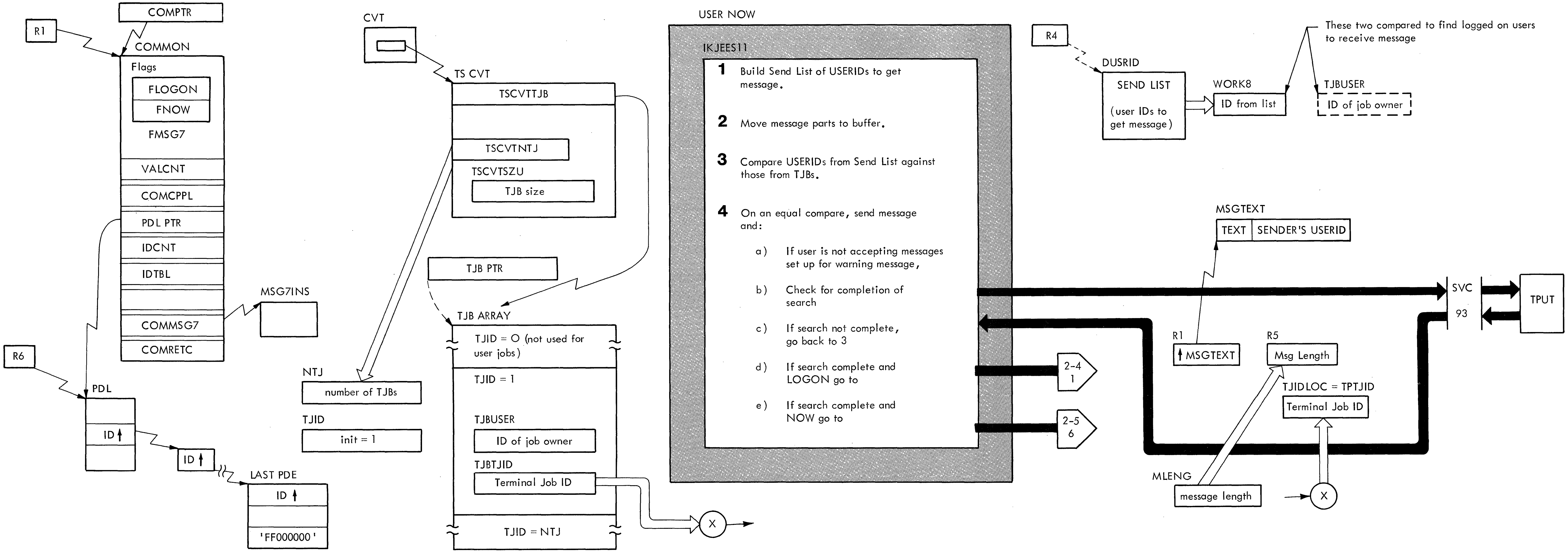


Diagram 2-1. Major Functions



Description		Routine	Label	FC
1	Common is built and initialized with pointer to the CPPL, the area for the SEND LIST, and the area for message 7 (user not accepting messages) inserts. IKJEES11 is called.	IKJEES10	IKJEES10	
2	Pointers to the command buffer and the PCL are put in the PPL and PARSE is invoked to check syntax of the command parameters and build the PDL. If the return code is > 0 go to send warning message and terminate. If the text length is greater than 115 bytes, the test is truncated and the warning message flag for truncated test (FMSGO) is set in COMMON. The TEXT is moved to MSGTEXT.	IKJEES11	DOPARS	
3	If USER is specified go to continue PDL examination and send messages to user. Otherwise process the OPERATOR request.		TUSER	
4	The routing code, which specifies the console to receive the message, is set in the WTOLIST. If the routing code is not specified, the default is to the master console. If the routing code is specified, it is converted to binary and used as an index to a table of routing codes (ROUTCD). The table entry is put in WTOLIST. The maximum length of a WTO is 72. The length of the message (text + the senders ID) is checked to see if two WTOs are required. The message is constructed in the WTOLIST and WTO is issued.		WTOL1 WTOL2 WTOL6	

Diagram 2-2. Initialize and WTO



1 Build Send List of USERIDs to get message.

2 Move message parts to buffer.

3 Compare USERIDs from Send List against those from TJBs.

4 On an equal compare, send message and:

a) If user is not accepting messages set up for warning message,

b) Check for completion of search

c) If search not complete, go back to 3

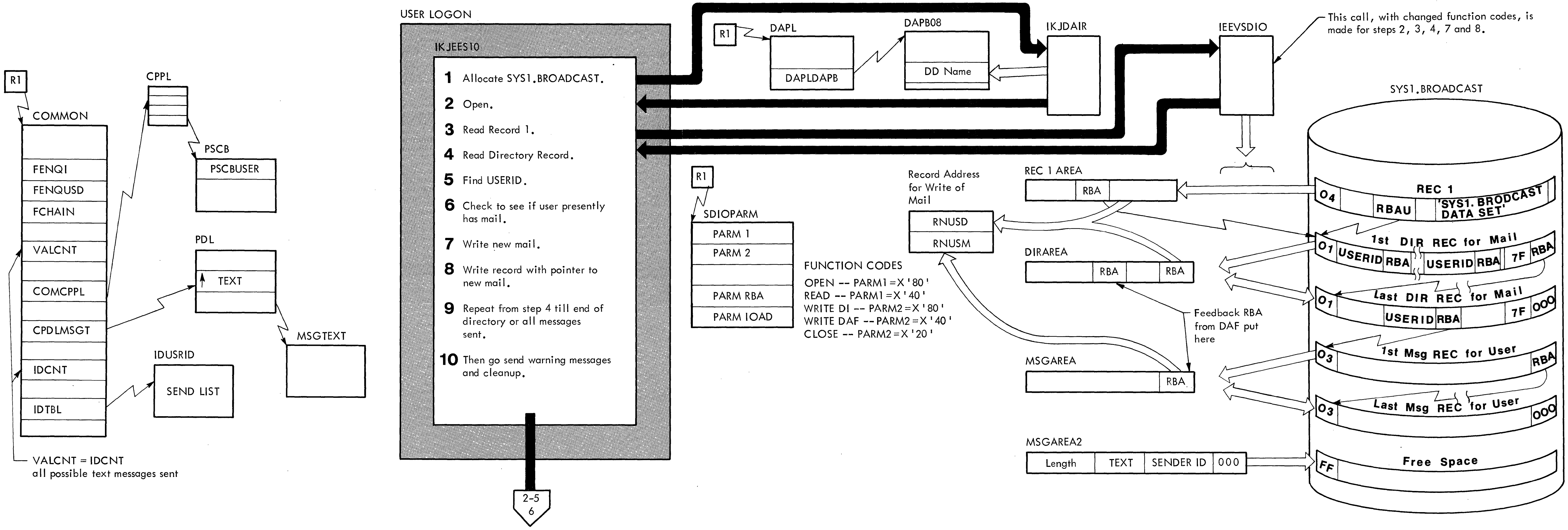
d) If search complete and LOGON go to

e) If search complete and NOW go to

Description	Routine	Label	FC
1 For each user that is to receive the message, there is a PDE that points to his USERID in the command buffer. The SENDLIST (IDUSRID) is built from these USERIDs. The maximum list length is 20. PDEs after 20 are ignored. The number of ID's in the list is put in IDCNT in COMMON.	IKJEES11	IDLP IDLP6	
2 The USERID of the sending user (from PSCB) is added to the text in MSGTEXT. The total length of the message is put in MLENG for use in the TPUT.		SKIPDUP	
3 The first TJB (TJID = 0) is not used for terminal jobs. The address of the TJB to be searched is calculated and set in TJBPTR. [Search address = array begin address + (TJID X size of TJB)]. To search the next TJB, TJID is incremented by 1 and the search address is recalculated: the end of the array has been reached when TJID (number of TJBs searched + 1) > NTJ (number of TJBs). For each TJB, each ID in the Send List is moved to WORK8 and compared to the ID of the job owner (TJBUSER).		TJBLOOP TJBL2 CHUSL	

Description	Routine	Label	FC
4 When a TJB belonging to a user in the Send List is found: The terminal job ID from the TJB (TJBTJID) is moved to TPTJID, the message length is moved to R5, and a pointer to MSGTEXT is moved to R1; then the TPUT macro is issued to send the message.		TPUT1	
a) A return code of X'0C' means the user is not accepting messages. If LOGON was specified (FLOGON in COMMON = 1) the message will be sent to the Broadcast DS, otherwise the "user-not-accepting" warning message flag (FMSG7) is set and the users ID is put in MSG7INS for use in the warning message.			
b) If either the message was accepted or the warning message flag was set, the ID in Send List is blanked, its processing is completed. VALCNT (the number of IDs processed) is incremented by 1. If VALCNT = IDCNT (number of IDs in list) then all ID processing is complete.		UPVAL	
c) If VALCNT ≠ IDCNT go back and process the next TJB.			
d) If VALCNT ≠ IDCNT and LOGON was specified (FLOGON = 1), go write messages to the Broadcast Data Set.			
e) If VALCNT ≠ IDCNT and NOW was specified (FNOW = 1), go send warning messages if any.			

Diagram 2-3. User Now



1 Allocate SYS1.BROADCAST.

2 Open.

3 Read Record 1.

4 Read Directory Record.

5 Find USERID.

6 Check to see if user presently has mail.

7 Write new mail.

8 Write record with pointer to new mail.

9 Repeat from step 4 till end of directory or all messages sent.

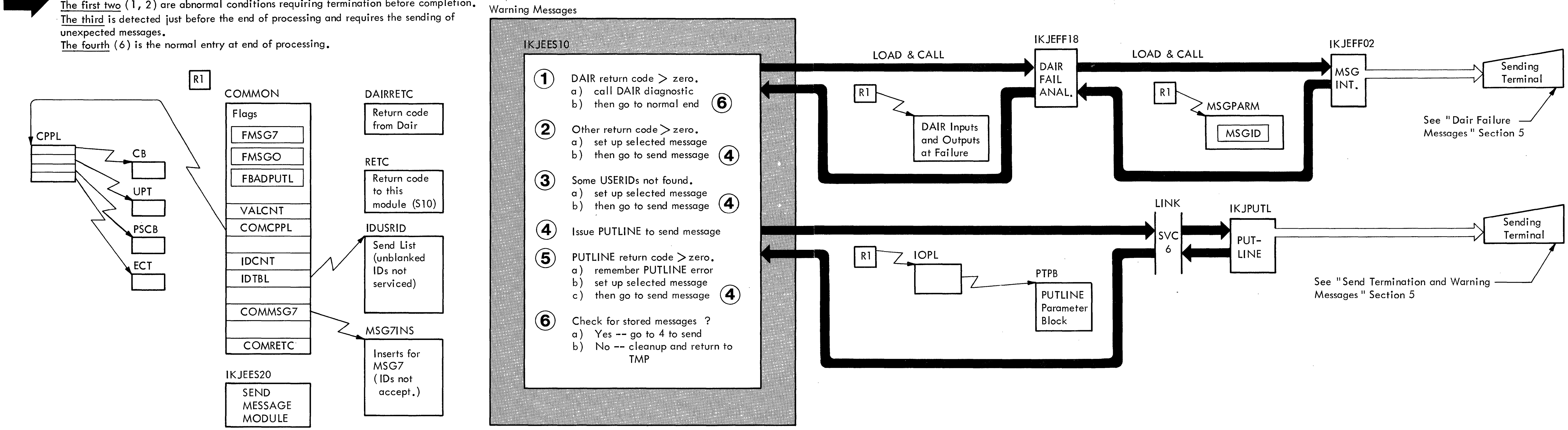
10 Then go send warning messages and cleanup.

Description	Routine	Label	FC
1 Dair is called, DAPL is fully set up. Since the data set is cataloged only the code '08', the data set name, and the disposition need be put in DAPB08. DAIR returns the Ddname in DAPB08.	IKJEES10	DODAIR	
2 SDIOPARM is set with the function code and the Ddname and IEEVSDIO is called to open the broadcast data set. If the return code = 0, the open flag in COMMON is set. If return code ≠ 0, go to send warning message and terminate.			
3 The function code is set for READ, the ↑ RBA for record 1 is set to PARMRBA and the address of RECIAREA is set in PARMIOAD. IEESVDIO returns record 1 to REC1AREA. If return code ≠ 0 go to send warning message and terminate.			
4 Before reading the DIRECTORY RECORD, MSGAREA2 is initialized with the record to be written (the message), the user ID of the sender is obtained from PSCBUSER in the PSCB and the TEXT from MSGTEXT. SDIOPARM is set with the function code for READ, the address of DIRAREA (to receive the record) and the ↑ RBA of the first directory record (obtained from RECIAREA which holds record 1). IEEVSDIO is called. If return code ≠ 0 go send warning message and terminate.		SENDER READUSD	
5 Each USERID in the directory record (now in DIRAREA) is compared against all IDs in the SEND LIST. When the record has been completely searched the next directory record (pointed to by the chain RBA in the present directory record) is read. The same steps are repeated until either: the last directory record (RBA field = 0) has been searched, or all IDs in the SEND LIST have been found (VALCNT = IDCNT). For either condition, processing is finished, go send warning messages.		SCANUSD	

Description	Routine	Label	FC
6 When a match is found (USERID from Send Test = USERID from directory record): The count of found IDs (VALCNT) is incremented, the RBA associated with the USERID in the directory record is examined. If the RBA ≠ 0, the user already has a mail record chain. FCHAIN in COMMON is set to '1' to record the fact and the user mail records (pointed to by RBA in directory record or mail record) are read until the last on the chain (RBA = 0) is found.		VALIDID CHAIN4	
7 Set the function code to write to free space (WRITE DAF). Set PARMRBA with the RBA of the first mail directory record to start the BDAM extended search. Set PARMIOAD with the address of MSGAREA2 (it contains the send message). Set PARMKEY with '03' the key written for a user mail record.		NOCHN1	
8 FCHAIN in COMMON is tested to see what record will point to new mail: if = 0 -- directory record must be written: put feedback from WRITE DAF into directory record, set PARM RBA to point to RNUSD (it contains the RBA of the directory record in which the USERID was found), set PARMIOAD with the address of DIRAREA (the record to be written), set PARMKEY to '01' (key for directory record, set function code to WRITE DI, and call IEEVSDIO to write the updated record.) if = 1 -- mail record must be written: put feedback RBA in mail message record, set PARMRBA to address of RNUSM, PARMKEY to '03' and call IEEVSDIO to WRITE 01.		CHAIN3	
9 If VALCNT ≠ IDCNT, go back and check next USERID in directory record against SEND LIST.		TESTWRIT	
10 If VALCNT = IDCNT, then all possible send messages have been sent. Go send warning messages.		TESTWRIT	

Diagram 2-4. User Logon

Entry to this function is at 1, 2, 3 or 6.
The first two (1, 2) are abnormal conditions requiring termination before completion.
The third is detected just before the end of processing and requires the sending of unexpected messages.
The fourth (6) is the normal entry at end of processing.



R1
CPPL
CB
UPT
PSCB
ECT
COMMON
Flags
FMSG7
FMSGO
FBADPUTL
VALCNT
COMCPPL
IDCNT
IDTBL
COMMSG7
COMRETC
IKJEES20
SEND MESSAGE MODULE
DAIRRETC
Return code from Dair
RETC
Return code to this module (S10)
IDUSRID
Send List (unblanked IDs not serviced)
MSG7INS
Inserts for MSG7 (IDs not accept.)
Warning Messages
IKJEES10
1 DAIR return code > zero.
a) call DAIR diagnostic
b) then go to normal end 6
2 Other return code > zero.
a) set up selected message
b) then go to send message 4
3 Some USERIDs not found.
a) set up selected message
b) then go to send message 4
4 Issue PUTLINE to send message
5 PUTLINE return code > zero.
a) remember PUTLINE error
b) set up selected message
c) then go to send message 4
6 Check for stored messages ?
a) Yes -- go to 4 to send
b) No -- cleanup and return to TMP
LOAD & CALL
IKJEFF18
DAIR FAIL ANAL.
R1
DAIR Inputs and Outputs at Failure
LOAD & CALL
IKJEFF02
MSG INT.
R1
MSGPARM
MSGID
Sending Terminal
See "Dair Failure Messages" Section 5
LINK
SVC 6
IKJPUTL
PUT-LINE
R1
IOPL
PTPB
PUTLINE Parameter Block
Sending Terminal
See "Send Termination and Warning Messages" Section 5

Description	Routine	Label	FC
1 The Common Dair Failure message routine is called to analyze the failure and send messages. Parameters are DAPL, which points to DAPB08, and DAIRRETC. The message Interface routine sends the message before returning to IKJEES10. After return, processing is as for a normal end.	IKJEES10	BADDAIR	
2 All other messages are issued by IKJEES10 using the PUTLINE macro. The putline parameter block (PTPB) contains buffers for message components and descriptor fields to describe the structure of the message. IKJEES10 analyzes return codes and failure conditions and sets up the PTPB with message components from IKJEES20. USERIDs are put in the buffer from blocks known to S10 and return code inserts are converted to EBCDIC before they are put in PTPB.		CONVRETC	
IKJPUTL, invoked by the PUTLINE macro uses the PTPB to compose the required message and sends it to the terminal.			
The non-zero return codes causing entry at (2) come from:			
TPUT -- the return code is analyzed by S11. Entry here means that the message required is -- 'Command System Error + Putline Error Code'		BADTPUT	
PARSE -- if the return code is 16, the "not enough main storage" message is sent. For all other codes the system error message with the PARSE return code is sent.		BADPARSE NOMAIN	
IEEVSDIO -- return code - X '01'; message - no room for MAIL in BROADCAST DS		BADIOA	
X '02'; can't open BROADCAST DS		NOTUSABL	
X '40'; not enough main storage		NOMAIN	
other ; BROADCAST DS synad error		NOTUSABL	

Description	Routine	Label	FC
3 At the end of the TJB search (NOW specified) or at the end of the User Directory search, if VALCNT ≠ IDCNT some IDs in IDUSRID have not been found. PTPB is set up to send : NOW -- 'USER(s) - user id(s) - NOT LOGGED ON'		MSG2OUT	
LOGON -- 'INVALID USURID(s) - user id(s) - + USER NOT AUTHORIZED FOR TSO' For both messages, the user id inserts are all non-blank positions in the SEND LIST.		SCUSD3	
4 IKJPUTL locates the sending terminal (from user information in IOPL), arranges the message segments and inserts as directed by PTPB, and sends the message.		WARNMSG	
5 A non-zero return code from IKJPUTL means: code = 16 -- not enough main storage to execute. code = other -- putline failure.			
In both cases the message being sent is lost. For the not enough storage code the NOMAIN message is set up in PTPB and another try is made to use PUTLINE. This is done repeatedly until some code other than 16 is received. (Storage becomes available).		NOMAIN	
For any other failure code (not 16, not 0), a bit in COMMON (FBADPUTL) is set to remember the failure and an attempt is made to send the 'Command System Error' message including the Putline error code. A second failure causes termination without further messages.		BADPUTL EXITO	
6 For a normal termination, the stored message bits in COMMON (FMSGO -- for message has been truncated and FMSG7 for users not accepting messages) are inspected and the required messages are sent. MSG7INS (pointed to by COMMON) furnishes the inserts for the not accepting message. In all cases control is returned to TMP with a return code of 0.		EXITCODE MSGOOUT MSG7OUT EXIT3	

Diagram 2-5. Warning Messages

Part 6: SUBMIT Command Processor

6

Contents

Illustrations

Charts

Diagrams

Figures

Section 1. Introduction

The SUBMIT command processor lets the user submit a job for interpretation and execution in the background. SUBMIT is a part of the Foreground-Initiated Background (FIB) facility of TSO. Figure 1-1 presents the four commands of the FIB facility and shows how they relate to each other and to the system.

Command Format

The format of the SUBMIT command is:

```
SUB[MIT]dslist[(NOTIFY)or(NONOTIFY)]
```

where

dslist is a list of data set names defining an input stream (JCL plus data).

NOTIFY is a default keyword requesting the final status of a JOB. JOB statements generated by the SUBMIT command processor include the NOTIFY parameter.

> When a job has finished processing in the background a message will be placed in the Broadcast data set for the terminal user. To receive this message the user must either issue the LISTBC command or relogon, whereupon he will receive any messages that might have been stored in the Broadcast data set during his last terminal session.

NONOTIFY is a keyword requesting that generated JOB statements not include the NOTIFY parameter; the terminal user will not have a final job status message placed in the Broadcast data set.

The FIB facility comprises four commands:

SUBMIT -- allows the terminal user to put existing Data Sets on the BRDRQ, stage Ⓐ below. From there the jobs defined by the Data Sets are processed as normal background jobs.

CANCEL/STATUS -- allows the terminal user to have jobs at stages Ⓑ Ⓒ Ⓓ Ⓔ marked Failed (and thus terminated) or to have status information about them sent to the terminal.

OUTPUT -- allows the terminal user to have a job, waiting at stage Ⓕ (MVT OUTPUT QUEUE) moved to another queue, delivered as output at the installation, or delivered as output at the terminal.

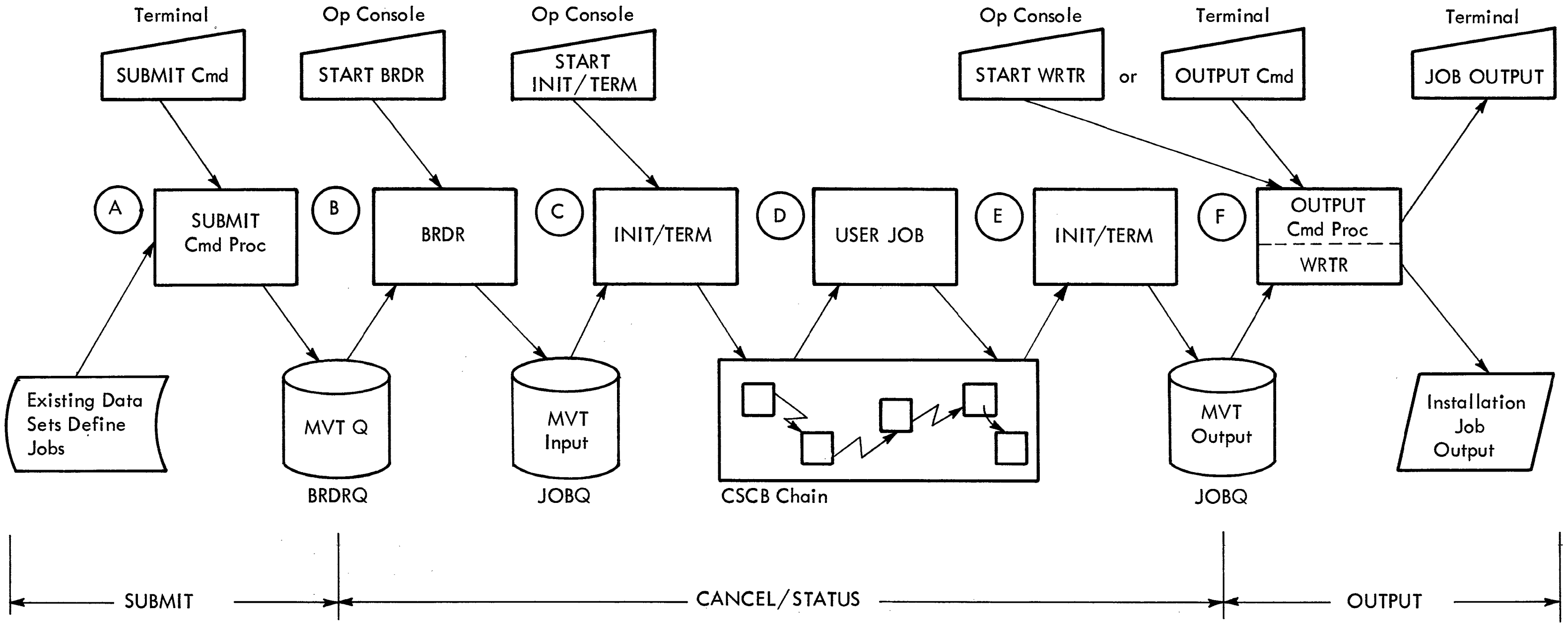


Figure 1-1. Foreground Initiated Background (FIB) Facility

Data Set Naming Conventions

A data set name within apostrophes is considered to be fully qualified and completely stated; the SUBMIT command processor will accept it as entered. A data set name not enclosed within apostrophes must conform to TSO naming conventions. It must contain at least two of the following fields:

- Userid (required)
- One or more user-supplied names (optional)
- A descriptive qualifier (required)

The total length of the data set name must not exceed 44 characters, including periods. The convential name is of the form:

```
USERID. (user supplied name). qualifier
```

The parenthesis indicates optional.

Data Set Restrictions

A data set specified in the dslist parameter must be a sequential direct access data set or a member of a partitioned data set. Logical record length must be 80 bytes.

Operational Considerations

More than one data set can be listed in the dslist parameter. Blocking factors may differ from one data set to another but the logical record length must be 80 bytes.

The input stream may contain more than one job. In this case, each job will be considered as a separate submission and an output data set will be created for each job.

Section 2. Method of Operation

This section summarizes the method of operation of the SUBMIT command processor. It includes method of operation diagrams which illustrate basic functions.

Summary

The SUBMIT command processor lets the user submit a job for interpretation and execution in the background. You must set up an input job stream (JCL plus data) on one or more data sets and include the data set name(s) as an operand of the SUBMIT command.

The SUBMIT command processor reads records from your data set(s), copies the records to a Submit output data set (one for each job), and places the Submit data set on queue number 55 of the MVT job queue (BRDR Q). A special background reader (BRDR) interprets the job, using the MVT reader/interpreter as a subroutine to build standard tables and control blocks. The job is later executed under control of the MVT initiator/terminator, just like any other MVT job.

Method of Operation Diagrams

The rest of this section consists of method of operation diagrams which illustrate basic functions. Each diagram contains a cross-reference table to help you find corresponding flowcharts or sections of code.

NOTE: The diagrams are at the back of this PLM.

Section 3. Program Organization

This section describes the program organization of the SUBMIT command processor. There are three parts to this section:

- Program Hierarchy Chart -- Figure 3-1.
- Function Tables -- which show the overall logic of individual routines.
- Flowcharts -- which show the detailed logic of individual routines.

Hierarchy

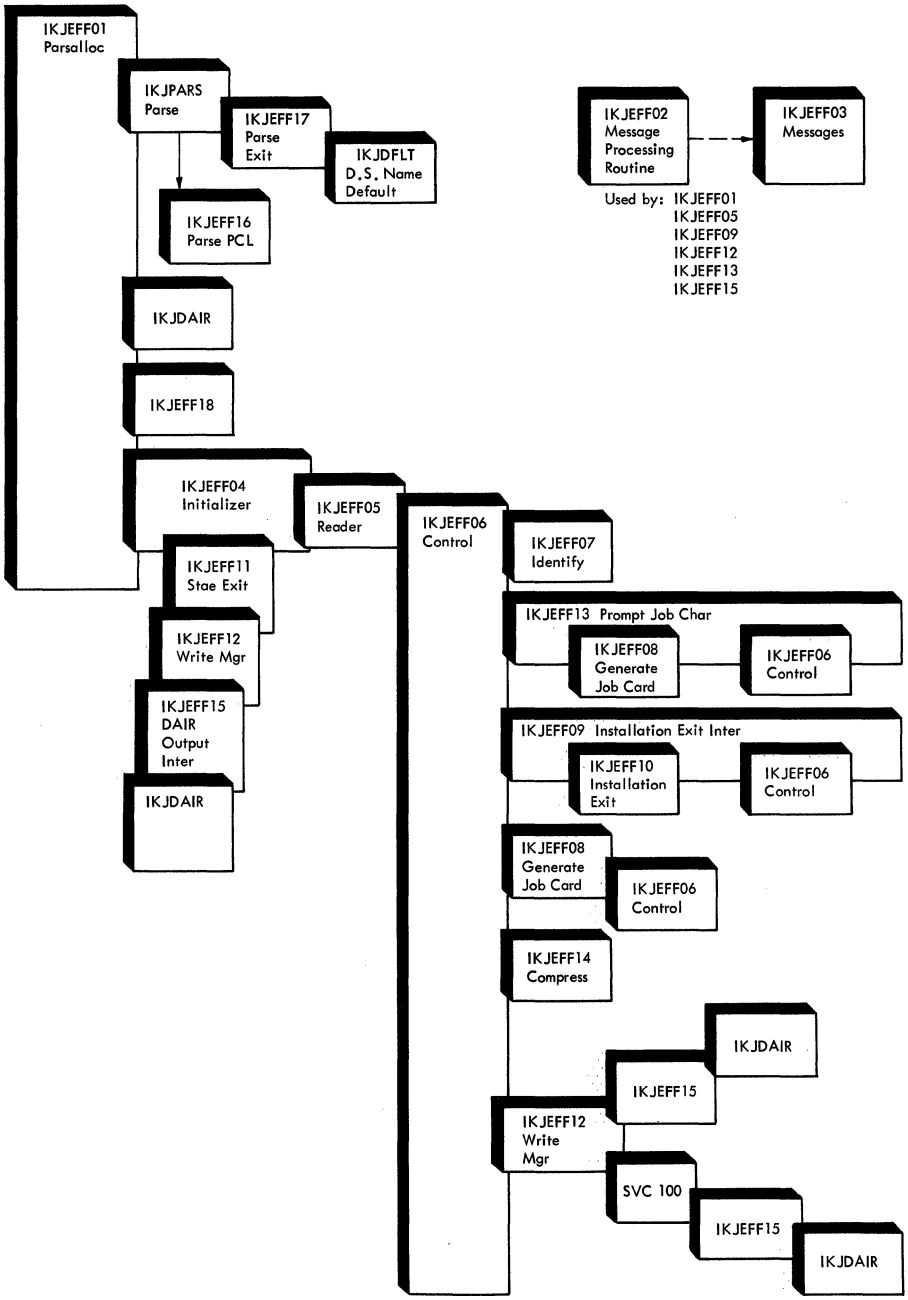


Figure 3-1. HIERARCHY

Hierarchy

The SUBMIT command processor is composed of 18 separate routines:

LOAD1 -- SUBMIT
- IKJEFF01 - Calls service routines to have command parsed and input data sets allocated.
- IKJEFF02 - Processes messages.
- IKJEFF03 - Contains messages.

LOAD2 -- IKJEFF04
- IKJEFF04 - Initializes tables and control blocks.
- IKJEFF05 - Reads records from the input data sets.
- IKJEFF06 - Processes records.
- IKJEFF07 - Identifies records as JCL or data.
- IKJEFF08 - Generates default JOB statement, if necessary.
- IKJEFF09 - Presents parameters to installation exit routine.
- IKJEFF11 - STAE exit routine.
- IKJEFF12 - Writes records to the Submit data set.
- IKJEFF13 - Prompts for additional character to form jobname, if necessary.
- IKJEFF14 - Compresses input record to form output record.

LOAD3
- IKJEFF10 - Installation exit routine.

LOAD4 -- IKJEFF10
- IKJEFF15 - Allocates Submit data set. Places Submit data set on the Job Queue.

LOAD5 -- IKJEEF16
- IKJEFF16 - Parse macros.
- IKJEFF17 - Parse exit.

LOAD6 -- IKJEFF18
- IKJEFF18 - Common Dair Failure Message Routine

IKJEFF00—Foreground-Initiated Background SVC (SVC 100)

Entry Points	The Writer Manager routine (IKJEFF12) issues SVC100. Upon entry, Register 1 points to the following parameter list: CPPL Function Byte 1 - Allocate 2 - Deallocate and Enqueue 3 - Deallocate and Scratch DCB Jobname Message Routine List Ptr
Operation	• Checks for the primary command (SUBMIT,CANCEL, STATUS, OUTPUT or OPERATOR). • Puts the program in key zero. • Links to one of the following routines: SUBMIT IKJEFF15 CANCEL IKJEFF51 STATUS IKJEFF51 OUTPUT IKJEFF61 or, if the primary command is OPERATOR, issues SVC 34.
Data Areas	None.
Routines Called	IKJEFF15 (for SUBMIT)
Registers	Upon exit, register 15 contains a return code, or X'80' if the user is not authorized to use FIB.

IKJEFF01—PARSE and ALLOCATE Routine

Entry Points	IKJEFF01 -- ATTACHed by the TMP. Upon entry, Register 1 points to the CPPL.
Operation	Prepares for SUBMIT command processing; checks SUBMIT parameters and allocates input data sets. • Builds a parameter list to be used when calling the Message Processor IKJEFF02. • Checks the user's authorization. • Loads IKJEFF16 and IKJEFF17. • Calls IKJPARS to check SUBMIT parameters and build a parameter list to be used when calling IKJDAIR. • Loads and calls IKJDAIR to allocate the input data sets. • Loads and calls the Initializer IKJEFF04 to start SUBMIT processing. • If there are errors, - calls the Message Processor IKJEFF02. - calls IKJSTCK to remove all elements from the stack.
Data Areas	Message Processor Parameter List, DAIR28, DAIR08, Parse Exit Work Area, CPPL, PDL, PDE
Routines Called	IKJEFF02, IKJEFF03, IKJEFF04, IKJPARS, IKJDAIR, IKJSTCK, IKJEFF18
Exits	Return to TMP through a single exit with a return code in Register 15: 0 - no errors 12 - errors
Registers	Upon exit, all registers except register 15 are restored.

IKJEFF02—Message Processor Routine

Entry Point	IKJEFF02 -- called by IKJEFF01, IKJEFF05, IKJEFF09, IKJEFF12, IKJEFF13, IKJEFF15. Upon entry, Register 1 points to the following parameter list: • Message Routine List • CPPL • ECB
Operation	• Selects a message from the Message Module CSECT coded with the IKJTSMSG macro. • Inserts variable text into the message. • Converts the value to printable hex. • Calls Putline IKJPUTL to send informational message. • Calls Putget IKJPTGT to send message requiring response from terminal user. • Calls Write-to-Operator WTO to send message to operator.
Data Areas	CPPL, Message Interface
Routines Called	IKJPTGT, IKJPUTL, WTO
Exit	Return to caller with return code in register 15 as returned from Putline or Putget (0 from WTO).
Registers	Upon exit, all registers are restored except register 15.

IKJEFF03—Foreground Message Module

Entry Points	Not executable -- referenced by the Message Processor routine IKJEFF02. This module contains messages coded by the IKJTXMSG macro instruction.

IKJEFF04—Initializer Routine

Entry Points	IKJEFF04 -- called by the Parse Allocate routine IKJEFF01. Upon entry, Register 1 points to the following parameter list: • Message Routine Interface List Ptr • Userid • DD Element Chain Ptr • Function Code Byte Bit 7 = NONOTIFY
Operation	• Initializes the SUBMIT Control Table. • Initializes the History Table. - sets the First switch on, - sets the Exit switch to 'JOB', - sets the Nonotify switch on if function code bit 7 is on. • Loads the Installation Exit routine IKJEFF10. • Issues the STAE macro, specifying the address of the STAE Exit routine IKJEFF11. • Calls the Reading routine IKJEFF05 to start SUBMIT processing. • Deletes the Installation Exit routine IKJEFF10. • If there are errors, call the STAE Exit routine IKJEFF11.
Data Areas	SUBMIT Control Table History Table
Routines Called	IKJEFF05, IKJEFF10, IKJEFF11
Exit	Return to caller.
Registers	Upon exit, all registers are restored except register 15.

IKJEFF05—Reading Routine

Entry Points	IKJEFF05 -- called by the Initializer routine IKJEFF04. Upon entry, Register 1 points to the following parameter list: • History Table • Current Statement Ptr • Message Routine List Ptr • DD Element Chain Ptr • Control Table Ptr
Operation	Reads the input data sets (one logical record at a time) and calls the Control routine IKJEFF06 to process it. • Finds the DAIR08 Block pointed to by the first element on the DD Element Chain. • Moves the ddname from the DAIR08 Block to the DCB. • Opens the DCB. • Reads one logical record. • Sets the Current Statement Ptr. • Calls the Control routine IKJEFF06 to process the record. • At end-of-file for the first data set, updates the DD Element Chain Ptr, gets the next DAIR08 Block and repeats the process. • At ultimate end-of-file, - passes a /* to the Control routine - passes a null statement to the Control routine - turns off the exit switches and passes a second null statement to the Control routine.
Data Areas	History Table, DAIR08, Message Interface
Routines Called	IKJEFF06, IKJEFF02(via address)
Exits	Return to caller.
Registers	Upon exit, all registers are restored.

IKJEFF06—Control Routine

Entry Points	IKJEFF06 -- called by IKJEFF05, IKJEFF08, IKJEFF09 and IKJEFF13. Upon entry, Register 1 points to the following parameter list: • Control Table PTR.
Operation	Copies the input data set(s) to a single output data set and sequeues a DSB/JFCB pointer to it. • Calls the Identify routine IKJEFF07 to identify the input statement. • If the first record is not a job statement, calls the Generate Job Card IKJEFF08 routine to generate a defult job statement. • If the exit switch for that statement is on, calls the Installation Exit Interface routine IKJEFF09 which will pass control to the Installation Exit routine IKJEFF10. • If the record is a job statement, calls the Prompt Job Character routine IKJEFF13 to prompt the user for an additional character if one is needed to form a jobname. • Calls the Writer Manager routine IKJEFF12 to deallocate a previous Submit Data Set (if any) and allocate a new Submit Data Set. • Calls the Compress routine IKJEFF14 to compress the record (squeeze out blanks and shift to the left). • Calls the Writer Manager routine IKJEFF12 to write the compressed record to the output data set.
Data Areas	CONTAB, History Table
Routines Called	IKJEFF07, IKJEFF08, IKJEFF09, IKJEFF12, IKJEFF13, IKJEFF14.
Exits	Return to caller.
Registers	Upon exit, all registers are restored.

IKJEFF07—Identify Routine

Entry Points	IKJEFF07 -- called by the Control routine IKJEFF06. Upon entry, Register 1 points to the following parameter list: • History Table • Current Statement Ptr • Jobname Field
Operation	Identifies: • If it is JCL: - indicates the kind of statement. - whether to expect a continuation. • If it is data, determines what kind of data; - DD * - DD DATA
Data Areas	History Table
Routines Called	None
Exit	Return to caller.
Registers	Upon exit, all registers are restored.

IKJEFF08—Generate Job Card Routine

Entry Points	IKJEFF08 -- called by Control IKJEFF06 or Prompt JOB Character IKJEFF13. Upon entry, register 1 points to the following list: • Pointer to History Table • Pointer to Current Statement Ptr • Pointer to Userid • Control Table Ptr

Operation	• Generates a default JOB statement:
	//userid JOB , default job statement
	// userid,
	// notify=userid,
	// msglevel=(1,1)
	• Calls Control IKJEFF06 to process the JOB statement.

Data Areas	CONTAB, Message Interface
Routines Called	IKJEFF06
Exits	Return to caller.
Registers	Upon exit, all registers are restored.

IKJEFF09—Installation Exit Interface Routine

Entry Points	Called by the Control routine IKJEFF06. Upon entry, Register 1 points to the following parameter list: • Current Statement Ptr • Userid • History Table • Message List Ptr • Submit Control Table Ptr • Exit Word • Exit Address
Operation	• Passes information to the Installation Exit routine IKJEFF10. • Calls the Control routine IKJEFF06, if necessary, to process a changed statement. • Calls the Message Processing routine IKJEFF02, if necessary, to process any message requests made by the Installation Exit.
Data Areas	MSGLIST, History Table, CONTAB
Routines Called	IKJEFF02 (via address), IKJEFF06, IKJEFF10 (via Address)
Exits	Return to Caller.
Registers	Upon exit, all registers are restored, except register 15 which contains a return code from IKJEFF10. 0 - continue 4 - return for another statement 8 - put out a message and return 12 - put out a message, get response and return 16 - abort Others - abort

IKJEFF10—Installation Exit Routine

Entry Points

IKJEFF10 -- Called by the Exit Interface routine IKJEFF09.
Upon entry, Register 1 points to a word which points to the following parameter list:

- Pointer to current statement if zero, entry is being made to get a statement. The current statement may be deleted by zeroing this pointer.
- Pointer to message to be put out if non-zero, entry being made to 'RETURN' the message so that the exit may free its buffer, if any. The format of the buffer is 'LL-TEXT' where LL is a two-byte length of the entire buffer.
- Pointer to response if non-zero, entry is being made to give a response to the message above. The format is 'LL-TEXT'. Submit will free the response buffer.
- Pointer to userid (8bytes, padded with blanks).
- Pointer to JCL exit switch bytes (these switch bytes begin on a byte boundary)

```
Byte 0 - Take exit for these switches may be changed
           by the exit.
    Bit 0 - JOB
        1 - EXEC (program or procedure)
        2 - DD
        3 - COMMAND
        4 - NULL
        5 - RESERVED
        6 - RESERVED
        7 - RESERVED
Byte 1 - This byte, if not zero, gives the card image
         column that the operand field starts in
         (i.e., the value is one-origined)
Byte 2 - This statement switches
    Bit 0 - is a JOB
        1 - is an EXEC
        2 - is a DD
        3 - is a COMMAND
        4 - is a NULL
        5 - OPERAND to be continued
        6 - Statement to be continued
        7 - Statement is a continuation word for
```

- Word for EXIT's use. This field is initialized to zeroes and retains whatever value the exit gives it. For example, it may be used to contain values and switches set in one entry to the exit to govern control and processing in the next entry to the exit.

IKJEFF10 -- Installation Exit Routine (Cont.)

Operation	User-Written can cancel jobs, establish priorities, change JCL, etc. (The IBM-supplied dummy zeros register 15 and branches on register 14.)
Data Areas	EXIT Switches.
Routines Called	None.
Exits	Return to IKJEFF09.
Registers	Upon exit, all register except 15 are restored. Register 15 contains a return code 0 - continue 4 - return for another statement 8 - put out a message and return 12 - put out a message, get response and return. 16 - abort Others - put out a message, abort

IKJEFF11—STAE Exit Routine

Entry Points	IKJEFF11 -- called by the Supervisor or by the Initializer, IKJEFF04. Upon entry, Register 1 points to a pointer to the Submit Control Table.
Operation	Calls the Write Manager routine to scratch the current Submit Data Set.
Data Areas	Control Table (CONTAB).
Routines Called	IKJEFF12
Exits	Return to caller.
Registers	Upon exit, all registers are restored.

IKJEFF12—Writer Manager Routine

Entry Points	IKJEFF12 -- called by the Control routine, IKJEFF06; or by the STAE EXIT, IKJEFF11. Upon entry, Register 1 points to the following parameter list: • Writer Manager Work Area Ptr • Function Byte 1 - start of a new job 2 - write block 3 - write, close and enqueue 4 - close and scratch • Current jobname • Output Block pointer • History Table • Message Processor Parameter List Ptr
Operation	• Calls the DAIR Entry Interface routine IKJEFF15 to allocate an output data set. • Obtains and initializes an Output Block. • Writes a Output Block to the data set. • Closes the data set. • Calls the Dair Entry Interface routine IKJEFF15 to deallocate the output data set via SVC 100.
Data Areas	History Table, Message Interface
Routines Called	IKJEFF02 (via address), IKJEFF15, IGC00100 (SVC 100)
Exit	Return to caller.
Registers	Upon exit, all registers are restored.

IKJEFF13—Prompt Job Character Routine

Entry Points	IKJEFF13 -- called by Control IKJEFF06. Upon entry, register 1 points to the following parameter list: • Message Routine List Ptr • Current Statement Ptr • Userid • Control Table Ptr • Current Jobname • Histroy Table
Operation	Compares the jobname against the userid and, if they are identical, prompts the user to supply an additional character to form a jobname of the form 'useridX', where X is the added character. • Calls the Control routine, IKJEFF06 to process the changed statement. • If unable to fit in the additional characters, calls the Generate JOB Card routine to generate a default statement.
Data Areas	Histroy Table, Message Interface, CONTAB
Routines Called	IKJEFF02, IKJEFF06, IKJEFF08
Exit	Return to caller.
Registers	Upon exit, all registers are restored.

IKJEFF14—Compress Routine

Entry Points	IKJEFF14 -- called by the Control routine IKJEFF06. Upon entry, register 1 points to the following parameter list: • Pointer to next output location • Length of Output Area • Input Block Ptr • Length of Input Block
Operation	• Compresses the Input Block statement (moves to the left, eliminates blanks). • Places the compressed statement in the Output Block. • Updates the Output Block Statement Ptr. • Decrements the space remaining in the Output Block.
Data Areas	Output Block
Routines Called	None.
Exit	Return to caller with return code in register 15: 0 - normal 4 - no room
Registers	Upon exit, all registers except register 15 are restored.

IKJEFF15—DAIR Output Interface Routine

Entry Points	IKJEFF15 -- called by the Writer Manager routine Upon entry, register 1 points to the following parameter list: • TMCT (CPPL) • Function Byte 1 - allocate 2 - deallocate and enqueue 3 - deallocate and scratch • DCB • Jobname • Message Routine Parameter List Ptr
Operation	Function 1 - allocate output data sets. • Calls IKJDAIR to allocate the Submit data sets. • Places a ddname in the caller's DCB. Function 2 - deallocate and enqueue Submit data sets • Calls IKJDAIR to deallocate the Submit data sets and to enqueue them on the System Job Queue. Function 3 - deallocate and scratch Submit data sets • Calls IKJDAIR to deallocate and scratch the Submit Data Sets.
Data Areas	DAPB30, DAPB18, CPPL, DCB, CVT, Queue Manager Master QCR, Message Interface
Routines Called	IKJDAIR, IKJEFF02 (via address)
Exit	Return to caller.
Registers	Upon exit, all registers are restored.

IKJEFF16—PARSE Entry Interface Routine

Entry Points	IKJEFF16 -- called by Parse/Allocate IKJEFF01. Upon entry, register 1 points to the following parameter list: • Parse Data Name Set PDE • Parse Exit Work Area
Operation	Builds a parameter list to be used when calling IKJPARS.
Data Areas	
Routines Called	None.
Exit	Return to Parse/Allocate IKJEFF01.
Registers	Upon exit, all registers are restored.

IKJEFF16—PARSE Control List CSECT

Entry Points	None.
	This CSECT is not executable. It is referenced by IKJPARS for the PCL and the entry point address of IKJEFF17.

IKJEFF17—PARSE Exit Interface Routine

Entry Points	IKJEFF17 -- called by IKJPARS. Upon entry, register 1 points to the following parameter list: Parse Data Set Name PDE Parse Exit Work Area
Operation	Builds DAIR08 Blocks and DD Element Chains to be used by the Parse/Allocate routine IKJEFF01 and the Reading routine IKJEFF05.
Data Areas	DAIR08 Block, PDE, PARSE EXIT Work Area, DD Element Chain
Routines Called	IKJEFF02, IKJDFLT.
Exits	Return to Caller.
Registers	Upon exit, all registers are restored, except register 15.

IKJEFF18—Common DAIR Failure Message Routine

Entry Point	IKJEFF18 -- on entry, register 1 points to a list of pointers to the following parameters: • The input parameter list used when the call to DAIR was made (the 'DAPL'), including the DAIR request parameter block. • A full word containing the DAIR return code from register 15. • A full word containing the address of message processor module IKJEFF02 or zeroes (in which case this routine will load and delete IKJEFF02). • A half word containing the caller number, 1 for General Caller.
Operation	IKJEFF18 diagnoses DAIR allocation and unallocation failures and puts out appropriate messages to the terminal. Input to this routine consists essentially of the input to and the output from DAIR. Thus if the return code from DAIR is non-zero, IKJEFF18 may be called to diagnose the error. The sequence of operation is: 1. The caller situation is identified (what operation was being performed). 2. A table relating return codes to their meanings for the callers situation is used. From this table a message ID for an appropriate message is obtained. 3. The message ID is passed to IKJEFF02 which send the message to the terminal.
Data Areas Used	CPPL DAPL DAIR Blocks
Routines Called	IKJEFF02
Exits	Both normal and error return to caller with register 15 set to zero.

Chart 01. SUBMIT Overview

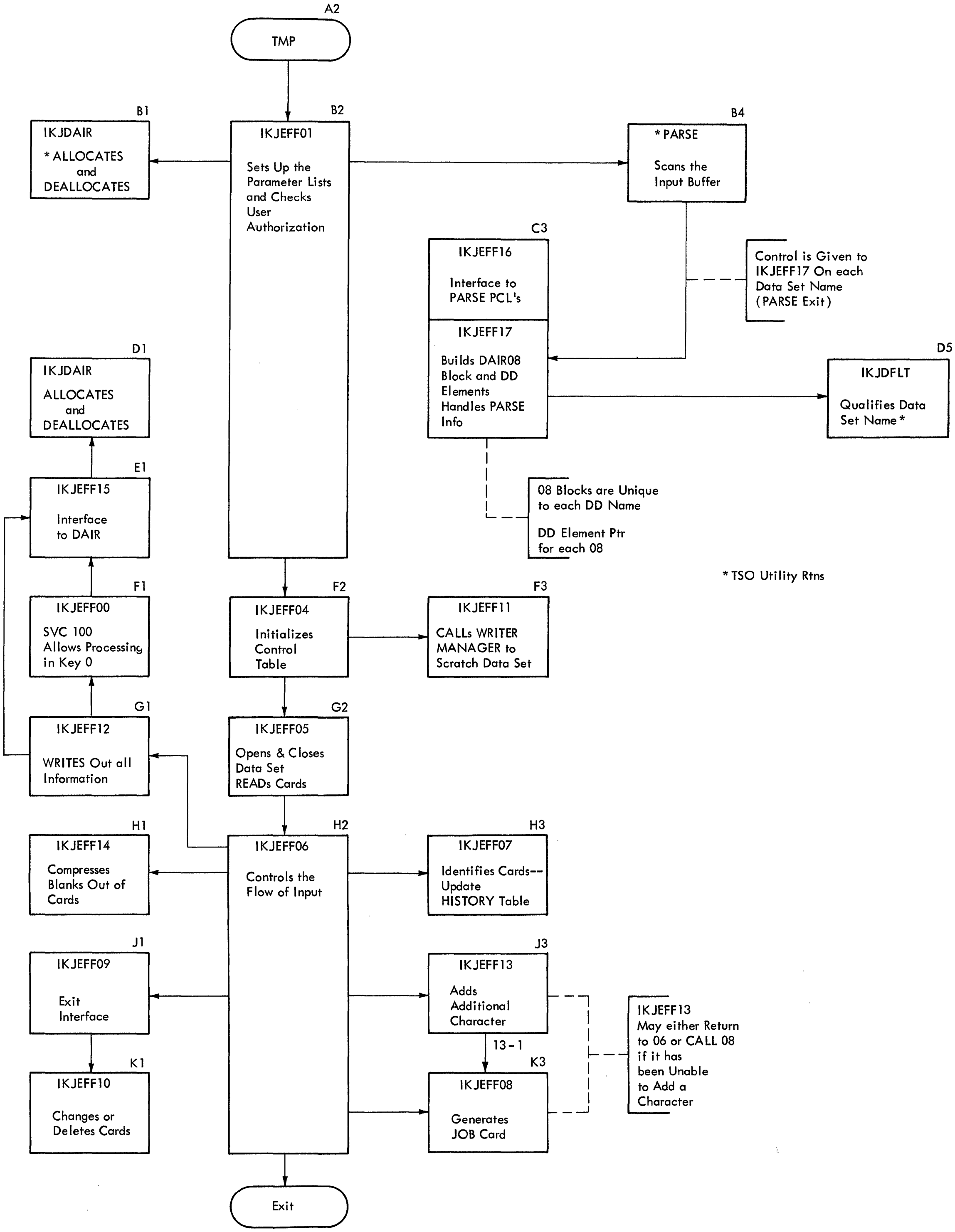

A2
TMP
B1
IKJDAIR
* ALLOCATES and DEALLOCATES
B2
IKJEFF01
Sets Up the Parameter Lists and Checks User Authorization
B4
* PARSE
Scans the Input Buffer
C3
IKJEFF16
Interface to PARSE PCL's
Control is Given to IKJEFF17 On each Data Set Name (PARSE Exit)
IKJEFF17
Builds DAIR08 Block and DD Elements Handles PARSE Info
D1
IKJDAIR
ALLOCATES and DEALLOCATES
D5
IKJDFLT
Qualifies Data Set Name *
E1
IKJEFF15
Interface to DAIR
08 Blocks are Unique to each DD Name
DD Element Ptr for each 08
* TSO Utility Rtns
F1
IKJEFF00
SVC 100 Allows Processing in Key 0
F2
IKJEFF04
Initializes Control Table
F3
IKJEFF11
CALLs WRITER MANAGER to Scratch Data Set
G1
IKJEFF12
WRITES Out all Information
G2
IKJEFF05
Opens & Closes Data Set READs Cards
H1
IKJEFF14
Compresses Blanks Out of Cards
H2
IKJEFF06
Controls the Flow of Input
H3
IKJEFF07
Identifies Cards-- Update HISTORY Table
J1
IKJEFF09
Exit Interface
J3
IKJEFF13
Adds Additional Character
IKJEFF13 May either Return to 06 or CALL 08 if it has been Unable to Add a Character
13-1
K1
IKJEFF10
Changes or Deletes Cards
K3
IKJEFF08
Generates JOB Card
Exit

Chart AA. IKJEFF01. PARSE and ALLOCATE

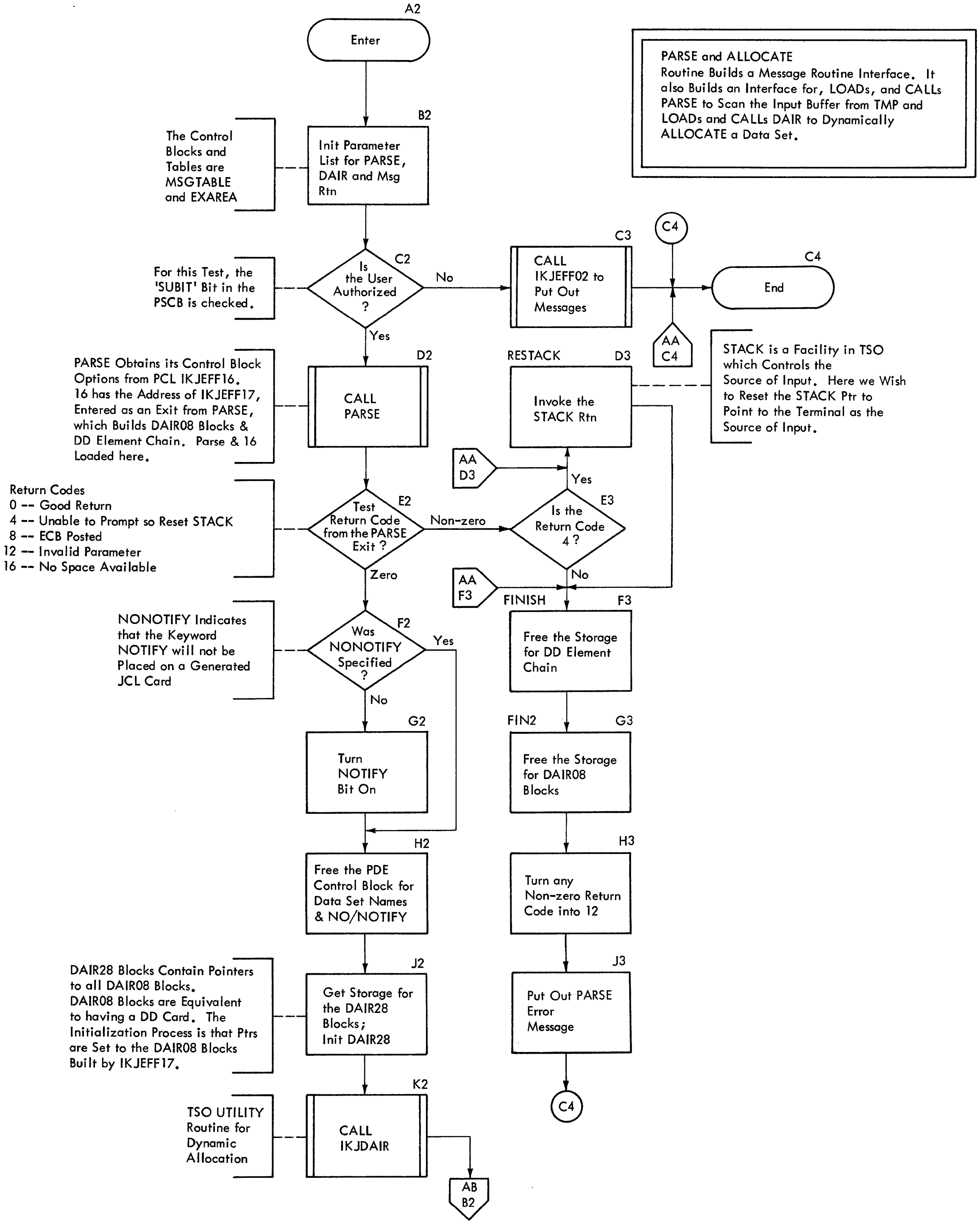

A2
Enter
B2
Init Parameter List for PARSE, DAIR and Msg Rtn
The Control Blocks and Tables are MSGTABLE and EXAREA
PARSE and ALLOCATE
Routine Builds a Message Routine Interface. It also Builds an Interface for, LOADs, and CALLs PARSE to Scan the Input Buffer from TMP and LOADs and CALLs DAIR to Dynamically ALLOCATE a Data Set.
C2
Is the User Authorized ?
For this Test, the 'SUBIT' Bit in the PSCB is checked.
No
C3
CALL IKJEFF02 to Put Out Messages
C4
C4
End
AA C4
Yes
D2
CALL PARSE
PARSE Obtains its Control Block Options from PCL IKJEFF16. 16 has the Address of IKJEFF17, Entered as an Exit from PARSE, which Builds DAIR08 Blocks & DD Element Chain. Parse & 16 Loaded here.
RESTACK
D3
Invoke the STACK Rtn
STACK is a Facility in TSO which Controls the Source of Input. Here we Wish to Reset the STACK Ptr to Point to the Terminal as the Source of Input.
AA D3
Yes
E2
Test Return Code from the PARSE Exit ?
Return Codes
0 -- Good Return
4 -- Unable to Prompt so Reset STACK
8 -- ECB Posted
12 -- Invalid Parameter
16 -- No Space Available
Non-zero
E3
Is the Return Code 4 ?
Zero
No
AA F3
FINISH
F3
Free the Storage for DD Element Chain
F2
Was NONOTIFY Specified ?
NONOTIFY Indicates that the Keyword NOTIFY will not be Placed on a Generated JCL Card
Yes
No
G2
Turn NOTIFY Bit On
FIN2
G3
Free the Storage for DAIR08 Blocks
H2
Free the PDE Control Block for Data Set Names & NO/NOTIFY
H3
Turn any Non-zero Return Code into 12
J2
Get Storage for the DAIR28 Blocks; Init DAIR28
DAIR28 Blocks Contain Pointers to all DAIR08 Blocks. DAIR08 Blocks are Equivalent to having a DD Card. The Initialization Process is that Ptrs are Set to the DAIR08 Blocks Built by IKJEFF17.
J3
Put Out PARSE Error Message
C4
K2
CALL IKJDAIR
TSO UTILITY Routine for Dynamic Allocation
AB B2

Chart AB. IKJEFF01. PARSE and ALLOCATE

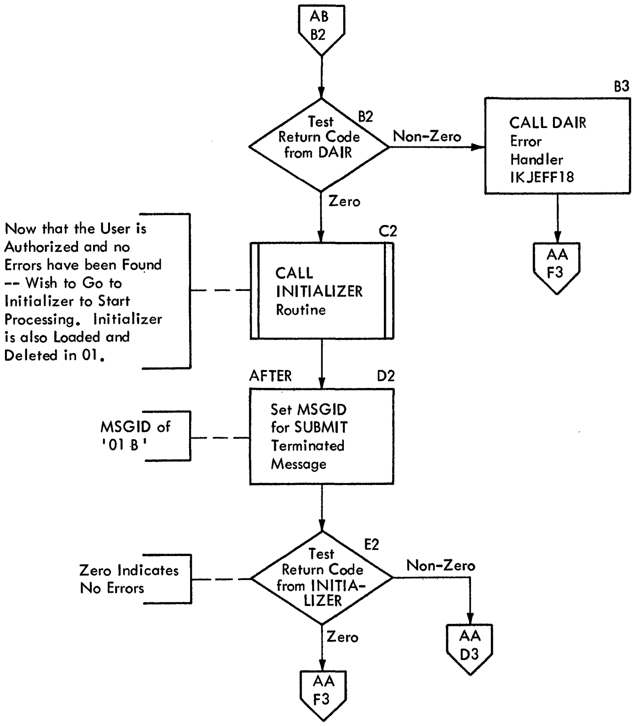
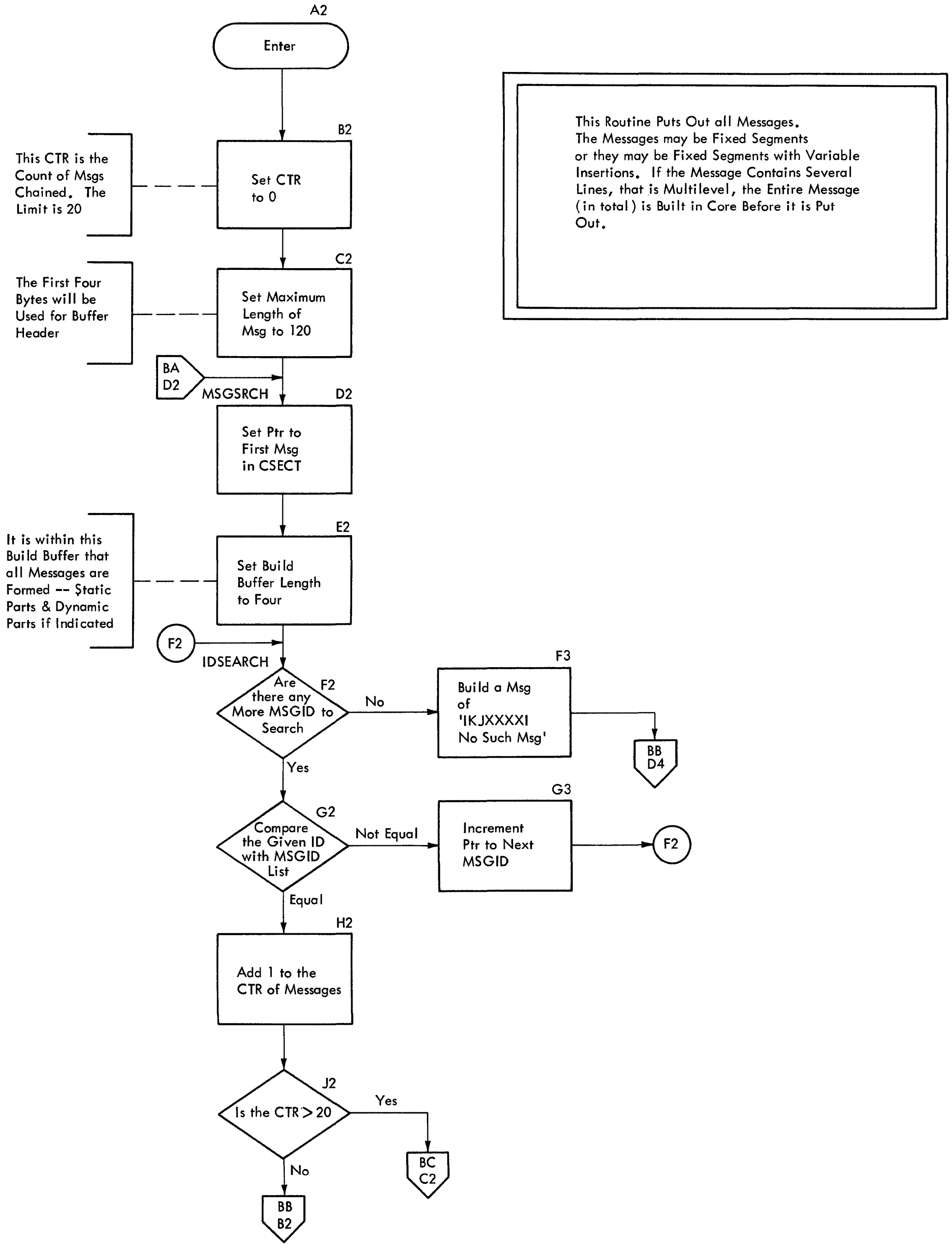

AB B2
Test Return Code from DAIR
B2
Non-Zero
B3
CALL DAIR Error Handler IKJEFF18
AA F3
Zero
Now that the User is Authorized and no Errors have been Found -- Wish to Go to Initializer to Start Processing. Initializer is also Loaded and Deleted in 01.
C2
CALL INITIALIZER Routine
AFTER
D2
MSGID of '01 B'
Set MSGID for SUBMIT Terminated Message
E2
Zero Indicates No Errors
Test Return Code from INITIA-LIZER
Non-Zero
AA D3
Zero
AA F3

Chart BA. IKJEFF02. MESSAGE PROCESSING


A2
Enter
B2
This CTR is the Count of Msgs Chained. The Limit is 20
Set CTR to 0
C2
The First Four Bytes will be Used for Buffer Header
Set Maximum Length of Msg to 120
BA D2
MSGSRCH
D2
Set Ptr to First Msg in CSECT
E2
It is within this Build Buffer that all Messages are Formed -- Static Parts & Dynamic Parts if Indicated
Set Build Buffer Length to Four
F2
IDSEARCH
F2
Are there any More MSGID to Search
No
F3
Build a Msg of 'IKJXXXXI No Such Msg'
BB D4
Yes
G2
Compare the Given ID with MSGID List
Not Equal
G3
Increment Ptr to Next MSGID
F2
Equal
H2
Add 1 to the CTR of Messages
J2
Is the CTR > 20
Yes
BC C2
No
BB B2

This Routine Puts Out all Messages. The Messages may be Fixed Segments or they may be Fixed Segments with Variable Insertions. If the Message Contains Several Lines, that is Multilevel, the Entire Message (in total) is Built in Core Before it is Put Out.

Chart BB. IKJEFF02. MESSAGE PROCESSING

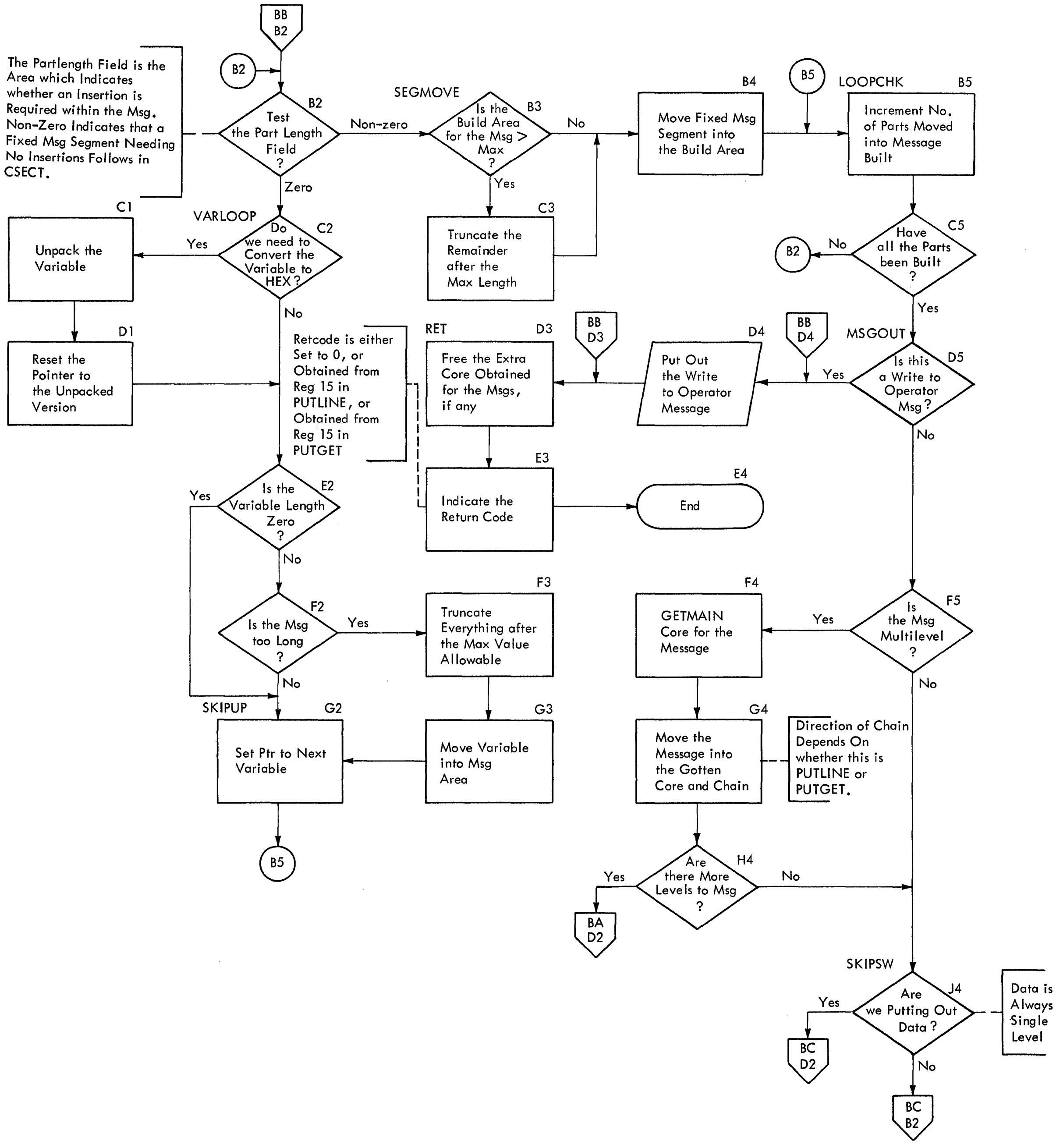

BB B2
B2
The Partlength Field is the Area which Indicates whether an Insertion is Required within the Msg. Non-Zero Indicates that a Fixed Msg Segment Needing No Insertions Follows in CSECT.
B2
Test the Part Length Field ?
Non-zero
Zero
SEGMOVE
B3
Is the Build Area for the Msg > Max ?
No
Yes
B4
Move Fixed Msg Segment into the Build Area
B5
LOOPCHK
B5
Increment No. of Parts Moved into Message Built
C1
Unpack the Variable
VARLOOP
C2
Do we need to Convert the Variable to HEX ?
Yes
No
C3
Truncate the Remainder after the Max Length
C5
Have all the Parts been Built ?
No
B2
Yes
D1
Reset the Pointer to the Unpacked Version
Retcode is either Set to 0, or Obtained from Reg 15 in PUTLINE, or Obtained from Reg 15 in PUTGET
RET
D3
Free the Extra Core Obtained for the Msgs, if any
BB D3
D4
Put Out the Write to Operator Message
BB D4
MSGOUT
D5
Is this a Write to Operator Msg ?
Yes
No
E2
Is the Variable Length Zero ?
Yes
No
E3
Indicate the Return Code
E4
End
F2
Is the Msg too Long ?
Yes
No
F3
Truncate Everything after the Max Value Allowable
F4
GETMAIN Core for the Message
F5
Is the Msg Multilevel ?
Yes
No
SKIPUP
G2
Set Ptr to Next Variable
G3
Move Variable into Msg Area
G4
Move the Message into the Gotten Core and Chain
Direction of Chain Depends On whether this is PUTLINE or PUTGET.
B5
H4
Are there More Levels to Msg ?
Yes
No
BA D2
SKIPSW
J4
Are we Putting Out Data ?
Yes
No
Data is Always Single Level
BC D2
BC B2

Chart BC. IKJEFF02. MESSAGE PROCESSING

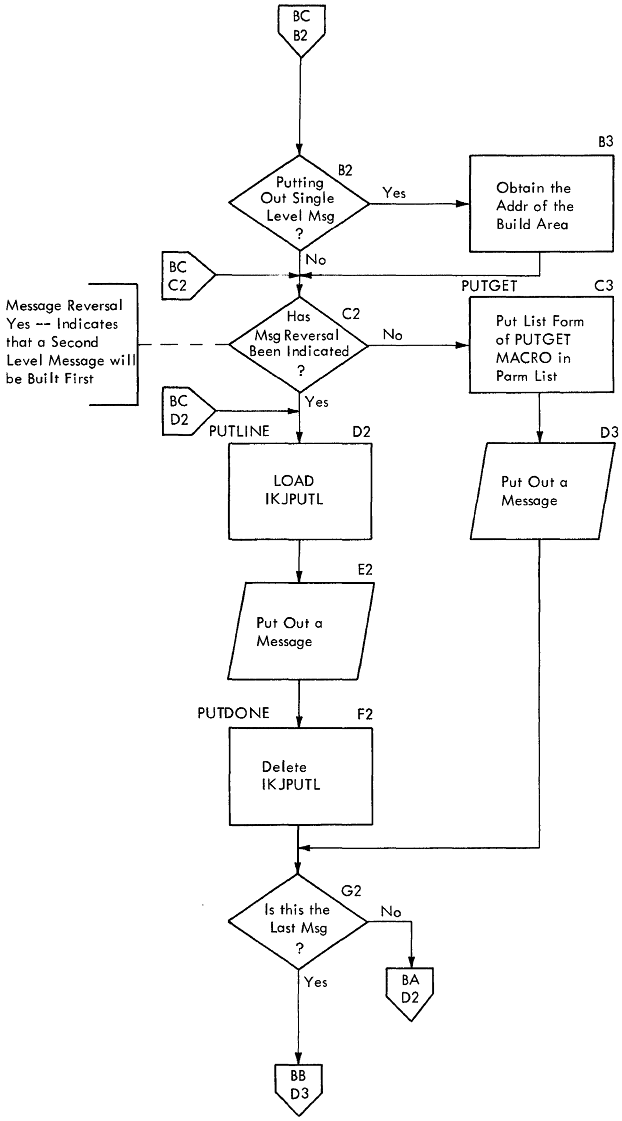

BC
B2
B2
Putting Out Single Level Msg ?
Yes
B3
Obtain the Addr of the Build Area
No
BC
C2
PUTGET
C3
Message Reversal Yes -- Indicates that a Second Level Message will be Built First
Has Msg Reversal Been Indicated ?
C2
No
Put List Form of PUTGET MACRO in Parm List
Yes
BC
D2
PUTLINE
D2
LOAD IKJPUTL
D3
Put Out a Message
E2
Put Out a Message
PUTDONE
F2
Delete IKJPUTL
G2
Is this the Last Msg ?
No
BA
D2
Yes
BB
D3

Chart CA. IKJEFF04. MESSAGE PROCESSING

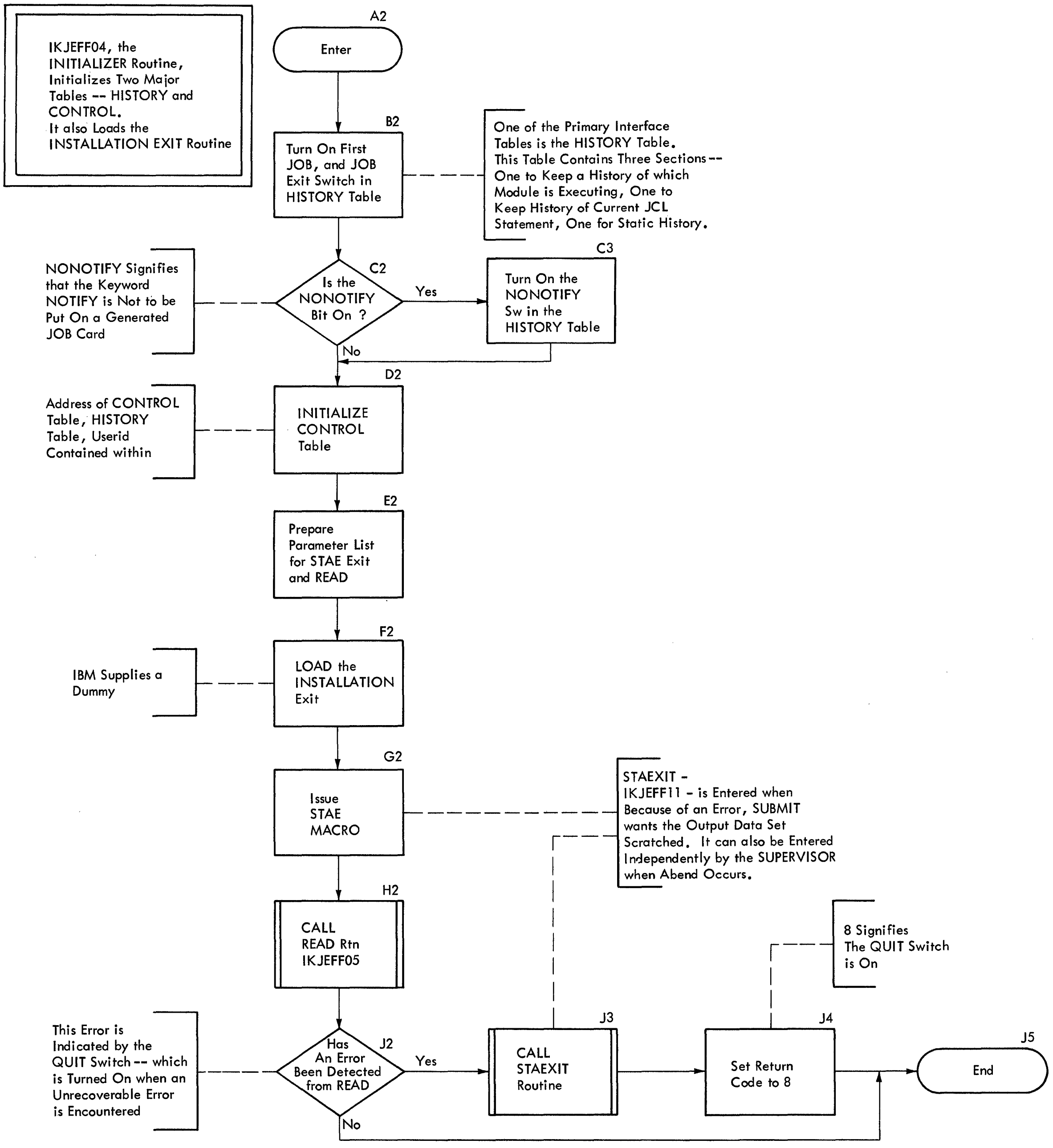

IKJEFF04, the INITIALIZER Routine, Initializes Two Major Tables -- HISTORY and CONTROL. It also Loads the INSTALLATION EXIT Routine
A2
Enter
B2
Turn On First JOB, and JOB Exit Switch in HISTORY Table
One of the Primary Interface Tables is the HISTORY Table. This Table Contains Three Sections -- One to Keep a History of which Module is Executing, One to Keep History of Current JCL Statement, One for Static History.
NONOTIFY Signifies that the Keyword NOTIFY is Not to be Put On a Generated JOB Card
C2
Is the NONOTIFY Bit On ?
Yes
No
C3
Turn On the NONOTIFY Sw in the HISTORY Table
D2
Address of CONTROL Table, HISTORY Table, Userid Contained within
INITIALIZE CONTROL Table
E2
Prepare Parameter List for STAE Exit and READ
F2
IBM Supplies a Dummy
LOAD the INSTALLATION Exit
G2
Issue STAE MACRO
STAEXIT - IKJEFF11 - is Entered when Because of an Error, SUBMIT wants the Output Data Set Scratched. It can also be Entered Independently by the SUPERVISOR when Abend Occurs.
H2
CALL READ Rtn IKJEFF05
8 Signifies The QUIT Switch is On
This Error is Indicated by the QUIT Switch -- which is Turned On when an Unrecoverable Error is Encountered
J2
Has An Error Been Detected from READ
Yes
No
J3
CALL STAEXIT Routine
J4
Set Return Code to 8
J5
End

Chart DA. IKJEFF05. READ

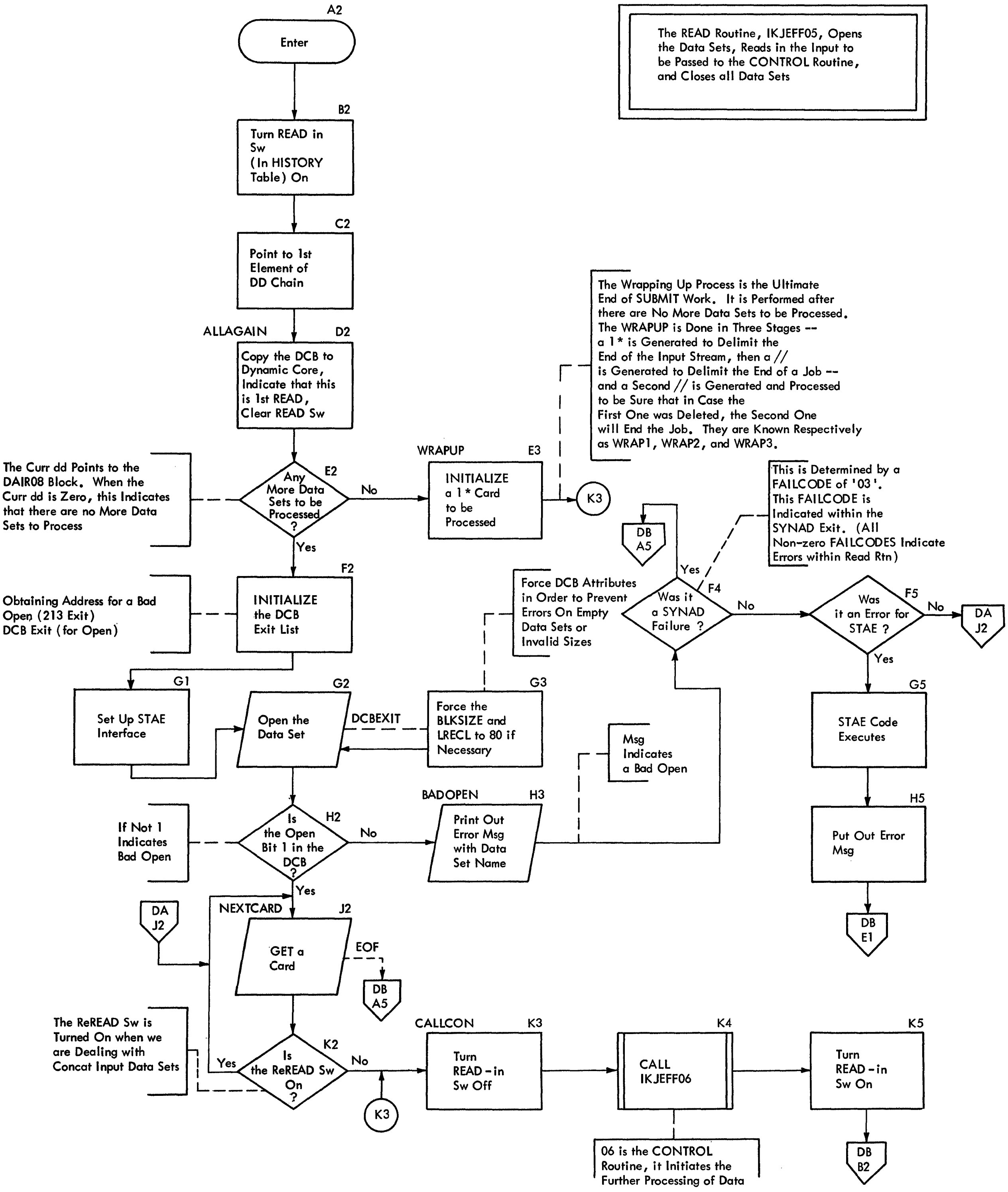

The READ Routine, IKJEFF05, Opens the Data Sets, Reads in the Input to be Passed to the CONTROL Routine, and Closes all Data Sets
A2
Enter
B2
Turn READ in Sw (In HISTORY Table) On
C2
Point to 1st Element of DD Chain
ALLAGAIN
D2
Copy the DCB to Dynamic Core, Indicate that this is 1st READ, Clear READ Sw
The Curr dd Points to the DAIR08 Block. When the Curr dd is Zero, this Indicates that there are no More Data Sets to Process
E2
Any More Data Sets to be Processed ?
No
Yes
WRAPUP
E3
INITIALIZE a 1 * Card to be Processed
K3
The Wrapping Up Process is the Ultimate End of SUBMIT Work. It is Performed after there are No More Data Sets to be Processed. The WRAPUP is Done in Three Stages -- a 1* is Generated to Delimit the End of the Input Stream, then a // is Generated to Delimit the End of a Job -- and a Second // is Generated and Processed to be Sure that in Case the First One was Deleted, the Second One will End the Job. They are Known Respectively as WRAP1, WRAP2, and WRAP3.
Obtaining Address for a Bad Open (213 Exit) DCB Exit (for Open)
F2
INITIALIZE the DCB Exit List
G1
Set Up STAE Interface
G2
Open the Data Set
DCBEXIT
G3
Force the BLKSIZE and LRECL to 80 if Necessary
Force DCB Attributes in Order to Prevent Errors On Empty Data Sets or Invalid Sizes
DB A5
Yes
F4
Was it a SYNAD Failure ?
No
This is Determined by a FAILCODE of '03'. This FAILCODE is Indicated within the SYNAD Exit. (All Non-zero FAILCODES Indicate Errors within Read Rtn)
F5
Was it an Error for STAE ?
No
DA J2
Yes
G5
STAE Code Executes
H5
Put Out Error Msg
DB E1
If Not 1 Indicates Bad Open
H2
Is the Open Bit 1 in the DCB ?
No
Yes
BADOPEN
H3
Print Out Error Msg with Data Set Name
Msg Indicates a Bad Open
DA J2
NEXTCARD
J2
GET a Card
EOF
DB A5
The ReREAD Sw is Turned On when we are Dealing with Concat Input Data Sets
K2
Yes
Is the ReREAD Sw On ?
No
K3
CALLCON
K3
Turn READ-in Sw Off
K4
CALL IKJEFF06
06 is the CONTROL Routine, it Initiates the Further Processing of Data
K5
Turn READ-in Sw On
DB B2

Chart DB. IKJEFF05. READ

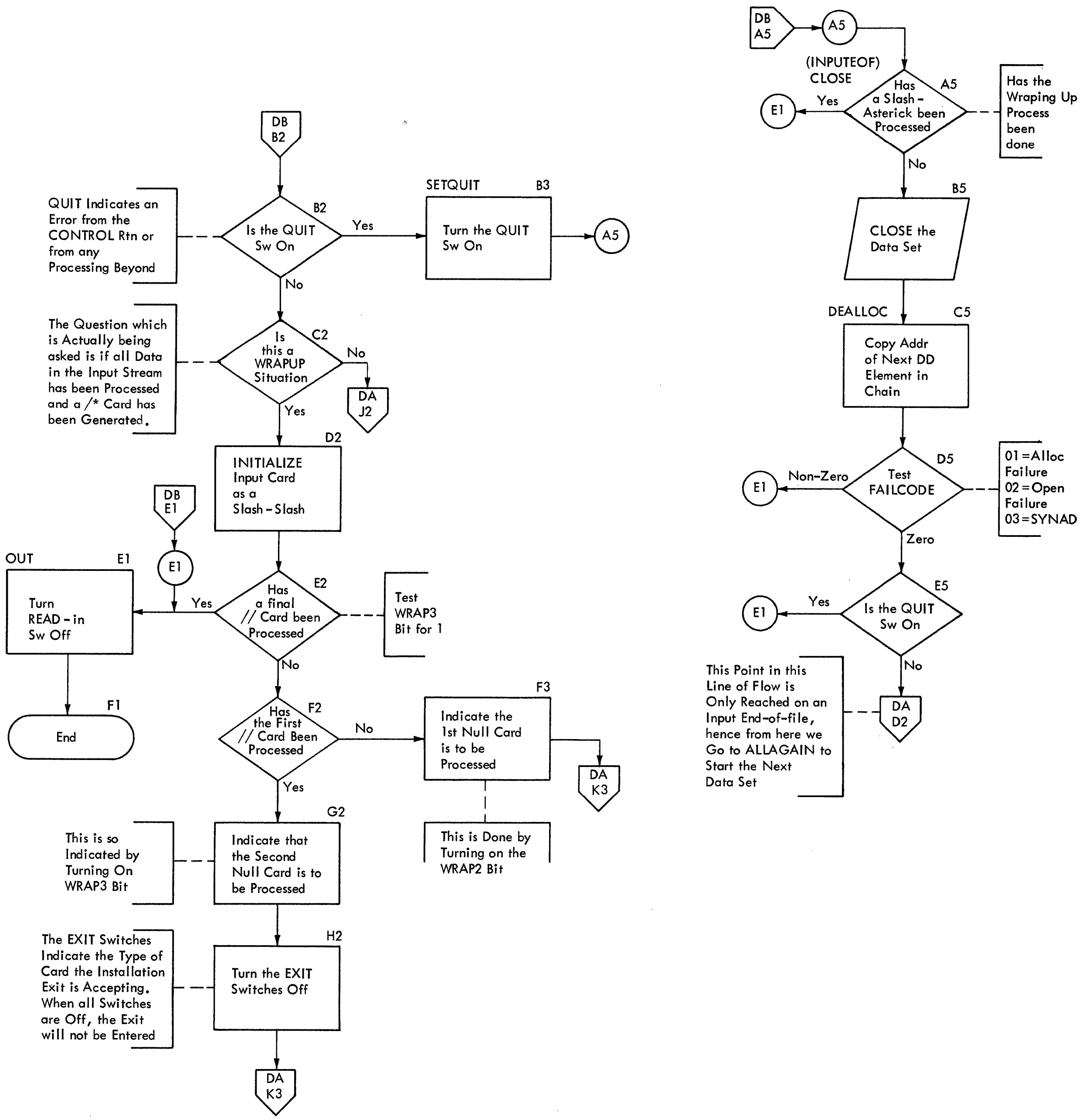

DB B2
QUIT Indicates an Error from the CONTROL Rtn or from any Processing Beyond
B2
Is the QUIT Sw On
Yes
SETQUIT B3
Turn the QUIT Sw On
A5
No
The Question which is Actually being asked is if all Data in the Input Stream has been Processed and a /* Card has been Generated.
C2
Is this a WRAPUP Situation
No
DA J2
Yes
D2
INITIALIZE Input Card as a Slash-Slash
DB E1
E1
OUT E1
Turn READ-in Sw Off
Yes
E2
Has a final // Card been Processed
Test WRAP3 Bit for 1
No
F1
End
F2
Has the First // Card Been Processed
No
F3
Indicate the 1st Null Card is to be Processed
DA K3
This is Done by Turning on the WRAP2 Bit
Yes
G2
This is so Indicated by Turning On WRAP3 Bit
Indicate that the Second Null Card is to be Processed
H2
The EXIT Switches Indicate the Type of Card the Installation Exit is Accepting. When all Switches are Off, the Exit will not be Entered
Turn the EXIT Switches Off
DA K3
DB A5
A5
(INPUTEOF) CLOSE
A5
Has a Slash-Asterick been Processed
Yes
E1
Has the Wraping Up Process been done
No
B5
CLOSE the Data Set
DEALLOC C5
Copy Addr of Next DD Element in Chain
D5
Test FAILCODE
Non-Zero
E1
01=Alloc Failure 02=Open Failure 03=SYNAD
Zero
E5
Is the QUIT Sw On
Yes
E1
No
This Point in this Line of Flow is Only Reached on an Input End-of-file, hence from here we Go to ALLAGAIN to Start the Next Data Set
DA D2

Chart EA. IKJEFF06. CONTROL

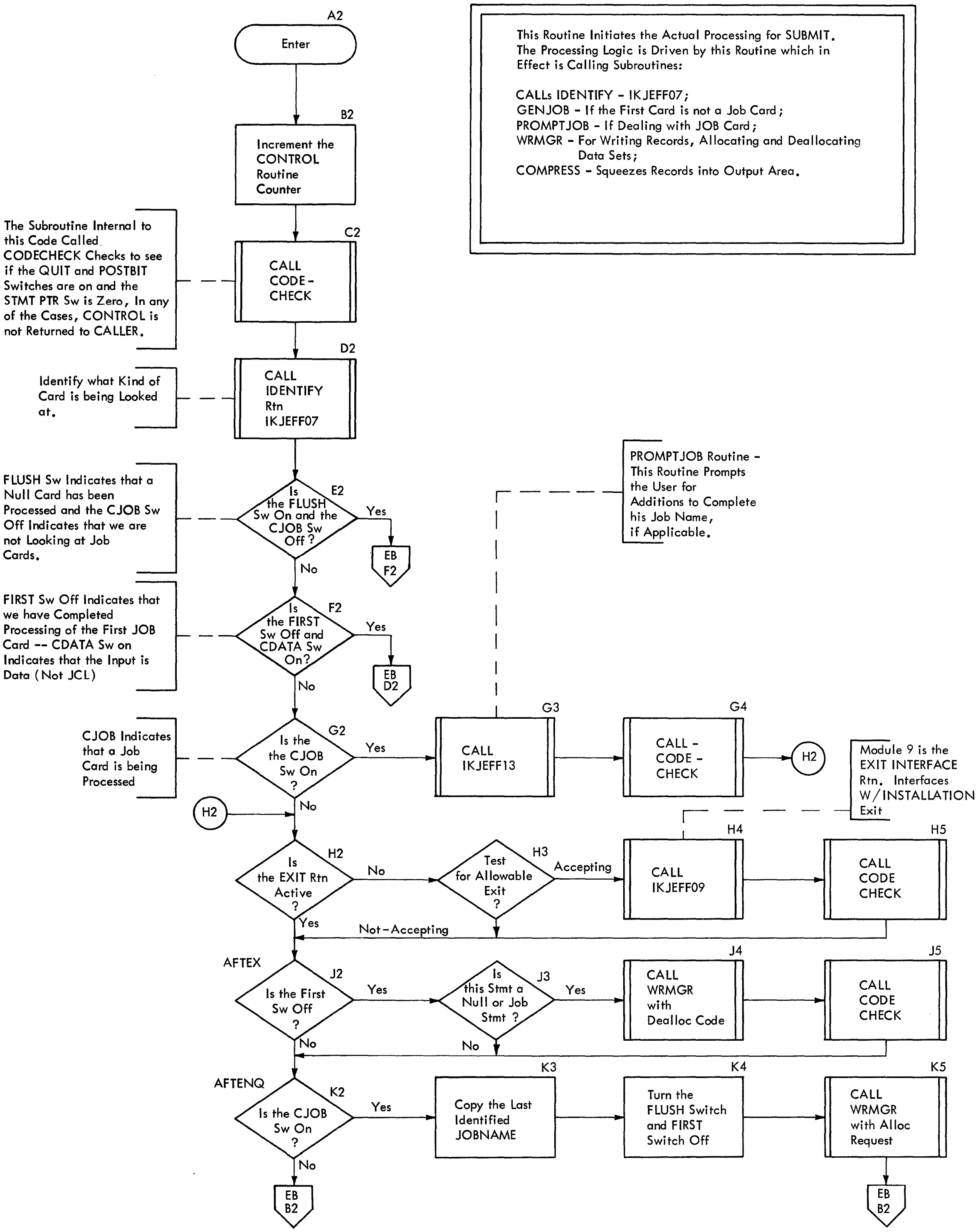

This Routine Initiates the Actual Processing for SUBMIT. The Processing Logic is Driven by this Routine which in Effect is Calling Subroutines:
CALLs IDENTIFY - IKJEFF07;
GENJOB - If the First Card is not a Job Card;
PROMPTJOB - If Dealing with JOB Card;
WRMGR - For Writing Records, Allocating and Deallocating Data Sets;
COMPRESS - Squeezes Records into Output Area.
A2
Enter
B2
Increment the CONTROL Routine Counter
The Subroutine Internal to this Code Called CODECHECK Checks to see if the QUIT and POSTBIT Switches are on and the STMT PTR Sw is Zero, In any of the Cases, CONTROL is not Returned to CALLER.
C2
CALL CODE-CHECK
D2
Identify what Kind of Card is being Looked at.
CALL IDENTIFY Rtn IKJEFF07
FLUSH Sw Indicates that a Null Card has been Processed and the CJOB Sw Off Indicates that we are not Looking at Job Cards.
E2
Is the FLUSH Sw On and the CJOB Sw Off ?
Yes
EB F2
No
FIRST Sw Off Indicates that we have Completed Processing of the First JOB Card -- CDATA Sw on Indicates that the Input is Data (Not JCL)
F2
Is the FIRST Sw Off and CDATA Sw On?
Yes
EB D2
No
PROMPTJOB Routine - This Routine Prompts the User for Additions to Complete his Job Name, if Applicable.
CJOB Indicates that a Job Card is being Processed
G2
Is the the CJOB Sw On ?
Yes
G3
CALL IKJEFF13
G4
CALL - CODE - CHECK
H2
Module 9 is the EXIT INTERFACE Rtn. Interfaces W/INSTALLATION Exit
No
H2
H2
Is the EXIT Rtn Active ?
No
H3
Test for Allowable Exit ?
Accepting
H4
CALL IKJEFF09
H5
CALL CODE CHECK
Yes
Not-Accepting
AFTEX
J2
Is the First Sw Off ?
Yes
J3
Is this Stmt a Null or Job Stmt ?
Yes
J4
CALL WRMGR with Dealloc Code
J5
CALL CODE CHECK
No
No
AFTENQ
K2
Is the CJOB Sw On ?
Yes
K3
Copy the Last Identified JOBNAME
K4
Turn the FLUSH Switch and FIRST Switch Off
K5
CALL WRMGR with Alloc Request
No
EB B2
EB B2

Chart EB. IKJEFF06. CONTROL

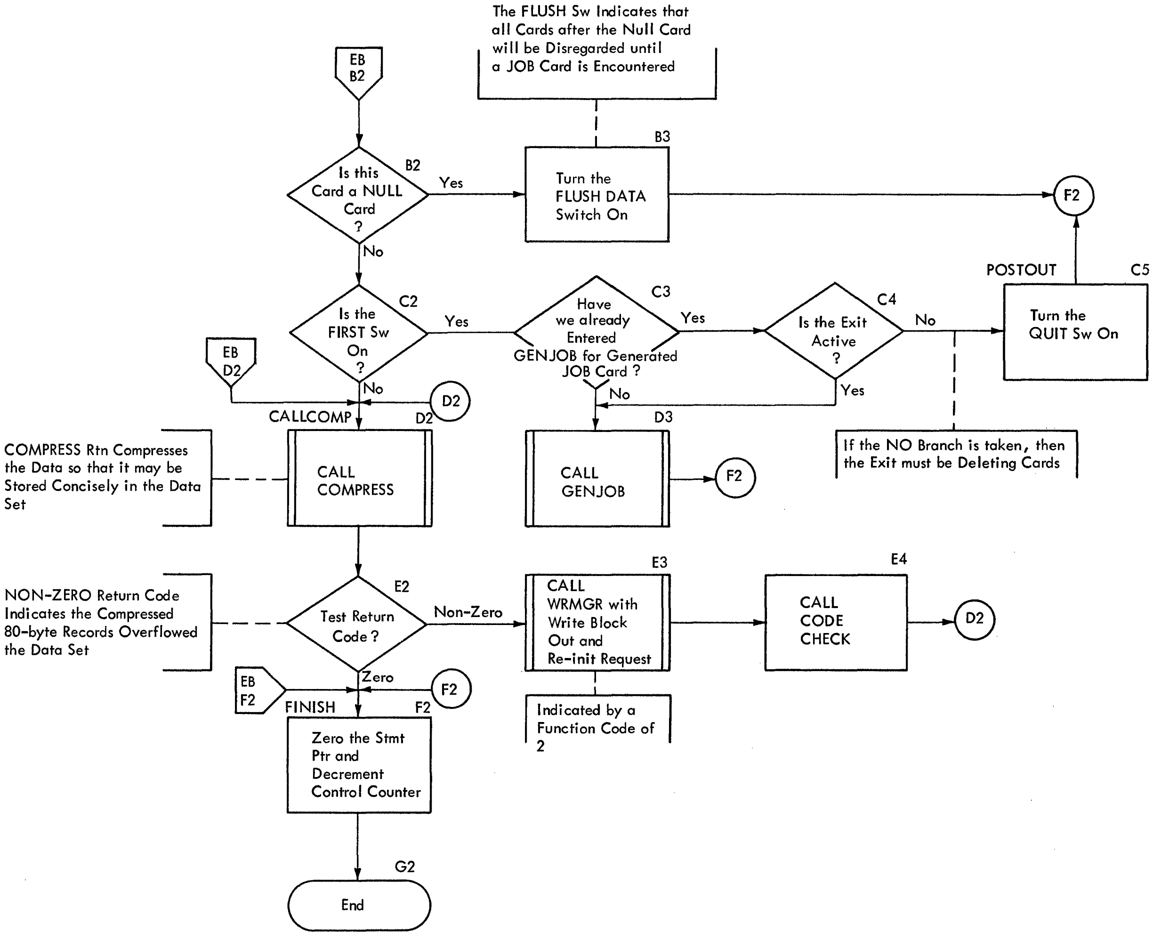
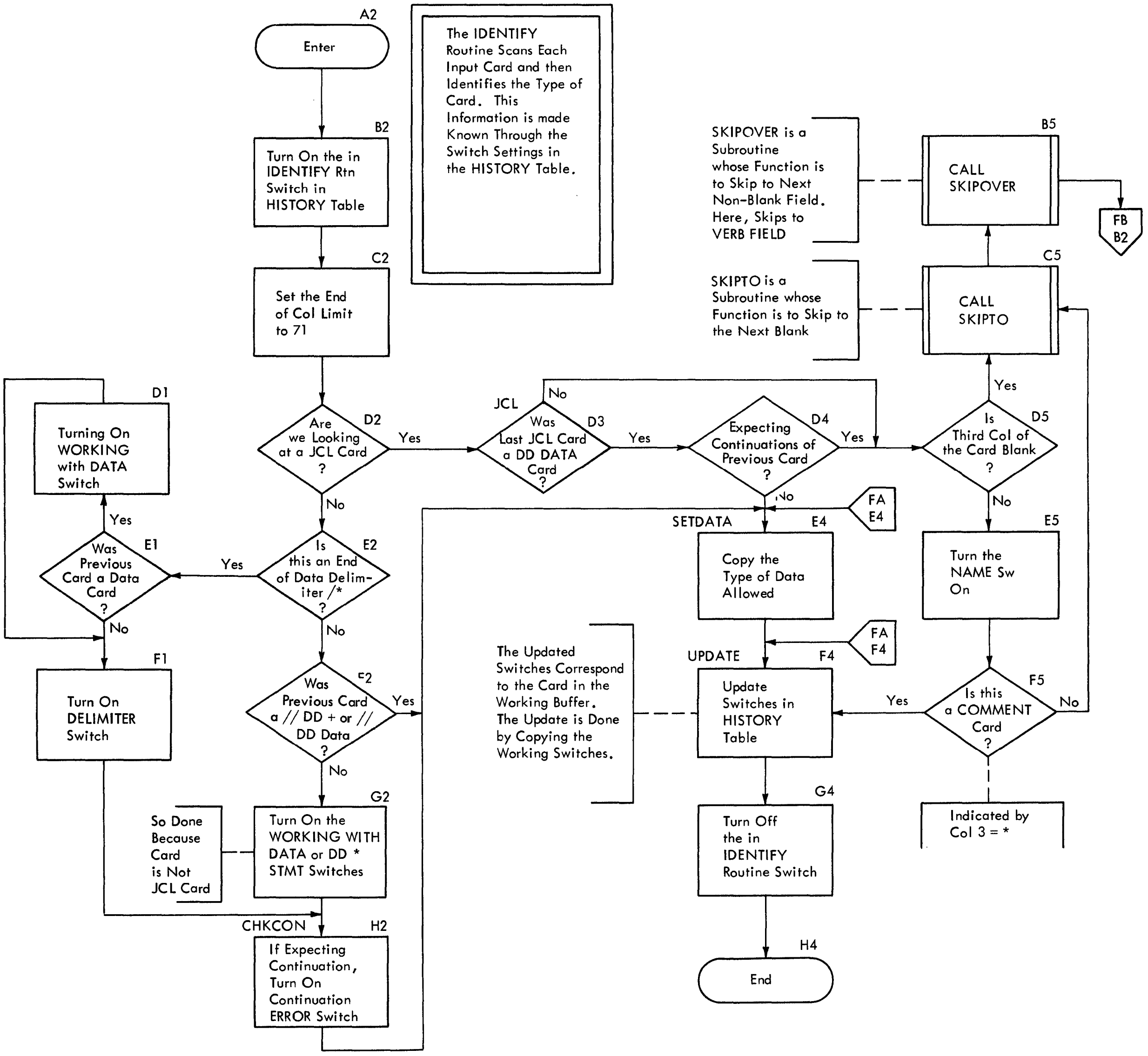
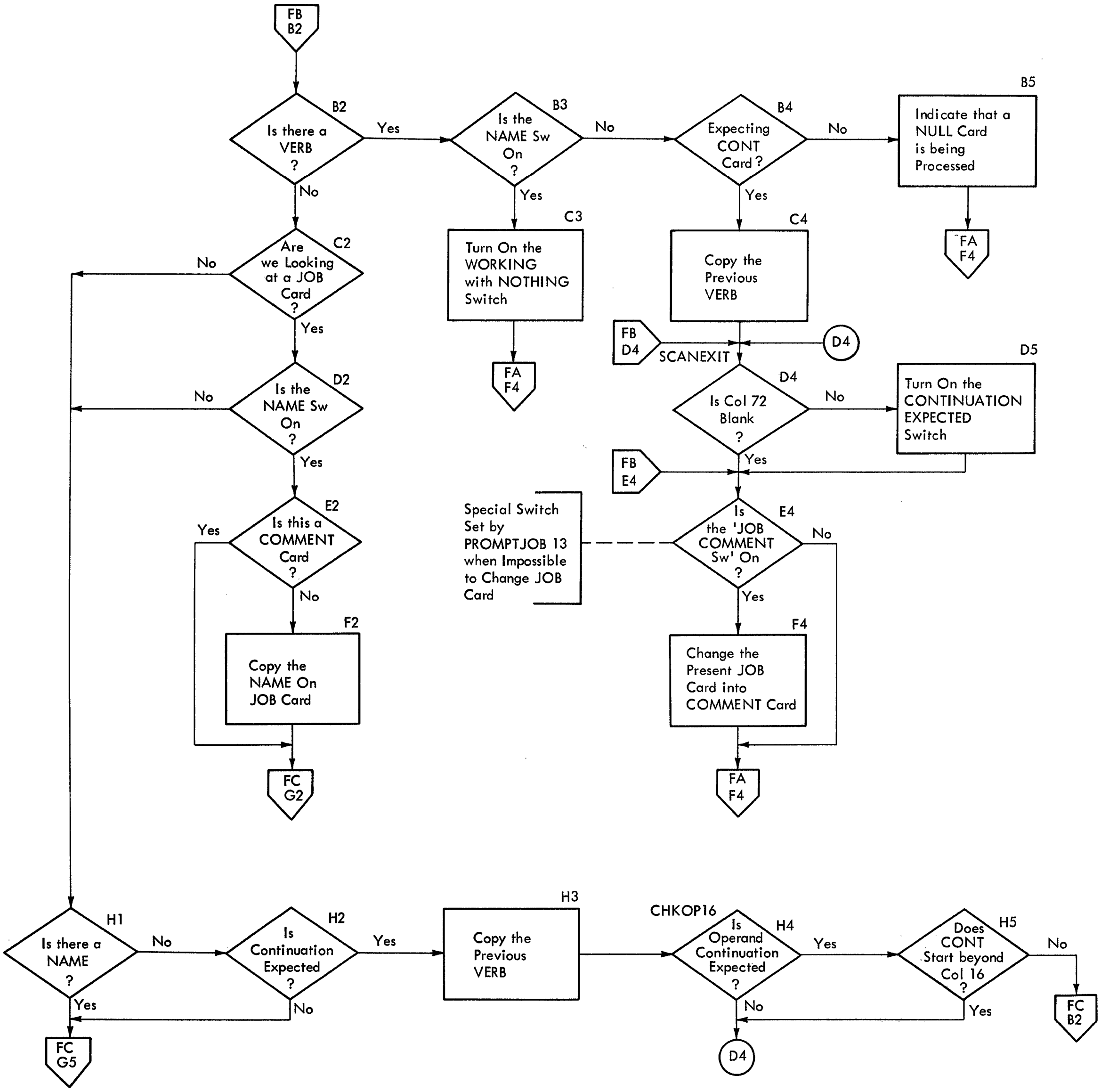

The FLUSH Sw Indicates that all Cards after the Null Card will be Disregarded until a JOB Card is Encountered
EB B2
B2
Is this Card a NULL Card ?
Yes
B3
Turn the FLUSH DATA Switch On
F2
No
POSTOUT
C5
C2
Is the FIRST Sw On ?
Yes
C3
Have we already Entered GENJOB for Generated JOB Card ?
Yes
C4
Is the Exit Active ?
No
Turn the QUIT Sw On
EB D2
No
D2
No
Yes
CALLCOMP
D2
D3
If the NO Branch is taken, then the Exit must be Deleting Cards
COMPRESS Rtn Compresses the Data so that it may be Stored Concisely in the Data Set
CALL COMPRESS
CALL GENJOB
F2
E3
E4
E2
NON-ZERO Return Code Indicates the Compressed 80-byte Records Overflowed the Data Set
Test Return Code ?
Non-Zero
CALL WRMGR with Write Block Out and Re-init Request
CALL CODE CHECK
D2
Zero
EB F2
F2
FINISH
F2
Indicated by a Function Code of 2
Zero the Stmt Ptr and Decrement Control Counter
G2
End

Chart FA. IKJEFF07. IDENTIFY


A2
Enter
The IDENTIFY Routine Scans Each Input Card and then Identifies the Type of Card. This Information is made Known Through the Switch Settings in the HISTORY Table.
B2
Turn On the in IDENTIFY Rtn Switch in HISTORY Table
C2
Set the End of Col Limit to 71
SKIPOVER is a Subroutine whose Function is to Skip to Next Non-Blank Field. Here, Skips to VERB FIELD
B5
CALL SKIPOVER
FB B2
SKIPTO is a Subroutine whose Function is to Skip to the Next Blank
C5
CALL SKIPTO
D1
Turning On WORKING with DATA Switch
D2
Are we Looking at a JCL Card ?
Yes
JCL
No
D3
Was Last JCL Card a DD DATA Card ?
Yes
D4
Expecting Continuations of Previous Card ?
Yes
D5
Is Third Col of the Card Blank ?
Yes
No
No
FA E4
SETDATA
E4
Copy the Type of Data Allowed
E5
Turn the NAME Sw On
Yes
E1
Was Previous Card a Data Card ?
Yes
E2
Is this an End of Data Delimiter /* ?
No
No
No
F1
Turn On DELIMITER Switch
F2
Was Previous Card a // DD + or // DD Data ?
Yes
No
The Updated Switches Correspond to the Card in the Working Buffer. The Update is Done by Copying the Working Switches.
FA F4
UPDATE
F4
Update Switches in HISTORY Table
F5
Is this a COMMENT Card ?
Yes
No
Indicated by Col 3 = *
So Done Because Card is Not JCL Card
G2
Turn On the WORKING WITH DATA or DD * STMT Switches
G4
Turn Off the in IDENTIFY Routine Switch
CHKCON
H2
If Expecting Continuation, Turn On Continuation ERROR Switch
H4
End


FB B2
B2
Is there a VERB ?
Yes
No
B3
Is the NAME Sw On ?
No
Yes
B4
Expecting CONT Card ?
No
Yes
B5
Indicate that a NULL Card is being Processed
FA F4
C2
Are we Looking at a JOB Card ?
No
Yes
C3
Turn On the WORKING with NOTHING Switch
FA F4
C4
Copy the Previous VERB
FB D4
D4
SCANEXIT
D2
Is the NAME Sw On ?
No
Yes
D4
Is Col 72 Blank ?
No
Yes
D5
Turn On the CONTINUATION EXPECTED Switch
FB E4
E2
Is this a COMMENT Card ?
Yes
No
Special Switch Set by PROMPTJOB 13 when Impossible to Change JOB Card
E4
Is the 'JOB COMMENT Sw' On ?
No
Yes
F2
Copy the NAME On JOB Card
F4
Change the Present JOB Card into COMMENT Card
FC G2
FA F4
H1
Is there a NAME ?
No
Yes
FC G5
H2
Is Continuation Expected ?
Yes
No
H3
Copy the Previous VERB
CHKOP16
H4
Is Operand Continuation Expected ?
Yes
No
D4
H5
Does CONT Start beyond Col 16 ?
No
Yes
FC B2

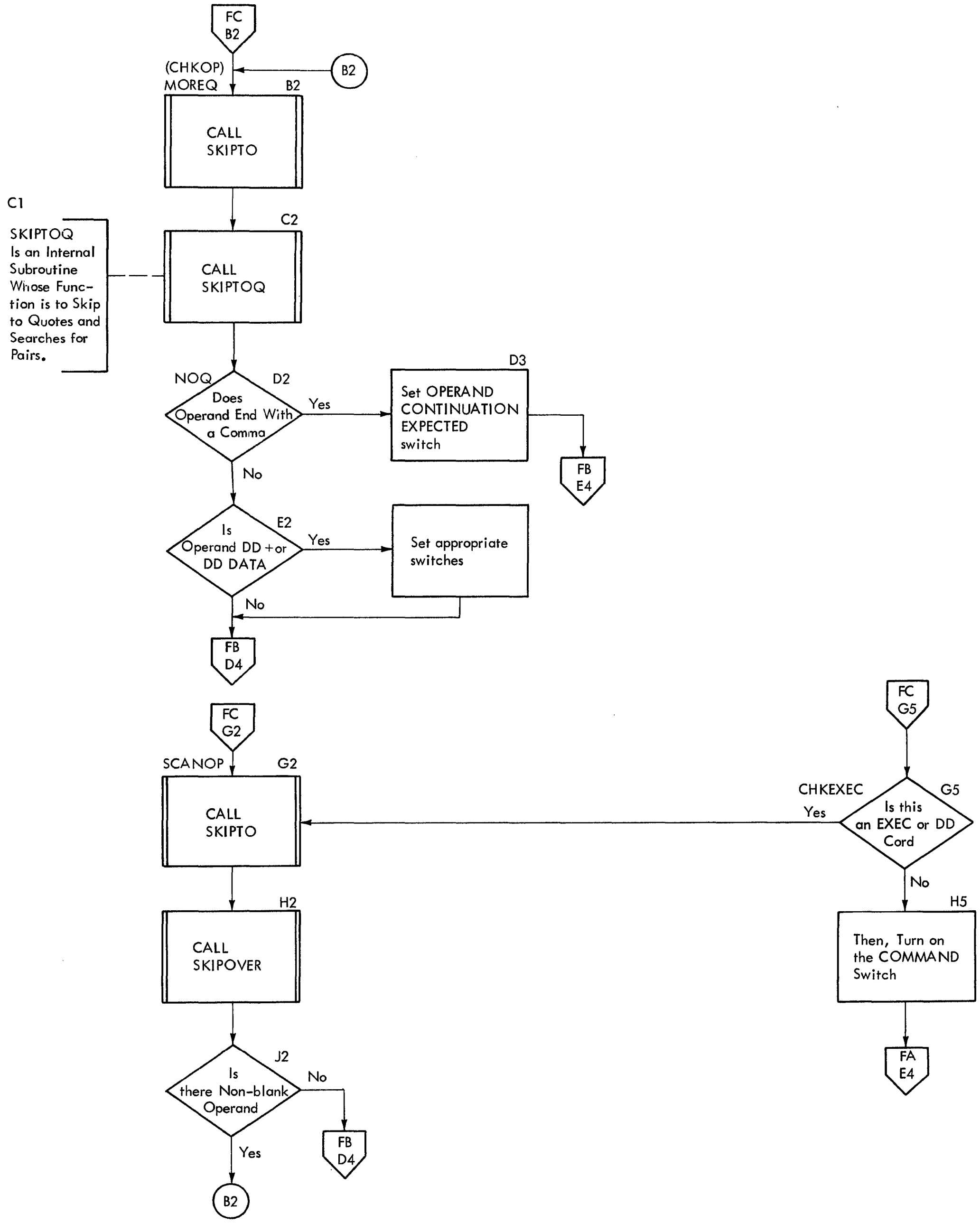
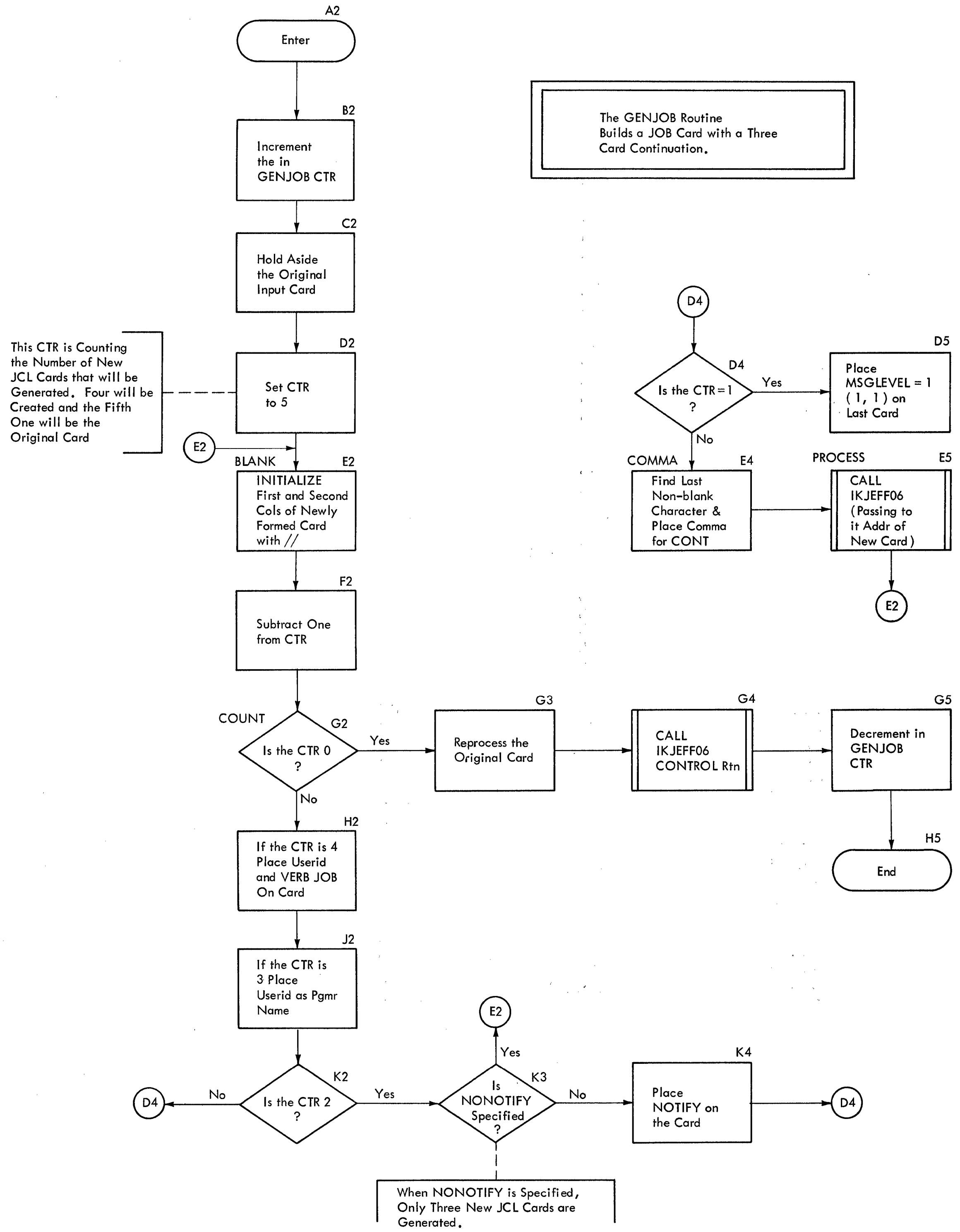

FC B2
(CHKOP)
MOREQ
B2
B2
CALL SKIPTO
C1
SKIPTOQ Is an Internal Subroutine Whose Function is to Skip to Quotes and Searches for Pairs.
C2
CALL SKIPTOQ
NOQ
D2
Does Operand End With a Comma
Yes
No
D3
Set OPERAND CONTINUATION EXPECTED switch
FB E4
E2
Is Operand DD +or DD DATA
Yes
No
Set appropriate switches
FB D4
FC G2
SCANOP
G2
CALL SKIPTO
H2
CALL SKIPOVER
J2
Is there Non-blank Operand
No
Yes
FB D4
B2
FC G5
CHKEXEC
G5
Yes
Is this an EXEC or DD Cord
No
H5
Then, Turn on the COMMAND Switch
FA E4

Chart GA. IKJEFF08. GENJOB


The GENJOB Routine Builds a JOB Card with a Three Card Continuation.

This CTR is Counting the Number of New JCL Cards that will be Generated. Four will be Created and the Fifth One will be the Original Card

When NONOTIFY is Specified, Only Three New JCL Cards are Generated.

A2
Enter
B2
Increment the in GENJOB CTR
C2
Hold Aside the Original Input Card
D2
Set CTR to 5
E2
BLANK
E2
INITIALIZE First and Second Cols of Newly Formed Card with //
F2
Subtract One from CTR
COUNT
G2
Is the CTR 0 ?
Yes
No
G3
Reprocess the Original Card
G4
CALL IKJEFF06 CONTROL Rtn
G5
Decrement in GENJOB CTR
H5
End
H2
If the CTR is 4 Place Userid and VERB JOB On Card
J2
If the CTR is 3 Place Userid as Pgmr Name
K2
Is the CTR 2 ?
No
D4
Yes
K3
Is NONOTIFY Specified ?
Yes
E2
No
K4
Place NOTIFY on the Card
D4
D4
D4
Is the CTR=1 ?
Yes
No
D5
Place MSGLEVEL = 1 (1, 1) on Last Card
COMMA
E4
Find Last Non-blank Character & Place Comma for CONT
PROCESS
E5
CALL IKJEFF06 (Passing to it Addr of New Card)
E2

Chart HA. IKJEFF09. EXIT INTERFACE

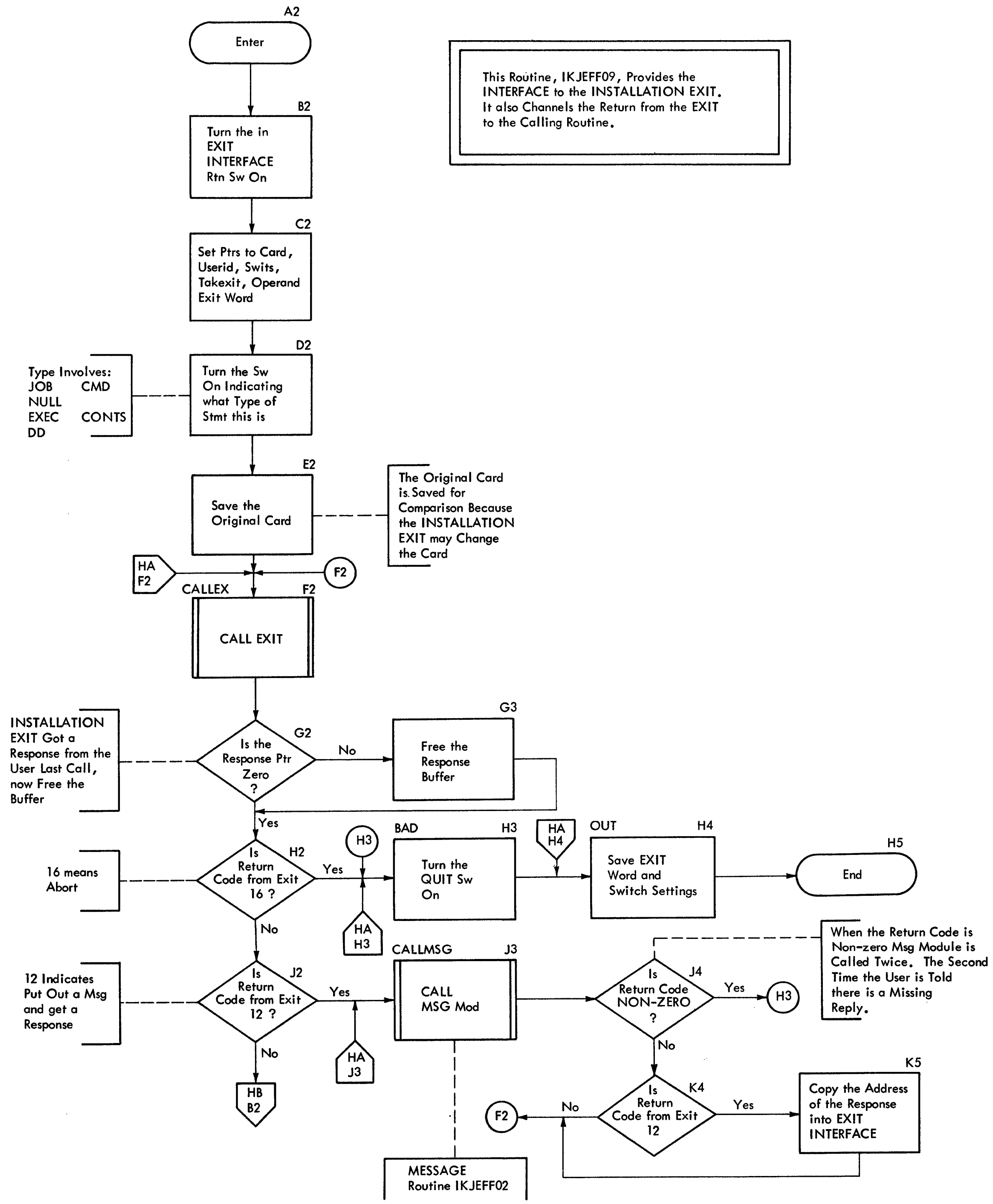

A2
Enter
This Routine, IKJEFF09, Provides the INTERFACE to the INSTALLATION EXIT. It also Channels the Return from the EXIT to the Calling Routine.
B2
Turn the in EXIT INTERFACE Rtn Sw On
C2
Set Ptrs to Card, Userid, Swits, Takexit, Operand Exit Word
D2
Type Involves: JOB CMD NULL EXEC CONTS DD
Turn the Sw On Indicating what Type of Stmt this is
E2
Save the Original Card
The Original Card is Saved for Comparison Because the INSTALLATION EXIT may Change the Card
HA F2
F2
CALLEX
F2
CALL EXIT
INSTALLATION EXIT Got a Response from the User Last Call, now Free the Buffer
G2
Is the Response Ptr Zero ?
No
G3
Free the Response Buffer
Yes
16 means Abort
H2
Is Return Code from Exit 16 ?
Yes
H3
BAD
H3
Turn the QUIT Sw On
HA H4
OUT
H4
Save EXIT Word and Switch Settings
H5
End
HA H3
No
12 Indicates Put Out a Msg and get a Response
J2
Is Return Code from Exit 12 ?
Yes
HA J3
CALLMSG
J3
CALL MSG Mod
J4
Is Return Code NON-ZERO ?
Yes
H3
When the Return Code is Non-zero Msg Module is Called Twice. The Second Time the User is Told there is a Missing Reply.
No
HB B2
No
K4
Is Return Code from Exit 12
Yes
K5
Copy the Address of the Response into EXIT INTERFACE
No
F2
MESSAGE Routine IKJEFF02

Chart HB. IKJEFF09. EXIT INTERFACE

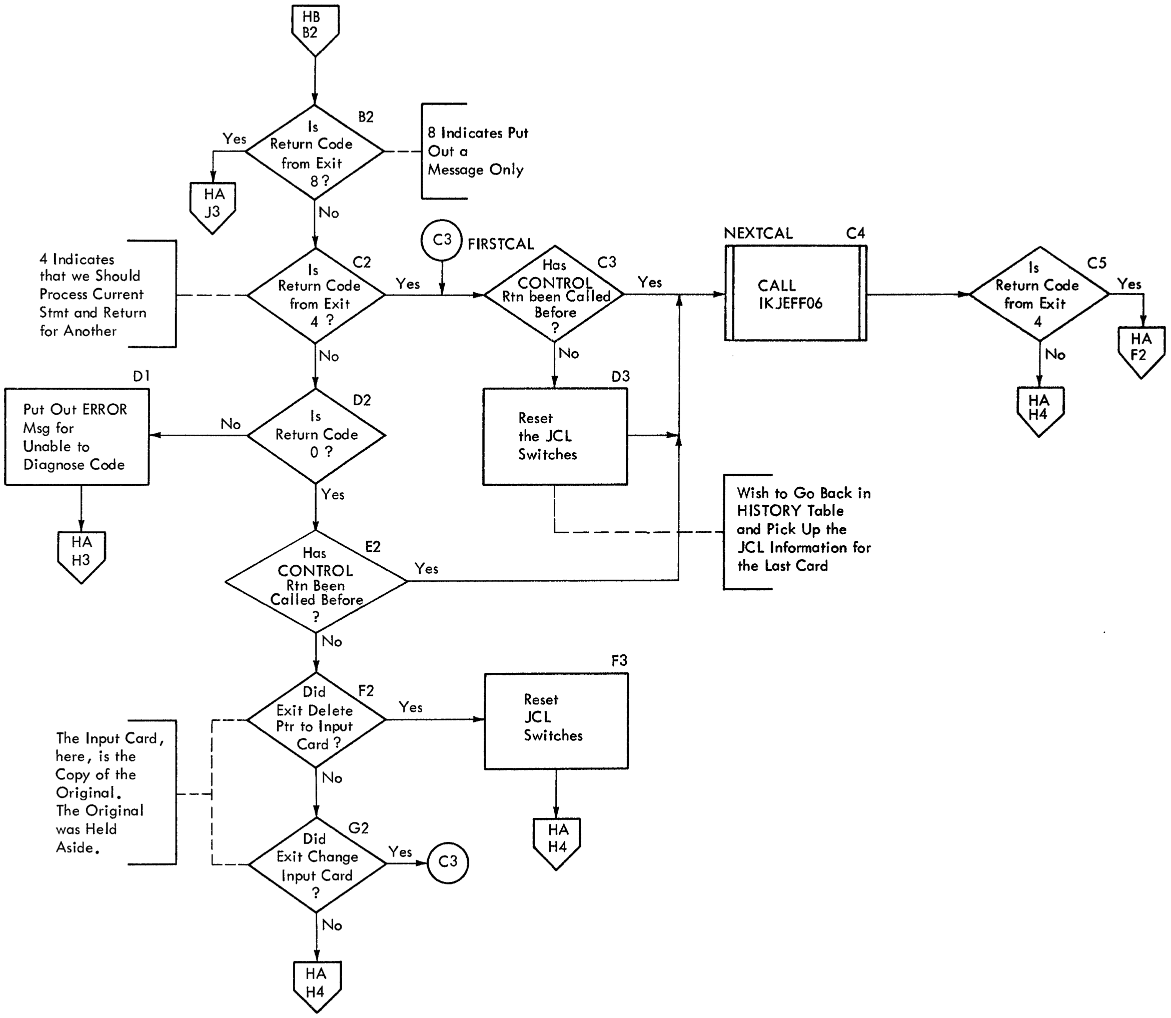

HB B2
B2
Is Return Code from Exit 8 ?
Yes
HA J3
No
8 Indicates Put Out a Message Only
4 Indicates that we Should Process Current Stmt and Return for Another
C2
Is Return Code from Exit 4 ?
Yes
C3 FIRSTCAL
C3
Has CONTROL Rtn been Called Before ?
Yes
NEXTCAL C4
CALL IKJEFF06
C5
Is Return Code from Exit 4
Yes
HA F2
No
HA H4
No
D1
Put Out ERROR Msg for Unable to Diagnose Code
No
D2
Is Return Code 0 ?
No
D3
Reset the JCL Switches
HA H3
Yes
Wish to Go Back in HISTORY Table and Pick Up the JCL Information for the Last Card
E2
Has CONTROL Rtn Been Called Before ?
Yes
No
F2
Did Exit Delete Ptr to Input Card ?
Yes
F3
Reset JCL Switches
HA H4
The Input Card, here, is the Copy of the Original. The Original was Held Aside.
No
G2
Did Exit Change Input Card ?
Yes
C3
No
HA H4

Chart IA. IKJEFF10. DUMMY EXIT

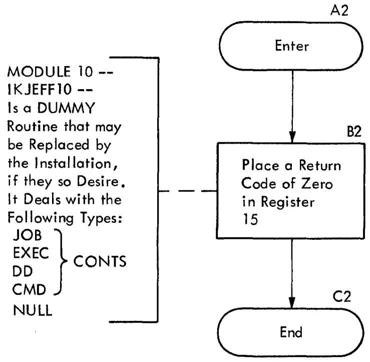


Chart JA. IKJEFF11. STAE EXIT

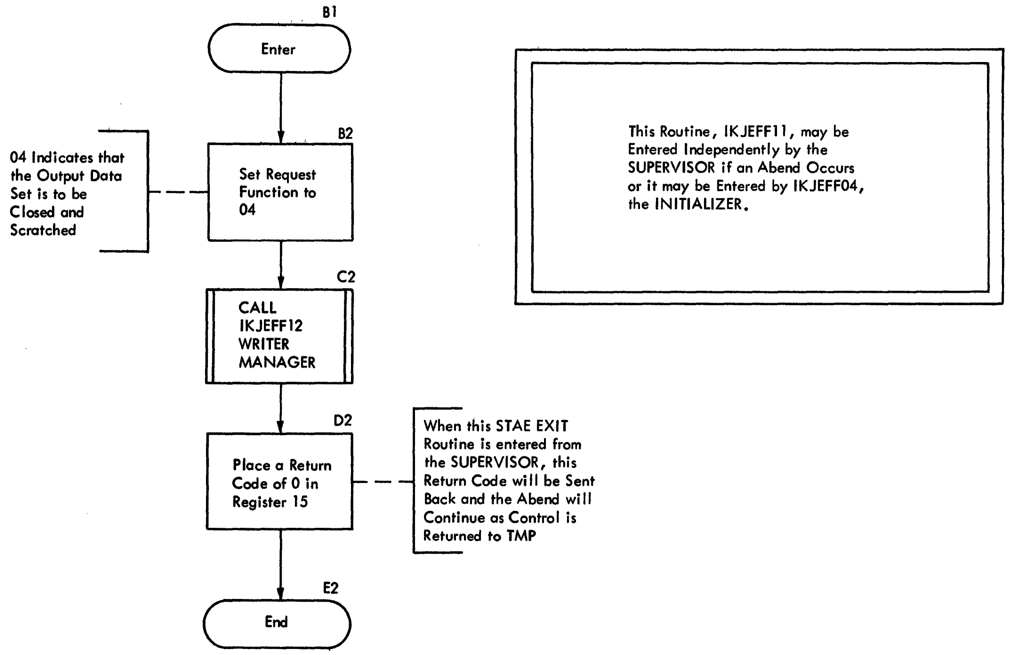

B1
Enter
B2
Set Request Function to 04
04 Indicates that the Output Data Set is to be Closed and Scratched
C2
CALL IKJEFF12 WRITER MANAGER
D2
Place a Return Code of 0 in Register 15
When this STAE EXIT Routine is entered from the SUPERVISOR, this Return Code will be Sent Back and the Abend will Continue as Control is Returned to TMP
E2
End
This Routine, IKJEFF11, may be Entered Independently by the SUPERVISOR if an Abend Occurs or it may be Entered by IKJEFF04, the INITIALIZER.

Chart KA. IKJEFF12. WRITER MANAGER

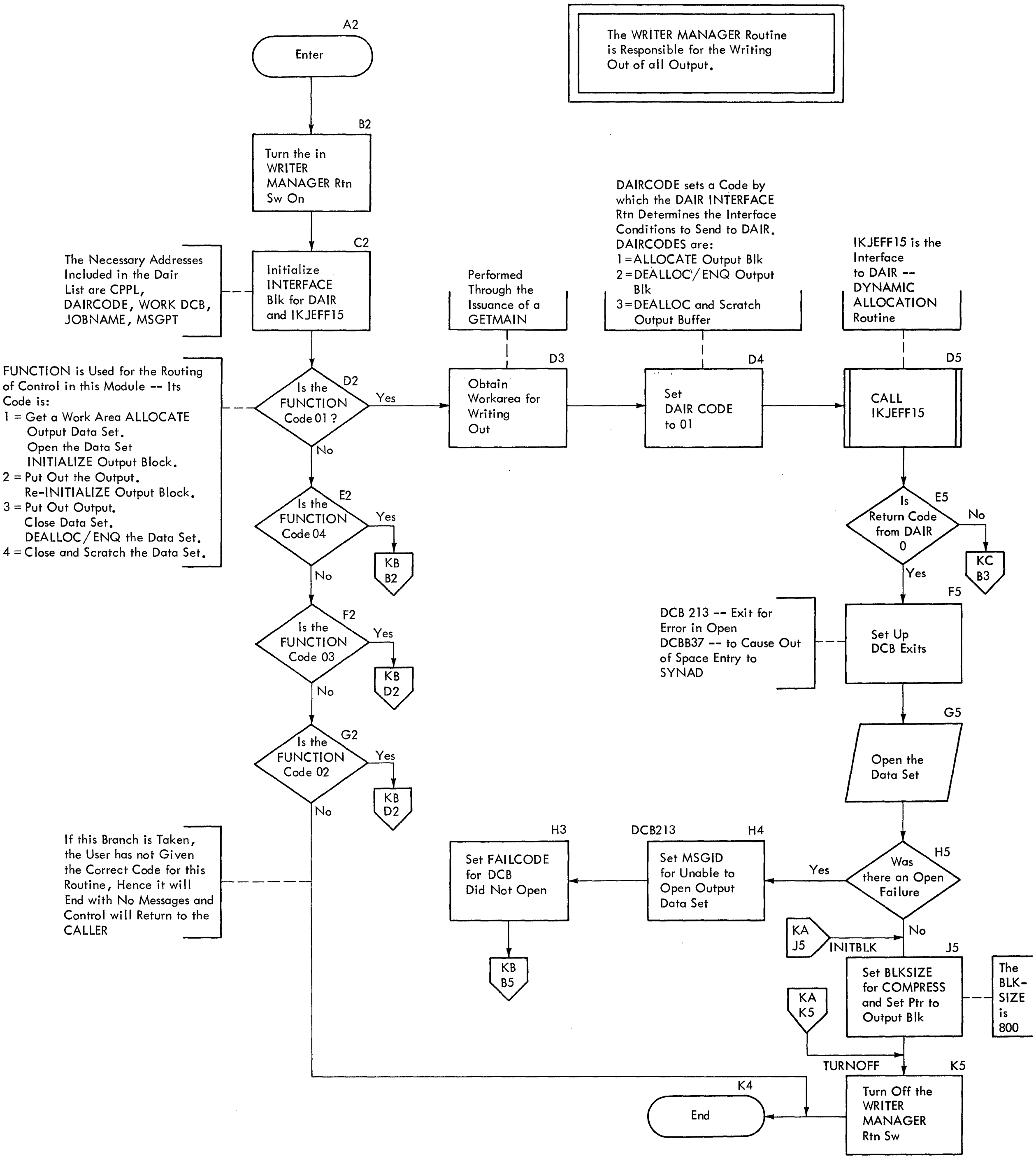

A2
Enter
The WRITER MANAGER Routine is Responsible for the Writing Out of all Output.
B2
Turn the in WRITER MANAGER Rtn Sw On
The Necessary Addresses Included in the Dair List are CPPL, DAIRCODE, WORK DCB, JOBNAME, MSGPT
C2
Initialize INTERFACE Blk for DAIR and IKJEFF15
Performed Through the Issuance of a GETMAIN
DAIRCODE sets a Code by which the DAIR INTERFACE Rtn Determines the Interface Conditions to Send to DAIR. DAIRCODES are:
1=ALLOCATE Output Blk
2=DEALLOC/ENQ Output Blk
3=DEALLOC and Scratch Output Buffer
IKJEFF15 is the Interface to DAIR -- DYNAMIC ALLOCATION Routine
FUNCTION is Used for the Routing of Control in this Module -- Its Code is:
1 = Get a Work Area ALLOCATE Output Data Set. Open the Data Set INITIALIZE Output Block.
2 = Put Out the Output. Re-INITIALIZE Output Block.
3 = Put Out Output. Close Data Set. DEALLOC/ENQ the Data Set.
4 = Close and Scratch the Data Set.
D2
Is the FUNCTION Code 01 ?
Yes
No
D3
Obtain Workarea for Writing Out
D4
Set DAIR CODE to 01
D5
CALL IKJEFF15
E2
Is the FUNCTION Code 04
Yes
KB B2
No
E5
Is Return Code from DAIR 0
No
KC B3
Yes
F2
Is the FUNCTION Code 03
Yes
KB D2
No
F5
DCB 213 -- Exit for Error in Open DCBB37 -- to Cause Out of Space Entry to SYNAD
Set Up DCB Exits
G2
Is the FUNCTION Code 02
Yes
KB D2
No
G5
Open the Data Set
If this Branch is Taken, the User has not Given the Correct Code for this Routine, Hence it will End with No Messages and Control will Return to the CALLER
H3
Set FAILCODE for DCB Did Not Open
KB B5
DCB213
H4
Set MSGID for Unable to Open Output Data Set
Yes
H5
Was there an Open Failure
No
KA J5
INITBLK
J5
Set BLKSIZE for COMPRESS and Set Ptr to Output Blk
The BLK-SIZE is 800
KA K5
TURNOFF
K5
Turn Off the WRITER MANAGER Rtn Sw
K4
End

Chart KB. IKJEFF12. WRITER MANAGER

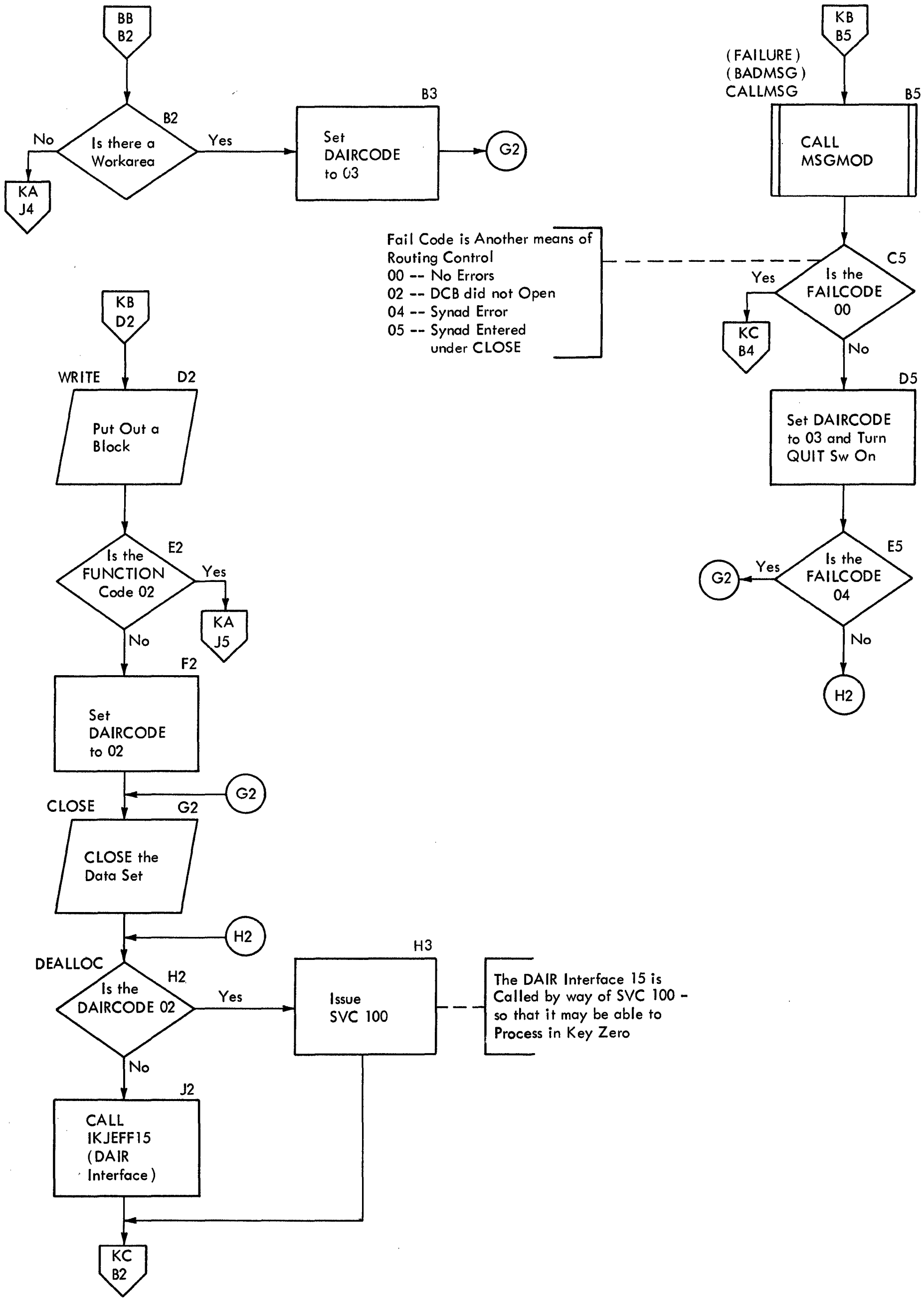


Chart KC. IKJEFF12. WRITER MANAGER

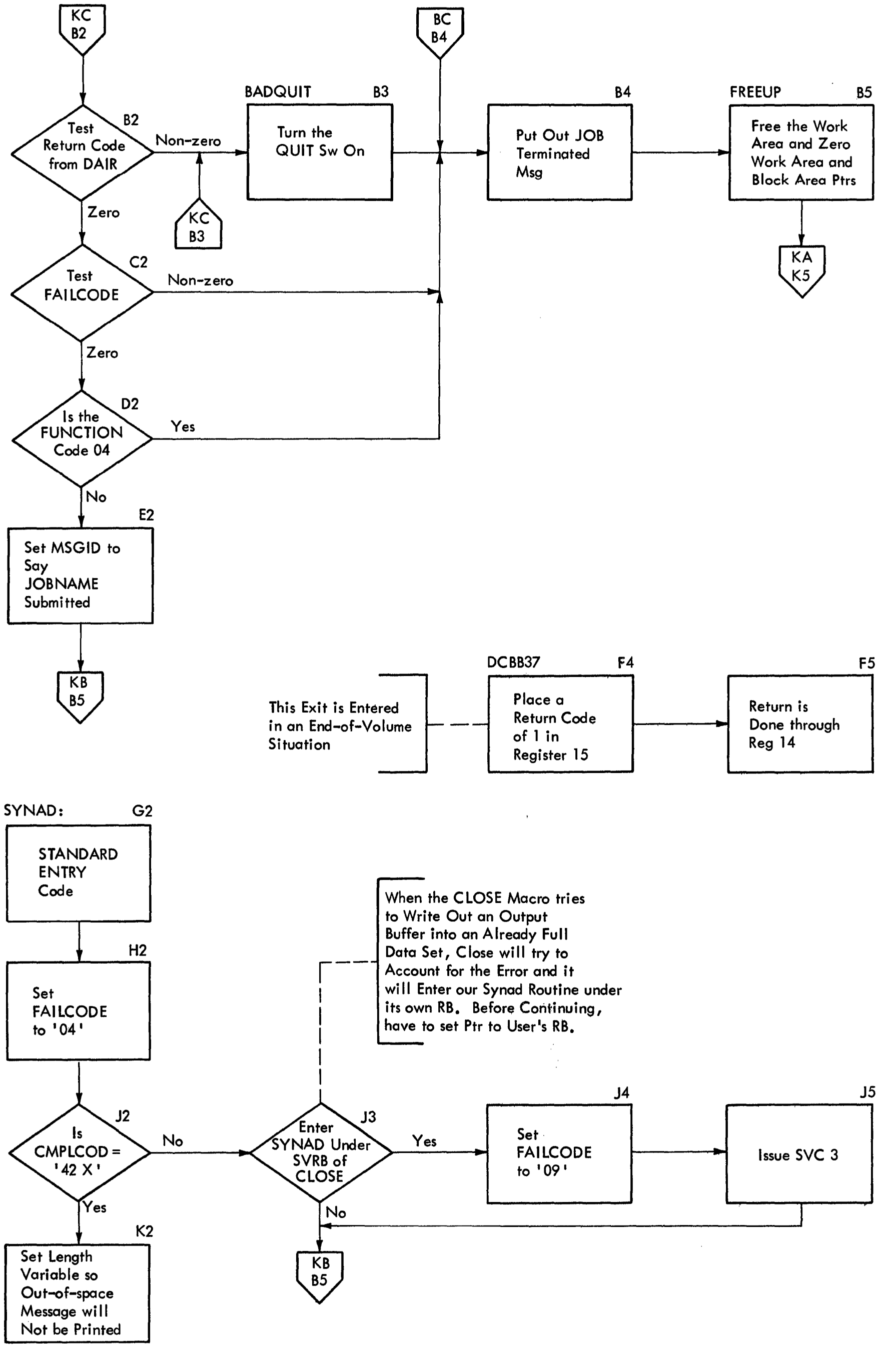
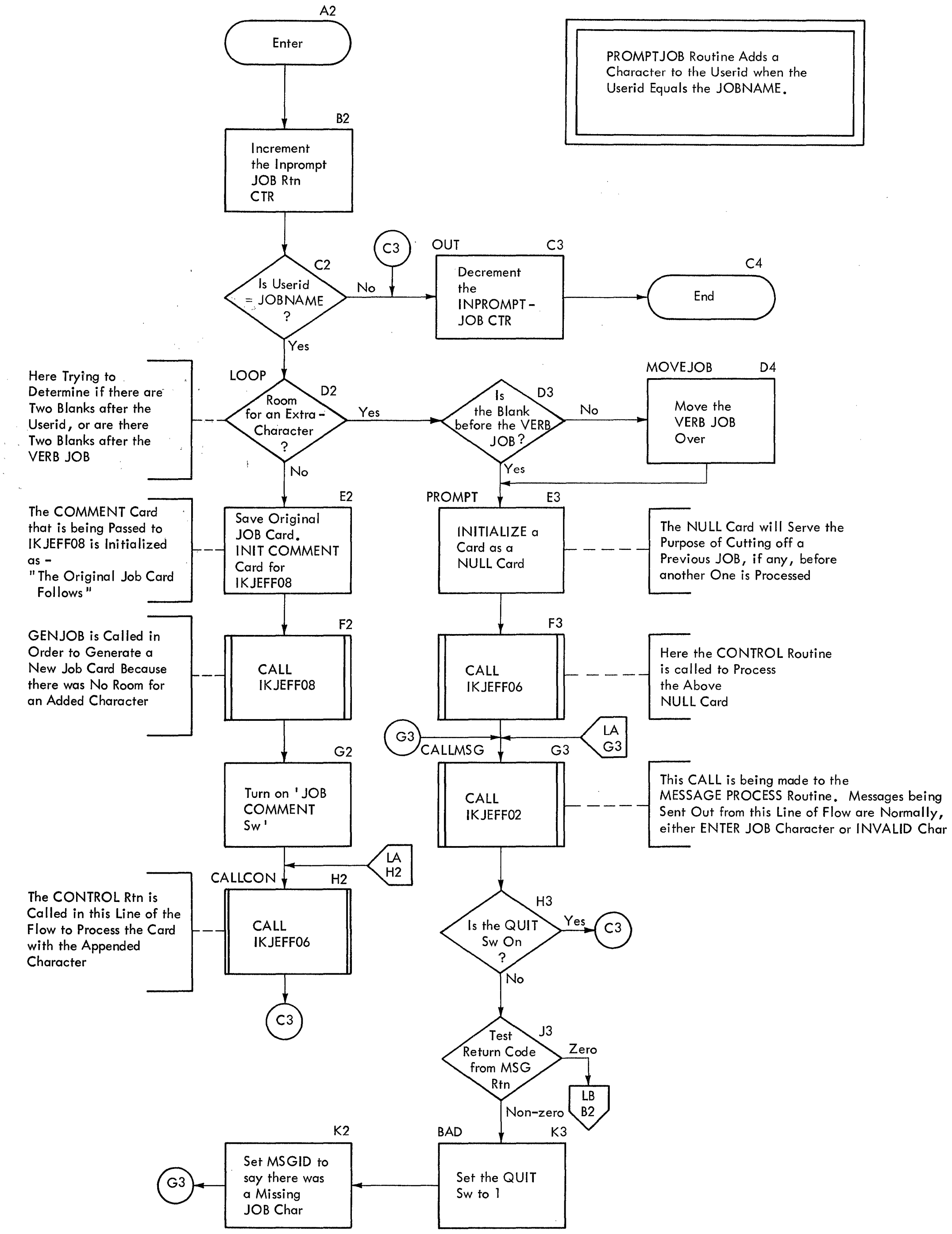


Chart LA. IKJEFF13. PROMPTJOB



PROMPTJOB Routine Adds a Character to the Userid when the Userid Equals the JOBNAME.
A2
Enter
B2
Increment the Inprompt JOB Rtn CTR
C2
Is Userid = JOBNAME ?
No
Yes
C3
OUT
C3
Decrement the INPROMPT-JOB CTR
C4
End
LOOP
D2
Room for an Extra-Character ?
Yes
No
Here Trying to Determine if there are Two Blanks after the Userid, or are there Two Blanks after the VERB JOB
D3
Is the Blank before the VERB JOB ?
No
Yes
MOVEJOB
D4
Move the VERB JOB Over
E2
Save Original JOB Card. INIT COMMENT Card for IKJEFF08
The COMMENT Card that is being Passed to IKJEFF08 is Initialized as - "The Original Job Card Follows"
PROMPT
E3
INITIALIZE a Card as a NULL Card
The NULL Card will Serve the Purpose of Cutting off a Previous JOB, if any, before another One is Processed
F2
CALL IKJEFF08
GENJOB is Called in Order to Generate a New Job Card Because there was No Room for an Added Character
F3
CALL IKJEFF06
Here the CONTROL Routine is called to Process the Above NULL Card
G3
CALLMSG
G3
LA G3
CALL IKJEFF02
This CALL is being made to the MESSAGE PROCESS Routine. Messages being Sent Out from this Line of Flow are Normally, either ENTER JOB Character or INVALID Char
G2
Turn on 'JOB COMMENT Sw'
LA H2
CALLCON
H2
CALL IKJEFF06
The CONTROL Rtn is Called in this Line of the Flow to Process the Card with the Appended Character
C3
H3
Is the QUIT Sw On ?
Yes
C3
No
J3
Test Return Code from MSG Rtn
Zero
LB B2
Non-zero
BAD
K3
Set the QUIT Sw to 1
K2
Set MSGID to say there was a Missing JOB Char
G3

Chart LB. IKJEFF13. PROMPTJOB

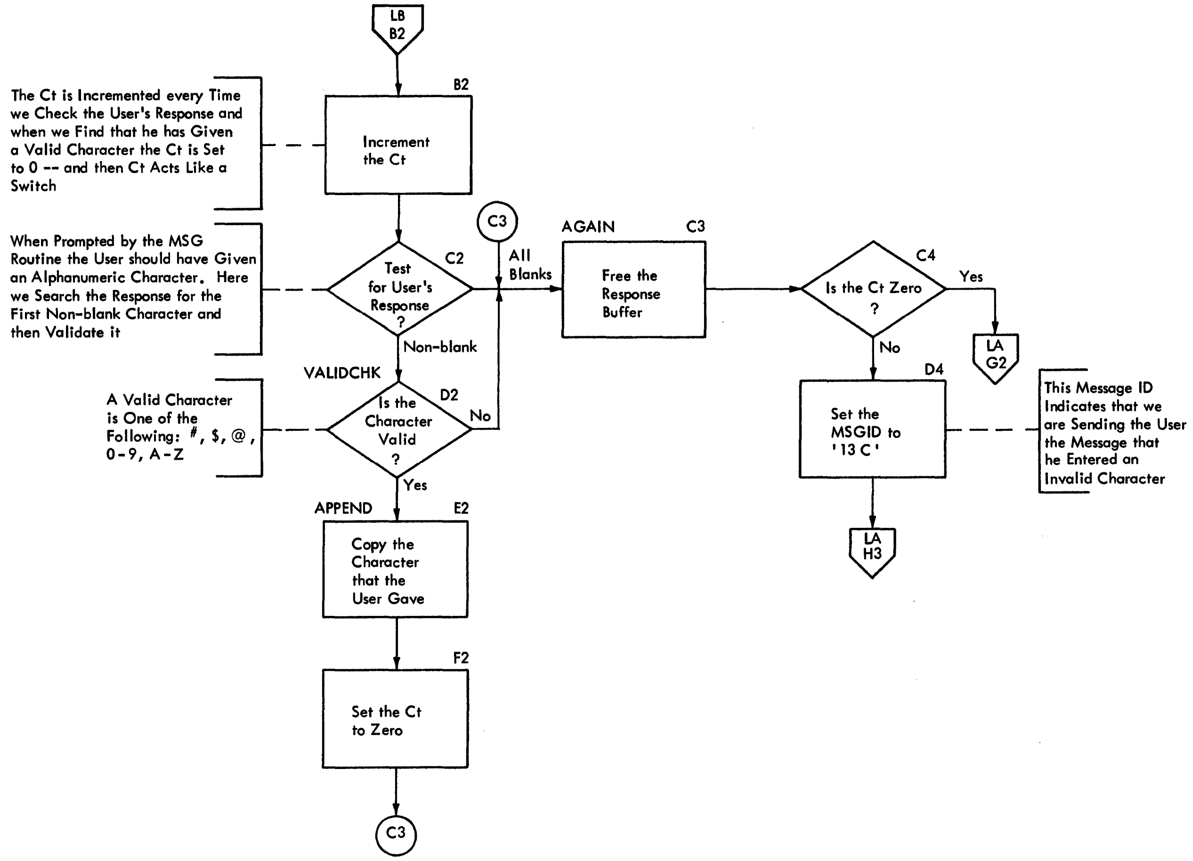
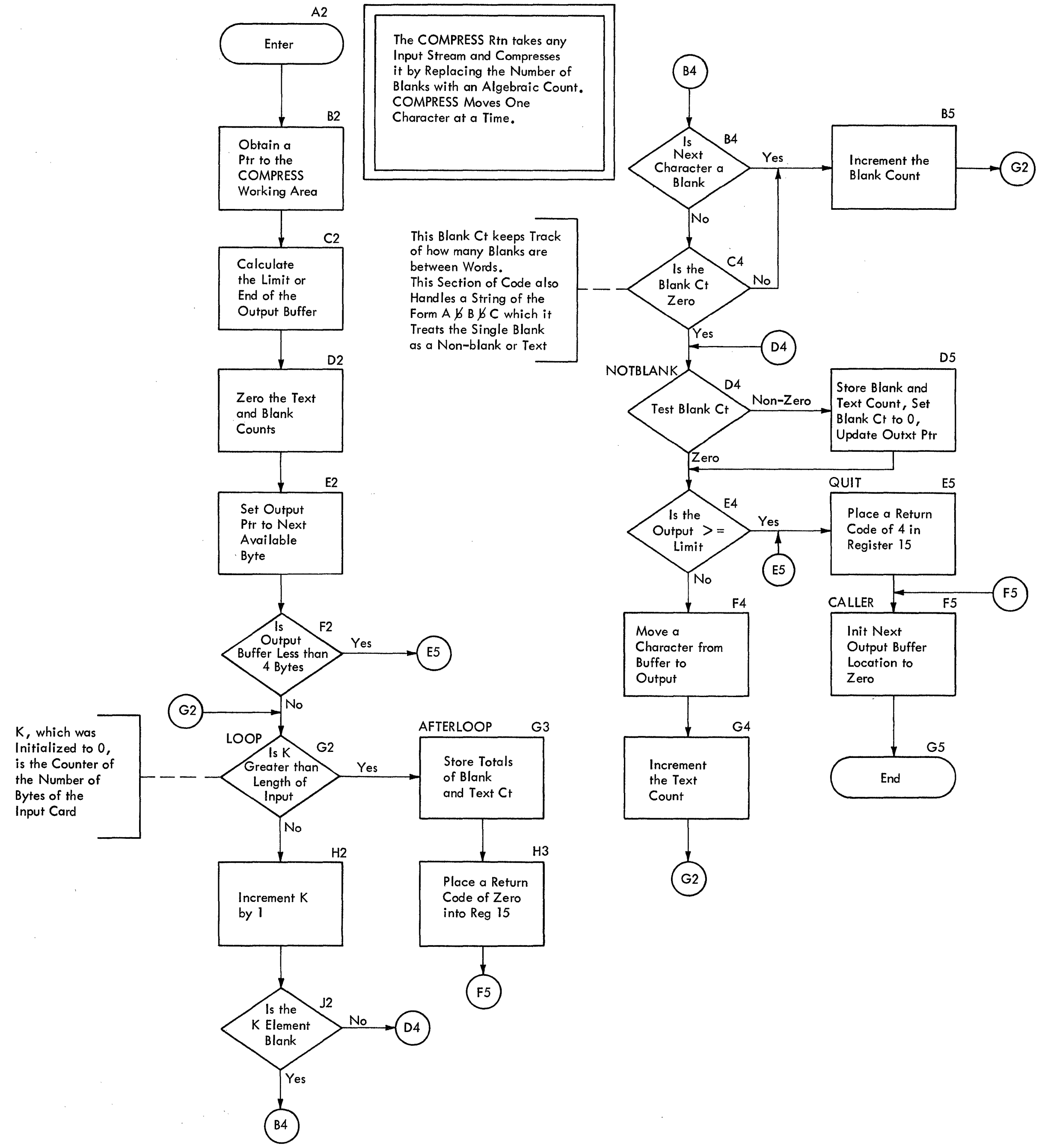

LB B2
B2
Increment the Ct
The Ct is Incremented every Time we Check the User's Response and when we Find that he has Given a Valid Character the Ct is Set to 0 -- and then Ct Acts Like a Switch
C2
Test for User's Response ?
When Prompted by the MSG Routine the User should have Given an Alphanumeric Character. Here we Search the Response for the First Non-blank Character and then Validate it
C3
All Blanks
AGAIN
C3
Free the Response Buffer
C4
Is the Ct Zero ?
Yes
LA G2
No
D4
Set the MSGID to '13 C'
This Message ID Indicates that we are Sending the User the Message that he Entered an Invalid Character
LA H3
Non-blank
VALIDCHK
D2
Is the Character Valid ?
No
A Valid Character is One of the Following: #, $, @, 0-9, A-Z
Yes
APPEND
E2
Copy the Character that the User Gave
F2
Set the Ct to Zero
C3

Chart MA. IKJEFF14. COMPRESS


The COMPRESS Rtn takes any Input Stream and Compresses it by Replacing the Number of Blanks with an Algebraic Count. COMPRESS Moves One Character at a Time.

This Blank Ct keeps Track of how many Blanks are between Words. This Section of Code also Handles a String of the Form A ƀ B ƀ C which it Treats the Single Blank as a Non-blank or Text

K, which was Initialized to 0, is the Counter of the Number of Bytes of the Input Card

A2
Enter
B2
Obtain a Ptr to the COMPRESS Working Area
C2
Calculate the Limit or End of the Output Buffer
D2
Zero the Text and Blank Counts
E2
Set Output Ptr to Next Available Byte
F2
Is Output Buffer Less than 4 Bytes
Yes
E5
No
G2
LOOP
G2
Is K Greater than Length of Input
Yes
No
H2
Increment K by 1
J2
Is the K Element Blank
No
D4
Yes
B4
AFTERLOOP
G3
Store Totals of Blank and Text Ct
H3
Place a Return Code of Zero into Reg 15
F5
B4
B4
Is Next Character a Blank
Yes
No
C4
Is the Blank Ct Zero
No
Yes
D4
NOTBLANK
D4
Test Blank Ct
Non-Zero
Zero
E4
Is the Output >= Limit
Yes
E5
No
F4
Move a Character from Buffer to Output
G4
Increment the Text Count
G2
B5
Increment the Blank Count
G2
D5
Store Blank and Text Count, Set Blank Ct to 0, Update Outxt Ptr
QUIT
E5
Place a Return Code of 4 in Register 15
F5
CALLER
F5
Init Next Output Buffer Location to Zero
G5
End

Chart NA. IKJEFF15. DAIR INTERFACE

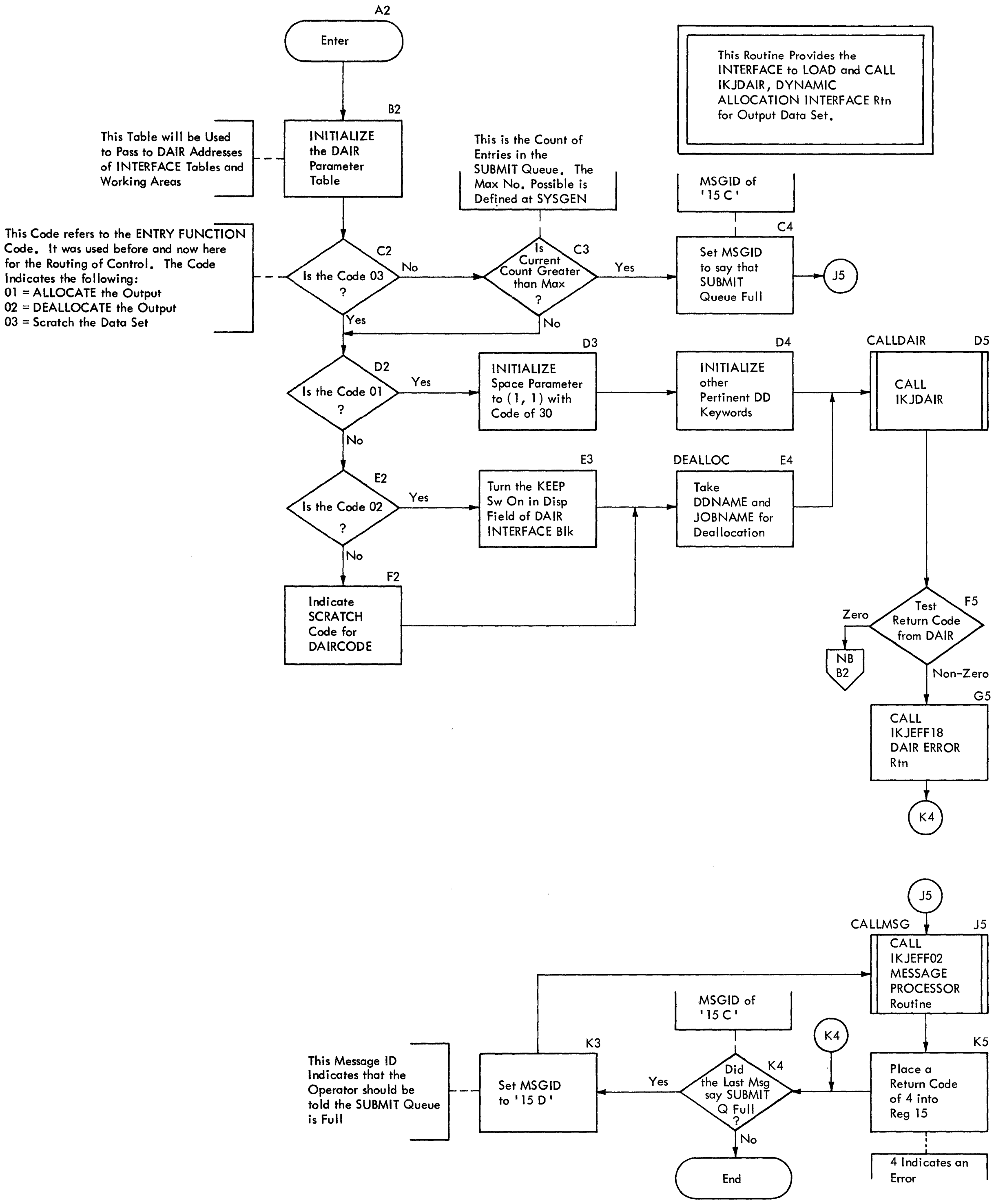

A2
Enter
B2
INITIALIZE the DAIR Parameter Table
This Table will be Used to Pass to DAIR Addresses of INTERFACE Tables and Working Areas
This Routine Provides the INTERFACE to LOAD and CALL IKJDAIR, DYNAMIC ALLOCATION INTERFACE Rtn for Output Data Set.
This is the Count of Entries in the SUBMIT Queue. The Max No. Possible is Defined at SYSGEN
MSGID of '15 C'
This Code refers to the ENTRY FUNCTION Code. It was used before and now here for the Routing of Control. The Code Indicates the following:
01 = ALLOCATE the Output
02 = DEALLOCATE the Output
03 = Scratch the Data Set
C2
Is the Code 03 ?
No
Yes
C3
Is Current Count Greater than Max ?
Yes
No
C4
Set MSGID to say that SUBMIT Queue Full
J5
D2
Is the Code 01 ?
Yes
No
D3
INITIALIZE Space Parameter to (1, 1) with Code of 30
D4
INITIALIZE other Pertinent DD Keywords
CALLDAIR
D5
CALL IKJDAIR
E2
Is the Code 02 ?
Yes
No
E3
Turn the KEEP Sw On in Disp Field of DAIR INTERFACE Blk
DEALLOC
E4
Take DDNAME and JOBNAME for Deallocation
F2
Indicate SCRATCH Code for DAIRCODE
F5
Test Return Code from DAIR
Zero
NB B2
Non-Zero
G5
CALL IKJEFF18 DAIR ERROR Rtn
K4
J5
CALLMSG
J5
CALL IKJEFF02 MESSAGE PROCESSOR Routine
MSGID of '15 C'
K4
K3
Set MSGID to '15 D'
This Message ID Indicates that the Operator should be told the SUBMIT Queue is Full
Yes
K4
Did the Last Msg say SUBMIT Q Full ?
No
End
K5
Place a Return Code of 4 into Reg 15
4 Indicates an Error

Chart NB. IKJEFF15. DAIR INTERFACE

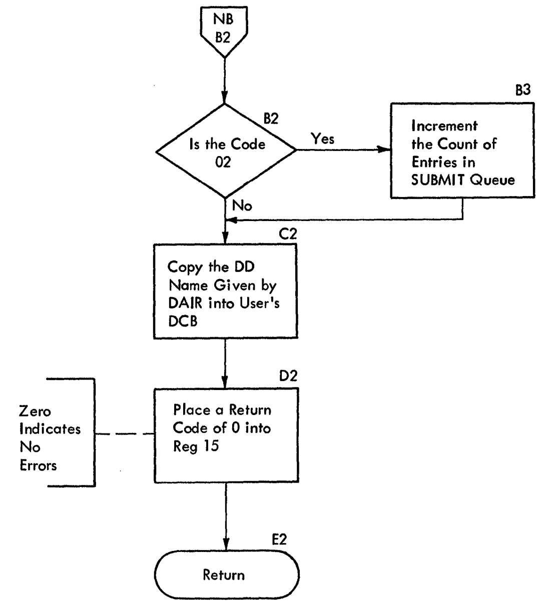
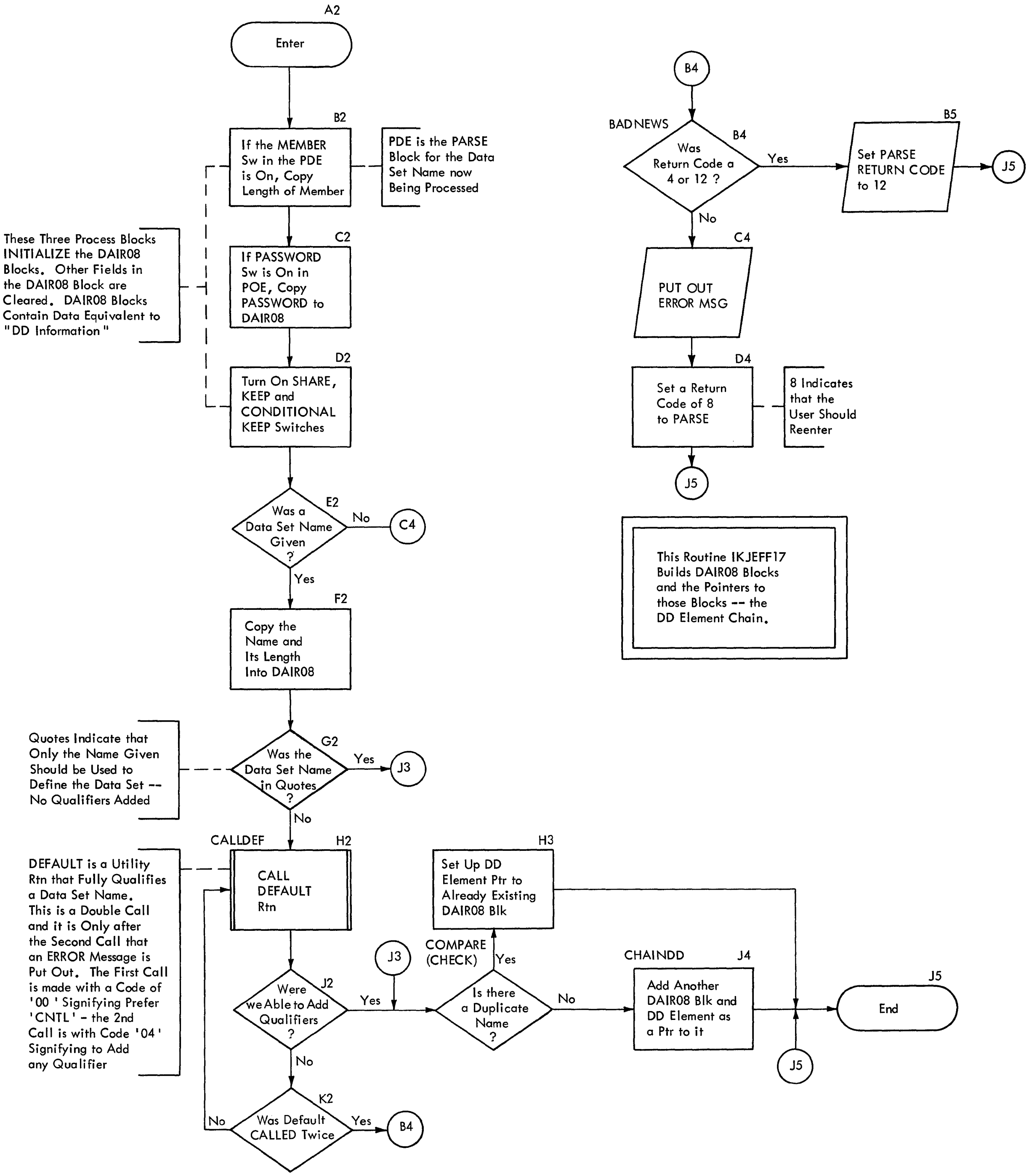

NB
B2
B2
Is the Code 02
Yes
B3
Increment the Count of Entries in SUBMIT Queue
No
C2
Copy the DD Name Given by DAIR into User's DCB
D2
Zero Indicates No Errors
Place a Return Code of 0 into Reg 15
E2
Return

Chart OA. IKJEFF17. PARSE EXIT


A2
Enter
B2
If the MEMBER Sw in the PDE is On, Copy Length of Member
PDE is the PARSE Block for the Data Set Name now Being Processed
These Three Process Blocks INITIALIZE the DAIR08 Blocks. Other Fields in the DAIR08 Block are Cleared. DAIR08 Blocks Contain Data Equivalent to "DD Information"
C2
If PASSWORD Sw is On in POE, Copy PASSWORD to DAIR08
D2
Turn On SHARE, KEEP and CONDITIONAL KEEP Switches
E2
Was a Data Set Name Given ?
No
C4
Yes
F2
Copy the Name and Its Length Into DAIR08
Quotes Indicate that Only the Name Given Should be Used to Define the Data Set -- No Qualifiers Added
G2
Was the Data Set Name in Quotes ?
Yes
J3
No
CALLDEF
H2
CALL DEFAULT Rtn
DEFAULT is a Utility Rtn that Fully Qualifies a Data Set Name. This is a Double Call and it is Only after the Second Call that an ERROR Message is Put Out. The First Call is made with a Code of '00 ' Signifying Prefer 'CNTL' - the 2nd Call is with Code '04' Signifying to Add any Qualifier
J2
Were we Able to Add Qualifiers ?
Yes
No
K2
Was Default CALLED Twice
No
Yes
B4
J3
H3
Set Up DD Element Ptr to Already Existing DAIR08 Blk
COMPARE (CHECK)
Yes
Is there a Duplicate Name ?
No
CHAINDD
J4
Add Another DAIR08 Blk and DD Element as a Ptr to it
J5
End
J5
B4
BADNEWS
B4
Was Return Code a 4 or 12 ?
Yes
B5
Set PARSE RETURN CODE to 12
J5
No
C4
PUT OUT ERROR MSG
D4
Set a Return Code of 8 to PARSE
8 Indicates that the User Should Reenter
J5
This Routine IKJEFF17 Builds DAIR08 Blocks and the Pointers to those Blocks -- the DD Element Chain.

Section 4. Directory

This section contains a routine directory organized alphabtically by entry name or routine. A data area directory is also provided, arranged alphabetically.

Routine Directory

Entry Name,	Load	Control	Cross Reference	
Routine Name	Module	Section	MO	FC
IKJEFF01	SUBMIT	IKJEFF01	2-1	AA,AB
IKJEFF02	SUBMIT, IKJEFF02	IKJEFF02		BA-BC
IKJEFF03	SUBMIT	data only		
IKJEFF04	IKJEFF04	IKJEFF04	2-3	CA
IKJEFF05	IKJEFF04	IKJEFF05	2-3	DA,DB
IKJEFF06	IKJEFF04	IKJEFF06	2-4	EA,EB
IKJEFF07	IKJEFF04	IKJEFF07	2-4	FA-FC
IKJEFF08	IKJEFF04	IKJEFF08	2-4	GA
IKJEFF09	IKJEFF04	IKJEFF09	2-4	HA,HB
IKJEFF10	IKJEFF10	IKJEFF10	2-4	IA
IKJEFF11	IKJEFF04	IKJEFF11		JA
IKJEFF12	IKJEFF04	IKJEFF12	2-5	KA-KC
IKJEFF13	IKJEFF04	IKJEFF13	2-4	LA,LB
IKJEFF14	IKJEFF04	IKJEFF14	2-4	MA

ROUTINE DIRECTORY (Cont.)

Entry Name, Routine Name	Load Module	Control Section	Cross Reference MO	 FC
IKJEFF15	IKJEFF04, IKJEFF15	IKJEFF15	2-5	NA,NB
IKJEFF16	IKJEFF16	data only	2-2	
IKJEFF17	IKJEFF16	IKJEFF17	2-2	DA
IKJEFF18	IKJEFF18	IKJEFF18		

Data Area Directory

Data Area	MO	FC
Control Table (CONTAB)	2-3,2-4	CA,EA,EB
DSNPDE	2-2	CA
History Table	2-3,2-4	CA,FA
PDL	2-2	AA
Parse Parameter List (PPL)	2-2	AA
WORKAREA	2-5	KA

Section 5. Data Areas

This section describes the data areas within the Submit Command Processor. Data areas are in alphabetical order.

CONTAB—Control Table

Size: 68 bytes

Constructed by: IKJEFF04

Used by: IKJEFF05, 06, 08, 09, 11 and 13 all define this table.
IKJEFF07, 10 and 12 use various fields only.

Contents: The Control Table contains fullword pointers to data areas used by all SUBMIT routines. The first twelve bytes contain the words "SUBMIT CONTROL TABLE" to help to find this table in a dump.

Only two routines reference the Control Table directly: Initializer and Control. Other routines are passed pointers to the pointers to the data areas they need.

Disp Dec	(Hex)	Field	Size (Bytes)	Use/Contents
0	(0)	TABID	12	Used for identification, initialized to 'SUBMIT TABLE'.
12	(C)	CONTABPT	4	Pointer to Control Table.
16	(10)	STMTPT	4	Pointer to Current Statement.
20	(14)	WRMGRPT	4	Pointer to Writer/Manager Work Area.
24	(18)	HISTPT	4	Pointer to History Table.
28	(1C)	OUTBLKPT	4	Pointer to Output Block.
32	(20)	JOBNAMPT	4	Pointer to Current Jobname.

CONTAB -- Control Table (Cont.)

Disp Dec (Hex)	Field	Size (Bytes)	Use/Contents
36 (24)	MSGLISPT	4	Pointer to Message Routine List.
40 (28)	USERIDPT	4	Pointer to Userid.
44 (2C)	TMCTPT	4	Pointer to Command Processor Parameter List (CPPL).
48 (30)	EXWORD	4	Exit Word.
52 (34)	EXITAD	4	Exit Address.
56 (38)	DDPTR	4	Pointer to DD Element Chain.
60 (3C)	COMECBPT	4	Pointer to Communication ECB.
64 (40)	INITSAVE	4	Pointer to Initializer Routine save area.

DSNPDE—Data Set Name Parameter Descriptor Element

Created by: IKJPARSE.

Used by: IKJEFF17.

Use: Formatted data set name parameters.

Disp Dec	(Hex)	Field	Subfield	Size (Bytes)	Use/Contents
0	(0)	PDEDSNPT		4	Pointer to data set name (44 char).
4	(4)	PDEDSNLN		2	Length of data set name.
6	(6)	PDEFLG1		1	
			PDEDSNB	1 bit	"1" if data set name present.
			PDEQUOTE	1 bit	"1" if data set name quoted.
7	(7)	*		1	Unnamed.
8	(8)	PDEMEMPT		4	Pointer to membername.
12	(C)	PDEMEMLN		2	Length of member name.
14	(E)	PDEFLG2		1	
			PDEMEMB	1 bit	"1" if member.
15	(F)	*		1	Unnamed.
16	(10)	PDEPASSP		4	Pointer to password.
20	(14)	PDEPASLN		2	Length of password.
22	(16)	PDEFLG3		1	
			PDEPASSB	1 bit	"1" of password.

History Table

Constructed by: IKJEFF04

Updated by: IKJEFF07

Used by: IKJEFF06

Contents: Three fields: one to record which module is executing, one to record the current JCL statement being processed, one for static history including the quit switch.

Disp Dec	 Hex	Field	Subfield	Size (Bytes)	Use/Contents
0	(0)	SUBTAB		12	Identification, initialized to History Tab.
12	(C)	WHATMOD		4	Indicates modules called that have not yet returned to caller.
12	(C)		INCONTRL	1	Control.
13	(D)		INPROMPT	1	Prompt.
14	(E)		INGEN	1	Generates.
15	(F)	*		1	Unnamed.
			INREAD	1 bit	Reader.
			INIDENT	1 bit	Identify.
			INEXIT	1 bit	Exit interface.
			INWRMGR	1 bit	Writer manager.
			INDAIRIN	1 bit	Dair interface.
			INMSG	1 bit	Message interface.
			INCOMPIN	1 bit	Compress.
16	(10)	JCLSWITS		12	Contains prior, current and static history of JCL statements.
16	(10)	PRIORJCL		4	Prior History; history of previous statement.

HISTORY TABLE (Cont.)

Disp	Field	Subfield	Size	Use/Contents
Dec (Hex)			(Bytes)	
		PVER8	2	Verb Type indicated by bit settings:
			Bit	
			0 PDELIM	Delimiter.
			1 PDATA	Data.
			2 PNULL	Null statement.
			3 PJOB	Job statement.
			4 PEXEC	Exec statement.
			5 PDD	DD statement.
			6 PCOMMAND	Command.
			7 PCOMMENT	Comment statement.
			8-15	Reserved (0).
18 (12)		PDTYPE	1	Data type indicated by bit settings:
			Bit	
			0 PDDAST	DD *
			1 PDDATA	DD DATA
			2-7	Reserved (0).
19 (13)		PMISC	1	Miscellaneous switches indicated by bit settings:
			Bit	
			0 PCONEX	Continuation expected.
			1 POPCONEX	Op Code and continuation expected.
			2 PCONERR	Jobname present.
			3 PNAME	Verb copied.
			4 PVERBCOP	Verb copied.
			5 PCOMCON	Make next statement a comment.
			6 PNOTHING	Unknown.
20 (14)	CURRJCL		4	Current History; listing of current JCL statement.

HISTORY TABLE (Cont.)

Disp Dec (Hex)	Field	Subfield	Size (Bytes)	Use/Contents
20 (14)		CVERB	2	Verb type indicated by bit settings:
			Bit	
			0 CDELIM	Delimiter.
			1 CDATA	Data.
			2 CNULL	Null statement.
			3 CJOB	Job statement.
			4 CEXEC	Exec statement.
			5 CDD	DD statement.
			6 CCOMMAND	Command.
			7 COMMENT	Comment statement.
			8 - 15	Reserved.
22 (16)		CDTYPE	1	Data type indicated by bit settings:
			Bit	
			0 CDDAST	DD *
			1 CDDATA	DD DATA
			2 - 7	Reserved (0).
23 (17)		CMISC	1	Miscellaneous switches indicated by bit settings:
			Bit	
			0 CCONEX	Continuation expected.
			1 COPCONEX	Op code and continuation expected.
			2 CCONERR	Continuation expected error.
			3 CNAME	Jobname present.
			4 CVERBCOP	Verb copied.
			5 CCOMMON	Make next statement a comment.
			6 CNOTHING	Unknown.
28 (1C)	STATIC		4	Contains Static History.

HISTORY TABLE (Cont.)

Disp Dec (Hex)	Field	Subfield	Size (Bytes)	Use/Contents
28 (1C)		GENL	1 Bit 0 QUIT 1 FLUSH 2 FIRST 3 NONOTIFY 4 - 7	General Switches; bit settings: Quit; terminate SUBMIT command processor. Flush the input stack. First record for a job. NONOTIFY parameter included in all generated JOB statements. Reserved (0).
29 (1D)		TAKEXITS	1 Bit 0 JOBX 1 EXECX 2 DDX 3 CMDX 4 NULLX	Take exits to the installation. Exit routine for the following statements; bit settings: Job statements. Exec statements. DD statements. Commands. Null statements.
30 (1E)		OPOFF	1	Operation offset.
31 (1F)		CODEFLG	1	Functions Code byte.

PDL—Parameter Descriptor List

Created by: IKJPARS

Used by: IKJEFF01 and IKJEFF17

Use: Formatted command parameters.

Disp Dec (Hex)	Field	Size (Bytes)	Use/Contents
0 (0)	overhead	8	Used by IKJPARS.
8 (8)	DSNPDE	28	Fully defined under DSNPDE.
36 (24)	NONOPDE	2	1 if notify 2 if no-notify

PPL—PARSE Parameter List

Created by: IKJEFF01

Used by: IKJPARS

Use: Interface to PARSE

Disp Dec (Hex)	Field	Size (Bytes)	Use/Contents
0 (0)	PPLUPT	4	Pointer to UPT.
4 (4)	PPLECT	4	Pointer to ECT.
8 (8)	PPLECB	4	Pointer to SUBMIT's ECB.
12 (C)	PPLPCL	4	Pointer to PCL.
16 (10)	PPLANS	4	Pointer to address of PDL.
20 (14)	PPLCBUF	4	Pointer to Command Buffer.
24 (18)	PPLUWA	4	Pointer to user work area.

WORKAREA—Writer Manager Work Area

Created by: IKJEFF12

Used by: IKJEFF14

Use: Contains block and output area required by Writer Manager (1040 bytes).

Disp Dec (Hex)	Field	Subfield	Size (Bytes)	Use/Contents
0 (0)	WORKDCB		96	
		*	37	Unnamed.
37 (25)		EXLST	3	Exit list pointer.
40 (28)		*	8	Unnamed.
48 (30)		OFLAGS	1	
		*	1 bit	Unnamed.
		*	1 bit	Unnamed.
		*	1 bit	Unnamed.
		OPENBIT	1 bit	
		*	1 bit	Unnamed.
		*	1 bit	Unnamed.
		*	1 bit	Unnamed.
		*	1 bit	Unnamed.
96 (60)	SAVEREG		64	Special save area used to recover registers in the SYNAD routine. For SYNAD, Reg 1 points to the DCB, so this save area is defined just after the DCB. Completion code is given in the high order byte of Reg 1.
96 (60)		BADR0	4	
100 (64)		BADR1	4	
100 (64)		COMPLCOD	1	
160 (A0)	BLOCK		808	Area for output block with pointers for use by compress routine; pointer to next available space remaining in bytes.

Section 6. Diagnostic Aids

This section contains a register usage summary and information on messages. A table lists the messages issued by each module by message ID. The messages and their meanings are then listed in message ID order.

Register Summary

All routines in this processor use the following four registers in the conventional ways:

R1 -- parameter list points

R13 -- register save area pointer

R14 -- return address

R15 -- return code

Messages (By Module)

IKJEFF01	IKJ56251+ IKJ56264
IKJEFF05	IKJ56265
IKJEFF09	IKJ56266 IKJ56280+ IKJ56282 IKJ56283
IKJEFF12	IKJ56250 IKJ56256 IKJ56270
IKJEFF13	IKJ56253+ IKJ56255 IKJ56257
IKJEFF15	IKJ520(WTO) IKJ56259 IKJ56270
IKJEFF17	IKJ56262 IKJ56252 IKJ56268 IKJ56269

IKJEFF18	IKJ56220I+ IKJ56221I+ IKJ56222I+ IKJ56223I+ IKJ56224I+ IKJ56225I+ IKJ56226I+ IKJ56227I+ IKJ56228I+ IKJ56229I+ IKJ56230I+ IKJ56231I+ IKJ56232I+ IKJ56233I+ IKJ56235I IKJ56236I IKJ56237I IKJ56238I IKJ56241I IKJ56243I IKJ56244I IKJ56245I IKJ56246I IKJ56247I IKJ56248I IKJ56249I

Messages (By Message ID)

Message Number	Messages
IKJ56220I	(1) NOT ALLOCATED, TOO MANY DATA SETS+ USE FREE COMMAND TO FREE UNUSED DATA SETS
IKJ56221I	(1) NOT ALLOCATED, REQUIRED VOLUME NOT MOUNTED+ VOLUME OR CVOL NOT ON SYSTEM AND CANNOT BE ACCESSED
IKJ56222I	(1) NOT ALLOCATED, REQUIRED VOLUME NOT MOUNTED+ VOLUME NOT ON SYSTEM AND CANNOT BE ACCESSED
IKJ56223I	COMMAND SYSTEM ERROR+ (routine) ERROR CODE XXX
IKJ56224I	INVALID SYSOUT CLASS
IKJ56225I	DATA SET -DSNAME- ALREADY IN USE, TRY LATER+ DATA SET IS ALLOCATED TO ANOTHER JOB OR USER
IKJ56226I	INVALID DATA SET -DSNAME- EXCEEDS 44 CHARACTERS
IKJ56227I	DATA SET 'dsname' ALLOCATED FOR SHARED USE ONLY+ DATA SET IS ALLOCATED TO ANOTHER JOB OR USER
IKJ56228I	DATA SET -DSNAME- NOT IN CATALOG
IKJ56229I	DATA SET -DSNAME- WILL CREATE INVALID CATALOG STRUCTURE+ A QUALIFIER CANNOT BE BOTH AN INDEX AND THE LAST QUALIFIER OF A DATA SET NAME
IKJ56230I	DATA SET 'dsname' NOT UNALLOCATED, MEMBER OF CONCATENATION
IKJ56231I	(1) NOT ALLOCATED, SYSTEM OR INSTALLATION ERROR+

IKJ56232I VOLUME ALLOCATED BUT DATA SET 'dsname' IS NOT ONE VOLUME+
CATALOG OR VOLUME INFORMATION INCORRECT, USE DELETE COMMAND

IKJ56233I HELP DATA SET NOT ALLOCATED, SYSHELP IS DUMMY+
MODIFY YOUR LOGON PROCEDURE

IKJ56235I MEMBER 'membername' SPECIFIED BUT 'dsname' NOT A PARTITIONED DATA SET

IKJ56236I FILE $\left\{\begin{array}{l}\text{'joblib'}\\ \text{'steplib'}\end{array}\right\}$ INVALID

IKJ56237I DATA SET 'dsname' NOT ON A DIRECT ACCESS DEVICE, NOT SUPPORTED

IKJ56238I (1) NOT FREED+
(2)

NOTES ON VARIABLES	DAIR CALL
(1) HELP DATA SET	24
UTILITY DATA SET	30 OR 18, SYSCLS GIVEN
DATA SET -DSNAME-	DSN PTR NOT ZERO
FILE -DDNAME-	IF DDNAME NOT BLANK
TERMINAL	1C
(2) SUBALLOCATED DATA SET	
GENERATION DATA GROUP	
PASSED DATA SET	
(3) DYNAMIC ALLOCATION	CODE NOT IN TABLE
CATALOG	CODE NOT IN TABLE
DAIR	CODE NOT IN TABLE
(4) CATALOG ERROR CODE 14	DYNAMIC ALLOCATION ERROR CODE XXXX
CATALOG I/O ERROR	

IKJ56241I (1) NOT ALLOCATED+
INVALID UNIT IN USER ATTRIBUTE DATA SET OR NO UNIT AVAILABLE

IKJ56243I DATA SET 'dsname' RESIDES ON MULTIPLE VOLUMES, NOT SUPPORTED

IKJ56244I DATA SET 'dsname' NOT ALLOCATED, DIRECTLY LARGER THAN PRIMARY QUANTITY

IKJ56245I DATA SET 'dsname' NOT ALLOCATED, NOT ENOUGH SPACE ON VOLUMES+
USE DELETE COMMAND TO FREE UNUSED DATA SETS

IKJ56246I FILE 'ddname' NOT ALLOCATED, IN USE

IKJ56247I FILE 'ddname' NOT FREED, IS NOT ALLOCATED

IKJ56248I DATA SET 'dsname' NOT ALLOCATED, REQUESTED AS NEW BUT CURRENTLY ALLOCATED

IKJ56249I DATA SET 'dsname' NOT ALLOCATED, CURRENTLY ALLOCATED WITH DISPOSITION OF DELETE

IKJ56250I JOB 'jobname' SUBMITTED

Explanation: Acknowledgement message. A job has been entered into the background.

User Response: None.

IKJ56251A COMMAND NOT AUTHORIZED

Explanation: Use of SUBMIT command as specified by account command is not authorized for this userid.

System Action: SUBMIT processing is terminated.

IKJ56251I YOUR INSTALLATION MUST AUTHORIZE USE OF THIS COMMAND

IKJ56253A ENTER JOBNAME CHARACTER

Explanation: A prompting message issued to complete a jobname which consists of the userid only.

System Action: The SUBMIT processor waits for the user's response.

User Response: Enter an alphanumeric character. If more than one character is entered. Only the first one is used.

IKJ56253I JOBNAME IS CREATED FROM USERID PLUS ONE ALPHANUMERIC CHARACTER

Explanation: This message appears if the user responds to message IKJ56253A with '?'.

IKJ56255A INVALID - CHARACTER 'character'

Explanation: The reply given to message IKJ56253A was not a valid alphanumeric character.

IKJ56255I REENTER-

Explanation: This message prompts the user for another character.

System Action: The submit processor waits for user response.

User Response: Enter one alphanumeric character.

IKJ56255I CHARACTER MUST BE ALPHANUMERIC

Explanation: This message appears if the user responds to message IKJ56255A with '?'.

IKJ56257I MISSING JOBNAME CHARACTER

Explanation: This message appears if the user has requested 'NOPROMPT' and therefore message IKJ56253A cannot be processed.

System Action: SUBMIT processing is terminated.

IKJ56259I BACKGROUND READER QUEUE FULL, TRY LATER

Explanation: The background reader queue (in the system jobqueue) does not contain sufficient space for a submitted job.

System Action: Submit processing is terminated. The system operator is notified.

User Response: The user must wait for a period of time and then re-submit the job.

IKJ56264I COMMAND SYSTEM ERROR+

Explanation: One of the service routines abended.

User Response: Check second-level error code.

IKJ56264A PARSE ERROR CODE XXX

Explanation: The Parse scan check found an error.

User Response: Look over the submit syntax and resubmit.

IKJ56264I COMMAND SYSTEM ERROR - DEFAULT ERROR CODE XXX

Explanation: An error was encountered in IKJDFLT.

User Response: Resubmit another data set name.

IKJ56265I+ INPUT DATASET 'data set name' NOT USABLE

Explanation: An error was encountered during the processing of an input data set.

System Action: SUBMIT processing is terminated.

IKJ56265I INPUT OPEN ERROR 'error code'

Explanation: Second level explanation for message IKJ56465I.

IKJ56265I SUBMIT TERMINATED ON JOB

Explanation: An error condition has been encountered.

System Action: Submit processing is terminated.

User Response: The error condition may be indicated by a previous message. In some cases the user may enter '?' to obtain additional information.

IKJ56266I UNABLE TO DIAGNOSE RETURN CODE XXXX

Explanation: Installation Exit sent to Exit Interface an unrecognizable return code.

System Action: Submit continue processing.

IKJ56267I SYNAD ERROR 'SYNAD INFO'

Explanation: Second level explanation for message IKJ56265I.

IKJ56268I INVALID DATA SET dsname, EXCEEDS 44 CHARACTERS

Explanation: Error message from Default Routine.

User Response: Enter data set name which is not greater than 44 characters.

IKJ56269I DATA SET dsname NOT IN CATALOG

Explanation: User entered a data set name not in the catalog.

User Response: Enter a valid data set name.

IKJ56270I OUTPUT DATASET FOR JOB 'jobname' NOT USABLE

Explanation: An output error has been encountered.

System Action: SUBMIT processing is terminated.

IKJ56270I OUTPUT OPEN ERROR 'error code'

Explanation: Second level explanation for message IKJ56270I.

IKJ56270I NOT ENOUGH DIRECT ACCESS SPACE

Explanation: Second level explanation for message IKJ56270I. Not enough space was available to copy the user's input data set(s).

System Action: SUBMIT processing is terminated.

User Response: Wait a period of time and then resubmit the job. Reduce the amount of data in the input stream by placing it in another data set and using a DD statement to point to it.

IKJ56272I SYNAD ERROR 'synad info'

Explanation: Second level explanation for message IKJ56270I.

IKJ56280A 'installation exit text'

Explanation: This message is from the installation exit. A response is called for.

System Action: The submit processor waits for the response requested by the message.

User Response: Give response.

IKJ56281I ENTER REPLY FOR EXIT-

Explanation: This message appears if the user responds to message IKJ56280A with '?'.

IKJ56282I MISSING REPLY FOR 'installation exit' FROM EXIT

Explanation: This message appears if the user has requested 'NOPROMPT' and therefore message IKJ56280A cannot be processed.

System Action: Submit processing is terminated.

IKJ56283I 'INSTALLATION EXIT TEXT'

Explanation: This message is from the installation exit.

System Action: Submit continues processing. No response is called for.

IEF404I 'jobname' ENDED

Explanation: Notify message issued when a job has finished executing in the background.

IEF453I 'JOBNAME' JOB FAILED

Explanation: Notify message issued when a job has finished executing in the background.

IKJ520I BACKGROUND READER QUEUE FULL

Explanation: Notification to system operator that the system jobqueue does not contain sufficient space for a submitted job.

System Action: Submit processing is terminated in the foreground. There is no background action.

Operator Response: The operator may wish to take action to free jobqueue records for use in the submit queue. He may do any of the following:

1. Increase the maximum number of jobs the system may contain at one time way of the 'MODIFY TSO' command.

2. Start a background reader task.

3. Stop background input readers.

4. Start background initiators and writers.

5. Increase the precentage of CPU time available to background tasks.

6. Decrease the number and sizes of foreground regions, to allow for more tasks in the background.

Index

Indexes to program logic manauals are consolidated in the publication IBM System/360 Operating System: Program Logic Manaul Master Index, Order No. GY28-6717. For additional information about any subject listed below, refer to other publications listed for the same subject in the Master Index.

The Submit Command Processor:

- Examines JCL statements read from the input data sets.
- Inserts or changes JOB statements when missing or unacceptable JOB statements are discovered.
- At the user's discretion, passes any JCL statement to a user written exit routine for processing.
- Builds and enqueues an output data set for each JOB.

The processor comprises four functional stages.

1. The syntax of the command parameters is checked, input control blocks are built, and input data sets are allocated.
2. Tables of addresses and processing information, to be passed between routines are built. Input statements are delivered for processing.
3. Each input statement is examined and processed as required. The limits of jobs defined by the resulting JCL statements are determined.
4. An output data set is allocated, written, and enqueued for each job.

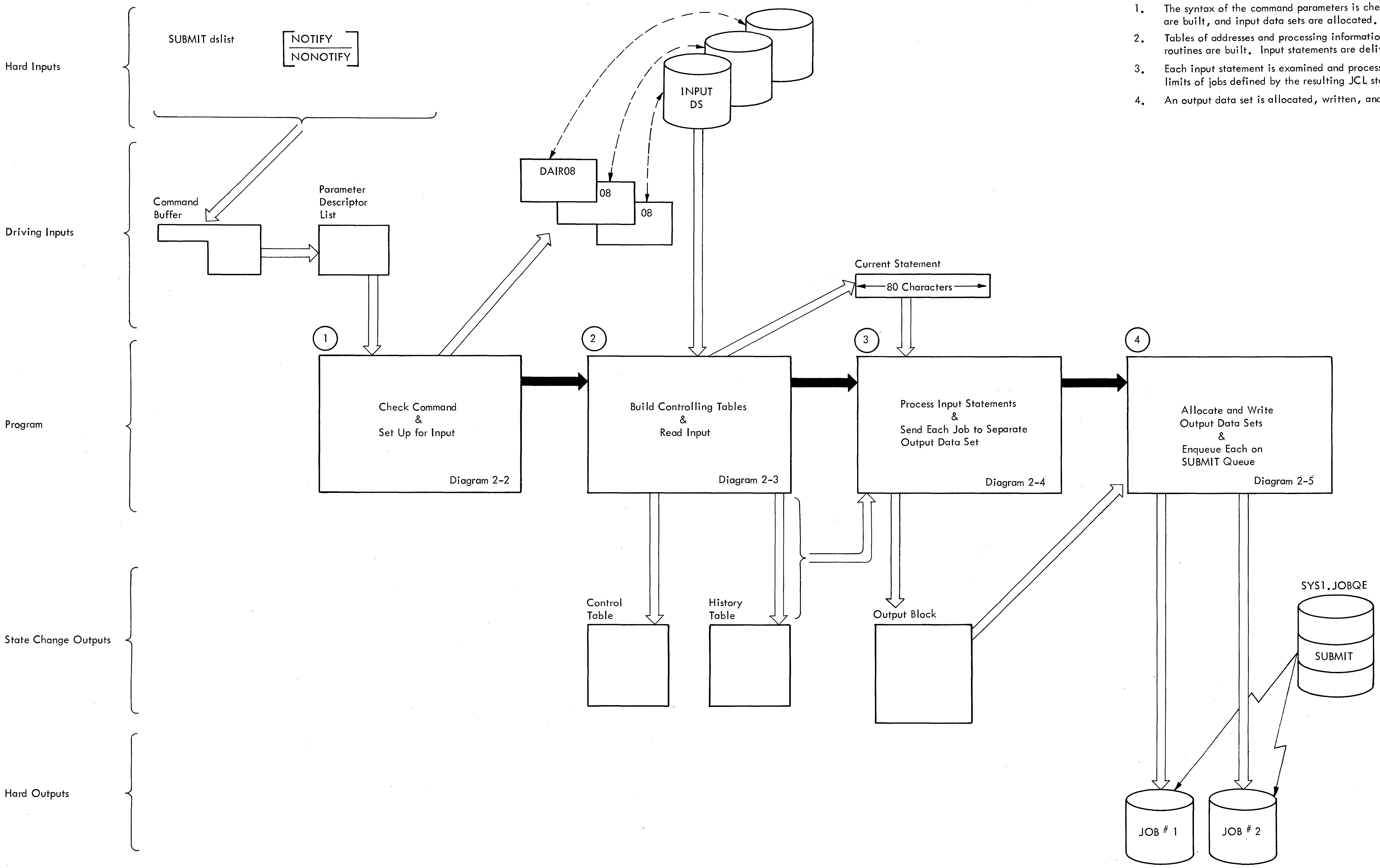
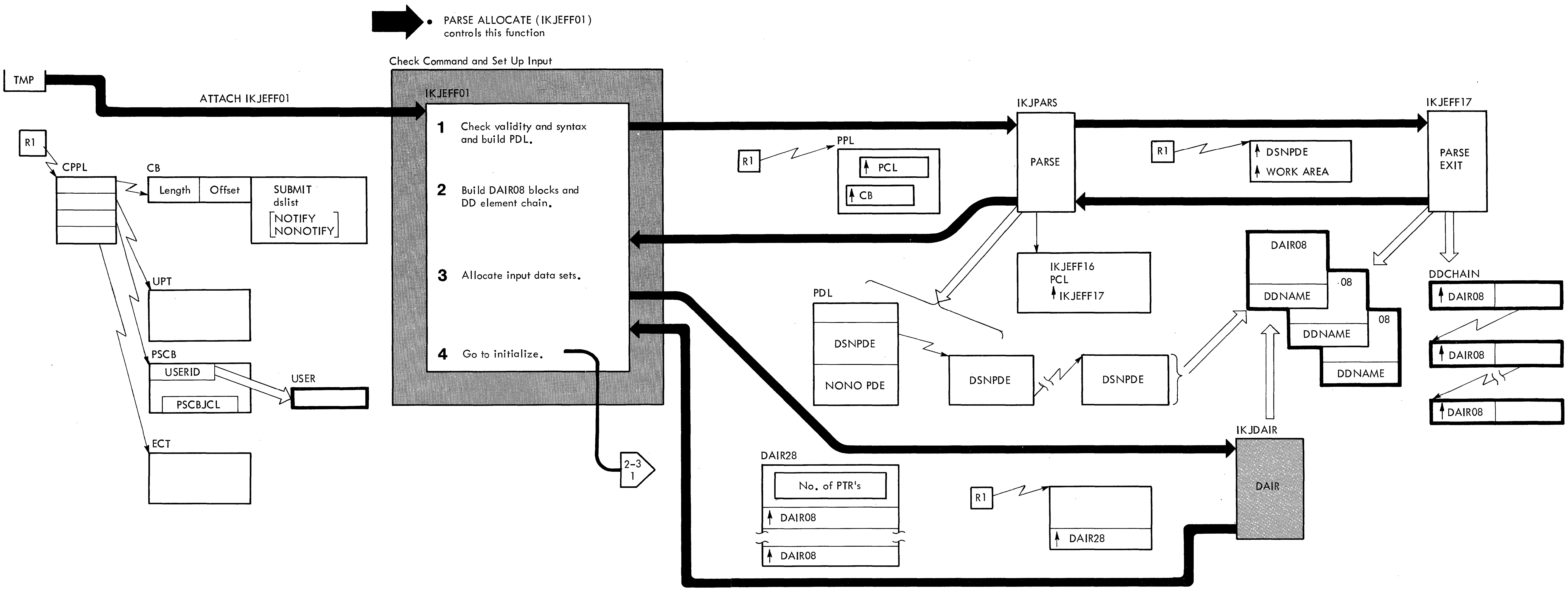


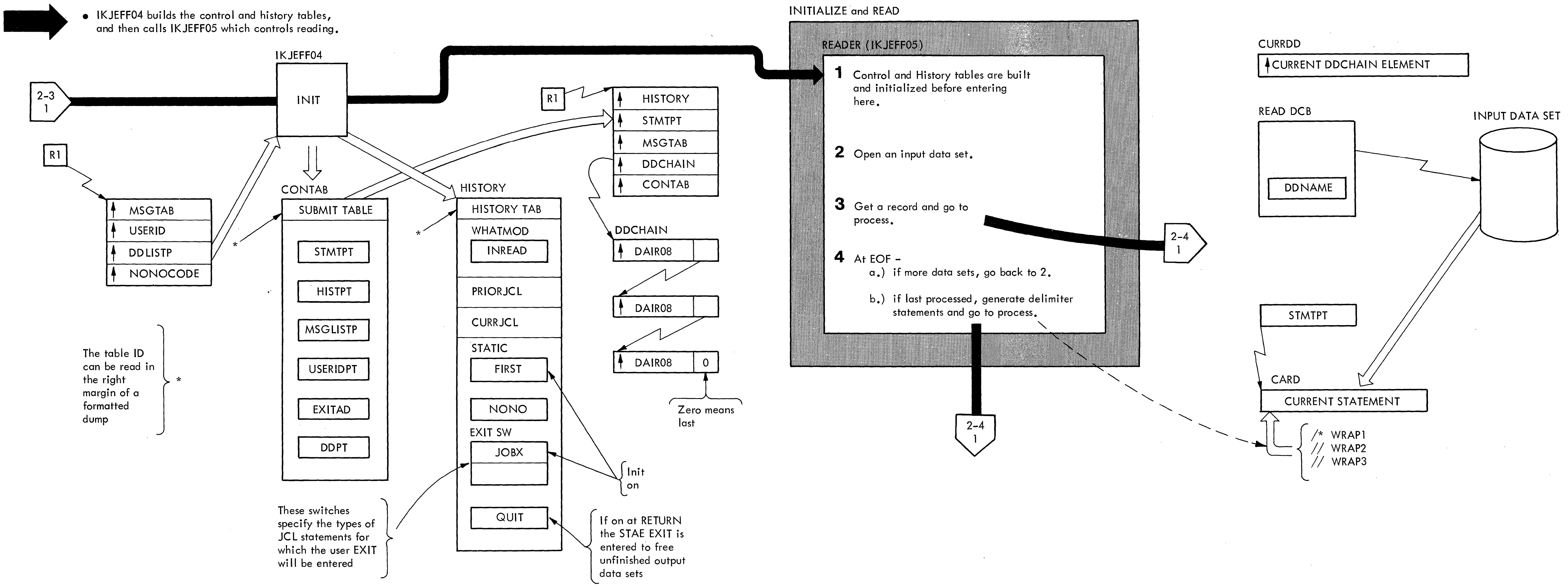
Diagram 2-1. Functional Summary



Description	Routine	Label	FC
1 The user ID from the PSCB is placed in USER for later use. The PSCBJCL bit is checked to see if the user is authorized to use SUBMIT '1' = yes. If no the job is terminated with a message to the user. Parse is called to check the command parameters and build the PDL. Parse prompts the user for a data set name if none is in the command buffer. It then builds the PDL containing one DSNPDE for each data set name furnished.	IKJEFF01	IKJEFF01	AA
2 As each DSNPDE is built, parse calls Parse Exit to build a DAIR08 block for each unique data set name and a chained DD element for each DSNPDE. (If a data set name appears twice in the dslist it will have only one DAIR08 built for it. However , a pointer to the DAIR08 will occur at two places in the DD Element Chain. This allows the same data set to be read in at any number of places in the input stream.)			
If Parse Exit is passed a DSNPDE for an unquoted data set name, IKJDFLT is called to check the name for conformance to TSO standards. The user is prompted to fully qualify or quote the data set name. If the user does not respond, a name of the form USERID. User assigned. CTRL is assigned.	IKJEFF17 IKJDFLT		OA

Description	Routine	Label	FC
3 A DAIR28 block containing the address of each of the DAIR08 blocks is built and a pointer to it is placed in DAPL, the DAIR parameter list. On one call DAIR allocates all of the input data sets. The DD name for each is delivered to the DAIR08. If the return code from DAIR ≠ 0, then IKJEFF18 is called to analyze the failure and send the appropriate message to the user before processing is terminated.	IKJEFF01		AA
4 IKJEFF04 is called with the addresses of the four areas in heavy outline (USER, FUNCTION, DDCHAIN, and the DAIR08 blocks) as parameters.	IKJEFF04		CA

Diagram 2-2. Check Command and Set-Up for Input

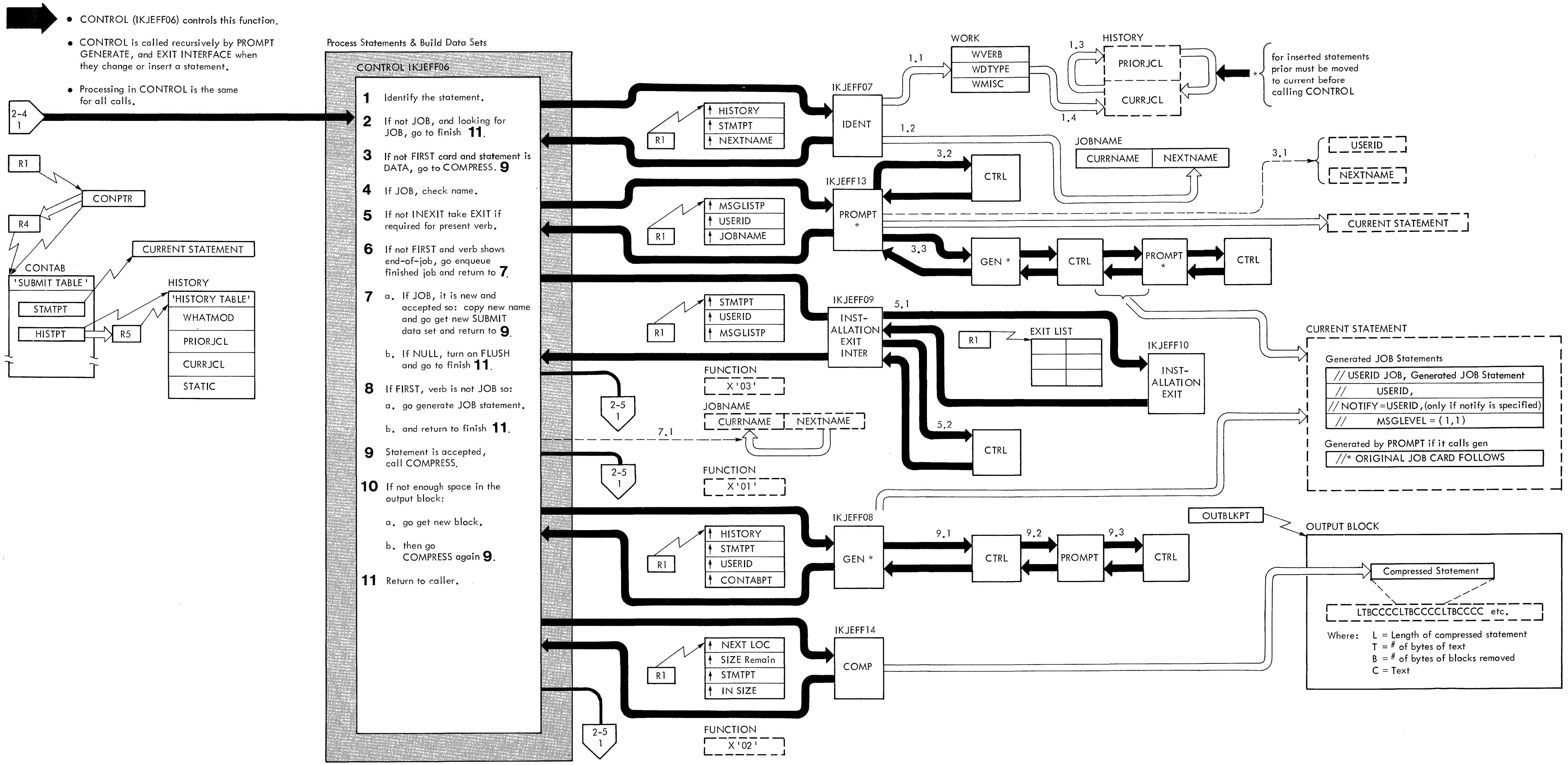

• IKJEFF04 builds the control and history tables, and then calls IKJEFF05 which controls reading.
IKJEFF04
INIT
2-3
1
R1
MSGTAB
USERID
DDLISTP
NONOCODE
CONTAB
SUBMIT TABLE
STMTPT
HISTPT
MSGLISTP
USERIDPT
EXITAD
DDPT
*
The table ID can be read in the right margin of a formatted dump
HISTORY
HISTORY TAB
WHATMOD
INREAD
PRIORJCL
CURRJCL
STATIC
FIRST
NONO
EXIT SW
JOBX
QUIT
These switches specify the types of JCL statements for which the user EXIT will be entered
Init on
If on at RETURN the STAE EXIT is entered to free unfinished output data sets
R1
HISTORY
STMTPT
MSGTAB
DDCHAIN
CONTAB
DDCHAIN
DAIR08
DAIR08
DAIR08
0
Zero means last
INITIALIZE and READ
READER (IKJEFF05)
1 Control and History tables are built and initialized before entering here.
2 Open an input data set.
3 Get a record and go to process.
4 At EOF -
a.) if more data sets, go back to 2.
b.) if last processed, generate delimiter statements and go to process.
2-4
1
2-4
1
CURRDD
CURRENT DDCHAIN ELEMENT
READ DCB
DDNAME
INPUT DATA SET
STMTPT
CARD
CURRENT STATEMENT
/* WRAP1
// WRAP2
// WRAP3

Description	Routine	Label	FC
1 The control table contains pointers to fields and tables used by different processing routines. Only this routine and the CONTROL routine (IKJEFF06) reference the table directly. Other routines receive pointers to only those elements required by their function. For instance, the address of STMTPT in CONTAB is passed to READER, after reading the address of CARD is returned to CONTAB for use by CONTROL. History contains these sections: a) WHATMOD -- Each routine called from here on sets a switch (or increments a counter) in WHATMOD when it is called, and turns it off when it RETURN's to its caller. This is done to keep a record of CALL's made, since processing in many instances means recursive use of programs. For example, CONTROL which is called by READER to process a statement may be called again, by a routine that CONTROL itself has called, before returning to READER. b) PRIOR and CURRENT history -- used primarily by IDENTIFY to record facts about the statement being examined. For example, verb type of continuation expected. Any routine changing a statement must set PRIOR to CURRENT before calling CONTROL to process the changed statement. c) STATIC history -- records history that remains constant across changes to the current statement and recursive calling . For example, the NONO switch, if 0, means that NOTIFY was specified and that all generated job statements must specify NOTIFY.	IKJEFF04	IKJEFF04	CA

Description	Routine	Label	FC
2 CURRDD is set with the address of the first DAIR08 and the DDNAME is moved from DAIR08 to the DCB. The data set is opened.	IKJEFF05	ALLAGAIN	DA
3 The record is read to CARD. The address of CARD is put in STMPT in CONTAB, and CONTROL is called.		CALLCON	DA
4 At EOF, the input data set is closed and CURRDD is set from the pointer field of the current DDCHAIN element. If CURRDD ≠ 0, go back and process the next data set. If CURRDD = 0 deliver the three wrap up statements to CONTROL. Before delivering the second null statement turn off all of the EXIT switches in HISTORY. This insures that the job being processed will be ended and the data set enqueued even if the user EXIT is deleting NULL statements.		INPUTEOF WRAPUP OUT	DB DA DA

Diagram 2-3. Initialize and Read

Description	Routine	Label	FC
1 The current statement is examined by IDENTIFY which: (1.1) Fills in WORK (same format as PRIORJCL and CURRJCL), (1.2) If the statement is a job card, copies the jobname to NEXTNAME, (1.3) Moves CURRJCL to PRIORJCL, and (1.4) Moves WORK to CURRJCL, and then returns to CONTROL.	IKJEFF07	IKJEFF07 UPDATE	FA FA
2 If FLUSH in STATIC is on, a null statement has ended a job, all statements until the next job card are ignored by simply ıeturning to the caller.	IKJEFF06	IKJEFF06	EB
3 If FIRST in STATIC is off, and a data card is found, it is part of an already identified job, and is compressed and sent to the output block.	IKJEFF06		EB
4 If the verb is JOB and the statement is not a continuation, then this is the first card of a new job. PROMPT is called to check the job name. If the jobname (in NEXTNAME) ≠ USERID (3.1), then the jobname is acceptable and an immediate return to CONTROL is made. If jobname = user ID, the user will be prompted to add an additional character to the jobname. Before prompting, a null card is generated and CONTROL is called to process it (3.2). The null ends the last job (if any) and causes the "job-submitted" message to be sent for the last job before the user is prompted on the next job. The message routine (IKJEFF02) is called, the character from the user's reply is added to the jobname in the current statement and CONTROL is called to process the changed statement. Upon return from CONTROL, PROMPT returns to CONTROL for the original call. If the original statement had no room for an added character, GENERATE (3.3) is called to furnish a default job statement. The original will be made a comment by IDENTIFY because PROMPT sets CCOMCON in CURRJCL. PROMPT adds the "originals follow" card after the generated job cards.	IKJEFF13	OUT PROMPT CALLMSG CALLCON OUT LOOP	LA LA LA LA LA LA LA
5 If INEXIT in WHATMOD is on, we are in CONTROL on a recursive call from EXIT INTERFACE, and we will not recall EXIT. If INEXIT is off and the TAKEXIT switch (in STATIC) for the current verb is on, EXIT INTER is called. The user written exit is entered and can make any desired changes to the job stream (5.1). The message routine and CONTROL are called by EXIT INTER to send messages and process changes made by EXIT.	IKJEFF06 IKJEFF09	CALLEX CALLMSG FIRSTCAL	EA HA HA
6 A job is ended by a null card or the job card for the following job. WRITER MANAGER is called with a function code of X'03' to enqueue the last job.	IKJEFF06	AFTEX	EA
7 a. If the card is JOB, CURRJCL is checked to make sure that the card is not a continuation, and then (7.1), the jobname is accepted and WRITER MANAGER is called with a function code of X'01' to start a new data set. Upon return, FIRST & FLUSH in STATIC are turned off if on.		AFTENQ CALLWRMG	EA EB
b. If the card is NULL, the FLUSH switch in STATIC is set. All cards until the next JOB card will be ignored.		AFTSTART	EB
8 Three or four (if notify is specified) cards are generated and then the original non-job card is reprocessed. CONTROL is called (9.1) recursively for the inserts and the original. PROMPT will be called (9.2) when the first inserted card is processed to get the additional character from the user. When GENERATE finally RETURNS to CONTROL it goes back to READER for another statement.	IKJEFF08	COUNT	GA
9 COMPRESS puts the output statement, in compressed form, into an output block furnished by WRITER MANAGER when it was called to get a SUBMIT data set.	IKJEFF14		MA
10 A return code of 4 from COMPRESS means that the output block did not contain enough room for the statement. WRITER MANAGER is called to write the last block and get a new one. Upon return COMPRESS is called to put the statement in the new block.			



- CONTROL (IKJEFF06) controls this function.
- CONTROL is called recursively by PROMPT GENERATE, and EXIT INTERFACE when they change or insert a statement.
- Processing in CONTROL is the same for all calls.

Process Statements & Build Data Sets

CONTROL IKJEFF06

1 Identify the statement.

2 If not JOB, and looking for JOB, go to finish **11**.

3 If not FIRST card and statement is DATA, go to COMPRESS. **9**

4 If JOB, check name.

5 If not INEXIT take EXIT if required for present verb.

6 If not FIRST and verb shows end-of-job, go enqueue finished job and return to **7**.

7 a. If JOB, it is new and accepted so: copy new name and go get new SUBMIT data set and return to **9**.

b. If NULL, turn on FLUSH and go to finish **11**.

8 If FIRST, verb is not JOB so:

a. go generate JOB statement.

b. and return to finish **11**.

9 Statement is accepted, call COMPRESS.

10 If not enough space in the output block:

a. go get new block.

b. then go COMPRESS again **9**.

11 Return to caller.

CURRENT STATEMENT

Generated JOB Statements

// USERID JOB, Generated JOB Statement
// USERID,
// NOTIFY=USERID,(only if notify is specified)
// MSGLEVEL = (1,1)

Generated by PROMPT if it calls gen

//* ORIGINAL JOB CARD FOLLOWS

* for inserted statements prior must be moved to current before calling CONTROL

OUTPUT BLOCK

Compressed Statement

LTBCCCCLTBCCCCLTBCCCC etc.

Where: L = Length of compressed statement
T = # of bytes of text
B = # of bytes of blocks removed
C = Text

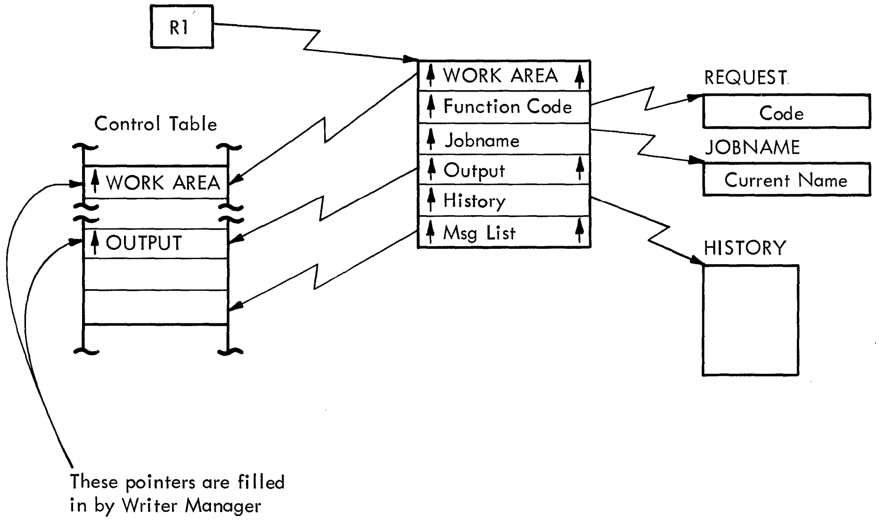
Diagram 2-4. Process Statements

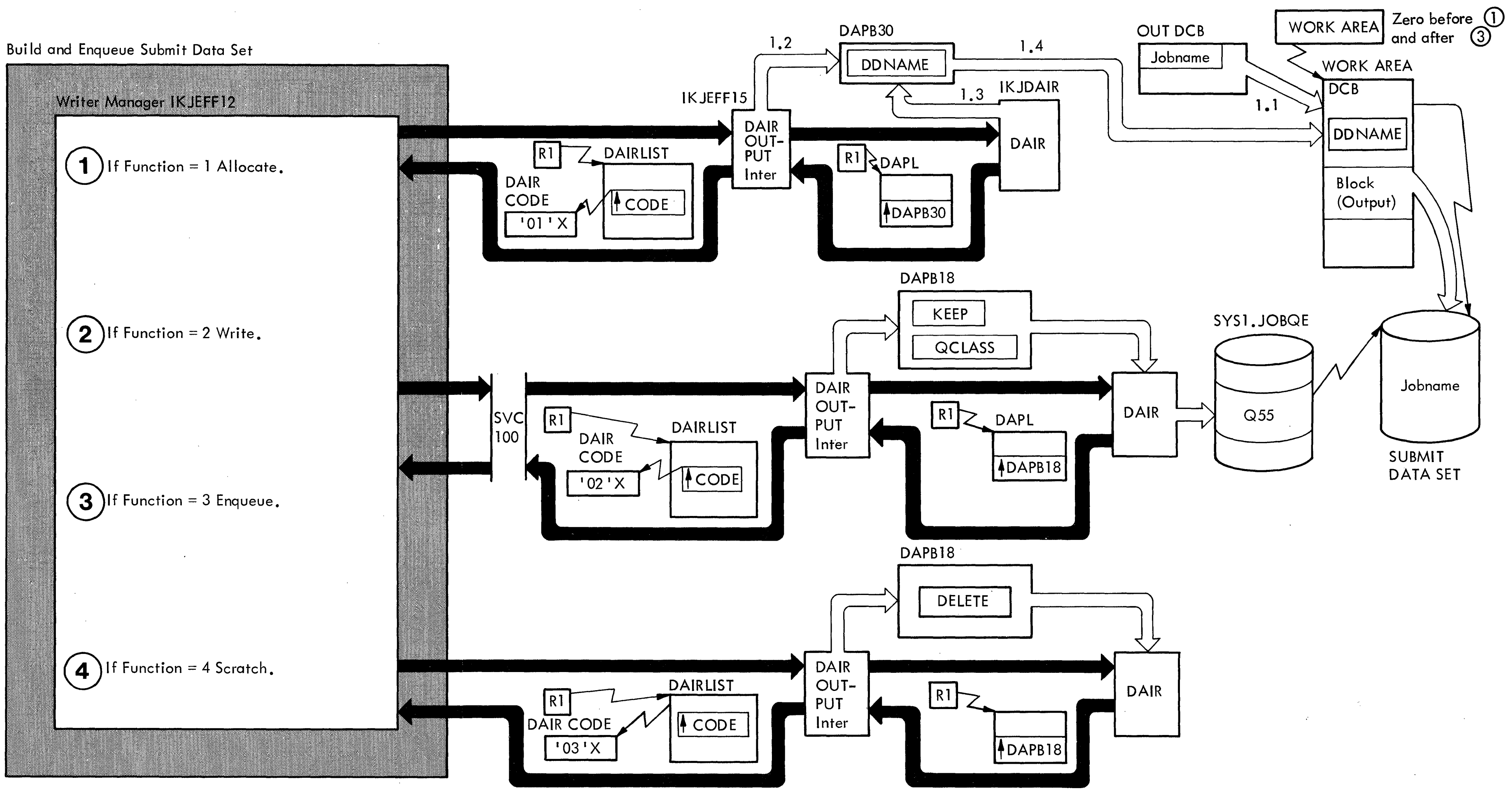
Writer Manager (IKJEFF12) controls this function.

- It is entered from Control (IKJEFF06) for functions 1, 2, and 3.
- For function 4 it is entered from the Stae Routine (IKJEFF11) under two conditions:

  a) If internal conditions requiring termination are detected, the command processor is in control and eventual return is to the TMP.

  b) If an Abend occurs, the Supervisor is in control and return is to the Supervisor to continue the Abend.





Description	Routine	Label	FC
1 A work area is set up. 1.1 -- The DCB is moved to the work area.	IKJEFF12		KA
1.2 -- A DAPB30 block is built.	IKJEFF15		NA
1.3 -- The data set is allocated and the DD name is put in DAPB30.	IKJDAIR		NA
1.4 -- Then the DD name is moved to the DCB in the work area and the data set is opened. Pointers to the DCB and the output block are put in the control table, the output block is initialized and return is made to CONTROL.	IKJEFF12	INITBLK	KA
2 The output block is written using the PUT macro and the output block is reinitialized. Return is made to CONTROL.	IKJEFF12	WRITE INITBLK	KB KA
3 The output block is written. The data set is closed. Dair output Interface is entered by way of SVC 100 (since Q55 is not a normal Sysout Q, Key Zero must be entered to enqueue the data set). Dair Interface sets up a DAPB18 block for KEEP and Q class 55 and DAIR is called to deallocate and enqueue the data set. The "job submitted" message is sent.	IKJEFF12 IKJEFF15 IKJDAIR	WRITE CLOSE DEALLOC	KB KB KB NA NA

Description	Routine	Label	FC
4 If the work area pointer is zero there is no data set to be scratched. (all started data sets have been enqueued). Return to CONTROL.	IKJEFF12	TURNOFF	KB KA
If the work area pointer is not zero the data set is closed and then Dair Output Interface is called with a DAIRCODE of 3 to deallocate and scratch the data set.		CLOSE DEALLOC	KB KB
Dair Interface builds a DAPB18 block, initializes it to request deletion and calls DAIR.	IKJEFF15		NA
Dair deallocates and scratches the data set.	IKJDAIR		NA
Writer manager sends the "submit terminated" message. The work area is freed and return is made to the caller.	IKJEFF12	FREEUP FREEUP1 TURNOFF	KA KC KA

Diagram 2-5. Build and Enqueue SUBMIT Data Set

Part 7: TERMINAL Command Processor

7

Contents

Illustrations

Charts

Diagrams

Figures

Section 1. Introduction

TERMINAL is a TSO Command Processor used to set the terminal control characters for the logged-on user of the system.

Functions

The TERMINAL Command Processor performs the following functions:

- Obtains the TERMINAL command string from the command buffer.
- Checks validity of line numbers, number of seconds, line size of the 2741, the number of display screen rows, and the display screen line size for the 2260.
- Scans Parse Descriptor List (PDL) for element pointer.
- Issues terminal status macros based on specified key options.
- Issues appropriate messages.

Environment

The TERMINAL Command Processor operates in a TSO user's region under the protection key assigned to that region. The processor's static data and instructions require about 2K of storage. Additional storage is used by the TSO service routines (IKJPARS and IKJPUTL). More definitive information about storage necessary for TSO and the TERMINAL Command Processor can be found in the Storage Estimates SRL, GC28-6551.

Physical Characteristics

The TERMINAL Command Processor consists of one load module. IKJEFT80 is the main processing module. It handles all the major functions of the processor. Control is returned to the terminal monitor program (TMP) following processing.

Operational Considerations

TERMINAL communicates with a terminal through the PUTLINE service routine (IKJPUTL).

The PARSE service routine (IKJPARS) scans and syntax checks the command string.

Input is the standard Command Processor Parameter List (CPPL). Upon receiving control from the TMP, register 1 points to the list.

The command buffer, pointed to by the CPPLCBUF , contains the TERMINAL command and its optional keyword parameters.

The TERMINAL command consists of the following:

- The command name TERMINAL or TERM (alias for TERMINAL).
- Optional keyword parameters to define operating characteristics (based on the type of terminal the user has). At least one of these keyword parameters must accompany the command name. Otherwise, the command is ignored.

  LINES(integer from 1-255) operand indicates an attention interrupt is to occur after the specified number of lines have been directed to the user's terminal. NOLINES operand specifies output lines count is not to be used to control the attention interrupt.

  SECONDS(integer from 1-255) operand indicates an attention interrupt is to occur after the specified number of seconds of program execution or continuous output to the user's terminal. NOSECONDS operand specifies that time is not to be used to control the attention interrupt. Any number specified will be rounded up to nearest multiple of 10 seconds.

  INPUT (string) operand specifies the character string entered at the end of any input line that will cause an attention interrupt. NOINPUT operand specifies no character string will cause an attention interrupt.

  BREAK operand indicates the user's terminal has the capacity of attention interrupt during input and output operations. NOBREAK operand specifies the user's terminal does not have both capacities.

TIMEOUT operand indicates the user's terminal keyboard will lock after a specified time interval of inactivity. NOTIMEOUT operand specifies the user's terminal keyboard will not lock due to inactivity.

LINESIZE (integer from 1-255) operand specifies the length of the line to be printed or displayed at the user's terminal.

SCRSIZE(n,m) operand specifies the screen dimensions of the display screen. "n" is the number of rows (1 - 255); "m" is the length of each row (1 - 255).

CLEAR(string) operand specifies a character string that can be used to erase the display station screen. NOCLEAR operand specifies no character string will be used to erase the screen.

Messages are printed at the terminal. These include diagnostic messages, informational messages, and prompting messages.

Section 2. Method of Operation

This section describes the logic of the TERMINAL Command Processor. It emphasizes the flow of data and control information through buffers and tables and contains a detailed functional description of the TERMINAL operation.

TERMINAL Command Processing Summary

TERMINAL processing begins when control is received from the TMP. If no operands are associated with the TERMINAL command, a message is issued. Return is to the TMP.

The PPL is initialized. The syntax of the TERMINAL command is checked. A special validity check is made on line number, number of seconds, and line size, if they are specified. Validity checks are also made on display screen line size and number of screen rows for 2260.

The PDL is checked for entries. If no entries are found, a message is issued. Control returns to the TMP.

Status macros are set and issued, based on user specified options.

Introduction to the Diagrams

The remainder of this section consists of the method of operation diagrams of the TERMINAL Command Processor.

The diagrams are designed to serve you in different ways:

- Initially, they quickly provide you with an understanding of the programming functions of the command processor.
- They show how the code supports the functions to be performed.
- They provide "pointers" to the actual implementations (by cross references to the flowcharts or program listings).

You can use them to become familiar with the system, or later, to find your way to specific implementation points in the program listings or other areas of the PLM.

Organization of Diagrams

Diagram 2-1 shows the functions of the controlling module. The diagram appears at the end of the PLM.

The processing block lists the functions performed in producing final output.

The descriptions provided at the left of the diagram list the major implementation steps performed in support of the processing functions.

The routine label and flowchart cross-referencing provided with the descriptions are "key" points designed to show you where to pick up the implementations in the program listings or flowcharts.

Symbols

This PLM uses the following symbols:

➡	Heavy black arrow indicates begin reading the diagram here.
⇨	White arrow indicates data transfer from one area to another.
➔	Black arrow shows flow of logic or passing of control.
→	Thin black arrow indicates pointer to an item.
⇢	Dotted arrow indicates reference to an item.
⬠	Off-page connector leads to a related diagram.
- - - -×	Getting or Freeing main storage.
➔□➔	Signifies passing of parameters.

Section 3. Program Organization

This section describes the TERMINAL command processor routine. Flowcharts are included following the discussion.

Hierarchy information is shown in Figure 3-1.

Hierarchy

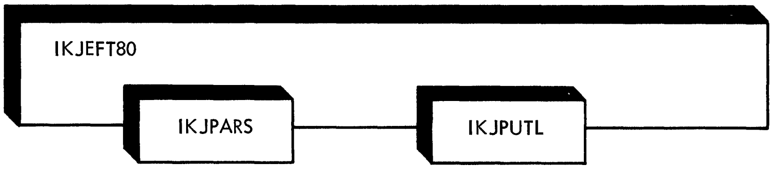


Figure 3-1. Control Module Hierarchy.

IKJEFT80

Entry points	IKJEFT80 is the primary entry point. LINNOCHK LINSZCHK Validity check routines SCHNK1 (Used by IKJPARS). SCHNK2 SECNOCHK
Operation	Checks validity of line numbers, number of seconds, line size of the 2741, the number of display screen rows, and the display screen line size for 2260. Scans PDL for element pointer. Issues terminal I/O controller (TIOC) macros based on specified keyword options. Issues appropriate messages.
Data areas used by IKJEFT80	PPL, CPPL, PDL.
Routines called	IKJPARS, IKJPUTL.
Exits	Normal return to caller, IKJEFT02. LINK to IKJPARS.
Registers	Upon entry, register 1 points to CPPL. Upon exit to the TMP, register 15 has a return code.

Cross reference	FC*	MO**
	AA	2-1
	AB	

FC* indicates flowcharts.
MO** indicates method of operation diagram.

Chart AA. TERMINAL Command Processing (1 of 2)

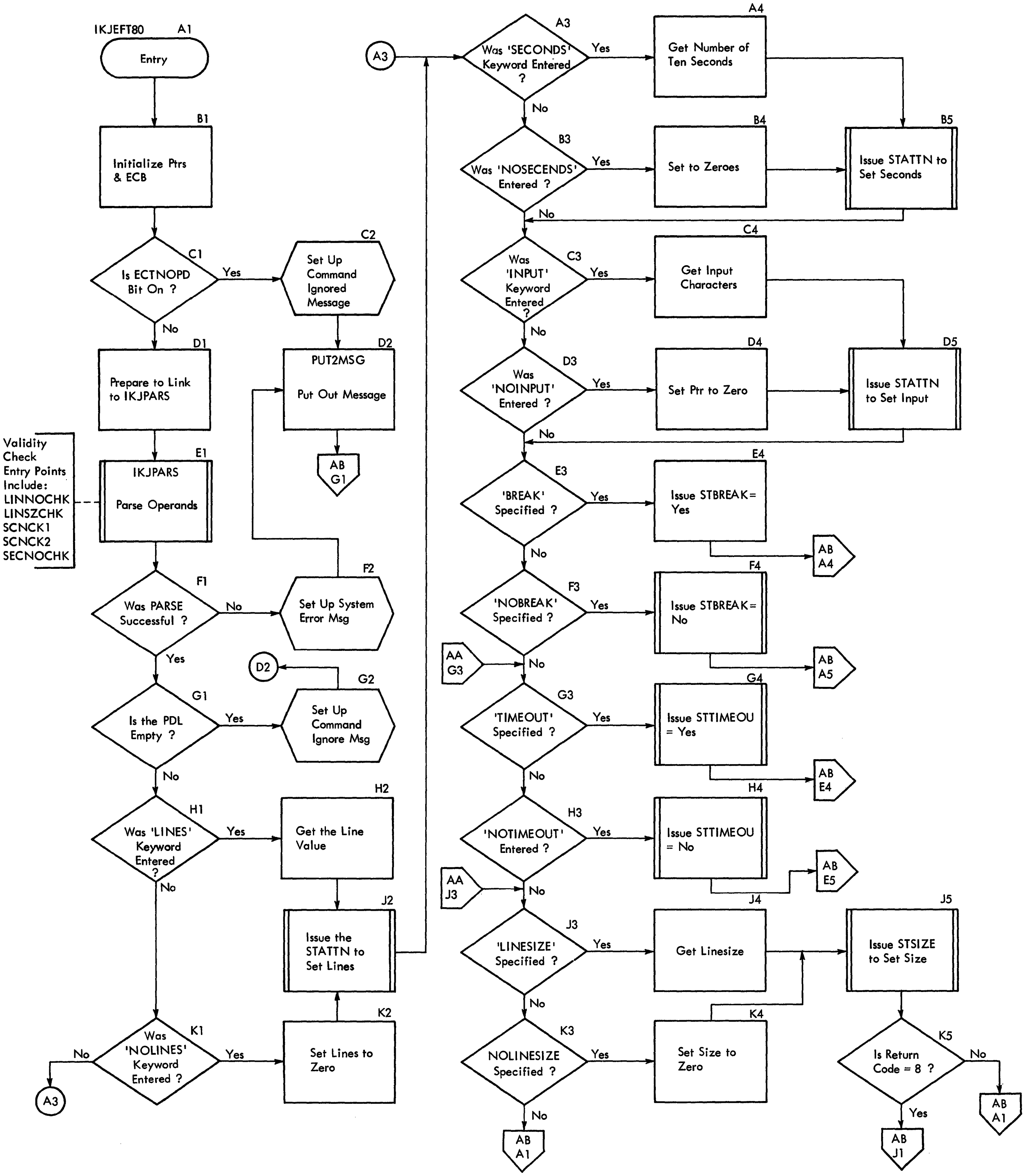
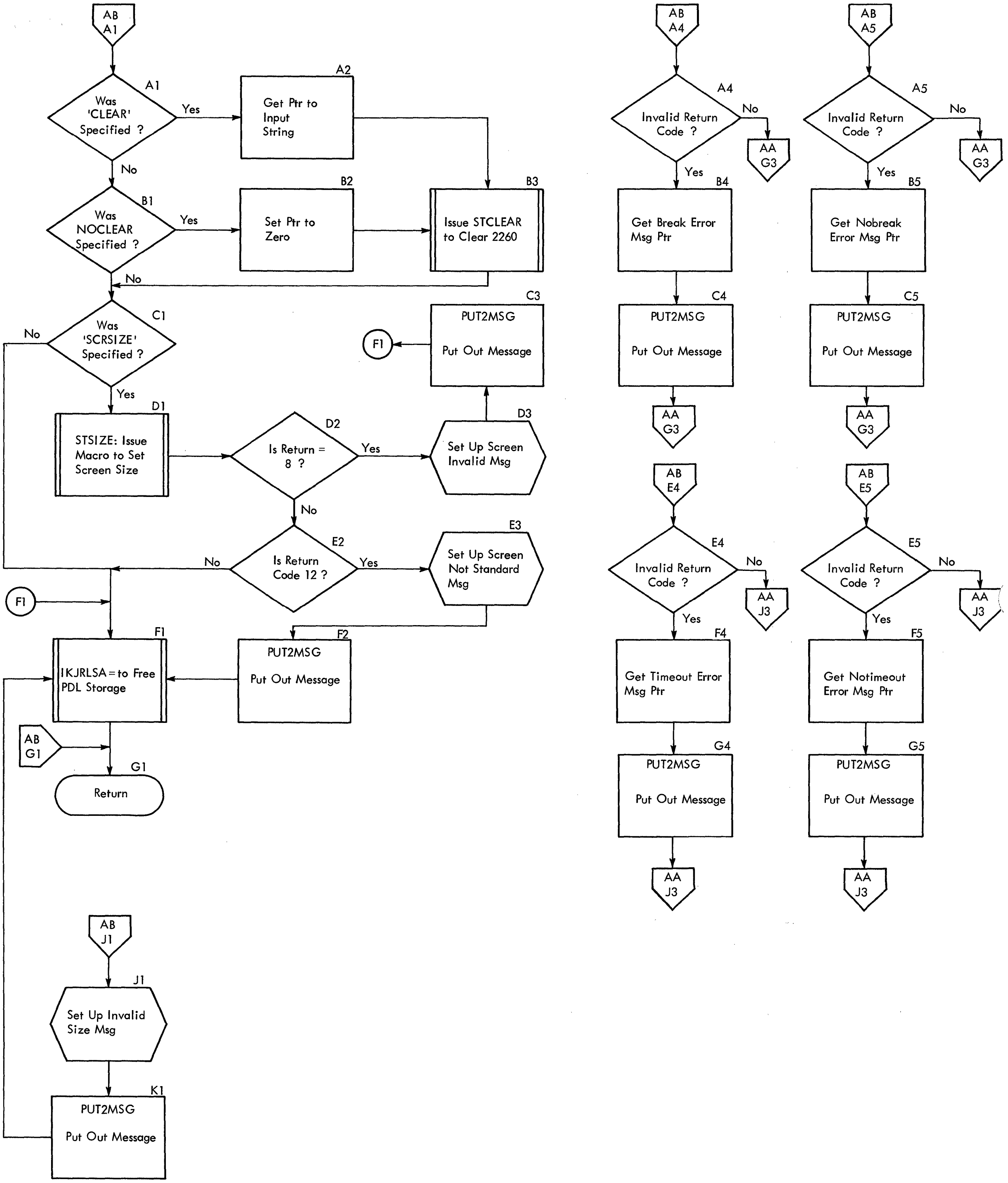


Chart AB. TERMINAL Command Processing (2 of 2)



Section 4. Directory

This section contains a routine directory organized alphabetically by entry point or routine name. A data area directory is also provided. It is arranged alphabetically. Cross references for the method of operation diagrams and flow charts are provided where applicable.

Routine Directory

Entry Point or Routine Name	Load Module	Control Section	Cross Reference	
			MO	FC
IKJEFT80	IKJEFT80	IKJEFT80	2-1	AA-AB
LINNOCHK	IKJEFT80	IKJEFT80	2-1	AA
LINSZCHK	IKJEFT80	IKJEFT80	2-1	AA
SCNCK1	IKJEFT80	IKJEFT80	2-1	AA
SCNCK2	IKJEFT80	IKJEFT80	2-1	AA
SECNOCHK	IKJEFT80	IKJEFT80	2-1	AA

Data Area Directory

Data Area	Cross reference	
	MO	FC
CPPL	2-1	AA
PDL	N/A	AA
PPL	N/A	AA

Section 5. Data Areas

This section contains the major data areas in the TERMINAL command processor. Pertinent information, such as which routine created the data area, which routine updates or uses the data area, and what the data area contains, is included.

CPPL—Command Processor Parameter List

Created by: TMP.

Used by: IKJEFT80.

Contents: A list of pointers.

Disp		Field	Size and Use
0	(0)	CPPLCBUF	4 bytes -- Points to command buffer.
4	(4)	CPPLUPT	4 bytes -- Points to the User Profile Table (UPT).
8	(8)	CPPLPSCB	4 bytes -- Points to the Protected Step Control Block (PSCB).
12	(C)	CPPLECT	4 bytes -- Points to the Environment Control Table (ECT).

Cross reference FC MO

AA N/A

PDL—PARSE Descriptor List (Local Description)

Created by: IKJPARS.

Used by: IKJEFT80.

Contents: IKJPARS Descriptor List. Pointed to by PDEPTR.

Disp	Field	Size and Use
0 (0)	STORPTR	4 bytes -- Points to PARSE storage for DSECT.
4 (4)	STORB	4 bytes -- Subpool length.
8 (8)	LNE	2 bytes -- Line keyword number.
10 (A)	SECSNO	2 bytes -- Seconds keyword number.
12 (C)	IPUT	2 bytes -- Input keyword number.
14 (E)	LINSZ	2 bytes -- Line size keyword number.
16 (10)	BRK	2 bytes -- Break keyword number.
18 (12)	TIMOUT	2 bytes -- Timout keyword number.
20 (14)	LINE1	4 bytes -- Points to line number.
24 (18)	LINE2	2 bytes -- Length of line number.
26 (1A)	LINE3	2 bytes -- Reserved.
28 (1C)	SECS1	4 bytes -- Points to number of seconds.
32 (20)	SECS2	2 bytes -- Length of number of seconds.
34 (22)	SEC3	2 bytes -- Reserved.
36 (24)	INPUT1	4 bytes -- Points to input.
40 (28)	INPUT2	2 bytes -- Length of input.
42 (2A)	INP3	2 bytes -- Reserved.

PDL -- Parse Descriptor List (Local Description) (Cont.)

Disp	Field	Size and Use
44 (2C)	LINESZ1	4 bytes -- Points to line size.
48 (30)	LINESZ2	2 bytes -- Length of line size.
50 (32)	LINS3	2 bytes -- Reserved.

Cross reference FC MO

AA N/A

PPL—PARSE Parameter List

Created by: IKJEFT80.

Used by: IKJPARS.

Contents: Parameter List.

Disp	Field	Size and Use
0 (0)	PPLUPT	4 bytes -- Points to UPT.
4 (4)	PPLECT	4 bytes -- Points to ECT.
8 (8)	PPLECB	4 bytes -- Points to ECB.
12 (C)	PPLPCL	4 bytes -- Points to PCL.
16 (10)	PPLANS	4 bytes -- Points to answer place.
20 (14)	PPLCBUF	4 bytes -- Points to command buffer.
24 (18)	PPLUWA	4 bytes -- Points to user work area.

Cross reference FC MO

AA N/A

Section 6. Diagnostic Aids

This section contains a register summary and the messages from the TERMINAL command processor.

Register Summary

Register	Use
1	Points to CPPL. Used as work register.
2	Work register.
3	Work register.
7	Work register.
12	Base register.
13	Save area register.
14	Return register.
15	Return code register.

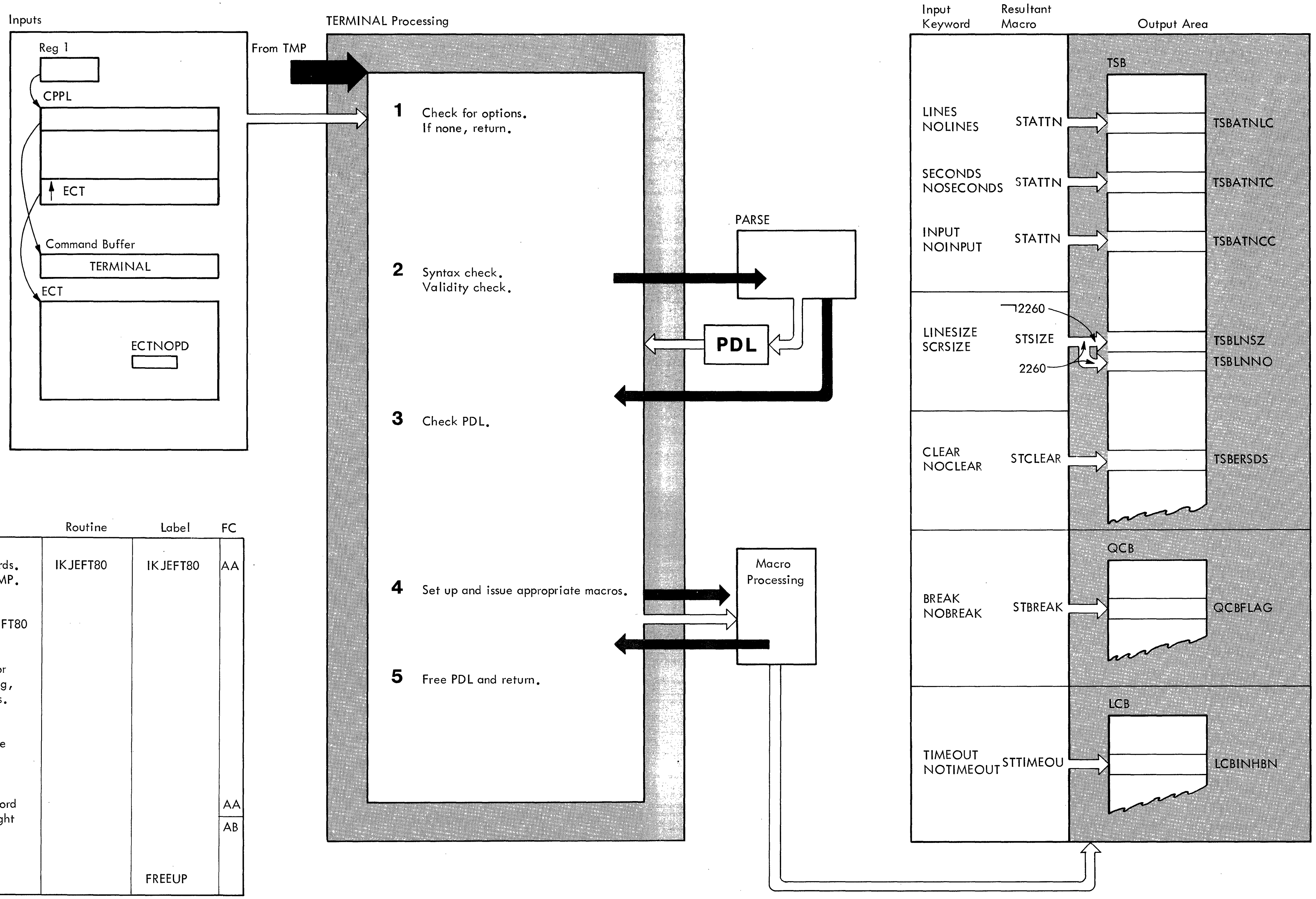
Messages

Message ID	Message Text
IKJ56655I	NO OPERANDS, COMMAND IGNORED
IKJ56656I	INVALID LINESIZE OPERAND, USE SCRSIZE
IKJ56657I	INVALID SCRSIZE OPERAND, USE LINESIZE
IKJ56658I	SCREEN SIZE NOT STANDARD, SCREEN CONTROL ERROR MAY OCCUR
IKJ56659I	BREAK INVALID FOR YOUR TERMINAL
IKJ56660I	NOBREAK INVALID FOR YOUR TERMINAL
IKJ56661I	TIMEOUT INVALID FOR YOUR TERMINAL
IKJ56662I	NOTIMEOUT INVALID FOR YOUR TERMINAL
IKJ56663I	INVALID NUMBER OF ROWS
IKJ56664I	INVALID LENGTH OF ROW
IKJ56667I *IKJ56667I	COMMAND SYSTEM ERROR+ PARSE ERROR CODE XXX

* indicates second level message for documentation only

Index

Indexes to program logic manuals are consolidated in the publication IBM System/360 Operating System: Program Logic Manual Master Index, Order No. GY28-6717. For additional information about any subject listed below, refer to other publications listed for the same subject in the Master Index.



Description	Routine	Label	FC
1 The ECTNOPD bit of the ECT is checked to see if there are any TERMINAL keywords. If none, the command is ignored, a message is printed and control returns to the TMP.	IKJEFT80	IKJEFT80	AA
2 The PARSE routine is given control to syntax check the command. In addition IKJEFT80 passes five entry points (LINNOCHK, SECNOCHK, LINSZCHK, SCNCK1, and SCNCK2) validity checking routines within IKJEFT80. If line or second keywords are provided, parse will temporarily pass control back to the appropriate routine for validity checking. When PARSE returns control to IKJEFT80 for mainline processing, it provides a return code that informs IKJEFT80 of the results of the validity checks.			
3 IKJEFT80 checks the PDL pointer (PDEPTR) to see if the PDL is empty. If it is, the terminal command is ignored, a message is written and control returns to the TMP.			
4 The TERMINAL status macros are set up and issued according to the specified keyword values. The end result is the placing of values in output fields, as shown at the right of this diagram.			AA AB
5 The PDL is freed and control returns to the TMP.		FREEUP	

Note: The PUTLINE macro is used for I/O operations to the terminal.

Note: The TSB, QCB, and LCB are initially set up by LOGON. These areas are subsequently referred to by the TCAM interface.

Diagram 2-1. TERMINAL Operational Characteristics

Part 8: TIME Command Processor

8

Contents

Illustrations

Charts

Diagram

Section 1. Introduction

The TIME Command Processor computes the total session time used by the terminal user.

Functions

The TIME Command Processor performs the following functions:

- Initializes pointers to the parameter lists used.
- Obtains the time from the PSCB fields, using the TIME macro.
- Converts the time to printable characters.
- Prints time for user with the PUTLINE (IKJPUTL) service routine.

Environment

The TIME Command Processor operates in a TSO user's region under the protection key assigned to that region.

The processor's static instructions and data require about 600 bytes of storage. An additional 1.5K is needed for the service routine IKJPUTL. More definitive figures on storage needed for the TSO option and the TIME Command Processor can be found in the publication IBM System/360 Operating System: Storage Estimates, GC28-6551.

The processor is called by the Terminal Monitor Program (TMP), rather than attached as most processors are.

Interfaces between this processor and the operating system are:

- The TMP that calls TIME.
- The TSO service routine IKJPUTL.
- The system macros. CALL, TIME, PUTLINE, TSEVENT, IKJPSCB.

Operational Considerations

Input to the processor is the Command Processor Parameter List (CPPL).

The command buffer, pointed to by CPPLCBUF, contains the TIME command. There are no positional operands or optional keywords.

Output consists of a message to the terminal user, giving him the execution and session time.

Section 2. Method of Operation

This section describes the logic of the TIME Command Processor. It emphasizes the flow of data and control information through buffers and tables, and contains detailed functional descriptions of the TIME operation.

Program logic information for the TMP and the TSO service routine used by the TIME Command Processor is contained in the IBM System/360 Operatinq System: Time Sharinq Option Terminal Monitor Proqram and Service Routines Proqram Loqic Manual, GY28-6770.

TIME Command Processing Summary

The TIME Command Processor computes the total session time used by the terminal user.

TIME receives control from the TMP by a CALL when a terminal user enters the TIME command. The processor takes the PSCB time fields and converts them to printable characters.

The resultant figures are written to the terminal user by the IKJPUTL service routine.

Introduction to the Diagram

The remainder of this section consists of the method of operation diagram that can be found at the back of the PLM.

The diagram is designed to serve you in different ways:

- Initially, it quickly provides you with an understanding of the programming functions of the command processor.
- It shows how the code supports the functions to be performed.

You can use it to become familiar with the system, or later, to find your way to specific implementation points in the program listings or other areas of the PLM.

Organization of Diagram

The processing block lists the functions performed in producing final output. The descriptions provided at the left of the diagram list the major implementation steps performed in support of the processing functions.

Symbols

This PLM uses the following symbols:

- ➡ Heavy black arrow indicates begin reading the diagram here.
- ⇨ White arrow indicates data transfer from one area to another.
- ➜ Black arrow shows flow of logic or passing of control.
- ——► Thin black arrow indicates pointer to an item.
- – – –► Dotted arrow indicates reference to an item.
- ⬠ Off-page connector leads to a related diagram.
- – – – –× Getting or Freeing main storage.
- ➡□➡ Signifies passing of parameters.

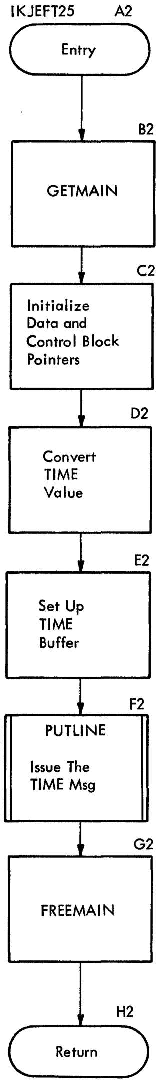
Section 3. Program Organization

Only one control section is required for the TIME Command Processor. No overall hierarchy is provided. The flowchart follows the information on the control section.

IKJEFT25

Entry point	Entered from IKJEFT02 or IKJEFT03 by a CALL macro instruction. IKJEFT25.
Operation	Obtains the amount of CPU time, and execution time from the PSCB, and computes the elapsed time for the terminal session from the PSCB. Gets a buffer for output message. Converts times to hours, minutes, and seconds. Prints converted times using the PUTLINE macro.
Data Areas used	Command Buffer, ECB, ECT, Parameter list (unnamed), PSCB, PTPB, PUTLBUF, PUTPL, TIMESMSG, TIMESBUF, UPT.
Routines called	IKJPUTL.
Exits	Normal return to caller.
Registers	Upon entry, register 1 points to a parameter list. Standard linkage conventions.

Chart AA. TIME Command Processing



Section 4. Directory

The routine directory and the data area directory normally found in this section of a PLM are not provided. This command processor contains one routine and 7 data areas.

Section 5. Data Areas

This section contains the major data areas in the TIME Command Processor. Pertinent information, such as which routine constructed the data area, which routine uses the data area, and what the data area contains is included.

Parameter List Passed by TMP

Constructed by: TMP.

Used by: IKJEFT25.

Contents: Parameter list pointed to by register 1 upon entry.

Displacement Dec.	Hex.	Field	Size	Use
0	(0)	unnamed	4 bytes	Points to command buffer.
4	(4)	unnamed	4 bytes	Points to User Profile Table (UPT).
8	(8)	unnamed	4 bytes	Points to Protected Step Control Block (PSCB).
12	(C)	unnamed	4 bytes	Points to Environment Control Table (ECT).

PSCB—Protected Step Control Block

Constructed by: LOGON.

Used by: IKJEFT25.

Contents: User attribute and accounting data. Pointed by the PSCBPTR.

Displacement Dec.	Hex.	Field	Subfield	Size (byte)	Use
0	(0)	PSCBUSER		7	Userid.
7	(7)	PSCBUSRL		1	Length of userid.
8	(8)	PSCBGPNM		8	Esoteric group name.
16	(10)	PSCBATR1		1	User attributes (for IBM use).
			PSCBCTRL		X'80' indicates OPERATOR command user.
			PSCBACCT		X'40' indicates ACCOUNT command user.
			PSCBJCL		X'20' indicates SUBMIT command user.
					Remaining 5 bits are reserved.
17	(11)	unnamed			Reserved
18	(12)	PSCBAT2		1	User attributes (for installation use).
19	(13)	unnamed		1	Reserved.
20	(14)	PSCBCPU		4	Cumulative CPU time session.
24	(18)	PSCBSWP		4	Cumulative time resident session.
28	(1C)	PSCBLTIM		4	Actual LOGON time of day.
32	(20)	PSCBTCPU		4	Total CPU time (accounting period).
36	(24)	PSCTSWP		4	Total time resident (accounting period).

PSCB -- Protected Step Control Block (Cont.)

Displacement Dec. Hex.	Field	Subfield	Size (Byte)	Use
40 (28)	PSCBTCON		4	Total time connected (accounting period).
44 (2C)	PSCBTCO1		4	Total time connected (accounting period).
48 (30)	PSCBRLGB		4	Points to relogon buffer.
52 (34)	PSCBUPT		4	Points to UPT.
56 (38)	PSCBUPTL		2	Length of UPT.
58 (3A)	PSCBRSZ		4	Region size requested in 2K units.
62 (3E)	PSCBU		8	Reserved for installation use.

PTPB—IKJPUTL Parameter Block

Constructed by: IKJEFT25.

Used by: IKJPUTL.

Contents: The IKJPUTL parameter block is used to return pertinent information and to control IKJPUTL functions. It is pointed to by PTPBPTR.

Displacement Dec. Hex.	Field	Size	Use
0 (0)	PTPBOPUT	4 bytes	Address of output line descriptor or data line.
4 (4)	PTPBFLN	4 bytes	Points to formatted line returned.

PUTLBUF—IKJPUTL Buffer

Constructed by: IKJEFT25.

Used by: IKJPUTL.

Contents: IKJPUTL message list. Pointed to by PUTPARMP.

Displacement Dec. Hex.	Field	size	Use
0 (0)	SEGNUM	4 bytes	Number of segments.
4 (4)	SEG1PTR	4 bytes	First segment pointer.
8 (8)	SEG2PTR	4 bytes	Second segment pointer.
12 (C)	SEG3PTR	4 bytes	Third segment pointer.
16 (10)	SEG4PTR	4 bytes	Fourth segment pointer.

PUTPL—IKJPUTL Parameter List

Constructed by: IKJEFT25.

Used by: IKJPUTL.

Contents: IKJPUTL parameter list. Found by adding the contents of register 12 to the displacement value of PUTPL.

Displacement Dec. Hex.	Field	Size	Use
0 (0)	UPTPTR	4 bytes	Points to UPT.
4 (4)	ECTPTR	4 bytes	Points to ECT.
8 (8)	ECBPTR	4 bytes	Points to event control block.
12 (C)	PUTPARMP	4 bytes	Points to PUTLBUF.

TIMEMSG—Primary Message Buffer

Constructed by: IKJEFT25.

Used by: IKJEFT25.

Contents: Primary message buffer. Occupies same space as MSGMOD.

Displacement Dec.	 Hex.	Field	Size	Use
0	(0)	TIMLEN	2 bytes	Length of primary segment.
2	(2)	TIMOFF	2 bytes	Offset to primary segment.
4	(4)	TIMMSG	42 bytes	Primary message segment.

TIMESBUF—Segmented Output Message Buffer

Constructed by: IKJEFT25.

Used by: IKJEFT25, IKJPUTL.

Contents: The segmented output message buffer. Found by adding the contents of register 12 and the displacement value of TIMESBUF in the cross reference table.

Displacement Dec. Hex.	Field	Size	Use
0 (0)	CPULEN	2 bytes	CPU formatted length.
2 (2)	CPUOFF	2 bytes	CPU time offset.
4 (4)	CPUSTIME	8 bytes	CPU time.
12 (C)	EXLEN	2 bytes	Execution time length.
14 (E)	EXOFF	2 bytes	Execution time offset.
16 (10)	EXSTIME	8 bytes	Execution time.
24 (18)	SESLEN	2 bytes	Session time length.
26 (1A)	SESOFF	2 bytes	Session time offset.
28 (1C)	CPUTTIME	8 bytes	Total session time.

Section 6. Diagnostic Aids

This section contains a register summary and the messages from the TIME Command Processor.

Register Summary

Register	Use
0	Work register
1	Parameter passing register
2	Work register
3	Work register
7	Work register
8	Work register
11	Base register for static code
12	Base register for dynamic storage
13	Save area register
14	Return register
15	Return code register

Messages

The only message that is written from this processor is

```
  IKJ56657I   CPU - HH:MM:SS EXECUTION - HH::MM::SS SESSION - HH:MM:SS
```

where H means hours, M means minutes, and S means seconds.

Index

Indexes to program logic manuals are consolidated in the publication IBM System/360 Operating System: Program Logic Manual Master Index, Order No. GY28-6717. For additional information about any subject listed below, refer to other publications listed for the same subject in the Master Index.

The TIME command processor uses PSCB time fields, which are in timer units (26.04 micro seconds) and converts them to a printable format of hours, minutes and seconds (HH:MM:SS). The PUTLINE service routine is used to write time information to the user.

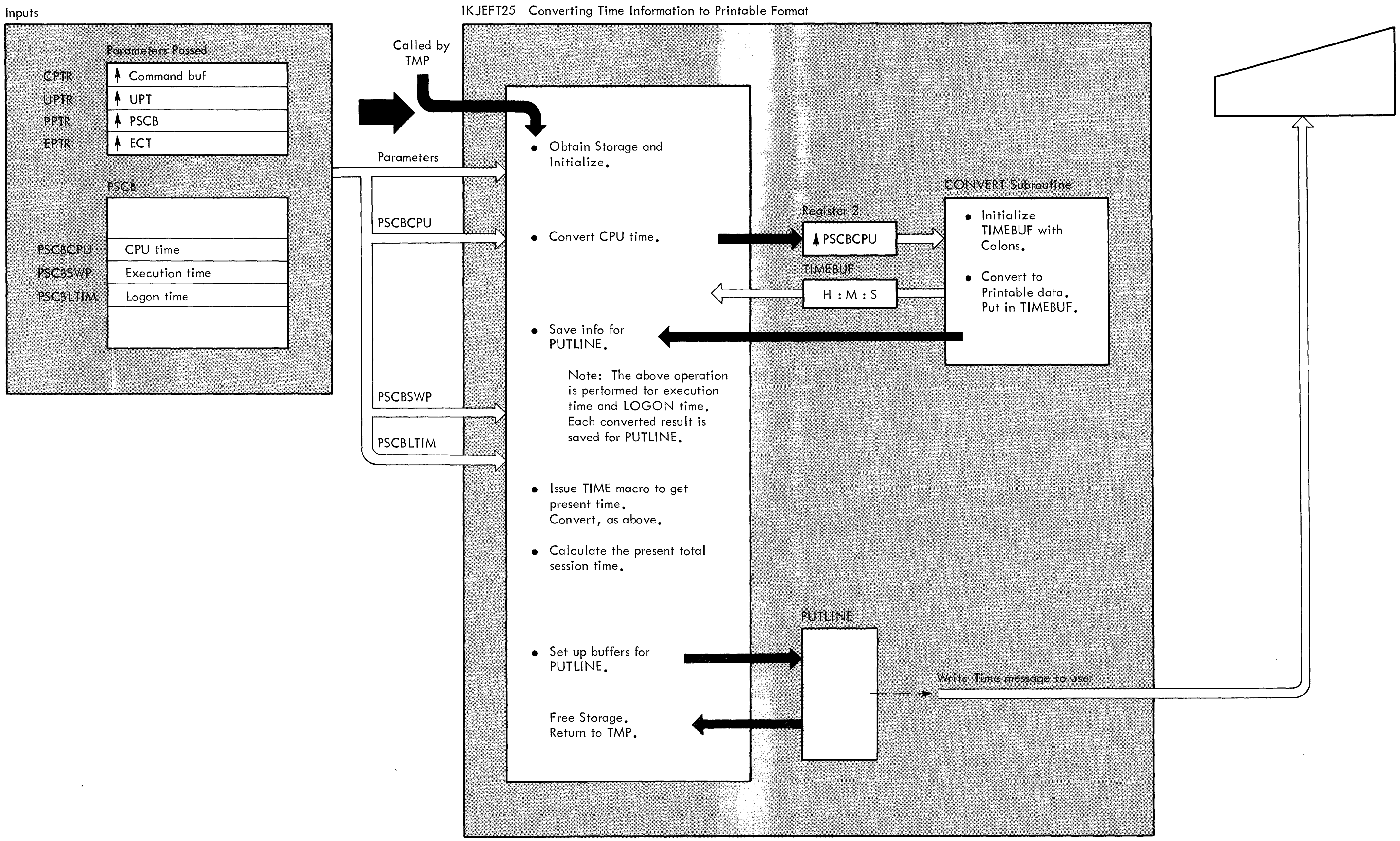

Inputs
Parameters Passed
CPTR
UPTR
PPTR
EPTR
Command buf
UPT
PSCB
ECT
PSCB
PSCBCPU
PSCBSWP
PSCBLTIM
CPU time
Execution time
Logon time
Called by TMP
Parameters
PSCBCPU
PSCBSWP
PSCBLTIM
IKJEFT25 Converting Time Information to Printable Format
• Obtain Storage and Initialize.
• Convert CPU time.
• Save info for PUTLINE.
Note: The above operation is performed for execution time and LOGON time. Each converted result is saved for PUTLINE.
• Issue TIME macro to get present time. Convert, as above.
• Calculate the present total session time.
• Set up buffers for PUTLINE.
Free Storage. Return to TMP.
Register 2
PSCBCPU
TIMEBUF
H : M : S
CONVERT Subroutine
• Initialize TIMEBUF with Colons.
• Convert to Printable data. Put in TIMEBUF.
PUTLINE
Write Time message to user

Diagram 2-1. TIME Command Processing

Part 9: WHEN/END Command Processor

9

Contents

Illustrations

Charts

Diagrams

Figures

Section 1. Introduction

WHEN/END is a TSO command processor designed to terminate a command or a command procedure, which is a prearranged sequence of TSO commands. Optionally, a new command may also be initiated.

The action taken upon execution of the WHEN command is determined by a comparison between the return code which has been set by the previous command processor and an integer (1 to 4 characters) indicated in the WHEN command.

Functions

The END command processing does the following:

- Marks the current procedure for deletion.
- Returns control to the caller.

The WHEN command processing does the following:

- Obtains the WHEN command string from the command buffer.
- Checks validity of the operator. An operator may be =, <, ¬=, >, <=, ¬>, >=, ¬<, EQ, LT, NE, NG, GE, or NL.
- Uses this operator to compare the previous return code with an indicated integer.
- If the comparison is false, causes execution to proceed as if the WHEN command had been a NOP.
- If the comparison is true, marks the current procedure for deletion.
- Checks for an action operand (new command).
- Places any new command on the input stack.
- Terminates the procedure if END is specified (Defaults to END if nothing is specified).
- Returns control to the caller. If WHEN is entered from a terminal, control is returned there. It returns there immediately if the action operand is END, or it returns there after execution of the command if the action operand is another command.

Environment

The WHEN/END Command Processor operates in a TSO user's region under the protection key assigned to that region. The processor's data and instructions require about 6K of storage. About 400 bytes of dynamic storage is used. Definitive requirements on space needed to operate using the WHEN/END Command Processor can be found in the Storage Estimates SRL, GC28-6551.

Physical Characteristics

The processor consists of one load module containing the control section (IKJEFE11) and the message control sections (IKJEFE15 and IKJEFE16). IKJEFE11 handles all major functions of the processor and returns control at the end of processing to the Terminal Monitor Program (TMP). The message control sections handle the mechanics of sending messages.

Operational Considerations

The WHEN/END Command Processor is normally invoked during the execution of a command procedure. The WHEN/END command consists of the following:

- The command name WHEN or the command name END. If END is specified, the procedure is terminated.

- The positional operands required for the WHEN command: SYSRC, which indicates that the return code in the Environmental Control Table (ECT) is to be tested; the operator to be used; and an integer to which the return code is to be compared.

- Any optional action operand. Any valid TSO command may be entered with its associated parameters.

Input is a return code from the previous command processor which is to be compared with an integer specified in the WHEN statement. The operator used for the comparison is also specified in the WHEN statement.

Output includes the same return code from the ECT as before WHEN executed.

Messages are printed at the terminal. These include diagnostic messages, and prompting messages (if not suppressed).

Section 2. Method of Operation

This section briefly summarizes the method of operation of the WHEN/END Command Processor. It includes diagrams which illustrate basic functions.

WHEN Command Processing Summary

WHEN command processing begins when control is received from the TMP. The general parameter list is initialized for PARSE and control is given to the IKJPARS service routine to check the syntax of the WHEN command. Upon successful completion, control returns to IKJEFE11, which checks the operator. If valid, this operator is used to compare the return code to the specified integer. If the comparison is true, control goes to IKJSTCK, which marks the current procedure (the top entry of the input stack) for deletion. Following this, the action operand (if any) is placed on top of the input stack. On an error diagnostic message is issued and control returns to the caller (either the TMP or the terminal).

END Command Processing Summary

The END command processing begins when control is received from the TMP. The general parameter list is initialized for STACK. If the command is valid, the current procedure is marked for deletion and control returns to the caller.

Introduction to the Diagrams

The remainder of this section consists of the method of operation diagrams of the WHEN/END command processor. The diagrams appear at the end of the PLM.

The diagrams are designed to serve you in different ways:

- Initially, they quickly provide you with an understanding of the programming functions of the command processor.
- They show how the implementations support the functions to be performed.
- They provide "pointers" to the actual implementations (by cross references to the flowcharts or program listings).

You can use them to become familiar with the system, or later, to find your way to specific implementation points in the program listings or other areas of the PLM.

Organization of Diagrams

Diagram 2-1 shows WHEN command processing. Diagram 2-2 shows END command processing.

The processing block lists the functions performed in producing intermediate or final output.

The descriptions provided at the left of the diagram list the major implementation steps performed in support of the processing functions.

The routine label and flowchart cross-referencing provided with the descriptions are "key" points designed to show you where to pick up the implementations in the program listings or flowcharts.

Symbols

This PLM uses the following symbols:

➡	Heavy black arrow indicates begin reading the diagram here.
⇨	White arrow indicates data transfer from one area to another.
➔	Black arrow shows flow of logic or passing of control.
→	Thin black arrow indicates pointer to an item.
⇢	Dotted arrow indicates reference to an item.
⬠	Off-page connector leads to a related diagram.
– – – –×	Getting or Freeing main storage.
➡□➡	Signifies passing of parameters.

Section 3. Program Organization

This section describes the **WHEN/END** Command Processor routines and how they relate to each other. Flowcharts are included at the end of the discussion.

Figure 3-1 shows the control module hierarchy. Figure 3-2 shows the message control hierarchy.

Hierarchy

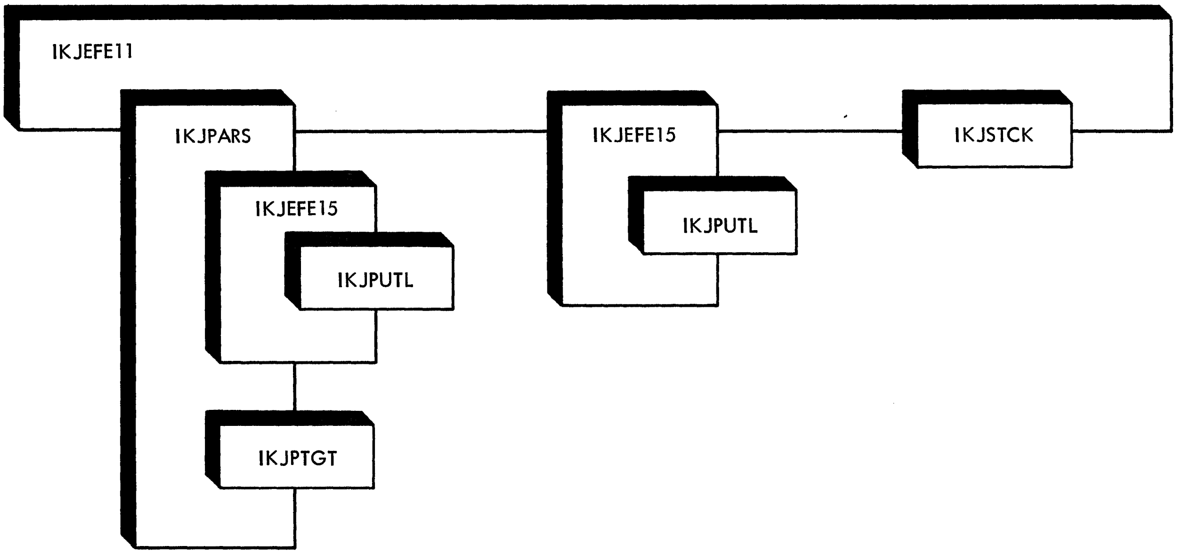


Figure 3-1. Control module hierarchy

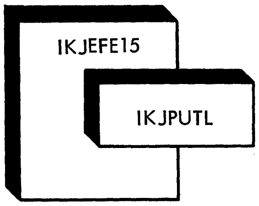


Figure 3-2. Message module hierarchy

IKJEFE11—Control Module

Entry point	IKJEFE11 -- from the TMP.
Operation	A check is made to determine if the command is END. If the command is END, the current element on the input stack is marked for deletion. Control returns to the user. If the command is WHEN, two parameters, an operator and an integer are passed to IKJEFE11. Operation is as follows: • Check to see if previous command processor was abended. If it has, a message is issued and control returns to the user. • Prepares buffer for IKJPARS. • Syntax and validity checks the WHEN command and the operator. • A comparison is made between the most recent return code and the integer. If the comparison is satisfied, the current element on the input stack is marked for deletion. If the comparison is not satisfied, the processor is terminated. • Any valid action operand found is put on the input stack. • Verification of placement is made and if unsuccessful a message is issued. • Control returns to the caller.
Data areas defined by IKJEFE11	IKJWHEN used as a general work area. IKJLSD IKJPPL
Data areas updated by IKJEFE11	IKJWHEN ECT (Indirectly. The TMP puts in the return code passed to it by IKJEFE11.)
Data areas used by IKJEFE11	IKJCPPL IKJPPL PCL IKJECT IKJSTPL PDL IKJLSD IKJWHEN IKJSTPB

IKJEFE11 -- Control Module (Cont.)

Routines called	LINK to IKJPARS (To check syntax.) IKJRLSA (To release PARSE parameter list.) LINK to IKJSTCK (To add or delete element from the input stack.)
Exits	RETURN to TMP. CALL to IKJEFE15.
Registers	Upon entry -- All registers are saved. Register Contents 1 Pointer to CPPL. 13 Pointer to register save area. 14 Return address. 15 Entry point address. Upon exit -- All registers are restored. Upon exit to IKJEFE15, register 1 has address of IKJWHEN. Upon exit to TMP, register 15 has the same return code as originally in the ECT.

Cross reference	FC*	MO**
	AA	2-1
	AB	2-2
	AC	

FC* indicates flowcharts
MO** indicates method of operation diagrams

IKJEFE15—Message Module

Entry point	IKJEFE15.
Operation	IKJEFE15 sends a message to the terminal.
Data areas defined by IKJEFE15	PTPB.
Data areas updated by IKJEFE15	PTPB (within IKJWHEN).
Data areas used by IKJEFE15	IKJWHEN IOPL PTPB.
Routines called	IKJPUTL (To send the message.)
Exits	To IKJEFE11 via a RETURN (BR 14).
Registers	Upon entry -- All registers are saved. Register 1 points to the work area set up in IKJEFE11 (IKJWHEN). Upon exit -- All registers are restored.

Cross Reference	FC	MO
	AA	2-1
	AB	2-2
	BA	

Chart AA. WHEN/END Command Processing Overview

Cross Reference: Diagram 2-1, 2-2.

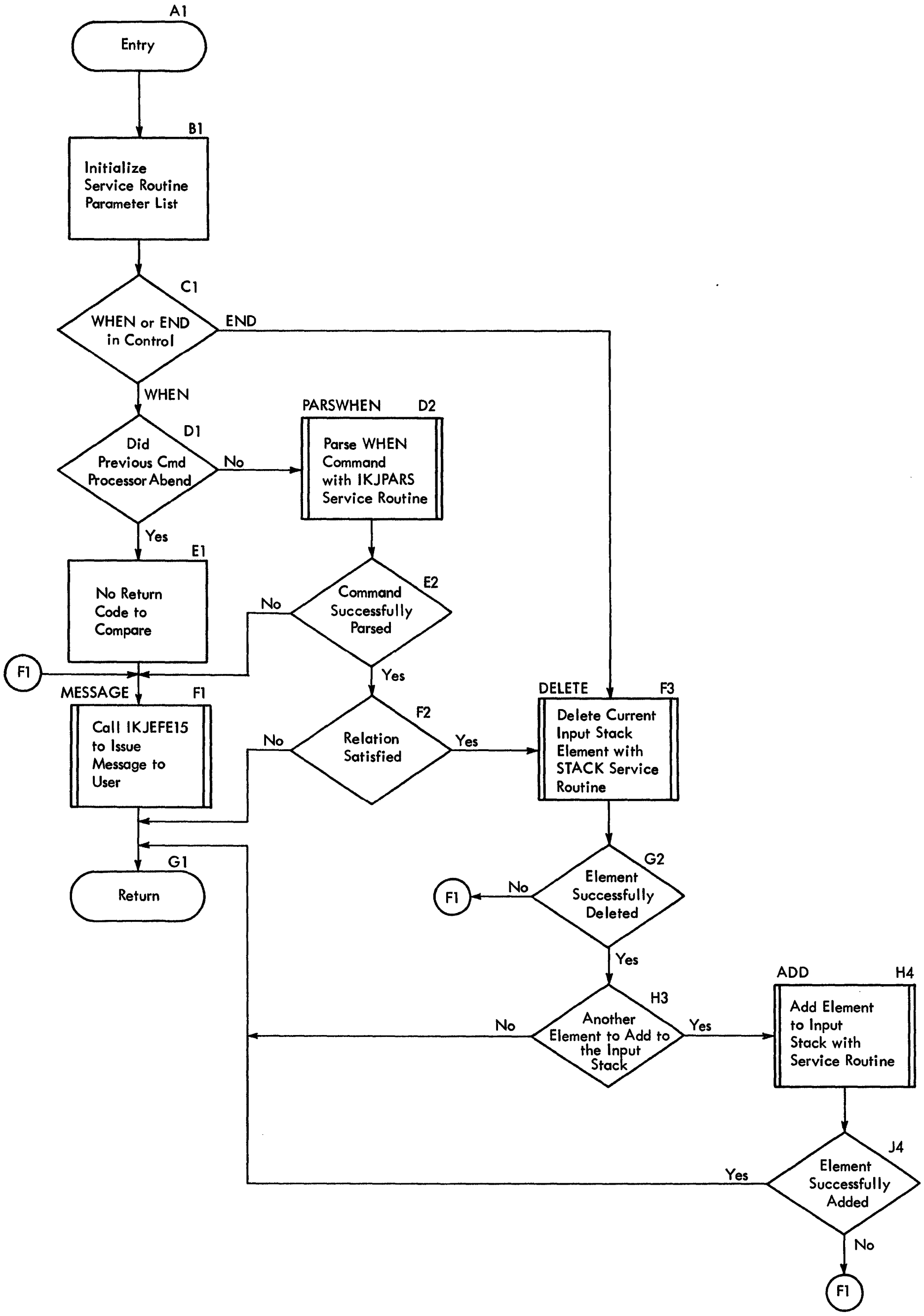

A1
Entry
B1
Initialize Service Routine Parameter List
C1
WHEN or END in Control
END
WHEN
D1
Did Previous Cmd Processor Abend
No
Yes
PARSWHEN D2
Parse WHEN Command with IKJPARS Service Routine
E1
No Return Code to Compare
E2
Command Successfully Parsed
No
Yes
F1
MESSAGE F1
Call IKJEFE15 to Issue Message to User
F2
Relation Satisfied
No
Yes
DELETE F3
Delete Current Input Stack Element with STACK Service Routine
G1
Return
G2
Element Successfully Deleted
No
F1
Yes
H3
Another Element to Add to the Input Stack
No
Yes
ADD H4
Add Element to Input Stack with Service Routine
J4
Element Successfully Added
Yes
No
F1

Chart AB. IKJEFE11. WHEN/END Command Processing (1 of 2)

Cross Reference: Diagram 2-1.

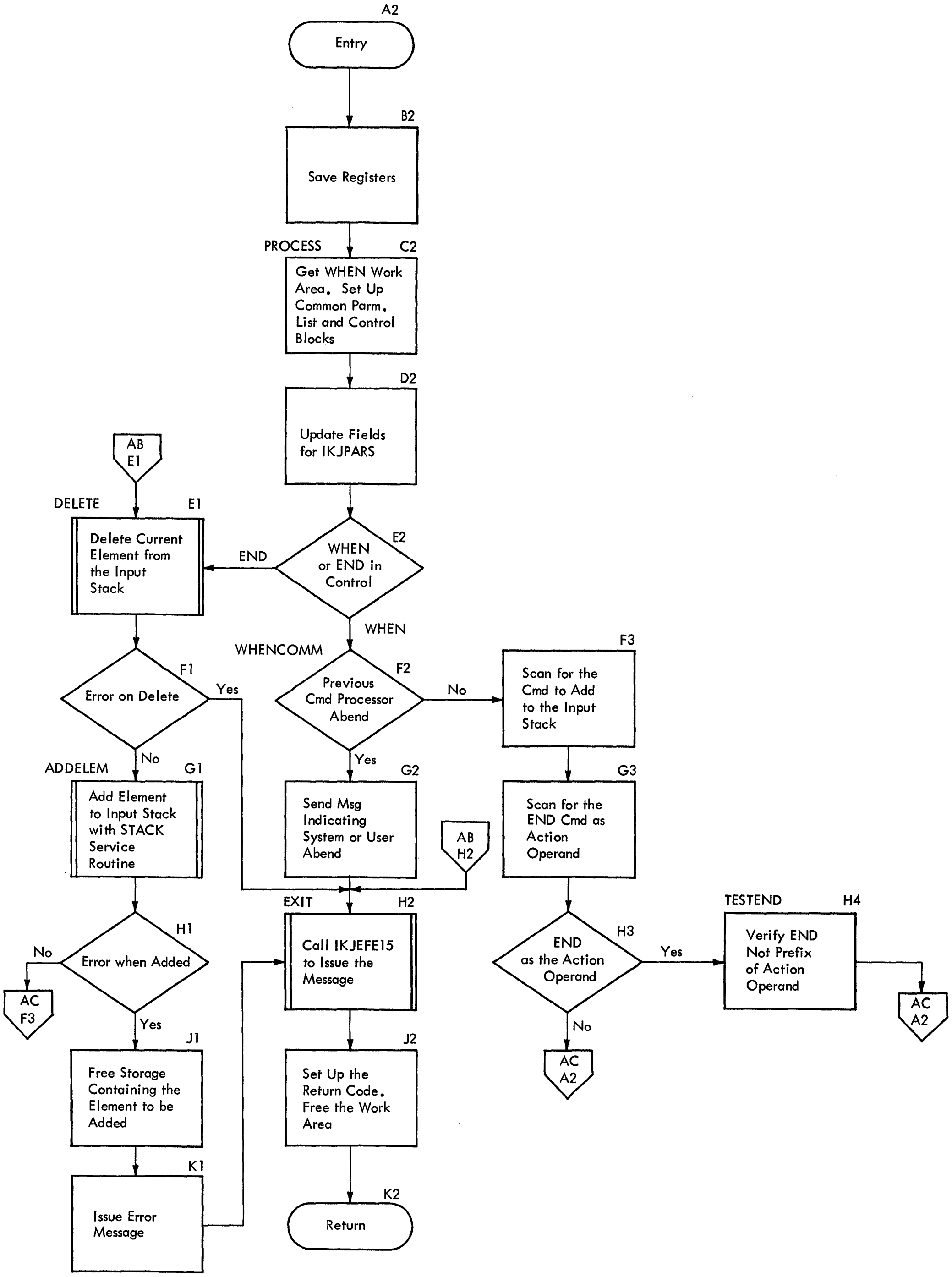

A2
Entry
B2
Save Registers
PROCESS
C2
Get WHEN Work Area. Set Up Common Parm. List and Control Blocks
D2
Update Fields for IKJPARS
AB E1
DELETE
E1
Delete Current Element from the Input Stack
END
E2
WHEN or END in Control
WHEN
WHENCOMM
F1
Error on Delete
Yes
F2
Previous Cmd Processor Abend
No
F3
Scan for the Cmd to Add to the Input Stack
No
ADDELEM
G1
Add Element to Input Stack with STACK Service Routine
Yes
G2
Send Msg Indicating System or User Abend
AB H2
G3
Scan for the END Cmd as Action Operand
EXIT
H2
Call IKJEFE15 to Issue the Message
H1
No
Error when Added
AC F3
Yes
H3
END as the Action Operand
Yes
TESTEND
H4
Verify END Not Prefix of Action Operand
AC A2
No
AC A2
J1
Free Storage Containing the Element to be Added
J2
Set Up the Return Code. Free the Work Area
K1
Issue Error Message
K2
Return

Chart AC. IKJEFE11. WHEN/END Command Processing (2 of 2)

Cross Reference: Diagram 2-1.

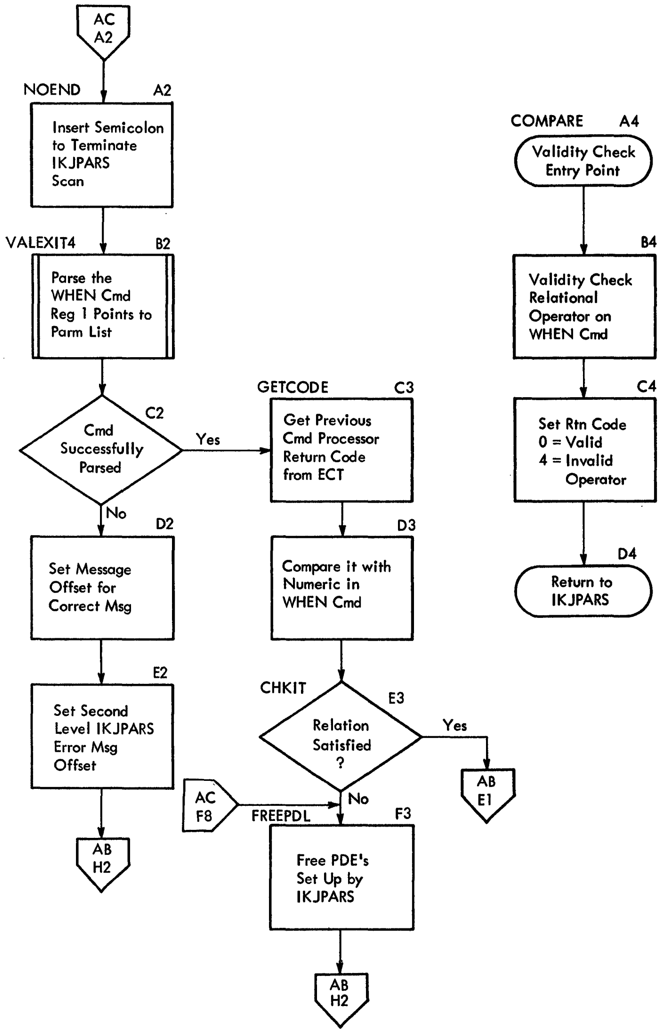
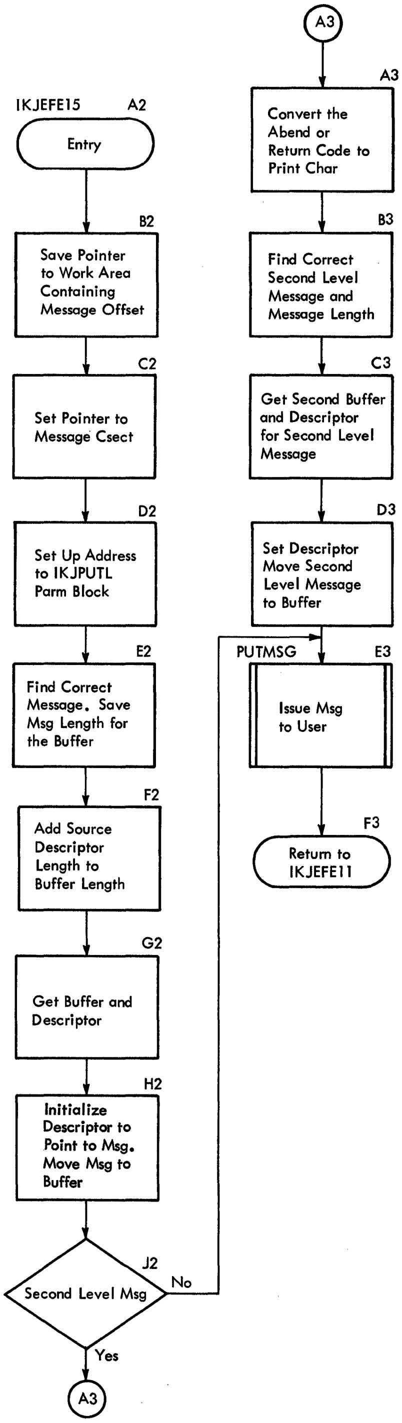

AC A2
NOEND A2
Insert Semicolon to Terminate IKJPARS Scan
VALEXIT4 B2
Parse the WHEN Cmd Reg 1 Points to Parm List
C2
Cmd Successfully Parsed
Yes
No
D2
Set Message Offset for Correct Msg
E2
Set Second Level IKJPARS Error Msg Offset
AB H2
GETCODE C3
Get Previous Cmd Processor Return Code from ECT
D3
Compare it with Numeric in WHEN Cmd
CHKIT E3
Relation Satisfied ?
Yes
AB E1
No
AC F8
FREEPDL F3
Free PDE's Set Up by IKJPARS
AB H2
COMPARE A4
Validity Check Entry Point
B4
Validity Check Relational Operator on WHEN Cmd
C4
Set Rtn Code 0 = Valid 4 = Invalid Operator
D4
Return to IKJPARS

Chart BA. IKJEFE15. Message Module

Cross Reference: Diagram 2-2.



IKJEFE15
A2
Entry
B2
Save Pointer to Work Area Containing Message Offset
C2
Set Pointer to Message Csect
D2
Set Up Address to IKJPUTL Parm Block
E2
Find Correct Message. Save Msg Length for the Buffer
F2
Add Source Descriptor Length to Buffer Length
G2
Get Buffer and Descriptor
H2
Initialize Descriptor to Point to Msg. Move Msg to Buffer
J2
Second Level Msg
No
Yes
A3
A3
A3
Convert the Abend or Return Code to Print Char
B3
Find Correct Second Level Message and Message Length
C3
Get Second Buffer and Descriptor for Second Level Message
D3
Set Descriptor Move Second Level Message to Buffer
PUTMSG
E3
Issue Msg to User
F3
Return to IKJEFE11

Section 4. Directory

This section contains a routine directory and a data area directory, both arranged alphabetically. Cross references for flowcharts and method of operation diagrams are provided for convenience, when applicable.

Routine Directory

Entry point, Routine name	Load Module	Control Section	Cross reference	
			MO	FC
IKJEFE11	IKJEFE11	IKJEFE11	2-1,2-2	AA
IKJEFE15	IKJEFE11	IKJEFE15	2-1,2-2	AB

Data Area Directory

Data Area	MO	FC
Command Buffer	2-1	AA
CPPL	2-1	AA
ECT	2-1	N/A
IKJWHEN	N/A	AA,AB,AC,BA
IOPL	N/A	AA,AB,BA
LSD	2-1	AA,AB
PDL	N/A	AA
PPL	2-1	AA
STPL	2-1,2-2	AA,AB

Section 5. Data Areas

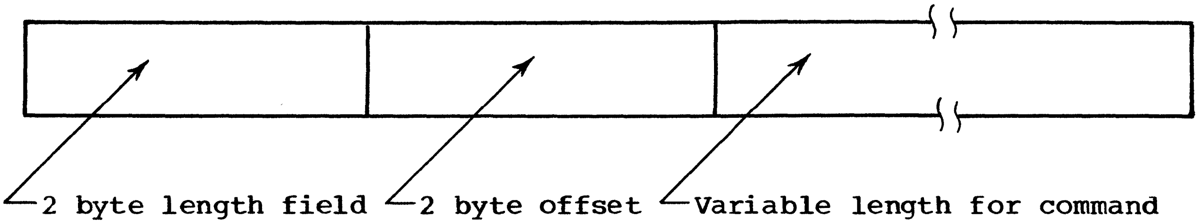
This section presents the major data areas in the WHEN/END Command Processor. Pertinent information, such as which routine created the data area, which routine uses or updates the area, and what the area contains, in included.

Command Buffer

Constructed by: TMP.

Used by: IKJEFE11.

Contents: Command buffer pointed to by CPPLCBUF in the CPPL.



Cross reference	FC	MO
	AA	2-1

CPPL—Command Processor Parameter List

Constructed by: TMP.

Used by: IKJEBLI1.

Location: The address of this list is contained in register one.

Disp Dec (Hex)	Field	Size (Bytes)	Use/Contents
0 (0)	CPPLBUF	4	Points to Command Buffer.
4 (4)	CPPLUPT	4	Points to UPT.
8 (8)	CPPLPSCB	4	Points to PSCB.
12 (C)	CPPLECT	4	Points to ECT.

Cross reference	FC	MO
	AA	2-1

ECT—Environment Control Table

Constructed by: The Environment Control Table (ECT) is built by the TMP and stored in a nonshared subpool.

Used by: Its fields can be modified by a command processor, or by a service routine, but cannot be freed.

Disp Dec (Hex)	Field	Subfield	Size (Bytes)	Use/Contents
0 (0)	ECTRCDF		1	High order bit on indicates a CP abend.
1 (1)	ECTRTCD		3	If the high order bit is on in ECTRCDF, the high order 12 bits hold the system abend code; the low order 12 bits hold the user abend code. Return code from previous CP.

ECT -- Environment Control Table (Cont.)

Disp Dec (Hex)	Field	Subfield	Size (Bytes)	Use/Contents
4 (4)	ECTIOWA		4	Address of I/O work area.
8 (8)	ECTMSGF		1	High order bit on indicates delete second level message.
9 (9)	ECTSMSG		3	Address of second level message chain.
12 (C)	ECTPCMD		8	Primary command name.
20 (14)	ECTSCMD		8	Secondary command name.
28 (1C)	ECTSWS		1	Switches.
		ECTNOPD		0 -- ON=No operands exist in command buffer.
		*		1 -- Reserved.
		ECTATRM		3 -- ON=LOGON/OFF requested TMP to LOGOFF user.
		ECTNMAL		4 -- ON=No broadcast notices to be received at LOGON.
		ECTNNOT		5 -- ON=No broadcast notices to be received at LOGON.
		*		6 -- Reserved.
		*		7 -- Reserved.
29 (1D)	*		3	Reserved.

Cross reference	FC	MO
	N/A	2-1

IKJWHEN—General Work Area (80 bytes)

Constructed by: IKJEFE11.

Updated by: IKJEFE11.

Used by: IKJEFE11.

Contents: A general work area.

Disp Dec (Hex)	Field	Subfield	Size (Bytes)	Use/Contents
0 (0)	WHPL		4	Points to the STACK parameter list.
4 (4)	Unnamed		4	Points to the PARSE parameter list.
8 (8)	Unnamed		28	General parameter list.
36 (24)	Unnamed		4	Points to STACK parameter block.
40 (28)	WHPBLOCK		20	General parameter block used for STACK, PARSE.
60 (3C)	WHPARAMS		4	Points to PARSE descriptor list.
64 (40)	WHATTECB		4	Service routine or ATTN routine ECB.
68 (44)	WHMSG1		1	Offset for message module.
69 (45)	WHMSG2		1	Secondary message index.
70 (46)	WHSWI		1	Status byte.
		WHEND	1 bit	Indicates END command in control.
			7 bits	Not used.
71 (47)	WHCHAR		1	First character of next command if delimiter was omitted.
72 (48)	WHENWAS		4	Not used.

IKJWHEN -- General Work Area (80 bytes) (Cont.)

Disp Dec (Hex)	Field	Subfield	Size (Bytes)	Use/Contents
76 (4C)	WHRCODE		4	Points to service routine return code. Indicates valid operation.
84 (50)	WHCOMM		4	Points to command to be added to input stack.
84 (54)	WHCMD		8	Name of command for message module.
88 (58)	WHGETM		4	GETMAIN size, subpool number for PUTL, STACK.
		WHSUBP	1	Subpool.
		WHFILL	1	Filler.
		WHLEN	1	Length.
96 (60)	WHWASIZ		4	Points to work area subpool and size.

Cross reference	FC	MO
	AA	N/A
	AB	
	AC	
	BA	

IOPL—Input/Output Parameter List

Constructed by: IKJEFE11.

Used by: IKJPUTL.

Contents: The address of this list is passed to IKJPUTL in register one.

Disp Dec (Hex)	Field	Size (Bytes)	Use/Contetns
0 (0)	IOPLUPT	4	Pointer to the UPT.
4 (4)	IOPLECT	4	Pointer to the ECT.
8 (8)	IOPLECB	4	Pointer to the ECB.
12 (C)	IOPLIOPB	4	Pointer to the I/O parameter block.

Cross reference	FC	MO
	AA	N/A
	AB	
	BA	

LSD—List Source Descriptor (16 bytes)

Constructed by: IKJEFE11.

Updated by: IKJEFE11.

Used by: IKJEFE11, STACK service routine.

Contents:

Disp Dec (Hex)	Field	Size (Bytes)	Use/Contents
0 (0)	LSDADATA	4	Points to record to be added to input stack.
4 (4)	LSDRCLEN	2	Record length, zero if variable length RECFM.
6 (6)	LSDTOTLN	2	Amount of storage to be freed.
8 (8)	LSDANEXT	4	Points to next record to be processed. Initialized to first recors by invoker. Updated by GETLINE/GETPUT.
12 (C)	LSDRSVRD	4	Reserved. LSD must be doubleword multiple.

Cross reference	FC	MO
	AA AB	2-1

PDL— Parameter Descriptor List

Constructed by: IKJPARS.

Used by: IKJEFE11.

Contents: IKJPARS puts the address of the PDL into the answer place pointed to by PPLANS in the PPL.

Disp Dec (Hex)	Field	Size (Bytes)	Use/Contents
0 (0)	PDECHAIN	8	Used to free the PDL.
8 (8)	PDERC	2	Offset to SYSRC.
10 (A)	Unnamed	2	Filler for macro expansion.
12 (C)	Unnamed	8	Filler for relational operation.
20 (14)	PDENUM	4	Integer to which last command processor's return code is compared.
24 (18)	PDENUML	2	Length of integer.
26 (1A)	PDENUMF	2	Flags for IKJPARS internal.

Cross reference	FC	MO
	AA	N/A

PPL—PARSE Parameter List

Constructed by: IKJEFE11.

Used by: IKJPARS.

Location: The address of this list is passed to IKJPARS in register one.

Disp Dec (Hex)	Field	Size (Bytes)	Use/Contents
0 (0)	PPLUPT	4	Points to UPT.
4 (4)	PPLECT	4	Points to ECT.
8 (8)	PPLECB	4	Points to ECB.
12 (C)	PPLPCL	4	Points to PCL.
16 (10)	PPLANS	4	Points to answer place.
20 (14)	PPLCBUF	4	Points to command buffer.
24 (18)	PPLUWA	4	Points to work area.

Cross reference	FC	MO
	AA	2-1

STPL—Stack Parameter List (16 bytes)

Constructed by: IKJEFE11.

Updated by: IKJEFE11.

Used by: IKJEFE11, STACK service routine.

Contents: List of addresses.

Disp Dec (Hex)	Field	Size	Use/Contents
0 (0)	STPLUPT	4	Points to UPT.
4 (4)	STPLECT	4	Points to ECT.
8 (8)	STPLECB	4	Points to command processor's ECB.
12 (C)	STPLSTPB	4	Points to STACK parameter block.

Cross reference	FC	MO
	AA	2-1
	AB	2-2

Section 6. Diagnostic Aids

Messages

This section contains the messages from the WHEN/END Command Processor.

Messages from the WHEN/END Command Processor Contained in IKEFE16.

Message ID	Message Text
IKJ56535I	* SYSTEM ERROR+
#IKJ56535I	PARSE ERROR CODE **
#IKJ56535I	STACK ERROR CODE **
IKJ56537I	* COMMAND NOT EXECUTED, NO RETURN CODE+
#IKJ56537I	PREVIOUS COMMAND ABENDED WITH USER ABEND CODE **
#IKJ56537I	PREVIOUS COMMAND ABENDED WITH SYSTEM ABEND CODE **

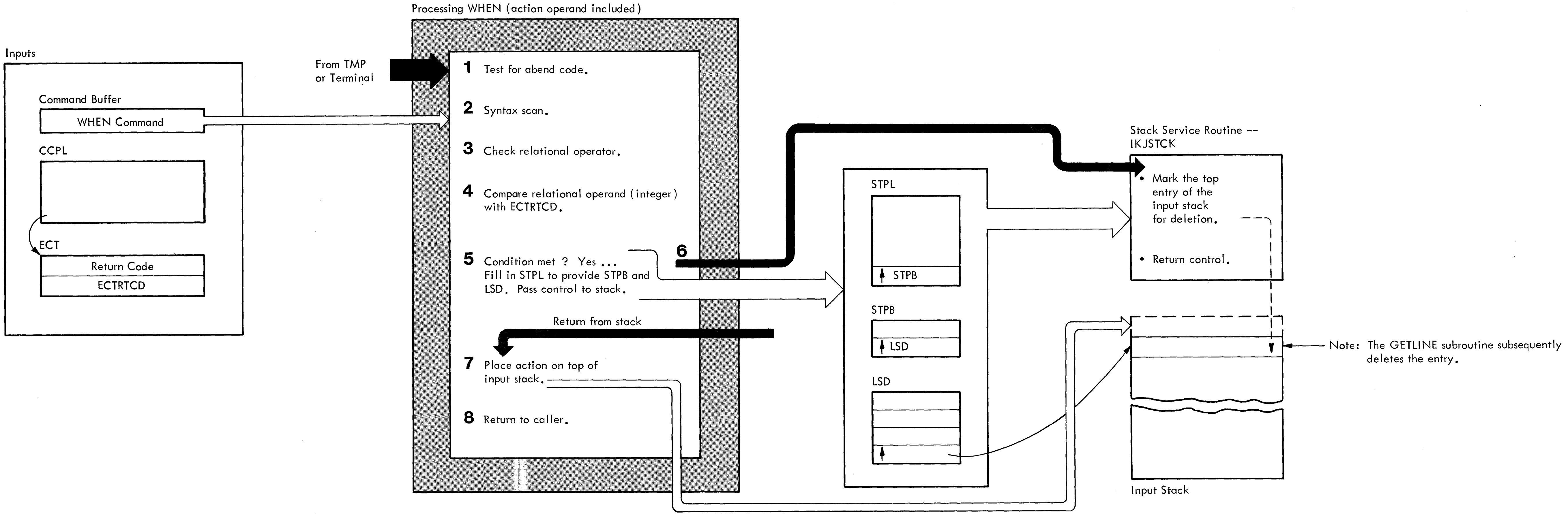
Note: indicates a second level message for documentation purposes only.

indicates the command name is inserted at the beginning of every first level message.

** indicates the return code is inserted at the end of every second level message.

Index

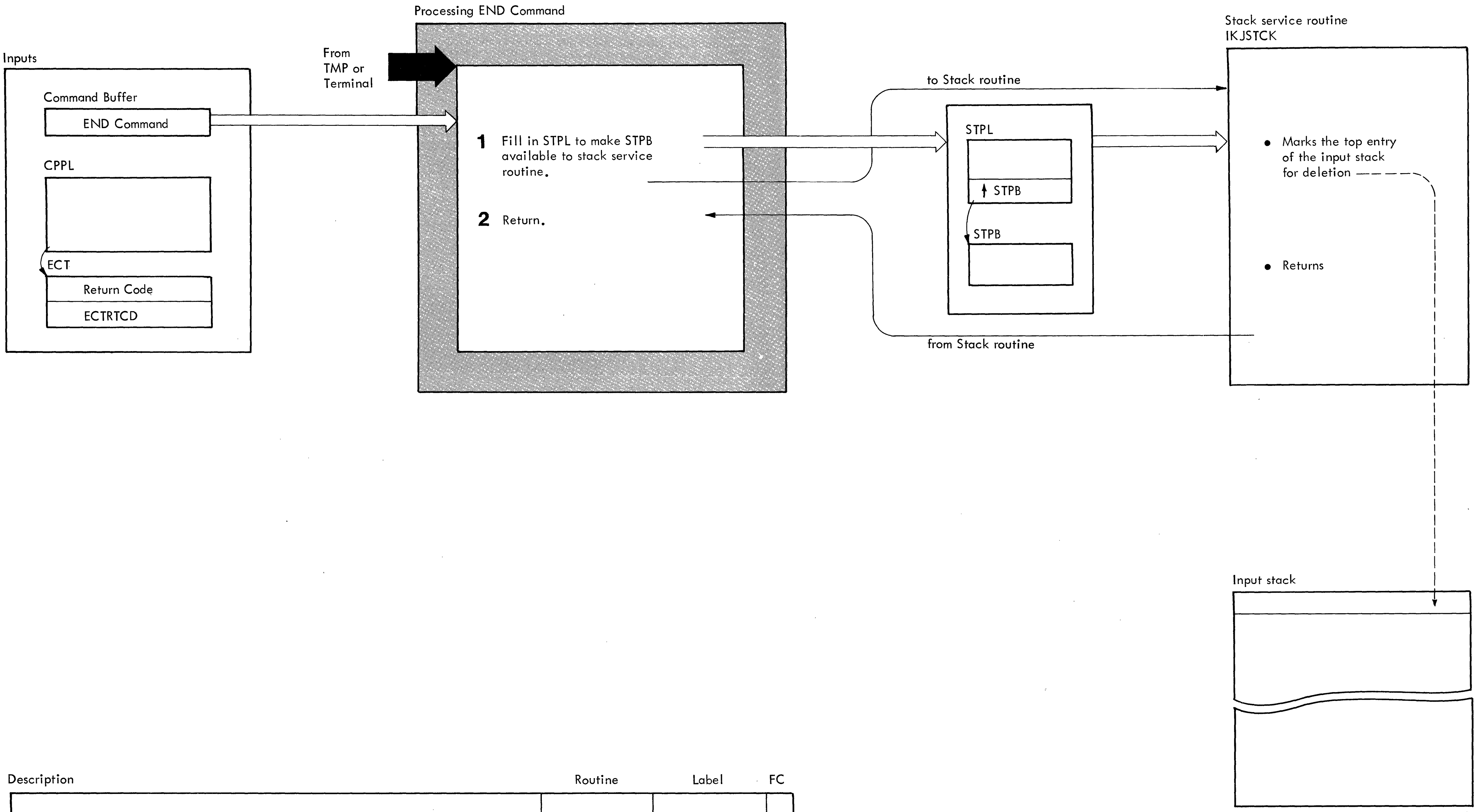
Indexes to program logic manuals are consolidated in the publication IBM System/360 Operating System: Program Logic Manual Master Index, Order No. GY28-6717. For additional information about any subject listed below, refer to other publications listed for the same subject in the Master Index.


Inputs
Command Buffer
WHEN Command
CCPL
ECT
Return Code
ECTRTCD
From TMP or Terminal
Processing WHEN (action operand included)
1 Test for abend code.
2 Syntax scan.
3 Check relational operator.
4 Compare relational operand (integer) with ECTRTCD.
5 Condition met ? Yes ... Fill in STPL to provide STPB and LSD. Pass control to stack.
6
Return from stack
7 Place action on top of input stack.
8 Return to caller.
STPL
STPB
STPB
LSD
LSD
Stack Service Routine -- IKJSTCK
• Mark the top entry of the input stack for deletion.
• Return control.
Input Stack
Note: The GETLINE subroutine subsequently deletes the entry.

Description	Routine	Label	FC
1 The PARSE Parameter List is initialized. Test for previous Command Processor abend. If previous Command Processor abend, issue message and return. If not, continue processing.	IKJEFE11		AB
2 Control then goes to the PARSE subroutine, which checks the syntax of the WHEN command.	IKJPARS IKJEFE11		AC AC
3 When the PARSE routine returns control to IKJEFE11, it provides a zero return code to indicate successful operation. (Non-zero causes a diagnostic message to be generated, and control is returned to the caller.)			
4 A validity check of the WHEN relational operator is made. Compares relational operand.			
5 If the condition is not met, control is returned to the caller. (The return code in the ECT is placed in register 15.) If the condition is met, the Stack Parameter List (STPL) is updated with the address of the Stack Parameter Block (STPB).			
6 The STPL is used by the Stack Service Routine to mark the top entry of the input stack for deletion.	IKJSTCK		AB

Description	Routine	Label	FC
7 If a return code of zero (successful operation) is passed from the stack routine, the WHEN action operand (if one is coded) is placed on top of the input stack. (If a non-zero return code is provided, the WHEN processor is terminated.) The Parse Parameter Descriptor List (PDL) is freed. (This list is set up by the parse subroutine. IKJEFE11 locates the PDL through the ANSWER place field of the PCL.)	IKJEFE11		AC
8 Control returns to the caller. (If WHEN was entered from a terminal, control is returned there upon termination of the command or command procedure specified in the action operand.)	IKJEFE15 IKJEFE11		BA AC BA

Diagram 2-1. WHEN/END Command Processing -- WHEN Command



Description	Routine	Label	FC
1 The general parameter list is initialized for the STACK service routine (to mark the top entry of the input stack for deletion).	IKJEFE11	IKJEFC11	AB
2 Control returns to the caller.	IKJEFE15		BA
	IKJEFE11		AB

Diagram 2-2. WHEN/END Command Processing -- END Command

READER'S COMMENT FORM

IBM System/360 Operating System:
TSO Command Processor PLM
Volume 6 — PROFILE, PROTECT, RENAME, RUN,
SEND, SUBMIT, TERMINAL, TIME, WHEN/END

Order No. GY28-6776-0

Please use this form to express your opinion of this publication. We are interested in your comments about its technical accuracy, organization, and completeness. All suggestions and comments become the property of IBM.

Please do not use this form to request technical information or additional copies of publications. All such requests should be directed to your IBM representative or to the IBM Branch Office serving your locality.

- Please indicate your occupation: ______________________________

- How did you use this publication?
  - □ Frequently for reference in my work.
  - □ As an introduction to the subject.
  - □ As a textbook in a course.
  - □ For specific information on one or two subjects.

- Comments (Please include page numbers and give examples.):

- Thank you for your comments. No postage necessary if mailed in the U.S.A.

YOUR COMMENTS, PLEASE . . .

This manual is part of a library that serves as a reference source for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Note: Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Cut Along Line

Fold Fold

FIRST CLASS
PERMIT NO. 81
POUGHKEEPSIE, N.Y.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY ...

IBM Corporation
P.O. Box 390
Poughkeepsie, N.Y. 12602

Attention: Programming Systems Publications
Department D58

Fold Fold

IBM®

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
[U.S.A. only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

IBM Technical Newsletter

File Number	S360-36 (OS Rel. 20.1)
Re: Order No.	GY28-6776-0
This Newsletter No.	GN28-2492
Date	June 1, 1971
Previous Newsletter Nos.	None

IBM System/360 Operating System:
Time Sharing Option
Command Processor Program Logic Manual
Volume 6
PROFILE, PROTECT, RENAME, RUN,
SEND, SUBMIT, TERMINAL, TIME, WHEN/END

This Technical Newsletter, a part of release 20.1 of IBM System/360 Operating System, provides replacement pages for the subject publication. These replacement pages remain in effect for subsequent releases unless specifically altered. Pages to be inserted and/or removed are:

INTRODUCTION	8.1
PROFILE	37-39 (remove 40-41)
PROTECT	39-42,45
RENAME	43-46 (remove 46.1)
RUN	33-36
SEND	45-47
TERMINAL	23-25
TIME	21-22 (remove 23)

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

Summary of Amendments

Revised indexes are provided for the PROFILE, PROTECT, RENAME, RUN, SEND, TIME, and TERMINAL command processor PLMs.

Diagram 2-1 (PROTECT Command Processing) on page 45 in the PROTECT command processor PLM is replaced.

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

IBM Corporation, Programming Systems Publications, P.O. Box 390, Poughkeepsie, N.Y. 12602

PRINTED IN U.S.A.

Release 20.1 (GY28-6776-0 modified by GN28-2492)

Item	Description	Areas Affected
Indexes	Revised indexes are provided for the PROFILE, PROTECT, RENAME, RUN, SEND, TIME, and TERMINAL command processor PLMs.	PROFILE pp. 37-39 PROTECT pp. 39-42 RENAME pp. 43-46 RUN pp. 33-36 SEND pp. 45-47 TERMINAL pp. 23-25 TIME pp. 21-22
PROTECT Diag.2-1	Diagram 2-1 is replaced.	PROTECT 45

Index

Indexes to program logic manuals are consolidated in the publication IBM System/360 Operating System: Program Logic Manual Master Index, Order No. GY28-6717. For additional information about any subject listed below, refer to other publications listed for the same subejct in the Master Index.

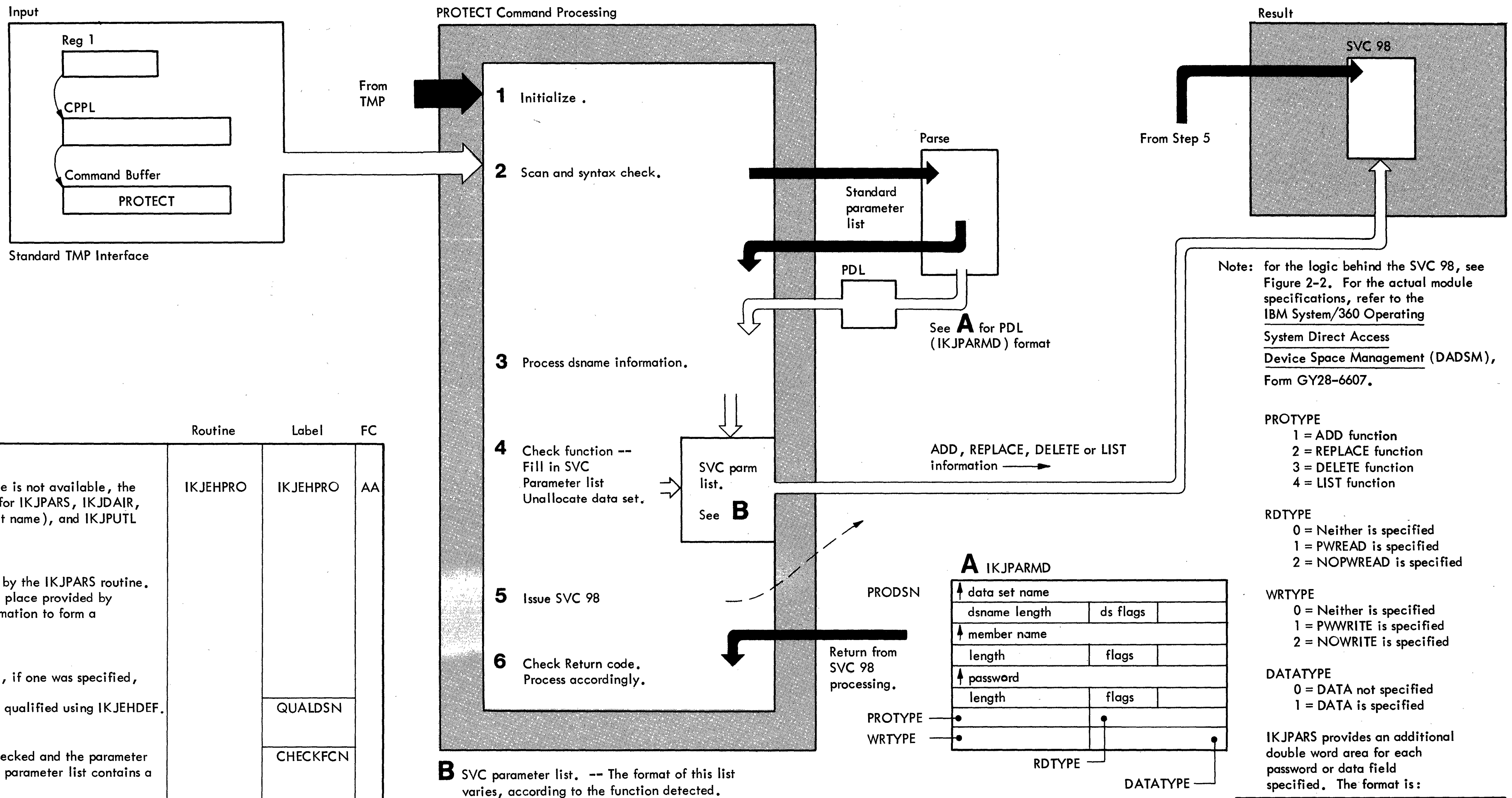
Note: Page numbers of the form "Intro/5" refer to pages in the "General Information" section of this publication.

(Pages 40-41 Deleted)

Index

Indexes to program logic manuals are consolidated in the publication IBM System/360 Operating System: Program Logic Manual Master Index, Order No. GY28-6717. For additional information about any subject listed below, refer to other publications listed for the same subject in the Master Index.

Note: Page numbers of the form "Intro/5" refer to pages in the "General Information" section of this publication.



Note: for the logic behind the SVC 98, see Figure 2-2. For the actual module specifications, refer to the IBM System/360 Operating System Direct Access Device Space Management (DADSM), Form GY28-6607.

PROTYPE
1 = ADD function
2 = REPLACE function
3 = DELETE function
4 = LIST function

RDTYPE
0 = Neither is specified
1 = PWREAD is specified
2 = NOPWREAD is specified

WRTYPE
0 = Neither is specified
1 = PWWRITE is specified
2 = NOWRITE is specified

DATATYPE
0 = DATA not specified
1 = DATA is specified

IKJPARS provides an additional double word area for each password or data field specified. The format is:

↑ password string	
string length	flags

Applicable names are:

ADDNEWPW
REPOLDPW
REPNEWPW
DELOLDPW
LSTOLDPW
STRDATA

Description	Routine	Label	FC
1 A GETMAIN is issued to request storage. If sufficient storage is not available, the program is abnormally terminated. Then general parameters for IKJPARS, IKJDAIR, IKJEHDEF (used if it is necessary to fully qualify the data set name), and IKJPUTL are initialized.	IKJEHPRO	IKJEHPRO	AA
2 Then the command is scanned and checked for correct syntax by the IKJPARS routine. IKJPARS places the address of the PDL (see A) in the answer place provided by IKJEHPRO. The PROTECT command processor uses this information to form a parameter list for the SVC 98 (PROTECT SVC) modules.			
3 After checking the IKJPARS return code the control password, if one was specified, is moved to the buffer. Then, if the data set name was not fully qualified, it is fully qualified using IKJEHDEF.		QUALDSN	
4 Next the function to be performed, found in PROTYPE, is checked and the parameter list for SVC 98 is filled in accordingly. The first byte of the parameter list contains a hexadecimal value indicating the function, as follows: X'01' ADD an entry to the password data set (PWDS). X'02' REPLACE an entry in the password data set. X'03' DELETE an entry from the PWDS. X'04' LIST protection, security counter, and optional data information of a protected data set. (The last 80 bytes of the PWDS entry for this data set password is placed in the 80 byte buffer pointed to by the SVC parameter list.)		CHECKFCN	
Unallocates the data set.		ISSUESVC	AA
5 Issue SVC 98 (see Diagram 2-2.)		SVC98	AB
	IGC0009H module		
6 Upon return from SVC 98, control is passed to the appropriate message processing and/or clean up locations, according to the return code provided by SVC 98.	IKJEHPRO	SVCRCTAB	

B SVC parameter list. -- The format of this list varies, according to the function detected.

X'01'

0	01	00 00 00
4	dslength	↑ dsname
8	Prot code	↑ New password
12	00	↑ Control password
16	String length	↑ String
20	Num of Vols	↑ Volume list
	1 byte	3 bytes

X'02'

0	02	00 00 00
4	dslength	↑ dsname
8	00	↑ Current password
12	Prot code	↑ New password
16	00	↑ Control password
20	String length	↑ String
24	Num of Vols	↑ Volume list

X'03'

0	03	00 00 00
4	dslength	↑ dsname
8	00	↑ Current password
12	00	↑ Control password
16	Num of Vols	↑ Volume list

X'04'

0	04	↑ Buffer (80 byte)
4	dslength	↑ dsname
8	00	↑ Current password

Diagram 2-1. PROTECT Command Processing

Index

Indexes to program logic manuals are consolidated in the publication <u>IBM System/360 Operating System: Program Logic Manual Master Index</u>, Order No. GY28-6717. For additional information about any subject listed below, refer to other publications listed for the same subject in the Master Index.

<u>Note</u>: Page numbers of the form "Intro/5" refer to pages in the "General Information" section of this publication.

(Page 46.1 Deleted)

Index

Indexes to program logic manuals are consolidated in the publication IBM System/360 Operating System: Program Logic Manual Master Index, Order No. GY28-6717. For additional information about any subject listed below, refer to other publications listed for the same subject in the Master Index.

Note: Page numbers of the form "Intro/5" refer to pages in the "General Information" section of this publication.

Index

Indexes to program logic manuals are consolidated in the publication IBM System/360 Operating System: Program Logic Manual Master Index, Order No. GY28-6717. For additional information about any subject listed below, refer to other publications listed for the same subject in the Master Index.

Note: Page numbers of the form "Intro/5" refer to pages in the "General Information" section of this publication.

Index

Indexes to program logic manuals are consoldiated in the publication <u>IBM System/360 Operating System: Program Logic Manual Master Index</u>, Order No. GY28-6717. For additional information about any subject listed below, refer to other publications listed for the same subject in the Master Index.

<u>Note</u>: Page numbers of the form "Intro/5" refer to pages in the "General Information" section of this publication.

Index

Indexes to program logic manuals are consolidated in the publication IBM System/360 Operating System: Program Logic Manual Master Index, Order No. GY28-6717. For additional information about any subject listed below, refer to other publications listed for the same subject in the Master Index.

Note: Page numbers of the form "Intro/5" refer to pages in the "General Information" section of this publication.

(Page 23 Deleted)

System/360 OS TSO Command Processor PLM — Vol. 6 (S360-36) Printed in U.S.A. GY28-6776-0

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
[U.S.A. only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]