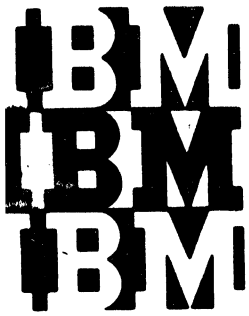


File No. S360-36
Order No. GY30-2011-1

Program Logic

IBM System/360 Operating System
Conversational Remote Job Entry
Program Logic Manual

Program Number 360S-RC-551

This publication describes the internal logic of the Conversational Remote Job Entry (CRJE) facility of the IBM System/360 Operating System (OS). CRJE provides Remote Job Entry capability for users at remote keyboard terminals that are connected to an IBM System/360 via communication lines.

This program logic manual is intended for use by personnel involved in program maintenance and by system programmers who are altering the system design.

PREFACE

This publication describes the structure of CRJE, its functions, the control flow among its routines, and the internal logic of its modules.

This manual has been organized and written to provide information for program maintenance and modification. The introduction provides background information necessary to understand the internals of CRJE: information such as the purpose of CRJE, its relationship to the operating system, its structure, system options, and a brief description of control blocks and data areas.

The method of operation section discusses the events or steps of the program that accomplish the functional objectives. Reference is made to the individual module that performs the event. The functional diagrams in this section show the flow of the program in achieving a particular function. These diagrams provide aid in getting to the proper module or control block.

The program organization section gives a detailed description of the events performed in each module. A flowchart of each module is also provided. These individual module descriptions and flowcharts provide a useful tool for the person using the book to perform maintenance or modification.

The data area layouts section contains information on all control blocks in CRJE. A map of each control block is provided and detailed information on the fields in each block. This section is arranged alphabetically.

All reference material is contained in the appendixes.

Readers should have a thorough knowledge of the IBM System/360 Operating System and should be familiar with the contents of the following publications:

IBM System/360 Operating System: Basic Telecommunications Access Method, GC30-2004.

IBM System/360 Operating System: Conversational Remote Job Entry, Concepts and Facilities, GC30-2012.

IBM System/360 Operating System: Conversational Remote Job Entry, Terminal User's Guide, GC30-2014.

IBM System/360 Operating System: Conversational Remote Job Entry, System Programmer's Guide, GC30-2016.

Readers may refer to the following publications for more detail in special areas:

IBM System/360 Operating System: FORTRAN IV Syntax Checker, Program Logic Manual, GY28-6831.

IBM System/360 Operating System: PL/I Syntax Checker, Program Logic Manual, GY33-8009.

Second Edition (March 1972)

This publication corresponds to Release 21.

This is a major revision of, and makes obsolete, GY30-2011-0 and Technical Newsletters GY30-2550 and GY30-2553. Changes to the text, and small changes to illustrations, are indicated by a vertical line to the left of the change.

The contents of this publication are subject to change from time to time. Changes will be reflected in periodically updated editions. Before using this publication, consult the latest System/360 SRL Newsletter, 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 is provided at the back of this publication for your comments. If the form is missing, comments may be addressed to IBM Corporation, Programming Publications, 1271 Avenue of the Americas, New York, New York 10020.

© Copyright International Business Machines Corporation 1970

Date of Publication: March, 1972
Form of Publication: Revision: GY30-2011-1

STAE Exit Support

New: Programming and Documentation

CRJE now supports a STAE exit for subtask abend conditions. The following changes have been made to reflect this support:

The descriptions of "Session Management" and "Closedown" (Method of Operation section) have been updated.

The following flowcharts have been updated:

IHKBGN	IHKSDQ
IHKCIP	IHKCMD
IHKCLN	IHKEOS
IHKSRV	IHKPUT
IHKLDC	IHKRER
IHKCCI	IHKSTS
IHKCC5	IHKSUB

Appendix E includes additional messages for STAE exit support.

80-character User Record Support

New: Programming and Documentation

CRJE now supports 80-character user library records. The following items have been updated to reflect this support:

- "Input/Output Flow" description (Method of Operation section)
- Flowcharts for modules IHKAWS, IHKSAV, and IHKBPM
- TUB (Terminal User Block)

Appendix B includes additional messages for 80-character user record support.

LOGOFF Exit

Maintenance: Programming and Documentation

The following descriptions have been updated to include handling of incomplete logon processing by the LOGOFF command processor (IHKLGF).

- The "LOGOFF EXIT" description (Method Operation Section)
- IHKCMD, IHKLGF, and IHKLGN module descriptions

JOBFAIL Name Description

Maintenance: Programming and Documentation

The name JOBFAIL cannot be used as a name for a remotely submitted job. This restriction has been noted in the "Notification of Job Output" description (Method of Operation section), and in the IHKDEQ module description.

Data Areas Updated

Maintenance: Programming and Documentation

The following data areas have been updated for technical accuracy:

AVT
DEF table
RJCT
TUB

Maintenance: Documentation Only

The PL/I fields of the DIR data area have been updated for technical accuracy:

Messages Updated

Maintenance: Programming and Documentation

Appendixes B and E now include updated messages.

New: Documentation Only

Appendix G, which contains the origin of central operator messages, has been added.

Miscellaneous

Maintenance: Documentation Only

The format of an entry in the system message file has been updated to show differences between entries for central installation messages and terminal user messages.

The IHKLST module description has been updated to correct the NONUM operand for the LIST subcommand.

Maintenance: Programming and Documentation

The "Command Exit" description (Method of Operation section) has been updated to point out additional information in the user command control table.

Method of Operation charts A, E, and H have been updated for technical accuracy.

Editorial changes that have no technical significance are not noted here.

Specific changes to the text made as of this publishing date are indicated by a vertical bar to the left of the text. These bars will be deleted at any subsequent republication of the page affected.

L 和 2

L 4

FIGURES

CHARTS

The Conversational Remote Job Entry (CRJE) facility of the IBM System/360 Operating System (OS) operates in a multiprogramming environment. It can run under MFT with at least 256K bytes of main storage or under MVT (or MP65) with at least 384K bytes of main storage. Direct-access storage space for CRJE tables, system libraries, the active area, and work areas is provided on a 2311 Disk Storage Drive, a 2314 Direct-Access Storage facility, or a 2319 Direct-Access Storage facility. Requirements for direct-access storage space depend upon the number of communication lines, the average size of the data sets being updated, the number of active users allowed at any one time, the number of delayed and/or broadcast messages allowed at one time, and the number of CRJE jobs submitted to OS for processing at any one time. User libraries require additional direct-access storage space.

PURPOSE OF CRJE

Conversational Remote Job Entry allows users at remote keyboard terminals to prepare and enter jobs to OS for processing. Data sets can be created and maintained through the facilities of CRJE. The data sets can contain program source statements, data, CRJE commands, data set names, or job control statements.

To submit data sets to OS for processing, the user simply lists the data sets in the order they are to be submitted. The user may request that the data sets be listed at his terminal. The output of jobs submitted can be printed at the user's terminal or directed to a central installation output writer.

The user invokes the functions of CRJE by CRJE commands. These commands are grouped by the functions they represent.

Session Management: The user's session is initiated by a LOGON command and terminated by a LOGOFF command.

Data Set Manipulation: Data sets are created and maintained by the EDIT command and the EDIT subcommands. A data set is created a line at a time by the EDIT NEW command, the Implicit subcommand, or the INPUT subcommand. Lines of a data set are inserted or replaced by the INPUT or Implicit subcommand and deleted by the DELETE or Implicit subcommand. Character strings within lines are replaced by the CHANGE subcommand. Two data sets are merged into one by the MERGE subcommand. The lines of a data set are renumbered by the RENUMBER subcommand. If a data set contains PL/1 or FORTRAN program source statements, the statements are scanned for syntax errors by the SCAN subcommand. The data set that has been updated is then saved in a user library by the SAVE subcommand.

Job Submission and Retrieval: Jobs are given to OS for processing when a SUBMIT command is entered at the terminal. If a job contains more than one data set, the data sets are listed in the order they are to be submitted. Jobs are canceled by the CANCEL command. Output of jobs is requested by the OUTPUT command. Interrupted output is obtained by the CONTINUE command.

Retrieval of Status Information: The user enters a LISTLIB or LISTDS command to obtain information about a data set in his user library. The information available includes the attributes and size of the data sets, the date last used, and whether the data sets have protection keys. The STATUS command is used to obtain information about jobs the user has submitted for processing.

Communication: The user can communicate with other terminal users and with the central operator by the SEND command.

A data set may contain a list of terminal commands. If an EXEC command is entered for this data set, the commands are executed as if they were being entered from the terminal.

The central operator controls the CRJE system. He issues the START command to initiate the system and the STOP command to terminate the system. He obtains information about users, jobs, or messages by the SHOW command. He sends messages to terminal users (MSG command), and he maintains the broadcast messages (BRDCST command). He removes job output in the CRJE SYSOUT class and processes it with a central installation output writer by the CENOUT command. He adds and deletes users by the USERID command.

RELATIONSHIP TO OPERATING SYSTEM

CRJE operates as a problem program under MVT or MFT with four separate tasks. The main CRJE task is attached by the OS master scheduler when a START command referring to a CRJE procedure is entered by the central operator. Control is given to the main CRJE task at the IHKBGN entry point of the START command processor and the task is given a protection key of zero. The open task is attached by the IHKEOS module. The other two tasks are service tasks of the main CRJE task and are attached by the CRJE start-up module IHKCIP.

CRJE uses the OS facilities for data management, task management, and job management. The following OS routines are used in job management: IEFQMDQ2, IEFQDELE, IEFQMSSS, IEFQMUNC, and IEFLOCDQ. OS data sets on direct-access storage devices are handled with the Basic Sequential and Basic Partitioned Access Methods (BSAM and BPAM). Task management is handled by the OS dispatcher. To allocate space for the job stream data set, the allocate module (IHKALC) uses the OS routine IGC0003B. The utility task issues a START RDR (SVC 34) on the job stream data set when submitting jobs to OS. The interface for starting the OS reader/interpreter on the job stream is handled by the utility task (IHKSRV).

Data is transmitted and received over the communication lines by the Basic Telecommunications Access Method (BTAM), which provides error recovery for the communication lines supported. BTAM issues I/O error messages to the central operator for irrecoverable line errors, and BTAM also issues messages giving line error statistics. The BTAM on-line terminal test facility is provided unless omitted by coding ONLNT=NO in the CRJELINE macro at CRJE assembly time.

CRJE and RJE may exist in the same system at the same time, each providing a separate function. The only routine they have in common is the IGC1503D routine for scheduling central commands.

When a central command is entered, the IGC1503D module gets control from SVC 34. The scheduling routine determines whether the command is for RJE or CRJE. The OS routine IGC0503D is used to write error messages at the central console.

SYSTEM STRUCTURE

CRJE is divided into four separate tasks: loader/controller task, utility task, OS data set open task, and main CRJE task (see Figure 1). The main CRJE task is attached by the OS master scheduler when CRJE is started. Control is first given to the START command processor (IHKBGN). The CRJE start-up module IHKCIP attaches the two service tasks (loader/controller and utility). The open module (IHKOPN) is loaded when an OS data set is to be accessed. The module is attached as a task by the EDIT command processor (IHKEOS). The ATTACH macro instruction specifies that the task can be neither rolled out nor cause another task to be rolled out. Each ATTACH macro specifies an event control block (ECB) to be posted by the OS control program when the attached task has terminated. This is done to ensure that each attached task is complete before CRJE detaches the task at CRJE closedown and returns to the OS control program.

All three of the attached tasks perform service functions for the main CRJE task. They are removed from the main CRJE task because they contain OS functions that issue WAIT macros in OS rather than in the CRJE dispatcher. Having these tasks external to the main CRJE task allows CRJE processing to take place while the OS WAIT macro is in effect for the attached task.

The tasks communicate by a series of ECBs. Each of the attached tasks has a list of ECBs that determine when there is work for that task to do. There is one ECB entry in the list for each CRJE subtask that uses that task. The post codes in the ECBs are addresses of parameter lists. Return is made by posting a return ECB that was passed as a parameter. The open task has one ECB that it posts when the task returns to the control program.

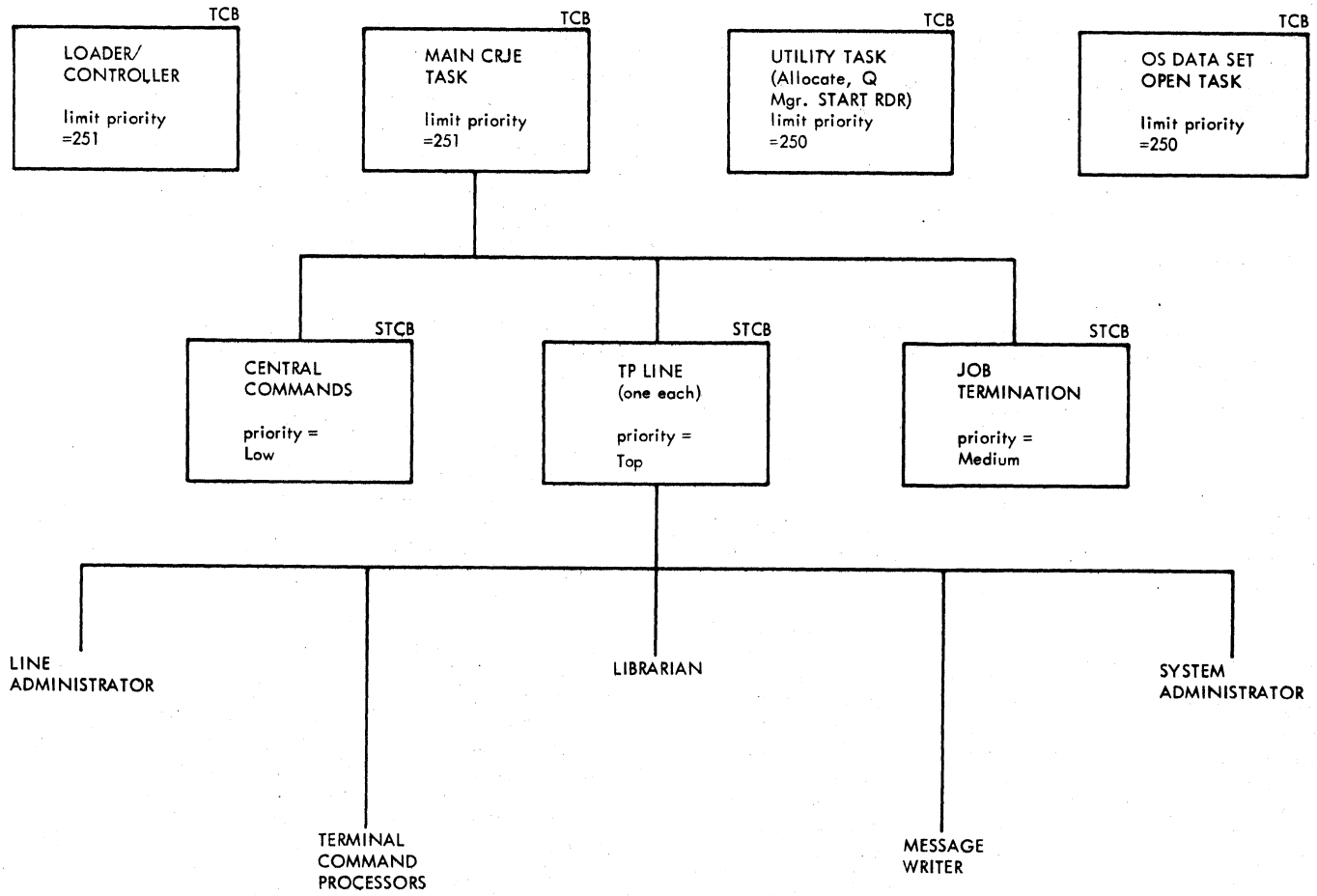


Figure 1. CRJE System Structure

LOADER/CONTROLLER TASK

The function of the loader controller task is to load and delete nonresident routines. The nonresident routines are loaded into the CRJE transient area.

The minimum size of the transient area is 8K bytes. It can be increased at start-up time in multiples of 2K bytes. This is done by specifying the amount to be increased in the PARM field of the EXEC statement for the CRJE procedure. All modules are loaded or deleted in blocks of 2K bytes.

The loader/controller task has a limit priority of 251, the same as the main CRJE task.

UTILITY TASK

The utility task has three functions: allocation of direct access space for the job stream of CRJE jobs; interfacing with the OS queue manager modules; and interfacing to start an OS reader on the submitted job stream. The OS reader/interpreter interface is handled as a separate task since CRJE allows only one reader/interpreter to operate in the system at a given time.

The reader/interpreter is invoked by issuing the START command internally. It is recommended that the central operator not issue a START RDRCRJE.S command while CRJE is executing; this will ensure that the reader/interpreter will be available to CRJE when needed. Only one system-assigned reader is allowed at any given time when operating in an MFT environment.

The utility task operates with a limit priority of 250, which is one less than the main CRJE task.

OPEN TASK

The open task handles the opening and scratching of OS data sets and members of

partitioned data sets, except user libraries. The open t sk is attached and detached by the IHKEOS module. The open task has a limit priority of 250, which is one less than the main CRJE task.

MAIN CRJE TASK

A subtask in CRJE is a sequence of instructions receiving control only from the CRJE dispatcher and yielding control only to the CRJE dispatcher. The main CRJE task is composed of a variable number of subtasks: the central command subtask, a subtask for each communication line, and the job termination subtask. The CRJE dispatcher controls the subtasks by means of an ECB list that contains an entry for each subtask.

Associated with each ECB list entry is a subtask control block (STCB). The CRJE dispatcher scans the ECB list for an ECB that is posted. A posted ECB means that the event the subtask is waiting for has completed. The dispatcher restores the registers for the subtask and returns program control to that subtask. When the CRJE dispatcher gets control again, it continues to look for posted ECBs. If at any time the CRJE dispatcher cannot find a posted ECB, it will issue a WAIT macro to give control to the OS dispatcher.

The central command subtask processes the commands entered by the central operator.

The job termination subtask dequeues remote jobs as they finish and queues notification messages for the user who submitted the job. The notification message includes an indication of whether the job completed normally or abnormally.

The routines that perform the functions of the communication line subtasks are divided into the following components: line administrator, librarian, message writer, system administrator, and terminal command processors.

The line administrator handles all servicing of the communication lines. It provides translation of the characters being sent or received from EBCDIC to the transmission code or from the transmission code to EBCDIC. It provides editing of lines sent and received. Editing consists of removing line control characters and backspace characters, translating to upper case, if requested, and adding the proper number of idle characters and new line characters.

The librarian provides access to the CRJE user libraries and the active area by means of macros. (See Appendix F for a list of these macros.) The user libraries are pre-allocated, partitioned data sets that contain the terminal users' permanent data sets. The active area is a pre-allocated data set that contains active files (one for each active user) and global files. Pre-allocation means that the data sets are allocated before the start-up of CRJE. The active files are assigned to active users for storage of temporary data sets. Each global file contains a member of the CRJE system library.

The message writer queues messages for or sends messages to the terminal user and the central operator. If a prepared message is not used, the message writer builds a message. If indicated on LOGON command, the message writer appends the standard message code to the message.

The system administrator controls the flow of terminal commands through the system and handles any abnormal situations, such as line failures, that result in terminating a user's session. The system administrator provides the command analysis function, which validates all terminal command verbs, builds a PPT (Parameter Position Table) containing the command and its parameters, and requests the loading of the command processor if the processor is nonresident. The disposition of system messages is controlled by the system administrator before requesting a new command from the line administrator.

The terminal command processors provide the functions requested by the commands. The processors use the librarian, the message writer, and the line administrator to process the commands.

CONTROL BLOCKS

The communication of information within CRJE is by means of control blocks. The information in these blocks and tables is stored in a compact form that is easily accessible. They have a standardized format so that the information is available to all parts of the CRJE system. The addresses maintained in the control blocks permit access to other blocks and tables. By using addresses in the AVT, all control blocks can be found. This is illustrated in Figure 2.

The KONBOX is not a CRJE control block and is not treated as such. It contains constants, control fields, and work areas required by the library I/O and active file I/O modules.

ADDRESS VECTOR TABLE (AVT)

This table contains the addresses of the entry points of the resident modules, the addresses of the control blocks, and any other addresses that the command processors need. The address vector table, IHKAVT, resides in main storage.

CRJE CONTROL TABLE (CCT)

There is only one CCT for the entire CRJE system. It is generated by the CRJETABL macro at CRJE assembly time. The CCT resides in main storage in the IHKMAC module.

This table contains installation-wide switches and options plus line control information for all teleprocessing lines in the CRJE network.

The CCT is reinitialized at each start-up. Some fields may be altered by the central operator by issuing central commands.

CONVERSATIONAL LINE BLOCK (CLB)

A CLB exists for each line in the CRJE system. They are generated by the CRJELINE macro at CRJE assembly time. They reside in main storage in the IHKMAC module. The CLB contains information about the terminal on that line. The information is necessary for the line administrator to perform BTAM operations on the line. Certain fields in the CLB are used for communicating line status and termination of line operations to the rest of the system. The CLB contains a pointer to the DECB for the line.

COMMAND DEFAULT TABLE (DEF)

The DEF contains the default options for the operands of each terminal command. When the default option of a command is to be used, the command processor checks the DEF table to find the default.

PARAMETER POSITION TABLE (PPT)

The function of the PPT is to pass the terminal command and its operands from the system administrator to the command processor. The system administrator allocates the PPTs as they are needed. One PPT is allocated for each line of input received from the terminal. If more than one PPT is needed for one command and all its operands, then the PPTs are chained together. They are variable in length with a minimum of 32 bytes. A pointer to the PPT is contained in the terminal user block.

REMOTE JOB CONTROL TABLE (RJCT)

The RJCT resides in the system library on disk. A copy of it is in the global file of the active area under the name CRJE.SYSLIB(JBTBLS). Each command processor interfaces with the Active File I/O (AFIO) of the librarian to read, write, delete, and search entries in the RJCT. An RJCT entry is initialized for each job that is submitted to OS for processing. The entry is deleted when the job is canceled or the output received. The job name is used as the key to locate entries in the global file member.

SUBTASK CONTROL BLOCK (STCB)

One STCB exists for each CRJE subtask. The STCB contains a pointer to its entry in the ECB list for the CRJE dispatcher, a pointer to the register save area, a pointer to the next STCB in the circle, and a dummy ECB that is always posted at start-up time. The CRJE dispatcher uses this information to determine which subtask is to be given control next.

The STCBs for the communication lines are generated by the CRJELINE macros. All of the STCBs reside in the IHKMAC module.

TERMINAL USER BLOCK (TUB)

A TUB is allocated by the line administrator when a user initiates his LOGON procedure; there is a TUB for each active user. The TUB contains the address of the CLB for the line of the terminal at which the user is logged on. The TUBs are also chained backward and forward.

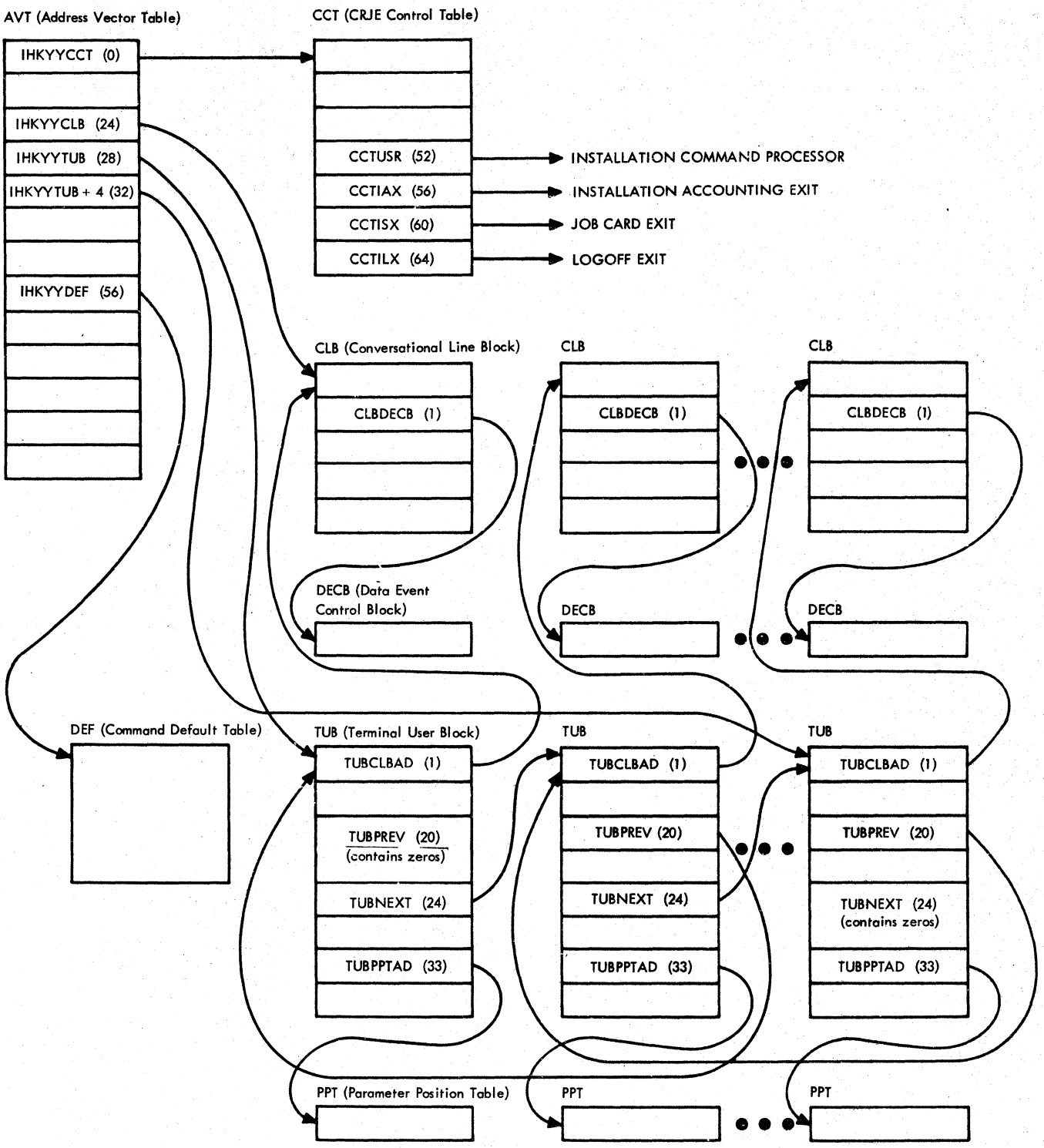


Figure 2. Control Block Pointers

The TUB is divided into 2 sections: terminal user section and the AFIO control section. The first section contains all status information concerning the terminal. Also in this section is the status information concerning the user's current session, including the attributes of the data set in his active file, the accounting information for his session, the options requested by the user, whether any messages exist for the user, and what type of commands are valid at this point in his session.

The AFIO control section contains information necessary for AFIO to operate within CRJE. This section also contains fields pertinent to library I/O.

NOTE: See the section, Data Area Layouts, for a map of the control blocks and a description of the fields.

SELECTED OPTIONS

INSTALLATION EXITS

Four exits in CRJE allow the installation to include routines of their own to meet special requirements. A routine can be added to process accounting information (LOGON exit); a routine can be added to obtain LOGOFF time (LOGOFF exit); and a routine can be added to check the JCL cards (JCL exit). An installation can also add commands and routines to process these commands.

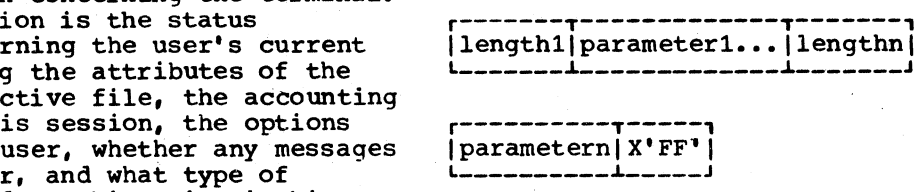
LOGON Exit: The LOGON exit gives the installation the opportunity to create a routine to process accounting information which is given as an operand on the LOGON command. The entry point to the routine must be specified in the ONEXIT operand of the CRJETABL macro.

If all the operands given on the LOGON command are correct, the LOGON processor (IHKLGN) checks the CCT to find an entry point address for the routine. If there is no routine, the entry point address is zero and the accounting information is ignored.

The following parameter list is provided for the LOGON exit and its address is passed in register one:

Word 1 - a pointer to the userid given on the LOGON command (this is a pointer to the TUBUSRID field in the TUB).

Word 2 - The address of the accounting information in the LOGON command, if present. This information has the following format:



The first byte gives the length of the first parameter and X'FF' indicates the end of the parameter list. These parameters are already stored in the PPT, so the installation exit is passed a pointer to this area.

If no accounting information exists, the address of a byte containing X'FF' is passed to the exit routine.

When the LOGON exit routine gives control back to the LOGON processor, the return code is checked. If 0, the user is allowed on the system and processing continues normally. If the return code is 4, the user is not allowed on the system. In this case a message is sent to the user notifying him of why he is not logged on, and control is given to the command analyzer to ask for another command.

LOGOFF Exit: The LOGOFF exit allows the installation to process information concerning session termination by using a routine of its own. The entry point must be specified in the OFFEXIT operand of the CRJETABL macro.

The LOGOFF processor (IHKLGF) checks to see if there is an entry point address in the IHKYYILX field of the AVT. If so, register one points to the following parameter list:

Word 1 - address of userid for the user whose session is being terminated. (This is the address of the TUBUSRID field in the TUB.)

Word 2 - address of first field in the CLB, which contains the type of terminal at which the user logged on.

Word 3 - address of field containing LOGON time, LOGOFF time, and elapsed time of session. This field also contains an indication of whether the session was terminated normally or abnormally.

This field is the same field used in giving the message writer the information for the LOGOFF message.

There are no return codes from the LOGOFF exit routine.

Note: If a processing error occurs during logon processing at the point (or any time thereafter) where CRJE gives control to the Logon exit, if one exists, CRJE gives control to the LOGOFF exit (if one exists) to get accounting information for the user currently logged on the system. If during logon processing where CRJE would have branched to the LOGON exit there is no exit, CRJE gives control directly to the LOGOFF exit (if it exists).

JCL Exit: This exit lets the installation provide a routine to examine and modify JCL statements of conversationally submitted jobs. The installation-provided routine can modify the JCL statements, but cannot add new 80-character records or delete records. Statements returned to CRJE from the exit routine are passed to OS after a check is made for duplicate jobname on the JOB card. The entry point for the exit routine must be provided in the JOBEXIT operand of the CRJETABL macro.

The SUBMIT command processor (IHKSUB) checks the CCT to get the entry point address for the JCL exit routine. If there is not an address, the JCL cards are sent to OS after a check is made for duplicate jobname on the JOB card. This routine is entered every time a JCL card is found. The address of the following parameter list is passed in register one:

Word 1 - address of JCL statement.

Word 2 - address of userid for user submitting the job.

A 0 return code from the JCL exit routine means that the job is to be passed to the operating system. A return code of 4 means that the job is not to be submitted. A return code of 8 means that the job is not to be submitted and the message supplied by the exit is to be sent to the user. Register 1 should contain the address of a 60-byte message to be sent to the user.

Command Exit: The installation is given the opportunity of adding terminal user commands and subcommands with routines to process them. The commands and subcommands must be specified in the USRMCMD and USRSCMD operands of the CRJETABL macro. The entry point of the installation-provided routine must be given in the CMDEXIT operand of the CRJETABL macro. Only one entry point is specified. This entry point is given control when any of the commands or subcommands are recognized by the command analyzer. The commands and subcommands that are added are put in the major command list and subcommand list. The entry point address of the routine is put in the AVT.

Register 1 points to a parameter list containing the address of the user command control table. This table is a section of the TUB. It contains the address of a 120-byte user buffer, the userid, and the data length. Upon entry to the command exit routine, the user buffer contains the command or subcommand with its operands after editing by the Line Administrator. Upon return from the command exit routine, the user buffer may contain a message (120 characters in length) to be sent to the terminal user.

A return code of 0 means that processing is complete. A 4 return code means that processing is complete and there is a message in the user buffer to be sent to the terminal.

PL/I AND FORTRAN SYNTAX CHECKERS

The installation has the option of including one or more versions of the PL/1 syntax checker and/or the FORTRAN syntax checker. The syntax checkers are included if the macro CHECKER is specified at the generation of the OS system supporting CRJE. The specific version of the syntax checker that is to be utilized is specified by a PARM field parameter of the EXEC statement in the cataloged procedure to start CRJE.

Source statements are scanned as lines are entered from the terminal or existing statements are scanned in response to a specific request. Specific requests are made by the SCAN subcommand. The automatic scan facility can be turned on by the EDIT command or SCAN subcommand. While the automatic scan is on, all lines entered in input mode are passed to the syntax checker. The automatic scan facility is turned off with the SCAN subcommand.

If SCAN is specified in the EDIT command, the data set must have the attribute of PL/1 or FORTRAN. If the character set and source margin are not specified, the default options are used.

SYSTEM AND USER LIBRARIES

CRJE ACTIVE AREA

The active area is allocated as a sequential data set on a direct-access device. Active data sets are files within the active area that are allocated to individual active users. Active files are basically sequential sets of unblocked records. Each record is 80 bytes long and has an associated 8-byte key, which is the line number. The keys and their associated records are logically arranged in increasing sequential order by key value.

Active files are allocated from within the active area one track at a time. Consequently, an active file may never have adjacent tracks assigned to it. The availability of a track for allocation to an active file is determined by examination of the track allocation table (TAT), which is constructed by the active area start-up module (IHKAST) at CRJE start-up time.

The active area consists of two types of files: global and private. Private files are active data sets that are assigned to users when they enter an EDIT command. Active data sets are referred to as active files. Active files are released when an END subcommand is entered by the user. The global files are allocated at start-up time and each member of the system library is copied as a global file.

Access to the files in the active area is provided with Active File I/O (AFIO) macros. The macros are listed in Appendix F and are discussed in detail in the Librarian section of Program Organization. These macros invoke the IHKAFI module, which is the AFIO control routine.

CRJE SYSTEM LIBRARY

The Conversational Remote Job Entry system library (CRJE.SYSLIB) is a partitioned data set that contains user, system, and broadcast messages, user verification information, and control tables for jobs that have been submitted to OS through CRJE.

The CRJE system library is a five-member partitioned data set containing the following files:

1. System Message File - CRJE.SYSLIB(SYSMSGS). This member consists of messages and diagnostics written to the terminal user or the central operator.

2. User Verification File - CRJE.SYSLIB(USERS). This file contains the userid and password of all potential active users and internal CRJE control information related to each user.

3. User Message File - CRJE.SYSLIB(USRMSGS). This file contains the delayed messages for users who were not active (logged on) when the messages were sent, or messages from a subtask other than the user's, or messages sent after one message had already been queued in storage for the user.

4. Broadcast Message File - CRJE.SYSLIB(BRDCST) . This file contains messages from the central operator that are of general interest to all system users.

5. Remote Job Control Tables - CRJE.SYSLIB(JBTBLS). This file contains status information for all active jobs currently in the CRJE system.

These five members of the system library are copied into the global files of the active area at start-up/initialization time. They make up the global files in the following order:

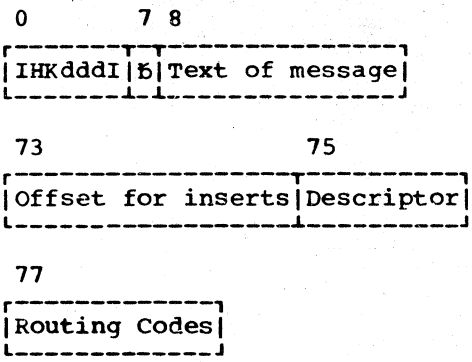
Global File #1 - SYSMSGS
Global File #2 - USERS
Global File #3 - USRMSGS
Global File #4 - BRDCST
Global File #5 - JBTBLS.

Global files #6, #7, and #8 are utility files that are dormant. When needed, they are created by the AFIO macro CREATE and are released by the AFIO macro RELEASE when their function is completed.

Global file #6 is used by the CRJE library condense module (IHKCDP). Global file #7 is used by the MERGE subcommand processor. Global file #8 is used by the RENUMBER subcommand processor.

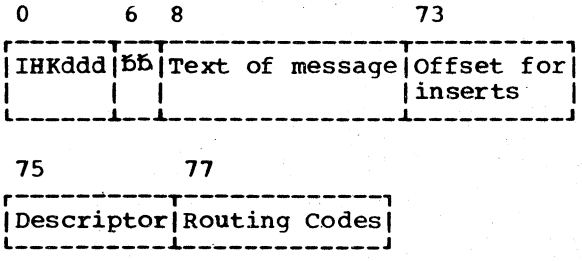
The format for the 80-byte records in the system message file is as follows:

For the central installation messages:



where ddd is the sequence number of the message.

For the terminal user messages:

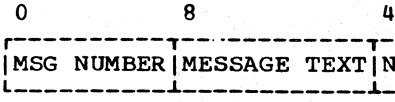


where ddd is the sequence number of the message.

Each entry in the user verification file is 88 bytes in the system library and 80 bytes plus an eight-byte key in the global file. This user verification file is discussed in detail in Data Area Layouts.

The user message file has the same format as the system message file except that the "offset for inserts" field is blanks, and there are no routing codes or descriptor codes. The key is the userid plus the sequence number, which is in the last 8 bytes of the record in the system library.

The broadcast message file has the following format:



Bytes 0-7 - contain the message number, which is in the form, BRDbnnnn, where nnnn can be 0-9999.

An installation-defined number of broadcast messages, with a maximum of 100, is maintained in this member. The large numbering scheme (0-9999) allows for the ordering of messages with a sufficient increment between numbers to facilitate adding messages anywhere within the file. Messages are inserted, replaced, and deleted within the active area.

The remote job control table (RJCT has the format that was previously discussed in the section on control blocks. Detailed information about each field is contained in the Data Area Layouts section.

USER LIBRARY

The user libraries are partitioned data sets with a name of CRJE.LIB.userid. A library must be allocated for each potential active user who plans to maintain permanent libraries. The user libraries must be allocated before the start-up of CRJE.

The size of the library depends on the estimated number of data sets that will be stored at any one time and on the size of the data sets.

User library directory entries are of variable length depending upon how they are created. The entries that are created by CRJE are 40 bytes; those created by CRBE are 30 bytes; and those created by utility programs are 12 bytes. Each one has a different format.

The EDIT command processor is the only command processor that recognizes all three types of data sets. The IHKED1 module checks the length of the directory entry; if it is not a CRJE or CRBE entry, the directory entry is created using the information given on the EDIT command and the default operands. If the entry was created by CRBE, it is expanded using the information already in the entry and the information in the EDIT command.

The formats of the three types of directory entries are discussed in section 5, Data Area Layouts.

INPUT/OUTPUT FLOW

INPUT

Input to the CRJE system from the terminal exists in two forms: commands and data. The user enters input in one of three modes: command mode, edit mode, and input mode. Data or the type of commands entered determines the mode in which the

user operates. Data is entered as lines of text in input mode to create a data set. This data set might be saved in the user library and used later as job input.

Commands

In command mode the user can enter any command, but not subcommands. The user is in command mode as soon as he logs on, and he stays in command mode unless he enters an EDIT command, after he enters an END subcommand in edit mode, or until he logs off.

The user enters edit mode when he issues an EDIT command that initiates updating operations on an existing data set. In edit mode the user can enter EDIT subcommands to delete, replace, and add lines to an old data set. A null line or an INPUT subcommand puts the user in input mode. From edit mode the user returns to command mode by entering an END subcommand. The user's active file is deleted when an END subcommand is entered, and EDIT subcommands are no longer accepted.

The user enters input mode from command mode by issuing an EDIT command that initiates the creation of a new data set in his active file. Input mode is entered from edit mode by issuing an INPUT subcommand or a null line. A data set is created in input mode by entering one line at a time. The user cannot go from input mode directly to command mode. A null line must be entered to go to edit mode; then an END subcommand puts the user in command mode.

Job input can be the data set being created by the user in his active file. Data sets that have been created by the user and then saved in his user library can also be job input. The input for one job can be a combination of the user's data sets stored in his library, his active file, and data sets stored in another user library. Whether input is a data set in the user's active file or a data set that the user has saved in his library, the data was initially created by the user or by another CRJE user. If another user's data set is included and is protected, the key must be provided before use of the data set is permitted.

Data

Data is entered into the user's active file in input mode. The user can specify that line numbers be displayed at his terminal as a prompt for a line of input. Whether the user receives the line-number prompts or not, the line numbers are still maintained in his active file. Input lines must be 80 characters unless line numbers are kept in the last 8 characters of the line, in which case 72 characters is the maximum. Input lines of more than 80 characters (or 72 if line numbers are specified) are truncated, and input lines of less than 80 characters (or 72 if line numbers are specified) are padded with blanks.

Note: When a user attempts to save input lines in an 80-character instead of an 88-character record user library, the input lines should not exceed 72 characters. If they do, the data in positions 73-80 is truncated, and a warning message is sent to the user.

OUTPUT

Output that can be received at a terminal is divided into two categories: data and messages. Data output can be the contents of a specific CRJE or OS data set, or it can be output of a submitted job. The terminal user must request data output. Besides messages from the central operator and messages from other users, there are other kinds of CRJE terminal user messages: error messages, information messages, and syntax checker messages.

Messages

The central operator can send messages to terminal users by the central command MSG. If the terminal user is not active, the central operator can request that the message be queued for later delivery, and the user will receive the message when he logs on. The central operator also creates and maintains the broadcast messages. These are messages that contain information of general interest to all users. The broadcast messages are sent if requested to each user when he initiates his session. The user may request the broadcast messages any time during his session by entering the LISTBC command.

Terminal users can send messages to other users by the SEND command. If the user is not active at the time the message is sent, it can be queued for later delivery. The message then becomes a delayed message and the user receives it when he logs on. These messages are not queued for later delivery unless the user sending the message designates them as such.

Most of the CRJE terminal user messages are dependent upon the input from the terminal. A message response to a command usually indicates an error in the command or an unexpected situation. Nonexistent information might be specified in the command, or the command might have a syntax error. An out-of-space condition or a data set that could not be found are examples of such unexpected situations. A message response to a command might be information. Commands such as STATUS, LISTLIB, and LISTDS request information about jobs and data sets. This information is given to the user in the form of messages. Messages that are sent to a user in response to a command entered are usually delivered to the terminal immediately following the command.

The only messages that are sent to the user in response to data entered are line number prompts, syntax checker error messages, and messages indicating unexpected situations such as out-of-space messages.

When conversationally submitted jobs terminate, and when the CRJE system closes down, notification messages are sent to user. Since these messages are not related to input entered at the terminal, they may be sent at any time. They will be printed at the terminal before a command is entered. The CRJE closedown message is printed immediately, regardless of what mode the user is in.

Data

By use of the OUTPUT command the user may request data output of a job he submitted conversationally to OS. Once the output of a conversationally submitted job is returned in full to the user, the data set is scratched. Thus, there is only one copy of output available.

The LIST subcommand lets the user get a copy of the contents of his active file. Since the data set in his active file is only scratched when an END subcommand is entered, the contents of the user's active file can be listed as many times as desired. Data sets are listed at the terminal one logical record per line or in 80-character lines. A line of an OS data set, with a maximum of 120 characters, is printed as one line.

The terminal user can discontinue the transmission of SYSOUT output data at his terminal. How he discontinues the output depends upon his particular terminal. When output requested by the OUTPUT command is discontinued, the data set is not scratched. The user can receive the remaining output by entering the CONTINUE command. If data output requested by a LIST subcommand is discontinued, the only way the user can get the remaining output is by entering another LIST subcommand.

CRJE GENERATION AND ASSEMBLY

The CRJE facility is incorporated into the operating system in the following manner: by copying the resident CRJE modules and BTAM modules from the component libraries into the telecommunications library (TELCMLIB), by copying the nonresident CRJE modules into the link library (LINKLIB), and by copying the initialization module IHKINT into the telecommunications library (TELCMLIB). The MACLIB must be specified since it contains the CRJE assembly macros, and the PROCLIB must be included because it contains the START procedure for CRJE; the assembler and the linkage editor must also be included. All of the facilities to support CRJE are specified with macros at the generation of the operating system.

After an operating system has been generated with the features required for CRJE, the CRJE system must be defined and generated. Generating the CRJE system involves the assembly of CRJE macros; the assembly of any installation exit routines; the creation of the CRJE load module; the allocation and initialization of CRJE data sets; and the provision of a procedure for the execution of CRJE and a procedure for the reader. This generating process is discussed in detail in IBM System/360 Operating System: CRJE System Programmer's Guide, GC30-2016.

Each installation is allowed to define its system configuration by three assembly macros residing on SYS1.MACLIB: CRJELINE, CRJETABL, and CRJEUSER. The operands of the macros provide the installation with the opportunity of specifying information about the system configuration as well as specifying system options.

The object module IHKMAC is produced when the CRJELINE macro and CRJETABL macro are assembled. This object module contains the CCT for the system and the CLB, STCB, DCB, and DECB for each line; the module resides on SYS1.TELCMLIB. The IHKMAC module is linkage edited with the installation exit routines, the AVT, and the preassembled resident CRJE modules. The output of this linkage edit forms the CRJE load module, IHKBGN.

CRJELINE

The CRJELINE macro describes the communications network supported by the CRJE system. For each communications line one macro must be assembled. For each line, a macro generates the following control blocks according to the operands specified on the macro:

- A BTAM DCB for each line group and a DECB for each line. (These are used in reading from and writing to the line.)
- A conversational line block (CLB).
- A DDNAME of the data set that will be used for writing a submitted job stream to disk.
- A subtask control block (STCB).
- An event control block (ECB) entry in the CRJE dispatcher's ECB list.
- An ECB entry in the ECB list of the utility task and an entry in the list of the loader/controller task.

CRJETABL

The CRJETABL macro specifies all installation-wide options selected for a particular system, such as the maximum number of broadcast messages, the maximum number of delayed messages, the maximum number of CRJE-submitted jobs that can reside in the CRJE system at one time, the number of central commands that can be queued and waiting to be processed at any one time, the installation exit routines, and the installation-provided commands and subcommands.

The maximum number of CRJE-submitted jobs that may reside concurrently in the CRJE system is equal to the number of records allowed in the remote job control table (RJCT) of the system library. The CRJE control table (CCT) is generated according to the operands specified. If the exit routines exist and are specified, the entry point addresses of the JCL exit routine, the LOGON exit routine, and the LOGOFF exit routine are put in the AVT.

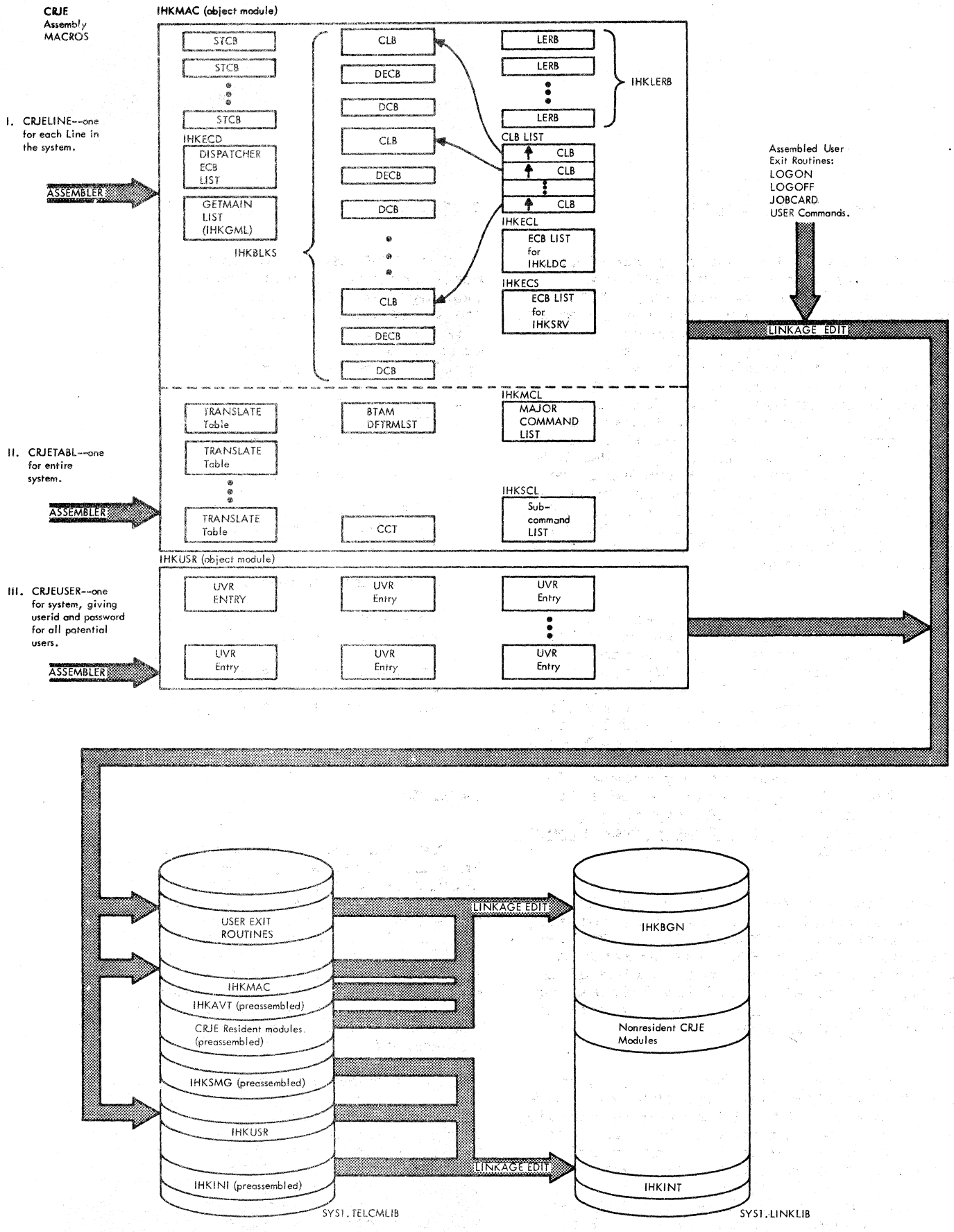


Figure 3. Defining the CRJE System

CRJEUSER

The CRJEUSER macro specifies the userid and password for all potential users. This macro generates one record in the CRJE.SYSLIB(USERS) member of the system library for each potential user. When a user logs on, his userid and password are checked for validity against the userids and passwords in the USERS global file.

The assembly of the CRJEUSER macro produces the object module IHKUSR, which resides on SYS1.TELCMLIB. This module contains a user verification record (UVR) for each potential active user. The IHKUSR module is linkage edited with the IHKINI module and IHKSMG module, which have been preassembled and placed on SYS1.TELCMLIB. The IHKINI module is the system library initialization module and the IHKSMG module contains all system messages. When these modules are linkage edited, the output forms the initialization module IHKINT, which resides on SYS1.LINKLIB. This initialization routine must be executed before the first start-up of CRJE.

Figure 3 (DEFINING THE CRJE SYSTEM) illustrates the assembled macros, the generated control blocks, and the linkage edit steps.

INITIALIZATION AND START-UP

INITIALIZATION

The initialization of CRJE begins before the start-up of the system. The CRJE system library initialization routine IHKINI is a CRJE utility program that must be executed before the first start-up of CRJE. This routine resides on SYS1.LINKLIB as the load module IHKINT with the entry point IHKINI. The load module consists of the CRJE system messages (IHKSMG), the list of users and passwords (IHKUSR), and the system initialization routine (IHKINI).

The initialization routine opens the system library and writes the system messages. The user entries, if any, are written in the system library and the directory entries are stored by the STOW macro. The remaining system library directory entries (user messages, broadcast messages, and job tables) are also stored. The CRJE system library is then closed and control is returned to the operating system.

If any I/O errors are encountered in this initialization, a message is sent to the central operator and processing discontinues.

START-UP

When the START command for CRJE is entered at the central console, the command scheduling routine of OS (SVC 34) gives control to the START command processor (IHKBGN).

Chart A illustrates the function of CRJE start-up and initialization, beginning at the time the START command is entered at the central console.

The START command processor issues an ENQ macro to be sure that CRJE is not already an active task in the operating system. If CRJE is not already an active task, the CRJE initialization routine (IHKCIP) inspects the parameters on the START command and performs the initialization of CRJE.

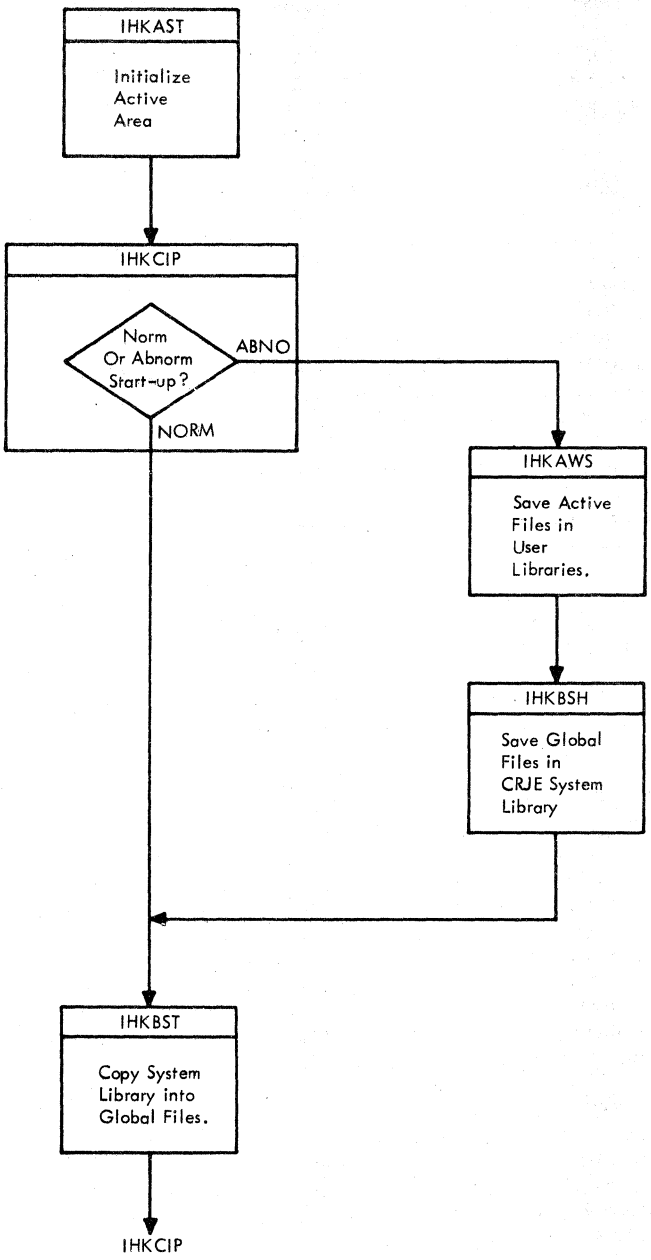
If the parameters on the START command are valid, an ATTACH macro is issued for the loader/controller task (IHKLDC) and for the utility task (IHKSRV). A LOAD macro is issued for the OS routines IEFQMDQ2, IEFQDELE, and IEFQMSSS.

As illustrated in Chart A, initialization depends upon the parameters specified on the START command. The process of initializing the active area depends upon whether the NORM or ABNO parameter is on the START command. NORM indicates a normal start-up. This means the previous closedown was normal and the copy of the system library in the global files was saved in CRJE.SYSLIB and all active files were saved in user libraries. In this case all that has to be done is initialize the active area and copy the members of CRJE.SYSLIB into the global files of the active area.

ABNO means the last closedown of CRJE was an abnormal closedown and the active area contains active files and system information that has not been saved. Therefore, this information must be saved and the system library updated and rewritten.

There are four modules that perform initialization on the active area: active area start-up/initialization module (IHKAST), active area recovery module (IHKAWS), library I/O shutdown module (IHKBSH), and library I/O start-up module (IHKBST). The following diagram illustrates how the modules work together to perform the complete active area

initialization under normal or abnormal conditions.



By use of the DEVTYPE macro, the IHKAST module checks to be sure that the device to which the active area data set has been assigned is a 2311, 2314, or 2319. The active area DCB is opened and the DEB is checked to be sure that no more than one extent has been allocated to the active area. A GETMAIN macro is issued for the track allocator table (TAT); and the table is initialized to show all of the tracks in the active area are available for allocation. Several fields in the KONBOX are initialized: such as number of heads per cylinder, maximum track length, block overhead, tolerance factor, track address of master index, length of master index, and mean cylinder number. The IHKAFI module and the IHKEXC module of AFIO uses these fields during the active file track allocation, deallocation, and master index track initialization.

If the start-up is abnormal, the IHKAWS module attempts to save the active files that existed at the time of the abnormal closedown. This operation depends upon gaining access to the active area.

The USERS global file is used to determine which users had an active file at the time of closedown. For users with an active file at closedown, the AFIO work area in the TUB is reinitialized with information from the user's active file entry in the master index track and from the user's own file index track. His user library is opened and a library I/O macro is used to point to the beginning of the user's active file. The user's active file is saved in his user library under the member name of ACTIVE. This processing is done for each user having an active file at the time of the last closedown.

The library I/O shutdown module (IHKBSH) saves the global files of the active area into the system library CRJE.SYSLIB. Librarian macros are used in the copy process. If the operation is terminated because of an error, the central operator is notified of the situation.

If the start-up is normal or if the start-up is abnormal, after recovery, the library I/O start-up module (IHKBST) copies the members of CRJE.SYSLIB into the global files. This module also searches the direct-access devices that contain user libraries. When a user library with the name CRJE.LIB.userid is found, the userid is checked for validity; and if valid, the DDNAME for the volume on which the library resides is put in the last eight bytes of that user's entry in the USERS global file.

The CRJE initialization module (IHKCIP) performs RJCT initialization according to the type of start-up--FORM, NFMT, or NONE. A START RDRCRJE command is issued for each job submission data set built by the SUBMIT processor and not processed by the OS reader. The remote job control table (RJCT) entries are inspected. All jobs not marked complete and not on SYS1.SYSJOBQE are deleted from CRJE and a message is queued for the user. To delete a job from CRJE means to delete the RJCT entry for that job and remove all references from SYS1.SYSJOBQE.

If the start-up specified is FORM, all RJCT entries are deleted. In this case there is nothing on SYS1.SYSJOBQE (reformatted at IPL); therefore, the RJCT entries are deleted for jobs marked

complete. If the start-up specified is NFMT (SYS1.SYSJOBQE not reformatted at IPL), all jobs marked complete are changed to not complete because all the pointers in the RJCT entries are incorrect. When the pointers are corrected, the jobs are marked complete again. If the start-up specified is NONE (no IPL since last start-up), an IOB (Input/Output Block) is built for each job marked complete.

An OPEN macro is issued for each line having a DD card. A X'40' is posted in the dummy ECB in the STCB for the lines successfully opened. A X'40' is posted in the stop acknowledgment ECB in the CLB for the lines not opened successfully.

The loader/controller now fills the transient area with nonresident modules. If the syntax checkers are specified, they are loaded and are given a chance to initialize their areas. A delete request is made for each module that was brought into the transient area by the loader/controller. This is done to reduce the request count to zero but physically leave the modules in the transient area.

The address of the communications ECB is put in the dispatcher's ECB list as the entry for the central commands. A message is sent to the central operator informing him that CRJE is now active.

After all initialization has been completed, the job termination handling routine (IHKSDQ) is given control to start processing on the lines. But, before any actual processing is done, the job end processor checks to see if any CRJE jobs have completed, and if so, dequeues the job from OS and queues a message for the user who submitted the job.

If a job is found on the output queue, a search is made for the RJCT entry for that job. If the RJCT entry is not found, the job is deleted because it is not a CRJE job.

If the RJCT entry is found for the job that is on the output queue, if the OS queue manager dequeue routine reads an SMB (System Message Block), and if the job is already marked complete, the job is deleted because it is a duplicate. If it is not marked complete, the job end processor marks it complete, queues a notification message for the user, and updates the RJCT entry for the job.

At this point the job termination handling routine attempts to retrieve the next job on the output queue. If there are no jobs to dequeue, the job termination handling routine waits for the ECB given to the OS queue manager by the IHKDEQ module. Control is returned to the point where the IHKDEQ module is invoked.

CRJE is now ready to perform processing on the communications lines.

SESSION MANAGEMENT

A session is the period of time that a user is active at a terminal. A user's session is initiated by a LOGON command, but may be terminated in any one of three ways. If he enters a LOGOFF command the session is terminated. If another user enters a LOGON command at the terminal before the first user enters a LOGOFF command, he is logged off automatically. An irrecoverable line error or closedown at the central installation causes automatic termination of a user's session.

Chart B illustrates the functional process of session management.

INITIATION

A command and its operands are put in the line buffer when entered at the terminal. The line administrator translates the text to EBCDIC and removes all line control characters and backspace characters. The system administrator examines the command. If it is invalid, it is rejected and an error message is sent to the user. Otherwise, a PPT (Parameter Position Table) is allocated and the parameters specified on the command are put in the PPT along with a code specifying the command.

If the command entered was a LOGON command, the system administrator checks the TUB to see if the previous terminal user was logged off. If the previous user was not logged off, the command analyzer issues a request for the loader/controller to load the LOGOFF processor. Control is then passed to the LOGOFF processor to log off the old user, but the PPT still contains the LOGON command code and parameters. The LOGOFF processor performs the LOGOFF functions and then checks the suppress bit in the CCT. If the bit is on, meaning that no more users can be logged on, the LOGON suppressed message is sent to the terminal, and the LOGOFF processor then returns to the command analyzer with a return code of 0 to indicate no LOGON is to be performed. If the suppress bit is not on, the LOGOFF processor returns to the command analyzer with the negative value of the LOGON command in register 15. The

command analyzer then requests the loading of the LOGON processor.

If the previous user was logged off (meaning that no automatic LOGOFF is to be performed), the system administrator checks the suppress bit in the CCT. If the suppress bit is on, this means that the central operator has requested that no more users be allowed to log on. So the LOGON command is refused and the user is notified. If the suppress bit is not on, the loader/controller is requested to load the LOGON processor.

The operands of the command are checked and the TUBBRD (broadcast messages requested) and TUBMID (message IDs requested) switches in the TUB are set depending upon the operands specified on the command. The user verification file manager (IHKUTM) is called to verify the userid and password. If no userid was entered or if the userid is not found or is already in use, then the user is prompted for a userid that is not in use. The user is prompted only once for a userid. If a correct one is not entered, he is notified that his LOGON is not accepted.

The IHKUTM module turns on the active bit in the user's verification record (UVR) in the global files. The password entered on the LOGON command is compared with the password in the user verification record. If they are not the same or if no password was specified on the LOGON command, the user is prompted for a password. Again the user is prompted only once. If the correct password is not specified, the LOGON command is not accepted and the user is notified.

If a LOGON user exit is provided, then the parameter list is set up to pass control to the exit. The first word of the parameter list points to the userid in the TUBUSRID field of the TUB. The second word points to the accounting information, or to a one-byte area containing X'FF' if no accounting information is given. Register 1 points to the parameter list, and the exit routine is then given control. If the return code from the exit routine is 4, the user is not allowed on the system. In this case a message is sent to the user, the active bit in the UVR is turned off, and a delete request is issued to have the loader/controller delete the LOGON processor from the transient area. If the return code is zero, the TIME macro is issued to get the LOGON time and it is stored in the TUB. The message writer then sends a LOGON message.

The IHKLGN module checks the CCT to find out if a SHOW SESSION for all LOGONs has been requested. If not, the UVR is checked to determine if a show session has been requested for this particular user. In either case, a message is sent to the central operator indicating the session.

The user is now successfully logged on and his userid is put in the TUB. Control is returned to the command analyzer, and a deletion request is issued for the LOGON processor.

TERMINATION

There are three possible reasons for the termination of a user's session: an error condition (active area I/O error, line error, or subtask abend), a LOGOFF command from the the terminal user, or a LOGON command from terminal user before the previous user issued a LOGOFF command. Normal termination of the CRJE system (STOP central command) causes the system administrator to issue a LOGOFF command for each active user. This LOGOFF command is handled in the same way as a LOGOFF command from a terminal. The reason for the termination of the session affects certain procedures in the LOGOFF process. If the reason for the LOGOFF is an active area I/O error, the TUBUTMN flag in the TUB is set. This means that no calls can be made to the IHKUTM routine, no show session messages can be sent to the central operator, and no LOGOFF message can be sent to the terminal user. Otherwise, processing is the same. If the reason for the LOGOFF is a line error, the TUBABEND flag in the TUB is set. The only change in the LOGOFF process in this case is that no messages can be sent to the terminal user. If the LOGOFF is automatic because of another user logging on, the only change in the process is that the LOGONs suppress bit in the CCT is checked before returning to the command analyzer.

The LOGOFF processor first gets the LOGOFF time by issuing the TIME macro. If an active area I/O error is not the cause of the LOGOFF, the IHKLGF module calls the IHKUTM module to verify the userid, to turn off the active bit in the UVR, and if abnormal termination is the cause of the LOGOFF (which was found by checking the ABEND bit in the TUB), to turn on the abnormal termination bit in the UVR. The show session bit in the CCT is checked and, if it is on, the message writer is requested to send the SHOW SESSION message to the central operator. If this bit is not on, the show session bit in the UVR is checked and if on, the SHOW SESSION message is sent to the central operator. The session time is calculated using the LOGON time, which is stored in the TUBTIME field

of the TUB, and the LOGOFF time. If abnormal termination is not the cause of the LOGOFF, the message writer sends the LOGOFF message to the user indicating LOGOFF time and total session time.

The parameter list is built for the LOGOFF exit, if one exists. The first word points to the TUBUSRID field in the TUB, which contains the userid. The second word points to a field containing the type of terminal at which the user was logged on. The third word points to a field containing the LOGON time, LOGOFF time, and session time. There are no return codes from the LOGOFF exit routine to be checked.

If main storage has been allocated for the IHKTAB module, it is now freed. The PPT is checked to see if this is an automatic LOGOFF caused by another LOGON. If it is, the command code in the PPT is that of the LOGON command. In this case the suppress bit in the CCT is checked; if it is on, the LOGONs suppressed message is sent to the terminal user, and control is returned to the command analyzer. If it is not on, the negative code for the LOGON command is put in register 15 and the TUB is initialized to zero except for the first 60 bytes. If the LOGOFF was automatic because of another LOGON, the command analyzer issues a delete request to the loader/controller for the LOGOFF processor and a load request for the LOGON processor. Processing then continues in the LOGON processor as described in the section on initiation.

DATA MANAGEMENT

Data management is divided into two areas. One area is the management of an active file in its entirety. This involves assigning space in the active area to a user and getting the data set into this space. The active file may contain an OS data set; it may contain a user's CRJE data set from his library; or it may contain a new data set the user creates. Managing the active file in its entirety also involves transferring it from the user's space in the active area to his user library, to OS for processing, to the user at his terminal, or deleting it altogether. Figure 4 illustrates these ways of getting a data set into the active area.

The second area of data management is manipulating the active file line by line or by groups of lines. This type of data management is discussed as the update function, which involves changing, deleting, or inserting lines or groups of lines or changing line numbers. The scan function is considered line-by-line data management.

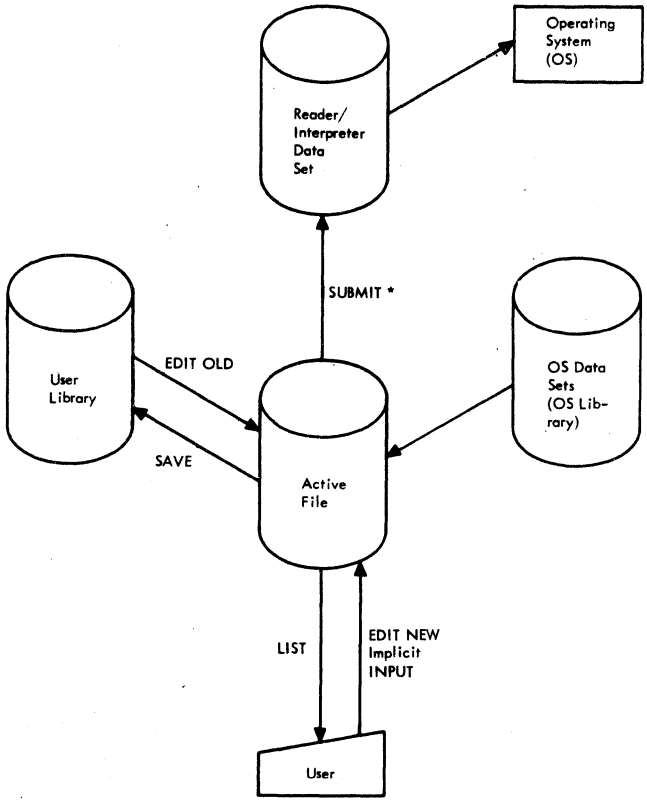


Figure 4. CRJE Data Management: Active File

CREATE FUNCTION

By entering an EDIT command with the NEW operand specified, the user indicates that he wants to create an active file. The command and its operands are put in a PPT by the command analyzer. Then the command analyzer branches to the command processor, which in this case is the IHKEDT module. The IHKEDT module checks the operands; if any operand is invalid, a message is sent to the user and control is returned to the command analyzer. If the dsname is valid, the IHKIRL module is entered to process the lines entered from the terminal. (See Chart C.)

A GETMAIN macro is issued and a directory entry is created. The address of

the GETMAIN area is put in the TUBDIRAD field. A CREATE macro is then issued to allocate space for an active file. The message writer prompts the user with a line number. His reply is a line of text that is put in his active file. The IHKAFI module is used to insert the lines of text into the user's active file. (This module is discussed in detail in the Librarian part of the Program Organization section of this publication.)

COPY FUNCTION

The copy function of data management within CRJE involves getting information into the active area space that is allocated to a user. The user issues an EDIT command with the OLD operand specified and the data set name. The data set can be a CRJE data set residing in a user library or an OS data set. (See Chart C.)

After the data set is copied into the space in the active area assigned to a user, it is referred to as his active file.

OS Data Set

The IHKEDT processor gets control with the command and its operands in a PPT. The operands are checked for validity. If any operand is invalid, a message is queued for the user and control is returned to the command analyzer. Otherwise, control is passed to the IHKEOS module, which calls the IHKIRL module to allocate space in the active area for an active file.

The IHKEOS routine, using the ATTACH macro, attaches the IHKOPN module, which issues a LOCATE macro and an OBTAIN macro to get the DSCB (Data Set Control Block). The DCB is set up in a GETMAIN area. The IHKOPN module then returns control to the IHKEOS routine. The IHKEOS module issues a GETMAIN macro for a read buffer, the size of which depends upon the blocksize of the data set to be read; then the data set is opened. The way in which the records are put in the buffer depends upon the record size and whether the records are fixed, fixed blocked, variable, or variable blocked. As the lines are built they are assigned line numbers using the default for starting line number and increment. When a block of ten lines is moved into the buffer, an AFIO macro is issued to insert the lines in the active file. This is continued until the entire data set is read and inserted into the active file. The IHKOPN module is then detached.

CRJE Data Set

The command processor (IHKEDT) receives control with the command and operands in a PPT. The operands are checked for validity. If any operand is invalid, a message is queued for the user and control is returned to the calling routine.

The CREATE macro is issued by the IHKED1 module to allocate space for an active file. The user verification file manager (IHKUTM) is called to verify the userid as the owner of the data set, to put the ddname of the library DD card into the KONBOX, and to queue the subtask for library I/O in order to access the user library. After this is done the CRJE data set is opened and the directory entry is made available.

The directory entry is inspected to see if it conforms to the CRJE format. If it does, changes are made in the directory entry according to the operands specified on the EDIT command, and the count of the number of times it was read is updated. If the directory entry does not conform to the CRJE format, it is expanded using the information already in the entry, the information in the operands of the EDIT command, and the EDIT default options.

When X's are found in the key field while reading the data set, the keys are sequenced. The default starting line number and default increment are used. When neither X's nor numerics are found in the key field, the key is set to 0 and a message is queued for the user. A library I/O macro is issued to read a block of records. An AFIO macro is issued to insert the records into the active file. This is continued until the complete data set is read and put into the active file. The IHKBPM module performs the reading of the CRJE data set, and the IHKAFI module initiates the function of inserting the lines in the active file.

UPDATE FUNCTION

The EDIT command initiates updating operations by getting a data set into the user's space in the active area. The second area of data management, which is done line-by-line, is requested by the EDIT subcommands. The functions that can be performed line-by-line are:

- Insert lines
- Replace lines
- Delete lines
- Replace character strings within lines

Combine a data set with the active file or copy lines from one place to another in the active file
Reassign line numbers to the lines
Request a syntax analysis of PL/I or FORTRAN source language statements
Indicate tab settings while entering lines

After the EDIT command is entered, only EDIT subcommands are accepted until an END subcommand is entered.

Entering Lines

Lines are entered into the active file by the Implicit subcommand or the INPUT subcommand. When the INPUT subcommand is entered, the INPUT processor (IHKIPT) gets control from the command analyzer, with the subcommand and its operands in a PPT. The numeric verification module (IHKNUM) checks the first two operands to see if they are all numeric. The INPUT subcommand processor saves the line number and increment in the PPT, and a switch is set depending upon whether R(eplace) or I(nsert) was specified. If the PROMPT operand was specified, the TUBLNPMT switch is set. If any errors are found in checking the operands, an error message is queued for the user, and control is returned to the command analyzer.

If no numeric operands are specified, the Implicit subcommand processor (entry point: IHKIRL01) inserts the lines of text at the end of the active file and uses the same increment as the other line numbers in the active file.

Otherwise an AFIO macro is issued to find the point in the active file at which the lines are to be inserted. If I was specified, the macro is for the specific line. If that line number already exists, an error message is queued for the user, and control is returned to the command analyzer. Otherwise the pointer is to the next lower line number. The Implicit subcommand processor (IHKIRL) prompts the user with a line number. The user responds with a line of text. An AFIO macro is then issued and the IHKAFI module of AFIO initiates the process of inserting the line into the active file. The user is then prompted with another line number. This is repeated until the user enters a null line or until the line number to be inserted is greater than the next line number in the active file.

If the R operand was specified, the AFIO macro is for the specific line number. If the line number does not exist, an error message is queued for the user and control is returned. Otherwise the Implicit subcommand processor (IHKIRL) prompts the user with the line number. The user responds with a line of text. An AFIO macro is issued to put this line of text in the active file. This process is repeated until a null line is entered or the end of the data set is reached. If a line number does not exist, the user is notified.

If neither the R nor the I operand is specified, the AFIO macro is issued for the specific line number. If the specific line number does not exist, the macro is for the next lower line number.

When the Implicit subcommand is entered, the IHKIRL module inserts or replaces the lines in the active file or deletes the line. An AFIO macro is issued for the line number. If the line already exists in the active file, another macro is issued and the line of text typed at the terminal overlays the old line already in the file. If the line does not exist, a macro is issued and the line of text is inserted after the next lower line number.

Deleting Lines

If there is no text in the Implicit subcommand, a macro is issued by the IHKIRL module to delete the line from the file. If the line does not exist, an error message is queued for the user. In either case control is returned to the command analyzer.

When the DELETE subcommand is entered, the numeric verification module (IHKNUM) checks that the operands (if any were specified) are all numeric. If one line number is specified, a macro is issued to delete the line from the active file. If the line does not exist in the active file, an error message is queued for the user, and control is returned to the system administrator. If no operands are specified on the subcommand, the last line in the active file is deleted. If two line numbers are specified on the subcommand, all of the lines in the active file with line numbers greater than or equal to the first line number specified and less than or equal to the second line number specified, are deleted.

The complete process of entering and deleting lines in the active file is illustrated in Chart D.

Changing Lines

The CHANGE subcommand is used to change character strings within a line. This subcommand is used when correcting statements having a syntax checker error. In this case the TUBCORRN switch must be set, and the line numbers specified on the subcommand must be within the range specified in the TUBIRLSA field. If they are not, an error message is queued for the user.

The system administrator puts the subcommand code with the operands in a PPT and passes control to the CHANGE subcommand processor (IHKCGN).

The IHKCGN module checks the operands specified. The IHKNUM module verifies that the line numbers given are all numerics. If any errors are found, an error message is queued for the user and control is returned to the command analyzer. If more than one line is specified, the first line is read into the AFIO buffer. The IHKCCS module scans the line for a character string that matches text1. When a match is found, the first part of the line is put in the work area. Text2 is then put into the work area. The remainder of the line after text1 (as it is in the AFIO buffer) is put in the work area. The entire line is then put back in the AFIO buffer. If ALL is specified in the subcommand, the complete process is repeated until the end of the line is reached.

The line always exists in the AFIO buffer as it appeared after the previous scan. This is to insure that the line will be correctly updated when the ALL operand is specified.

If text2 is shorter than text1, the line is condensed and padded with blanks. If text2 is longer than text1, the line is expanded and text2 is inserted. If the line exceeds the maximum length after text2 is inserted, the line is truncated and an error message is queued for the user.

When the complete line is scanned, an AFIO macro puts the line back in the active file. If two line numbers are specified, the next line is read and the process is repeated until the last specified line is reached or EOD is detected. Control is then returned to the command analyzer.

If only one line number is specified, the same process is done for that one line. If the line does not exist or if there is no match on text1 within the line or lines scanned, an error message is queued for the user and control is returned to the IHKCMD module. (The change process is illustrated in Chart E.)

Merging Lines

The MERGE subcommand is used to insert lines from a user's data set into the active file or to insert lines that are in the active file to another place in the active file. A complete data set may be inserted or just the lines specified. The lines may be inserted at the end of the active file or after a specified line within the active file.

The system administrator passes control to the IHKMGE processor with the subcommand code and the subcommand operands in a PPT. The operands are checked for validity. The IHKNUM module checks the line number operands for all numerics. If any errors are found, a message is queued for the user and control is returned to the system administrator.

If one or three line number operands are specified, a utility file is created and all the lines in the active file are read and written into the utility file. The active file is then released and a new one created. The IHKAFI module initiates the manipulation of lines in the active file when AFIO macros are issued.

If the single asterisk (*) is specified on the subcommand, control is passed to the IHKMAA module to process a merger of the active file into the active file. If a utility file exists, all the lines of the utility file, up to and including the line specified as the line number after which the merger is to be made, are inserted into the new active file. Then the entire utility file when no or only one numeric operand is specified, or the specified range when two or three numeric operands are specified, is inserted at the end of the new active file and the lines are resequenced. The lines in the utility file after the line number at which the merger was made are inserted at the end of the new active file. The lines in the new active file are resequenced, and the utility file is then released.

If there is a data set name specified on the subcommand, the IHKMUF module handles the merging process. If the directory entry for the data set to be merged is not of the CRJE or CRBE format and a line range is specified, the command is rejected. If the library I/O module (IHKBPM) cannot find the data set in a user library, an error message is queued for the user and control is returned to the command analyzer. If

the data set is found and is in a user library, it is opened; or if the owner's userid and any protection key have been specified, it is opened.

The insertion of the lines into the active file is done in the same way as described before, by overlapping AFIO macros and library I/O macros. If a range of lines is specified, only these lines are inserted; otherwise the entire data set is inserted. If lines from the active file have been read into a utility file, they are reinserted at the end of the active file. All lines are resequenced when inserted into the active file. Control is returned to the command analyzer. (Chart F illustrates the merge function.)

Renumbering Lines

The RENUMBER subcommand is used to request that new line numbers be assigned to the lines of the active file. Control is passed to the RENUMBER subcommand processor (IHKRNR) with the subcommand code and the operands in a PPT. The IHKNUM module first checks the operands for all numerics. If an error is found, an error message is queued for the user, and control is returned to the command analyzer. If no operands are specified, the default option is ten for both line number and increment. The operand values are stored in the last 40 bytes of the user buffer, which is used as a work area.

A block of records is read from the active file, and the line numbers are resequenced. If line numbers are contained in the last eight characters of the line, they are also changed. A utility file is created and, when all the lines in the block are resequenced, the block is written to the utility file. The process of reading a block, resequencing the line numbers, and writing the block in the utility file is continued until EOD is encountered on the active file. The utility file is then released and control is returned to the command analyzer. The RENUMBER processor issues AFIO macros to manipulate the lines in the active file and utility file.

If the RENUMBER subcommand is issued for an active file that has no lines in it, the increment operand becomes the default increment for prompting the user with the line numbers in input mode. If the RENUMBER subcommand is issued for an active file that has no lines in it, the subcommand has the effect of changing the default increment. (Chart E illustrates the renumber process.)

Setting Tabs

The TABSET command is used to define input tab settings or output tab settings. If the command is entered to indicate OFF for either input or output tabs, the processor (IHKTAB) deallocates the area in which the tab settings are stored and sets the tab address pointer in the TUB to zero.

When the command is entered with tabs, they are checked for length, ascending sequence, and numerics. A maximum of 10 tab settings is accepted. If more are submitted, the first valid 10 are taken and an error message is queued. A tab setting of 0 or 1 is not accepted, and a setting greater than 80 for input or 120 for output is rejected. As the tabs are checked they are stored in a work area. After all the tab settings have been checked, they are moved to the user buffer. If no errors are found, the number of settings is put in the first halfword of the area containing the settings. If the tab address pointer in the TUB is 0, a GETMAIN macro is issued for a tab area. If the tab address pointer is not zero, the tab settings are put in that area. If any errors are encountered, a message is queued for the user before returning to the command analyzer.

When the line administrator reads a line of text from the terminal and encounters a horizontal tab character, the TABSET edit module (IHKLAT) is called if the TUBTABAD field is nonzero. The IHKLAT module gets the address of the area containing the tabs from the TUBTABAD field. The line is edited by adding the proper number of blank characters whenever a horizontal tab character is encountered.

When the line administrator is ready to write a line of text to the terminal, the line edit write module (IHKLEW) is called if the TUBOUTAB field is nonzero. The IHKLEW module gets the address of the area containing the tabs from the TUBOUTAB field. The line is edited by substituting either the horizontal tab character and idle characters, or the horizontal tab character plus backspace characters and idle characters to replace strings of blank characters.

SCAN FUNCTION

The scan function can be invoked while lines are being entered in input mode, or it can be invoked after the lines are entered, which is referred to as delayed scan mode. The scan function is invoked in input mode by specifying the SCAN operand

on the EDIT command or by entering the SCAN subcommand with only the ON operand specified. It is turned off by entering the SCAN subcommand with the OFF operand. Delayed scan mode is invoked by entering the SCAN subcommand with line numbers designating the range of lines to be scanned or by the SCAN subcommand with no operands.

Scanning lines in input mode means the IHKIRL module accepts lines from the terminal as long as there are any continuation lines (continuation character: -). The continuation character is removed and each line is put into the active file. When a line without a continuation character is detected or when maximum continuation is reached, the line is put into the active file and the line number range since the last non-continuation line is passed to the syntax checker interface (IHKSYN). A pointer to these line numbers is put in the TUBIRLSA field. Therefore, when scanning lines in input mode the syntax checker interface always assumes that it receives line numbers specifying a complete statement. If it is not a complete statement, an error message is returned.

The syntax checker interface retrieves the lines from the active file, chains them together, builds the parameter list, and enters the checker. The checker returns to the interface each time an error is found. The interface queues the error message and returns to the checker until the complete statement is scanned. When the complete statement is scanned, the interface returns to the IHKIRL module. If there were errors, the IHKIRL module then returns to the command analyzer. If there were no errors, the IHKIRL module prompts the user with the next line number, and input mode continues.

In the case of errors, the command analyzer sends the error messages to the user. The user has several choices of reply: null line, Implicit subcommand, CHANGE subcommand, or several Implicit subcommands followed by a null line. A null line means the lines with errors will not be corrected; input mode continues. The IHKCGN module processes any corrections made with the CHANGE subcommand. If the Implicit subcommand is entered, the IHKIRL module is given control from the command analyzer. The IHKIRL module checks to see if the line is in the range of lines being scanned. If so, the line replaces a line in the active file, and the TUBCORCN bit in the TUB is set to indicate that a correction was made. The CHANGE subcommand processor makes corrections in the same way as the Implicit subcommand processor. The terminal user indicates by a null line that he has finished his corrections. The interface is then entered again and the process continues.

When the SCAN subcommand is entered with line numbers specified (or without any operands, meaning that the entire active file is to be scanned), the SCAN subcommand processor (IHKSCN) checks the CCT for the presence of either the PL/1 or FORTRAN syntax checker. If the proper syntax checker is not available, an error message is queued for the user, and control is returned to the command analyzer. If the syntax checker is available, the line number operands given on the subcommand are moved to the user buffer. The IHKNUM routine checks the line numbers for all numerics. The IHKSCN module moves the line numbers back to the PPT and puts a pointer to them in the TUBIRLSA field. The IHKRNQ module is called to queue the SCAN processor for the syntax checker interface.

For the FORTRAN syntax checker the interface reads a line ahead to check for continuation or for a comment; then it chains the lines together, builds the parameter list, and branches to the checker. For PL/1 one line at a time is sent to the checker. If the last line is not the end of a statement, the checker returns and the interface chains another line and enters the checker again. If the maximum number of continuation lines is reached before the end of the statement, the interface queues an error message for the user and the lines collected are sent to the checker.

When an error is encountered, the checker returns and the interface queues the error message for the user and returns to the checker until the complete statement group is scanned. If there are errors queued when the statement group has been completely scanned, the interface returns to the SCAN processor, which sends the messages to the user. The SCAN processor then goes back to the interface, which returns to the checker with the next lines to be scanned. The scan is finished when end-of-data is reached or the last line number has been scanned. The SCAN processor then returns to the command analyzer.

LIST FUNCTION

The LIST subcommand is used to display, at the user's terminal, certain lines or all the lines in the active file. (Chart G illustrates the list function, save function and scratch function.) If the line number operands are present, they are

checked to see that they are all numerics by the IHKNUM routine. An error causes a message to be queued and control to be returned to the command analyzer. If the line number operands are not present, the entire active file is to be listed. Otherwise only the line or range of lines specified is listed. (See Chart G.)

The line numbers are moved to a GETMAIN area and a switch is set in the first byte of the PPT if NUM is specified (or is the default). The LIST subcommand processor (IHKLST) issues AFIO macros and reads one line at a time from the active file. The line is moved to the user buffer, and a CWRITE macro is issued to send the line to the terminal. If more than one line is to be listed, the next line is read from the active file and sent to the terminal. This process continues until EOD is detected in the active file or until the next line number in the active file is greater than the specified last line number in the range. Control is returned to the command analyzer when the lines have been listed.

SAVE FUNCTION

The SAVE subcommand is the only means the user has of keeping his active file. The active file is stored in his user library. The IHKSAV module checks the operands on the subcommand. If an error is found, a message is queued for the user and control is returned to the command analyzer. If the OLD keyword is specified on the EDIT command, and if either the data set name is not specified on the SAVE subcommand, or if the data set name on the SAVE subcommand is the same as the name on the EDIT command, the data set in the library is replaced by the one in the active file. If the NEW keyword is specified on the EDIT command or if the data set name on the SAVE subcommand is different from the data set name on the EDIT command, a check is made for an existing CRJE data set with the same name. If a duplicate data set name is found, the user is prompted for a new data set name. If the user responds with a null line, then the active file replaces the existing data set in the user library.

AFIO macros are issued to read a block of records from the active file. Another macro is issued to write the block of records into the user library. If there is not enough space in the user library, it is condensed. An attempt is made again to write the block of records. If there is still not enough space available, a message is sent to the user indicating that his library is full and requesting the name of a data set to be deleted. The user may respond with a data set name or a null line. A null line terminates the save process. (See Figure 5.)

If a data set name is entered, it is checked for validity. If invalid, the user is notified and is asked to enter another name. If the data set name is valid, it is deleted from the user library. Then the user is again prompted for the name of a data set to be deleted. This process of deleting data sets is continued until the user enters a null line. The user's library is then condensed again if at least one data set was deleted, and an attempt is made to write the block of records into his library. If there is still no room, the complete process is repeated until space is available or until the user terminates the process by entering a null line when first prompted.

When space is available, the block of records is written in the user library. A block of records is read and written until EOD is encountered in the active file. Return is to the command analyzer.

SCRATCH FUNCTION

The END subcommand terminates creating and updating operations on the active file and the active file is deleted. The END subcommand processor (IHKEND) checks the TUBDIRAD field to determine whether main storage has been allocated for a directory entry. If main storage has been allocated, then the area is freed and the field is set to zero. A RELEASE macro is issued for the active file and the following switches are set to zero: TUBEXIT, TUBSCN, TUBFOR, TUBPL1, TUBLNUMN, and TUBLONGN. Control is then returned to the command analyzer. (See Chart G).

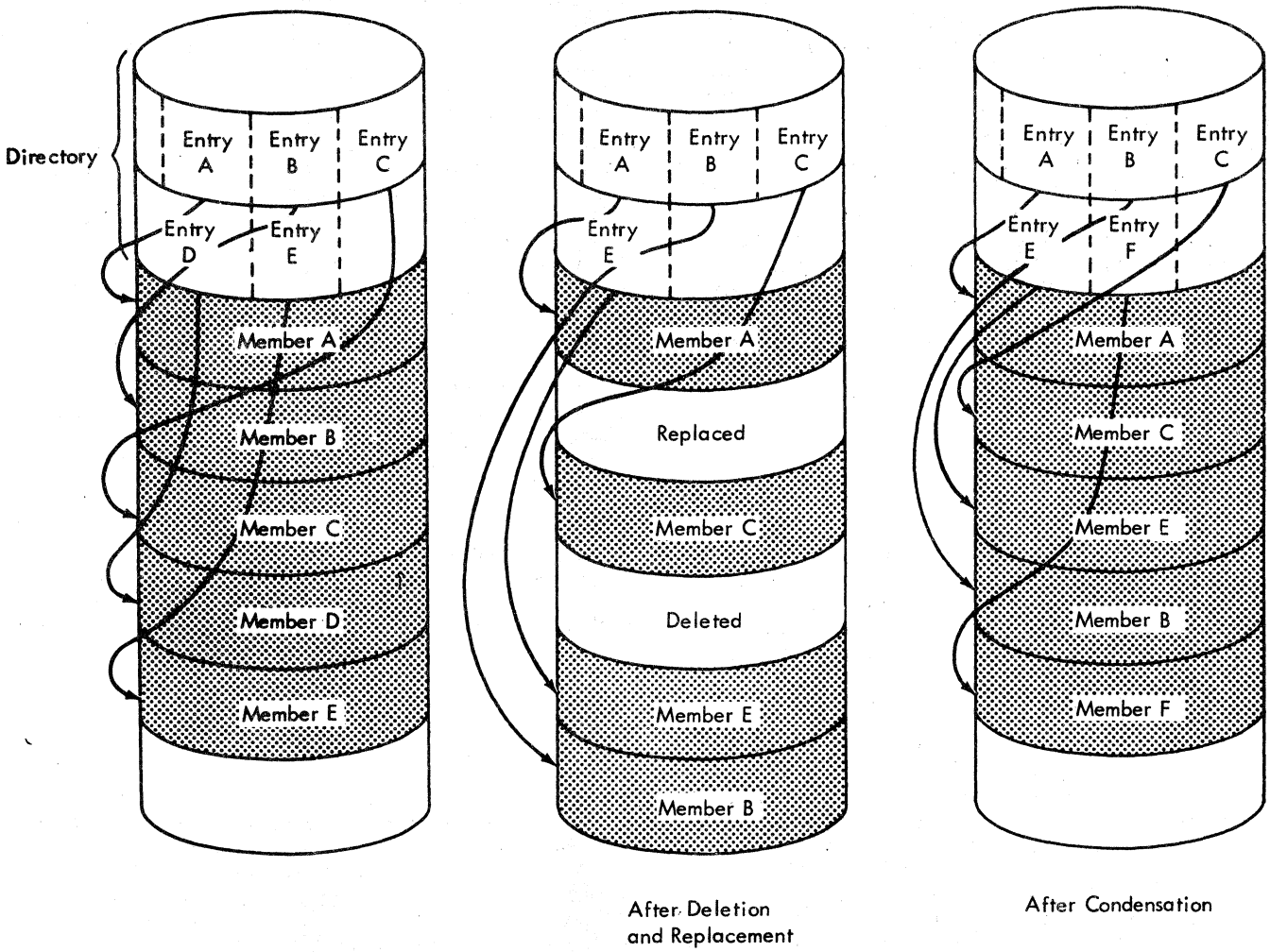


Figure 5. Library Condensation

JOB MANAGEMENT

SUBMISSION OF JOBS

The SUBMIT command allows users to submit jobs to OS for processing. The SUBMIT command processor allows up to ten data set names (plus dsnames found in DSLIST files) to be combined into a sequential data set and entered into the OS job input stream. If more than ten data set names are specified, only the first ten are used. The data sets specified must be CRJE data sets. If the user is in EDIT mode, his active file may be entered in the input stream. The data sets are put into the input stream in the order they are listed. Together they may make up one job or several jobs.

Chart H illustrates the functional aspect of job submission.

The SUBMIT input record processor (IHKGET) and the allocate module (IHKALC) perform special services for the SUBMIT processor. The IHKGET module syntax checks the dsnames submitted and positions the file to the beginning of these data sets. The IHKALC module allocates the sequential reader/interpreter input data set and builds the START RDRCRJE command.

The IHKSUB module inspects all the records that are to be submitted for JOB statements and DD data statements. All JCL statements are passed to the JCL exit, if it exists. The JCL exit can request that the SUBMIT processor accept the statement, or the exit routine can request that the SUBMIT be aborted with or without a message to the user. The inspection of JCL is suspended while DD data input is being processed.

When the first JOB card is found, the IHKALC module is called to allocate the reader/interpreter data set.

All records except the JOB statement are blocked when they are written to the reader/interpreter data set. This prevents a loss of data if the reader (RDRCRJE) is rolled out in MFT. After the JOB card has been written, the TTR of the statement is saved for abnormal processing.

When an instream procedure is encountered, the reader is rolled out in MFT. The rollout causes the record containing the instream procedure to be repeatedly processed "n" times, where "n" is the number of records remaining in the block to be processed. For example, if the instream procedure is the third record in the block of data to be processed (the first two records of the block having been processed already), the reader will be rolled out, and the record will be processed a total of eight times, representing the processing of records three through ten. To avoid unpredictable results, instream procedures should not be used when submitting input under MFT.

When a complete job containing an EXEC statement has been collected, the RJCT entry for the job is written in the JBTBLS global file. If no EXEC statement was received for the job, the SUBMIT is aborted.

If the SUBMIT is to be aborted because of a bad return code from the IHKGET routine, the error message returned by the IHKGET module is sent to the user, and the IHKGET module is deleted. If the abort is for any other reason, the IHKGET module is called to force end-of-input and the RJCT base register is cleared.

In a normal SUBMIT termination the RJCT base is tested. If it is nonzero, the RJCT entry for the last job is written. If no jobs were collected, a message is sent to the user informing him that no jobs were found. The CRJE service routine (IHKSRV) is called to close the reader/interpreter data set DCB and to start the reader procedure (RDRCRJE) on the data set. Then the save area and work area are freed before control is returned to the command analyzer (IHKCMD).

NOTIFICATION OF JOB OUTPUT

At start-up time the IHKSDQ module passes control to the IHKDEQ module to look for work on the CRJE SYSOUT queue. The IHKDEQ module posts the IHKSRV module with a request to invoke the IEFQMDQ2 routine, which attempts to dequeue a job from the CRJE SYSOUT queue. If there are no jobs on the queue, the OS queue manager keeps an ECB, which it will post when an entry is queued on the SYSOUT queue. The IHKDEQ module then returns control to the IHKSDQ module, which sets up the appropriate ECB address and return address for waiting in the dispatcher. The IHKSDQ module then waits for the ECB given to the OS queue manager (IEFQMDQ2) by the IHKDEQ module.

If the IHKDEQ module finds work on the SYSOUT queue, regular job end processing is performed. This is explained later in this section.

When OS executes a job entered through CRJE, an entry is queued on the CRJE SYSOUT queue, if the user submitting the job

specified the CRJE SYSOUT class. If, however, the user specified a SYSOUT class other than CRJE, no entry is made in the CRJE SYSOUT class. When a job is queued by OS on the CRJE SYSOUT queue, the ECB that the IHKDEQ module gave to the OS queue manager gets posted. This gives control to the IHKSDQ module, which in turn invokes the job end processor (IHKDEQ). (See Chart I.)

The job end processor first sets a flag to indicate to the closedown module that job termination is in progress. The IHKSRV module is posted with a request to invoke the IEFQMDQ2 routine, which will attempt to dequeue a job from the CRJE SYSOUT queue. If a job is dequeued by the IEFQMDQ2 routine, the IHKAFI module is used to search for the RJCT entry with the jobname of the job that was dequeued. If the RJCT entry is not found, the jobname is checked for JOBFAIL (this is the jobname that OS returns after encountering JCL errors on the JOB statement). If it is not, the job does not belong to CRJE. Control is given to the second entry point (IHKRER01) of the IHKRER module to delete the job. The OS queue manager read routine (IEFQMSSS) reads the SMB/DSBs for the job. The data set scratch routine (IHKRER02) scratches the associated data sets. The OS queue manager delete routine (IEFQDELE) deletes the OS queue space. When this is completed, the job end processor issues another dequeue request to the IHKSRV module for the next job on the SYSOUT queue.

If the RJCT entry is not found but the jobname is JOBFAIL, IHKDEQ scans the SMBs for the original jobname. If the jobname is found, search is made for the RJCT entry. If the jobname is not found, the job is deleted. For this reason, the user is cautioned against the use of JOBFAIL as a jobname.

If the RJCT entry is found, it is checked to see if the job is marked complete. If it is already marked complete (the job is a duplicate), the delete job routine (IHKRER01) of the IHKRER module is called to delete this job. A dequeue request is then issued for the next job on the CRJE SYSOUT queue.

If the RJCT entry is not already marked complete, it is now marked complete and a notification message is sent to the user who submitted the job. The necessary information for locating the output of the job is placed in the RJCT entry, and the entry is written in the global file. Then another dequeue request is issued for the next job on the CRJE SYSOUT queue.

When there are no more jobs on the CRJE SYSOUT queue, control is returned to the IHKDEQ module, but the OS queue manager dequeue routine keeps the ECB, which will be posted when OS queues a job on the CRJE SYSOUT queue.

RETRIEVAL OF JOB OUTPUT

The OUTPUT command allows the user to obtain output of jobs he has submitted. Only OS data sets in the CRJE SYSOUT class and job management messages, if MSGCLASS specified the CRJE SYSOUT class, are available. After the syntax of the command is checked and the operands are verified, the RJCT entry for the job is read into main storage. The first SMB or DSB for the job is read from the CRJE SYSOUT class. If an SMB was read and SMSG was specified on the command, the job management messages are sent to the terminal user. When the end of an SMB is reached, the next SMB or DSB is read and processed until all for this job are processed. (See Chart J.)

If a DSB is read, it contains the TIOT entry for the DD statement. This TIOT is used by the OUTPUT processor to create a new TIOT referred to by the OPEN, CHECK, and CLOSE macros to provide access to the SYSOUT data set. BSAM is used to read the SYSOUT data. The OUTPUT processor deblocks the data and transmits it to the remote terminal, one logical record at a time. After all the data has been transmitted, it is scratched. The next SMB or DSB is read and processed until there are no more for this job.

When all the SMBs or DSBs are read and processed, the IEFQDELE routine deletes the entry on the SYSOUT queue. The IHKAFI module is then used to delete the RJCT entry for this job.

The TTR of the current block being transmitted to the terminal is saved by the OUTPUT processor when an unsuccessful write to the line is detected. This permits resumption of processing from that point after a discontinue/continue sequence.

When a CONTINUE command is entered at the terminal, the same process as for the OUTPUT command is performed. If a record is a scratched data set, then it has already been read and the TTR of the last block read is reinitialized. The next data set is then read and processed.

CANCELING A JOB

When a CANCEL command is entered, the IHKRER module processes the command. The RJCT entry is found and read to verify the existence of the job to be canceled. A check is then made to be sure that the user canceling the job is the user who submitted the job. If the job is not found or the requesting user did not submit the job, an error message is sent to the user informing him of the situation.

If the job is not complete, an OS CANCEL command is issued for the job. The job delete bit in the RJCT entry is turned on and the RJCT entry is written in the global file. The OS CANCEL command causes the job to terminate abnormally. The IHKDEQ module gets control when the job terminates. The dequeue routine, after checking the job delete bit, passes control to the IHKRER01 entry point where the job is deleted. The IHKDEQ routine sends a message to the user indicating that the job was canceled.

If the job is complete, the associated data sets are scratched, the OS queue space is deleted, and the RJCT entry is deleted.

SYSTEM INQUIRY

There are commands available to both the terminal user and the central operator for obtaining information about the CRJE system. The terminal user may obtain information about all the data sets in his user library by the LISTLIB command. Information about one particular data set in his library is obtained by the LISTDS command. The terminal user can keep track of his jobs by the STATUS command.

The SHOW command allows the central operator to have information about the CRJE system printed at his console. The following information can be requested, but only one type at a time:

- A list of all CRJE jobs and their status (whether the job is complete)
- Status of a particular job
- A list of all valid CRJE users with an indication of whether the user is active, the user's line address, and the time each active user has been logged on
- An indication of whether a particular user is active
- A list of currently active users and the amount of time each has been active and each user's line address
- A display of the number of currently active users
- A list of the current values of all line error accumulators for all lines being used by CRJE
- A list of error and transmission counts for a specified line
- Notification as users log on and off
- Notification when a specific user logs on and off
- A list of all broadcast messages
- A list of the delayed messages for all users or a specific user

DATA SETS

A terminal user issues a LISTLIB command to get a list of all the CRJE data sets in his library and the attributes of each. If STATUS is specified on the command, the size of each data set and whether it is protected is included in the list. If HISTORY is specified as an operand, the following information is given about each data set: creation date (first time saved), date last modified (last time saved), and the number of times accessed since created. All of this information may be obtained in the same manner for a particular data set by the LISTDS command.

The LISTDS command processor (IHKLDS) handles both the LISTLIB command and the LISTDS command. The processor first determines which command was entered, checks the operands, and sets the appropriate flags in the PPT. Library I/O is used to read lines of the data set if the command is LISTDS, or to read a block of directory entries if the command is LISTLIB. A file is created in the active area, and a header message is obtained from the message writer and inserted at the beginning of the active file. Information is obtained from the directory entry and inserted into the user buffer. The information in the user buffer is aligned according to the labels of the header. The operand flags are tested to determine exactly what information was requested. Any unnecessary information is blanked out. Then the line of information is inserted into the active file.

When the information has been inserted into the active file for the requested data sets, control is passed to the LIST subcommand processor (IHKLST). The IHKLST module lists the contents of the active file at the terminal and then releases the active file.

The central operator cannot request through CRJE any information about a data set in a user library.

JOBS

To obtain information about jobs he has submitted, the terminal user issues the STATUS command. The user may request the status of one job by specifying the jobname on the command, or he may request the status of all the jobs he has submitted by issuing the command with no operands. The status information given is whether a job is scheduled for execution, is currently being executed, has not been executed, or is not in the system. If a job is scheduled for execution, its position on the job queue is given.

The STATUS command processor checks the operand of the command. If a jobname is specified, the RJCT entries are checked to find out if the job is in the system. If a jobname is not specified, the RJCT entries for all jobs submitted by the user are checked. If a job is marked complete, a job complete message with a normal or abnormal indication is returned to the user.

Otherwise the IEFLOCDQ routine checks the OS job queue. If a job is found belonging to the user, its position on the queue is returned. Otherwise the CSCBs are searched to determine whether a job is executing. If it is, the appropriate message is sent to the user. Otherwise the RJCT entry is checked again to determine whether a job is complete.

The central operator uses the SHOW command to request certain information about jobs in the system. He can request a list of all CRJE jobs with an indication of whether the jobs are complete. He can also use the SHOW command to inquire about a particular job.

MESSAGES

The central operator communicates with terminal users by sending messages, and users communicate with the central operator in the same way. Terminal users can communicate with each other by sending messages.

The SEND command directs a message from one terminal user to another or to the central operator. The central operator uses the MSG command to send messages to terminal users. The central operator also communicates with terminal users through broadcast messages. Broadcast messages contain information that is of interest to all terminal users but they are only sent to a user upon request.

TERMINAL USER MESSAGES

The message writer handles most messages to terminal users. The message writer sends a message that is already built or builds one, using information that the processor provides and previously-prepared text. The processor requesting the message sends the message writer all needed information in parameter lists. The message writer gets the previously-prepared text from the SYSMSGS global file. The message may be sent to the user immediately or queued for later delivery.

Most command processors branch to the message writer to queue messages for the user. When the system administrator gets control back from a processor, it branches to the message writer to send all messages that are queued for the user. There are exceptions, however. Some processors interface directly with the line administrator in sending information or messages to the terminal. In input mode the IHKIRL module prompts the user with line numbers by issuing a CWRITE R macro. The LIST processor builds a file and sends it directly to the terminal. If the line administrator encounters any errors in sending information to the terminal, it has messages within its own module that it can send to the user.

Only one message for each user can be queued in main storage. The second message to be queued is kept in the USRMSGS global file. The messages are kept in the order in which they are issued for each user. The TUBMSG bit in the TUBFLG1 field is turned on to indicate there are messages in the USRMSGS global file for this user. When the system administrator branches to the message writer to send all messages, the message queued in main storage as well as all the messages in the USRMSGS global file for this user are sent.

If the user is inactive when messages are queued for him, all the messages become delayed messages and are written in the USRMSGS global file. In this case the UVRDMSG bit in the UVRCNTL1 field of his user verification record is turned on. When the user logs on, he receives all of these delayed messages.

The user may request that the message ID be attached to the message when it is printed at his terminal. To get the message IDs, the user must specify MSGID on his LOGON command.

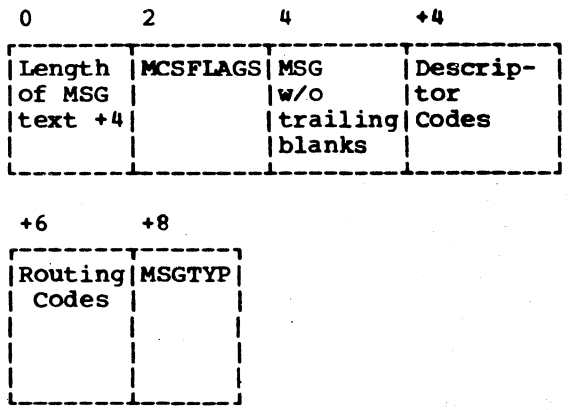
A message that a user sends to another user with a SEND command is never queued in main storage. Even if no messages are queued for the user, the message is written

in the USRMSGS global file and the TUBMSG bit is turned on.

CENTRAL OPERATOR MESSAGES

Messages are built and sent to the central operator, or a supplied message is sent to the central operator. A message for the central operator is never queued by CRJE. The execute form of the WTO macro is used to print messages at a console.

The required format of a message area referred to by the execute form of the WTO macro is as follows:



MCS requires the MCSFLAGS field, the descriptor code, and the routing code for each message. The descriptor code and routing code are contained in bytes 75-78 of the message area in the SYSMSGS global file. The MCSFLAGS field is inserted by the message writer when the message is built and sent. The MSGTYP field is added by the message writer for messages to be sent to consoles that have issued the general SHOW SESS command. System messages may have a maximum length of 72 bytes. After the length of the text is determined, the descriptor code and routing code are moved to the area immediately following the message text. For more detailed information on the requirements of MCS, see the section Multiple Console Support Interface.

The central operator can issue broadcast messages to provide users with system information. If CRJE is not active, he can put the messages in the BRDCST member of the system library by a utility program. When CRJE is active, he can issue the BRDCST command. By this command the central operator can add or delete broadcast messages in the BRDCST global file.

Every terminal user has access to the broadcast messages, but the messages are sent only upon request. When the user logs on, if he does not specify NOBC, all the broadcast messages are listed at his terminal after his delayed messages. At any time (except in edit mode) the user may request a listing of the broadcast messages. The LISTBC command is used for this purpose. This command causes the broadcast message bit in the TUB to be turned on by the system administrator.

The next time the system administrator branches to the message writer to send all available messages for this user, the message writer recognizes that the bit is on and all the broadcast messages are sent to the user.

INTERFACES

OS READER/INTERPRETER INTERFACE

The OS reader/interpreter interface involves building a reader procedure and cataloging it in the SYS1.PROCLIB. The procedure must be named RDRCRJE and should be the same as the installation's standard unblocked reader procedure with three exceptions. These exceptions and more detailed information about the procedure are given in the CRJE System Programmer's Guide, GC30-2016.

When a SUBMIT command is entered, the allocate routine issues a GETMAIN macro for 104 bytes and builds the START RDRCRJE command. The address of the area containing the command is put in the TUBIRLSA field. Space is obtained and the input job stream is built. The IHKSRV module then issues an SVC 34 if no reader is currently active. If a reader is active, a STIMER macro is issued for two seconds. After two seconds a check is again made for a currently active reader.

CENTRAL COMMAND INTERFACE

When an RJE or CRJE central command is entered through the central console or the card reader, it is recognized and SVC 34 gives control to the proper job management routine. This routine builds a command input buffer (CIB), which is chained to either the RJE or CRJE command queue.

The IHKCCI routine provides the interface between the RJE/CRJE scheduling routine and the command processors. The

scheduling routine (IGC1503D or IGC0703D) posts the communications ECB in the CRJE CSCB (command scheduling control block). This causes the CRJE dispatcher to give control to the interface routine (IHKCCI). The address of the CIB, which contains the command, is in the SPL (start parameter list) and the address of the SPL is in the AVT. The command and its operands, if necessary, are examined to determine which processor is needed.

Since all the processors are nonresident, a load request for the processor needed is passed to the loader/controller. When the processor is loaded, control is passed to it with register 1 pointing to a two-word parameter list containing the address of the AVT and the address of the CIB. When the processor is finished and control is returned to the interface routine, the loader/controller deletes the processor and the interface routine dequeues the CIB and returns control to the dispatcher.

SYNTAX CHECKER INTERFACE

Lines entered in input mode are put into the active area by the command processor. At the end of a statement, indicated by a carrier return (CR) with no hyphen at the end of a line, the command processor enters the interface, passing to it the beginning and ending line numbers of the group. The SCAN subcommand processor passes the beginning and ending line numbers of the requested scan to the interface. Both processors use the same entry to the interface. In the TUB is the address of the work area containing the beginning and ending line numbers.

Continuation lines for both PL/1 and FORTRAN statements in input mode are indicated by a -CR at the end of the line. The line number range passed to the interface in input mode represents one statement. For PL/1 input mode, the lines in the range are read one at a time and chained until the end of data or maximum continuation is reached; the group of lines is then passed to the checker to be scanned. For FORTRAN input mode, the next line is read and kept, while the current line is passed to the checker with the return-12-expected bit set. For standard FORTRAN, the checker always returns a code of 12 when this bit is set, indicating that another line is to be passed. The current line is freed, the next line already read is set up, and the succeeding line is read. When the end of data is reached, the return-12-expected bit is not set, and the current line is passed to the checker. The group collected by the checker is then scanned.

The line number range passed to the interface in delayed scan mode may be more than one statement. For PL/1, the lines are read one at a time, chained to the previous line if there is one, and passed to the checker with the return-12-expected bit set. If the last line is not the end of a statement, the checker returns a code of 12, indicating which, if any, lines in the group may be freed. When the end of a statement is found, or when maximum continuation or the end of data is reached, the lines are scanned. On maximum continuation, the interface queues an error message.

For FORTRAN, the interface reads a line ahead, checking for a "C" in column 1 to indicate a comment, or a nonblank and nonzero character in column 6 to indicate continuation. For a group of comments, or several within a continuation group, all but the first are are freed immediately. When the next line is a comment or continuation, the return-12-expected bit is set and the current line is passed to the checker. On return 12, the line is freed, the next line already read is set up, and the succeeding line is read. When the next line is not a comment or a continuation, or when the end of data is reached, the return-12-expected bit is turned off and the current line is passed. The collected group is then scanned.

Both syntax checkers return to the interface on errors and may be reentered to continue scanning the statement. The interface queues all error messages until the statement is completely scanned, then returns to the processor to have the messages sent to the user. The interface returns to the SCAN subcommand processor only when there are messages to be sent to the user and at the end of the scan. In input mode the terminal user may make corrections and cause the lines to be sent back to the checker. Lines entered with the Implicit subcommand are not scanned, except when corrections are being made.

MULTIPLE CONSOLE SUPPORT (MCS) INTERFACE

The message writer handles most of the functions necessary to support multiple consoles, which involves specifying descriptor codes and routing codes for all messages sent to the central operator. Routing codes define the type of console and descriptor codes specify the type of message. These codes are determined in advance for each prepared CRJE message and

are contained in bytes 75-78 of the message area in the system library.

Interfacing with MCS involves determining whether MCS is in the system. The CVTUCM field of the communications vector table (CVT) points to the UCM (unit control module) base. The UCMMCS bit in the UCMMODE field of the UCM base is on if MCS is in the system. The UCM base is followed in main storage by all the UCM entries. A UCM entry exists for each console. Composite consoles (separate input and output devices) have two UCM entries; the first UCM entry is for the input device and points to the second entry which is for the output device. Each UCM entry contains the console ID in the UCMID field. The following entries in the UCM base are used to find the UCM entry for a specific console:

UCMVEA - address of first UCM entry,
UCMVEZ - size of UCM entries,
UCMVEL - address of last UCM entry.

Most CRJE central messages are sent as a response to central commands. Therefore, the message goes to the console from which the command was issued. The module calling the message writer indicates the requesting console by storing the console ID in the high order byte of the word of binary zeros pointed to by the second parameter and by setting the high order byte of the message number to X'20'. The message writer then sets the routing code to zero, the MCSFLAGS field to X'4000', and loads the requesting console ID in register 0 (right-justified).

When delayed or broadcast messages are displayed on the console, the IHKCC4 module (SHOW BRDCST, SHOW MSGS, and MSG D=userid Central Command Processor) stores the descriptor code (X'0800') and the routing code (zeros) at the end of the message text, sets the MCSFLAGS field to X'4000', and passes the console ID in the high-order byte of the word of zeros to the IHKMSG02 entry point of the message writer for delayed messages or to the IHKMSG03 entry point for broadcast messages.

The terminal user may send a message to a specific console by specifying the routing code (1-16). When this number is passed to the message writer, it is in binary and the message writer converts it so that the correct bit gets turned on. The message writer also sets the MCSFLAGS field to X'8000'. If a routing code is specified for a non-existent console, the message goes to the master console. If the message goes to all consoles that have issued a general SHOW SESS, the module calling the message writer sets the high-order byte of the message number to X'08'. The message writer sets the MCSFLAGS field to X'1000' and adds the MSGTYP field (X'0800') to the message area. Multiple console support does the processing to send this message to all requesting consoles. In all other cases the message writer uses the routing codes stored with the message and sets the MCSFLAGS field to X'8000'.

The descriptor code for responses to central commands is the immediate command response (ICR) with a code of X'0800'. The only other descriptor code used by CRJE is system status (SS) with a code of X'1000'. The system status code indicates status such as disk error or out of main storage.

CRJE SYSTEM ERROR PROCEDURES

CLASSIFICATION

Certain recognized errors, from which the system cannot recover, occur in CRJE. Other errors cause error recovery procedures to be followed, which help the system to recover. These errors are listed below along with the results:

- GETMAIN failure - requires the terminal user to reenter a command.
- Active area out of space - results in notification to the terminal user and the central operator, and termination of processing of the last command entered.
- User library out of space - results in notification to the terminal user, and in addition, the user is prompted for data sets to be deleted from his library.
- User library I/O errors - result in making the user library as inoperative and notifying the terminal user and central operator of the error condition. No other attempt is made to read the library during this CRJE session.
- Active area I/O errors - result in the termination of CRJE; the active users and central operator are notified of the shutdown.
- Communications line errors - result in BTAM error recovery including text-read error retry (EROPT=RWC); failure to recover results in notifying the central operator and reinitializing the line.

- Start-up errors - result in termination of CRJE.
- Shutdown errors - result in recording these errors at the central operator's console.

RECOVERY

Following are the error handling procedures for the errors listed above:

GETMAIN Failure

When a GETMAIN failure occurs in a line subtask, the command processor returns to the system administrator with a return code of 8. Files that are partially copied into the active area must be released. The system administrator notifies the user by sending an out-of-space message immediately.

If a GETMAIN failure occurs in the line administrator in trying to obtain a line save area, return is made to the system administrator with a nonzero return code. The system administrator will notify the central console and pass control to the CRJE dispatcher to wait on the line ECB until it is activated by the MODIFY central command.

If a GETMAIN failure occurs in the line administrator in trying to obtain a terminal user buffer, the line administrator notifies the terminal user, if possible, and returns to the system administrator. The system administrator tests for a CLB and sends a message to the central operator. Then control passes to IHKDSP to wait on the line until it is activated. If a GETMAIN failure occurs in the central command subtask, no procedure is required. All required buffers, work areas, and save areas are contained within the load module.

Active Area Out of Space

If this error occurs in a line subtask, a return code of 20 is sent to the system administrator. The system administrator then notifies the central operator. The command processor releases private files and queues a notification message for the terminal user.

If the error occurs in the central command subtask, the central operator is notified. Any required messages pertaining to the command are sent by the message writer.

The following procedure is attempted by the message writer when it encounters an out-of-space condition while attempting to write delayed messages:

- The mesage is lost.
- For an active user the message lost bit (TUBLMSG) in the TUB is set.
- For an inactive user the message lost bit in the user verification record is set.
- A MESSAGE LOST message is sent to the user if the message lost bit in the TUB is set; then the bit is reset.

  Note: The LOGON processor must set the message lost bit in the TUB if the message lost bit in the UVR is set. The LOGOFF processor must set the UVR message lost bit if the TUB message lost bit is set.
- A return code of 20 is sent to the calling routine.

User Library Out of Space

This error is recognized by the SAVE subcommand processor. When the error occurs, the processor prompts the user for data sets to be deleted from his library. After several data sets are deleted the user library is condensed to provide more space, and another attempt is made to save the active data set.

User Library I/O Errors

These errors are encountered by the library I/O modules. The following procedures define the error processing required of the modules that use the library I/O facilities:

- The IHKUTM module is called to mark the UVR, indicating the user library is inoperative (UVRLITIN).
- BTAM start-up (IHKBST) resets this indication at start-up.
- The IHKUTM module returns a code of 24 if the library is marked inoperative following queuing requests (X'80').

- A LIBRARY INOPERATIVE message must be queued for the terminal user.
- A message must also be sent to the central operator notifying him which library is inoperative.

Active Area I/O Errors

When one of these errors occurs in a line subtask, a return code of 12 is sent to the system administrator by the command processor. The system administrator sets the abnormal stop bit (CCTATERM) in the CCT if it is not already set. The suppress LOGON bit in the CCT is also set. A STOP CRJE command is executed internally. A message is sent to the central operator notifying him of the abnormal closedown. A similar message is also sent to the terminal user. The user is logged off. The LOGOFF processor bypasses any calls to the IHKUTM module. A CREAD R macro is issued to reset the line and free the terminal user buffer. The stop acknowledgment ECB in the CLB (CLBSAECB) is posted. The line subtask waits for the line ECB. If the abnormal stop bit in the CCT is set, the line administrator returns the error code to the processor.

When an active area I/O error occurs in the central command subtask or the job terminator subtask, the abnormal stop bit in the CCT is set, and additional central commands are rejected.

Communications Line Errors

All line errors are returned to the system administrator for the appropriate action. The system administrator notifies the central operator and executes a SAVE command for the active file if the subtask is in edit mode. An END subcommand is then executed to get out of edit mode. A LOGOFF command is also executed. If the line can be recovered, a CREAD I macro is issued to free the TUB and the line buffer and to reinitialize the line; otherwise a CREAD R macro is issued to free the TUB, line buffer, and line save area and to flag the line as inoperative.

Start-up Errors

Errors encountered during start-up terminate CRJE with the central operator being notified whenever possible.

Shutdown Errors

The system library may be out of space if the library allocation is not consistent with the variable system parameters specified at CRJE assembly time. If this occurs, a message is sent to the central operator and the shutdown process continues.

If an I/O error occurs in the active area during shutdown, an error message is sent to the central operator.

CENTRAL OPERATOR CONTROL OF CRJE

The central operator initiates and terminates the operation of CRJE. By means of the central commands, the central operator has the capability of dynamically controlling the CRJE system. In controlling the CRJE system the central operator has the following capabilities:

Maintaining the broadcast messages;

Removing job output in the CRJE SYSOUT class and processing it with a central installation output writer;

Sending messages to CRJE terminal users;

Deleting delayed messages for a user not currently logged on the system;

Requesting a display of CRJE system information;

Controlling availability of the CRJE system to terminal users by adding or deleting authorized users, by suspending or resuming new sessions, and by activating or deactivating communication lines.

CRJE and RJE use the same scheduling routine, IGC1503D. When CRJE and RJE are in the system at the same time, a C must be specified on the central commands for CRJE. When a central command is recognized for CRJE or RJE, SVC 34 gives control to the scheduling routine. The IGC1503D scheduling routine determines whether the command is for CRJE or RJE. A 96-byte command input buffer (CIB) is then built and chained to the proper command buffer queue. If the central command is for CRJE, and CRJE is not active, an error message is sent to the central operator. After the command has been processed, the scheduling routine turns on the completion bit of the communications ECB in the CRJE CSCB. The dispatcher then gives control to the interface routine.

The IGC1503D routine does not process the MODIFY command. The CSCB marking routine of OS recognizes the MODIFY command and determines that it is for CRJE. The IGC0703D routine sets up the command input buffer (CIB) with the command and its operands. The IGC0703D routine then posts the ECB, thereby giving control to the IHKCCI routine.

The interface routine (IHKCCI) issues an EXTRACT macro to retrieve the address of the CIB (command input buffer). The IHKCCI routine then examines the command and its operands and determines which nonresident routine is needed to process the command. A load request for the appropriate routine is sent to the loader/controller. Control is then passed to the routine with register 1 pointing to a two-word parameter list. When the command is processed and control returns, the nonresident processor is deleted and the CIB is dequeued. The dispatcher then regains control.

There are eight nonresident processors for the central commands. The processors are composed according to size and similarity in processing rather than according to external function.

Following is a list of the CRJE central command processors with the external functions they perform and the estimated size of each module:

Module Name	Function	Size
IHKCC1	SHOW USERS SHOW JOBS	2K
IHKCC2	SHOW LERB SHOW BRDCST MODIFY	2K
IHKCC3	BRDCST	2K
IHKCC4	SHOW MSGS MSG D=userid	2K
IHKCC5	CENOUT	2K
IHKCC6	SHOW SESS SHOW SESSREL	2K
IHKCC7	USERID	2K
IHKCC8	MSG SHOW ACTIVE	2K

Each module is discussed in detail in the Program Organization section.

RESOURCE MANAGEMENT

RESIDENT AND NONRESIDENT MODULES

The nonresident modules are loaded into the transient area by the loader/controller when requested. The minimum size of the transient area is 8K. It can be increased at start-up time by multiples of 2K. The nonresident routines are grouped together in 2K load modules and the loader/controller loads routines into the transient area in terms of 2K. The number of modules necessary to make 2K are linkage edited together into one load module.

Every nonresident module has a 12-byte entry in the nonresident module table (IHKMOD). This table contains information such as whether the routine is reentrant or serially reusable; whether the routine is currently in the transient area; whether the module is a "trouble module," the number of routines in the load package; and the name and entry point address of the routine.

A "trouble module" is a module that requests the loading of another module before it is deleted itself, or a module that queues itself for library I/O or uses the IHKUTM module to be queued for library I/O. The number of "trouble modules" allowed in the transient area at one time is the number of 2K blocks assigned to the transient area minus one. The "trouble module count" in the first byte of the module table contains the number of "trouble modules" allowed in the transient area. Each time a "trouble module" is loaded, one is subtracted from the count; when a "trouble module" is deleted, one is added. If the count is zero, no more "trouble modules" can be loaded.

The block table contains an entry for each 2K block in the transient area. Each entry contains a block request count, which is increased for load requests and decreased for delete requests. Also, in each entry is the code of one of the routines currently occupying that 2K block.

The LOAD request that is sent to the loader/controller is for the particular module needed, not for the entire load module. When a load request is received by the loader/controller, a check is made to see if the module is already in the transient area. If it is already there but no LOAD macro has been issued for it, then a macro is issued to find the address of the entry point.

Before any modules can be loaded into the transient area, a block must be found with no requests for any of its modules; this load module is deleted. Then a 2K load module can be loaded into this block

of the transient area. The process of loading and deleting modules into and from the transient area is described more fully in the module description of the loader/controller.

CONTROL OF SERIALLY REUSABLE MODULES

Control of serially reusable modules depends upon whether the module is resident or nonresident or whether it is one of the resources handled by the queuing module (IHKRNQ). Control of the nonresident serially reusable modules is handled by the loader/controller. The librarian queue module (IHKRNQ) controls the following serially reusable resources: AFIO extended work area, user libraries, global files, job submission, and the syntax checker interface.

The loader/controller handles the control of serially reusable modules that are nonresident. A bit in the first byte of each entry in the module table indicates whether the module is reentrant or serially reusable. Another bit indicates whether the routine is locked (meaning someone is currently using it). When a load request is found for a routine already in the transient area, the flag is checked to see if the module is reentrant. If it i , the entry point address is inserted in the return ECB and this ECB is posted. If the module is serially reusable, the flag indicating whether the module is already in use is checked. If the module is already in use, the event count of the WAIT macro is incremented and the request is flagged as waiting for a module.

The librarian queue module handles the control of the user libraries, global files, syntax checker interface, and the AFIO extended work area by queuing requests for the resources as they are issued. Each queue has a queue control element associated with it. Each user (or line) also has an individual queue element associated with the queue control element. The queue control elements are in the KONBOX and all of the queue elements, except the ones for the syntax checker interface which are gotten dynamically from main storage, are in the TUB. When a resource is needed, control is passed to the IHKRNQ module with register 1 pointing to a parameter list containing the address of the queue element and the address of the queue control element for the resource desired.

The IHKRNQ module determines if the specified queue is empty. If it is empty, the queue element is placed as the first in the queue for that resource and control is returned to the caller. If the queue is not empty, the queue element provided by the caller is placed on the end of the queue and a call is made to the CRJE dispatcher to wait until the queue element for this request is posted. Once the queue element is posted, the dispatcher returns control to the IHKRNQ module. The IHKRNQ module then returns control to the calling routine. The queue module (IHKRNQ) never returns control to the calling routine until it is the first on the queue for the requested resource.

CLOSEDOWN

The process of closing down the CRJE system is essentially the same whether the closedown is normal or abnormal. Normal closedown occurs whenever the central operator enters a STOP command at the central console. Abnormal termination occurs whenever an error occurs from which the system cannot recover.

ABNORMAL CLOSEDOWN

Three types of errors cause abnormal closedown of the CRJE system: (1) start-up errors, (2) active area I/O errors, and (3) subtask abends.

The central operator is notified, if possible, when a start-up error occurs.

Active users and the central operator receive notification when an active area I/O error occurs. The command processor sends a return code of 12 to the system administrator. The system administrator sets the stop and abnormal bits in the CCT if they are not already on. The suppress LOGON bit in the CCT is also turned on. CRJE is then stopped by executing an internal STOP command. From this point on, the process of abnormal closedown is the same as for normal closedown.

Subtask abends are reported to the central operator when they occur. For the service and loader/controller tasks, the STAE exits for these tasks post all outstanding requests and set the stop bits for the main task. The subtask abnormal termination messages are written to the the central operator by the closedown routine (IHKCLN) for the main task.

NORMAL CLOSEDOWN

The OS command scheduling routine of SVC 34 recognizes a STOP command from the central console and turns on the stop bit in the ECB of the CSCB (command scheduling control block) for the CRJE task. The IGC0703D module checks the stop bit, processes the command, and turns on the completion bit of the ECB. When the IHKCCI module discovers that the stop bit is on, the CRJE stop routine is loaded and control is passed to this module (IHKSTP).

The stop routine sets the stop bit in the CCT and scans the STCBs for the lines. If the stop acknowledgment ECB in the CLB is not posted, the system administrator forces a LOGOFF for the user, does general cleanup for the line, and posts the stop acknowledgment ECB. When all the stop acknowledgment ECBs are posted, the global files in the active area are written to the CRJE system library by the library I/O shutdown module (IHKBSH). The stop ECB for the job terminator is posted, indicating that STOP processing is complete and that the job termination module (IHKSDQ) can return to the START command processor (IHKBGN). Control is then returned to the IHKCCI module, which waits in the dispatcher, thereby giving the IHKSDQ module control.

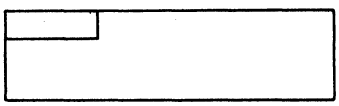
Chart K illustrates the functional process of closing down the CRJE system.

The IHKSDQ module checks whether a STOP command is pending. If it is, the module waits for its stop ECB to be posted. it returns to the START command processor (IHKBGN), which in turn passes control to the closedown routine. The closedown module (IHKCLN) informs the attached tasks of the closedown, unchains the job terminator ECB from the OS queue manager, issues a DELETE macro for certain OS routines, closes all the lines, detaches the attached tasks, notifies the central operator of the closedown, and returns control to the START command processor (IHKBGN).

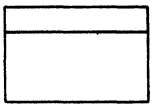
The START command processor (IHKBGN) issues a DEQ macro to indicate that CRJE is no longer an active task. Control is then returned to the OS system control program.

LEGEND FOR METHOD OF OPERATION CHARTS:

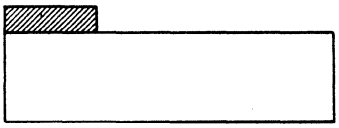
LEGEND for Method of Operation Charts



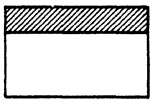
Main CRJE processing routine



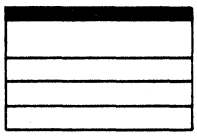
Secondary CRJE routine



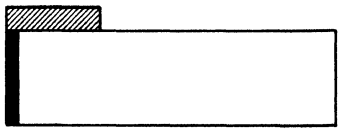
Main OS processing routine



Secondary OS routine



CRJE Control Block



OS Control Block

Control Flow

Data Reference or Movement

Chart A. Start-Up and Initialization

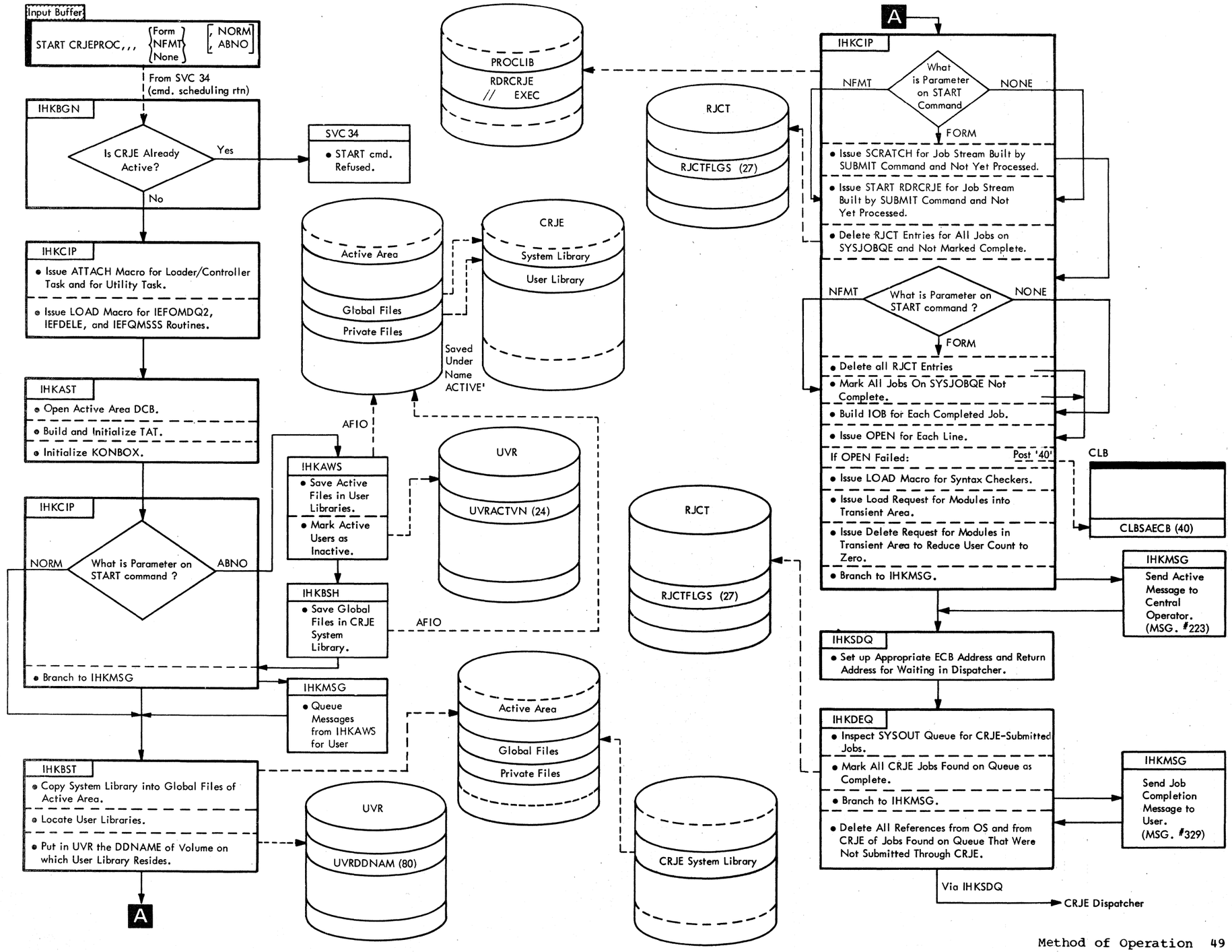


Chart B. Session Management (Part 1 of 2)

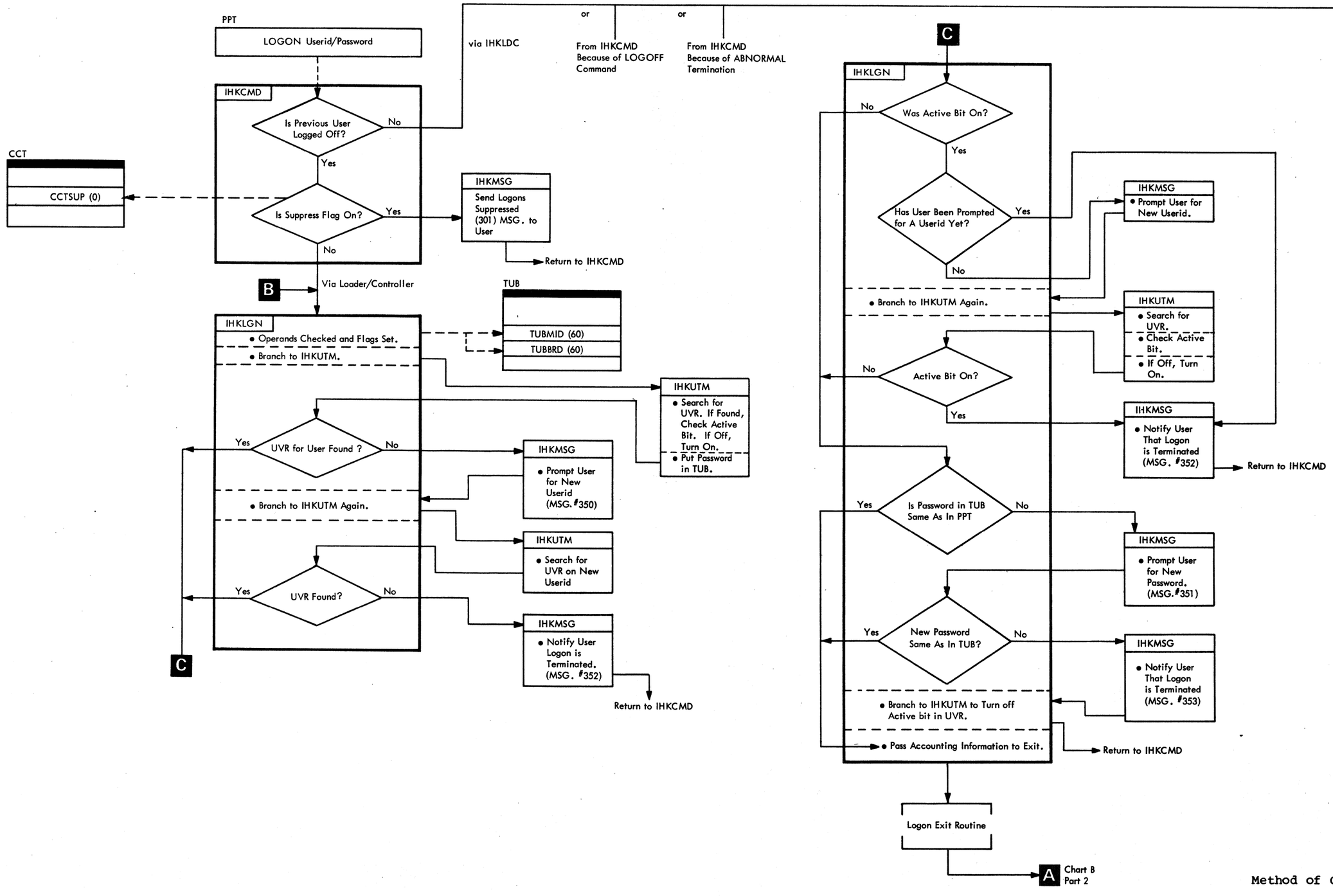


Chart B. Session Management (Part 2 of 2)

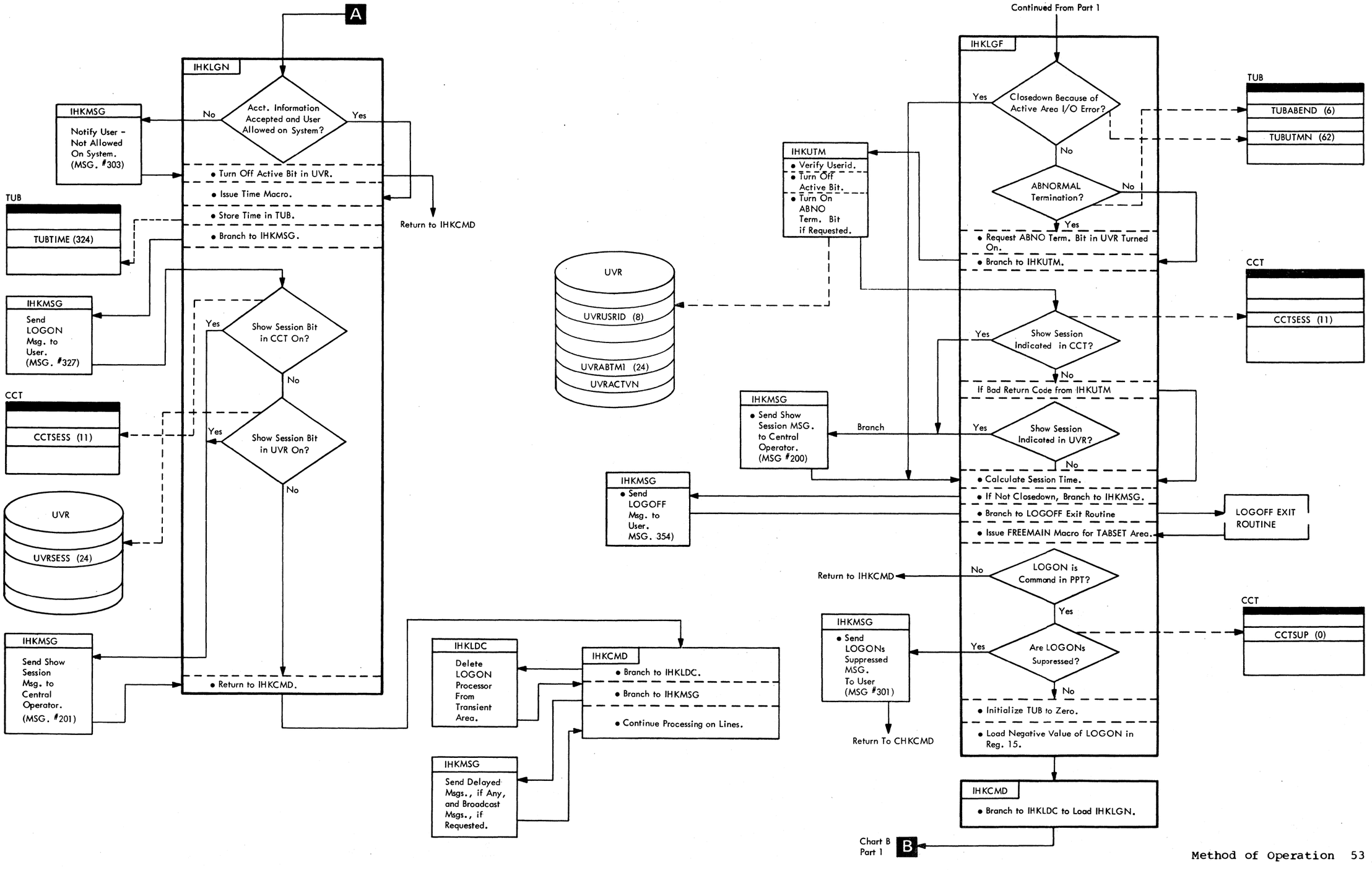


Chart C. Data Management: Create and Copy Functions

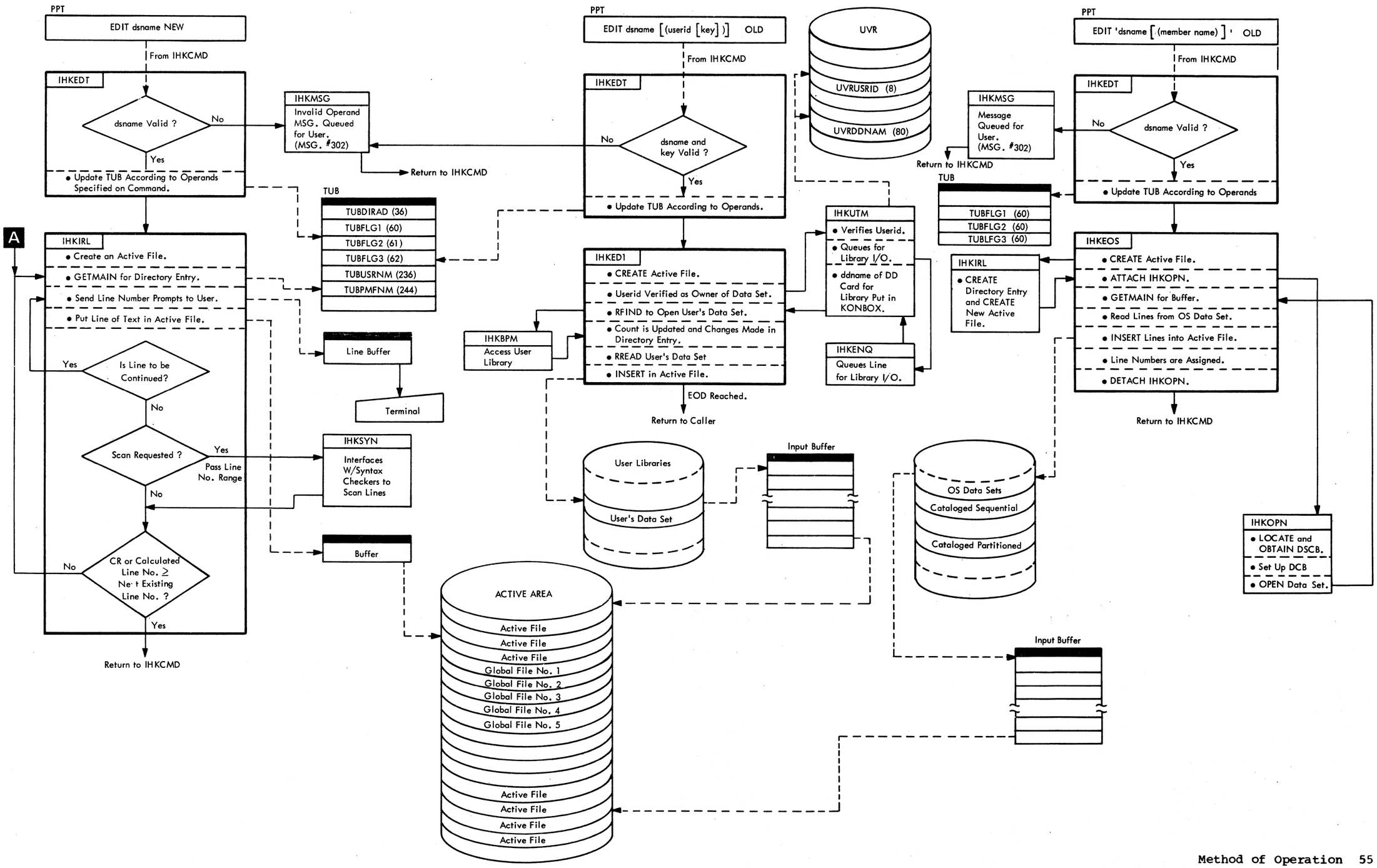


Chart D. Input and Delete Update Functions

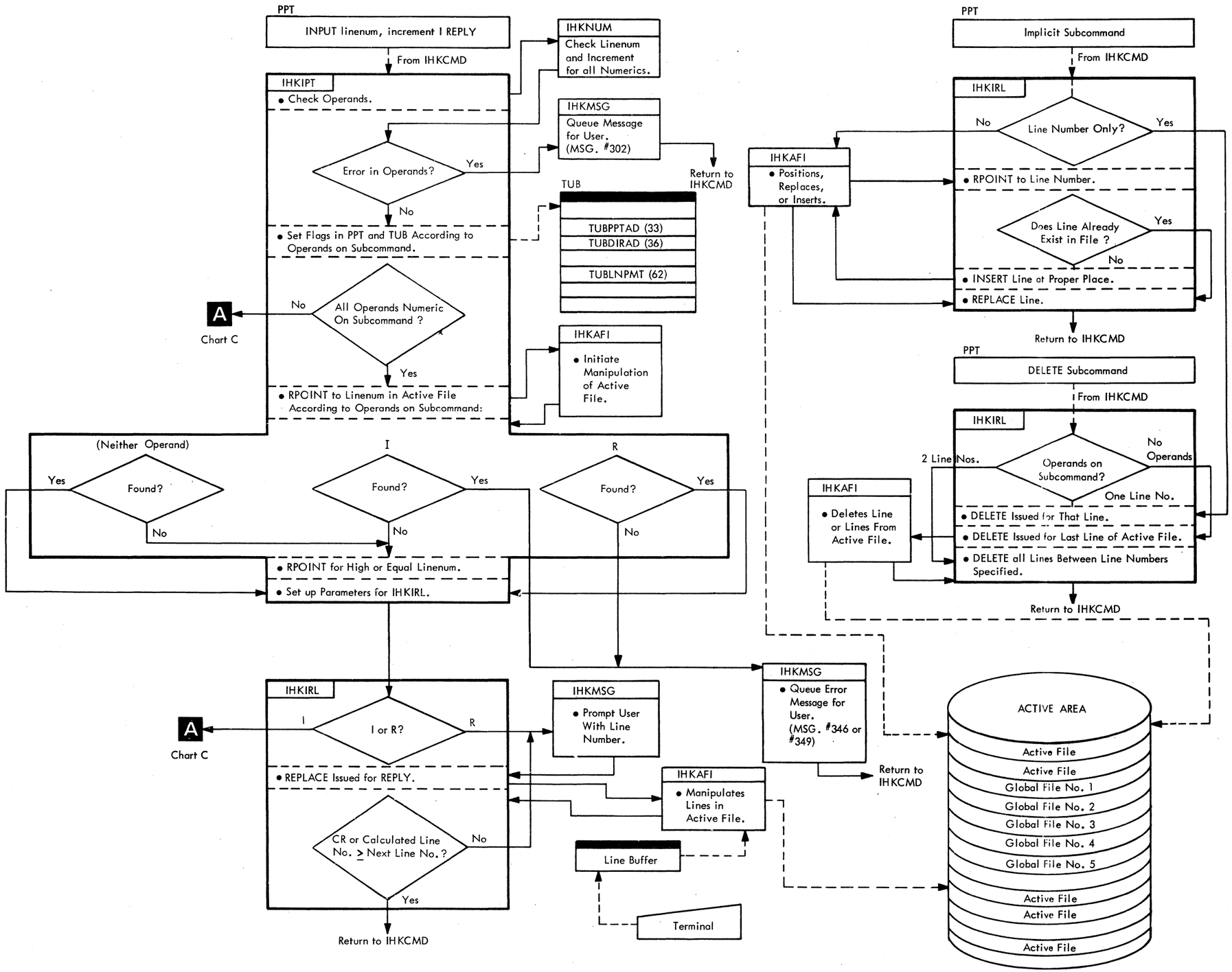
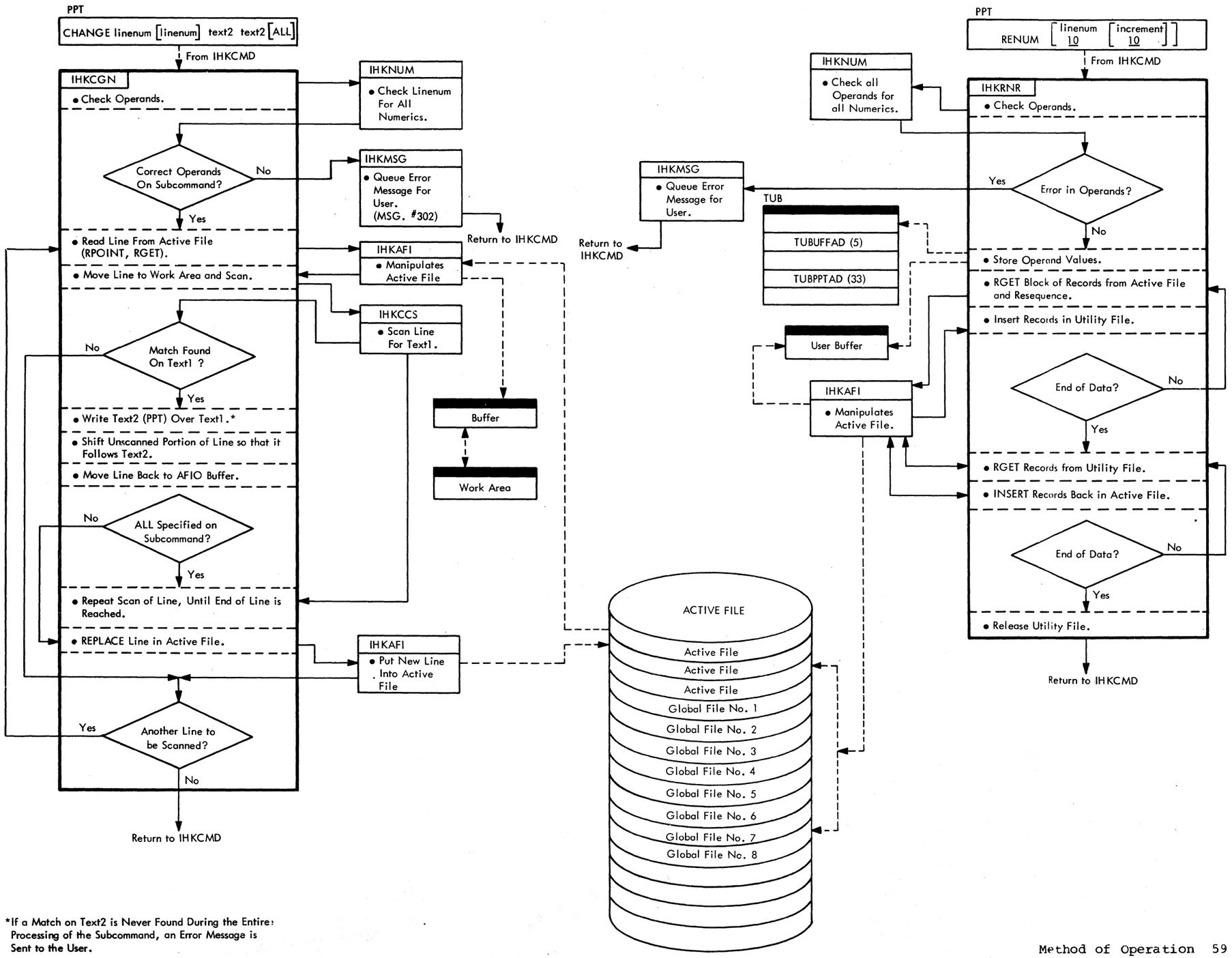


Chart E. Change and Renumber Update Functions



*If a Match on Text2 is Never Found During the Entire Processing of the Subcommand, an Error Message is Sent to the User.

Chart F. Merge Update Function

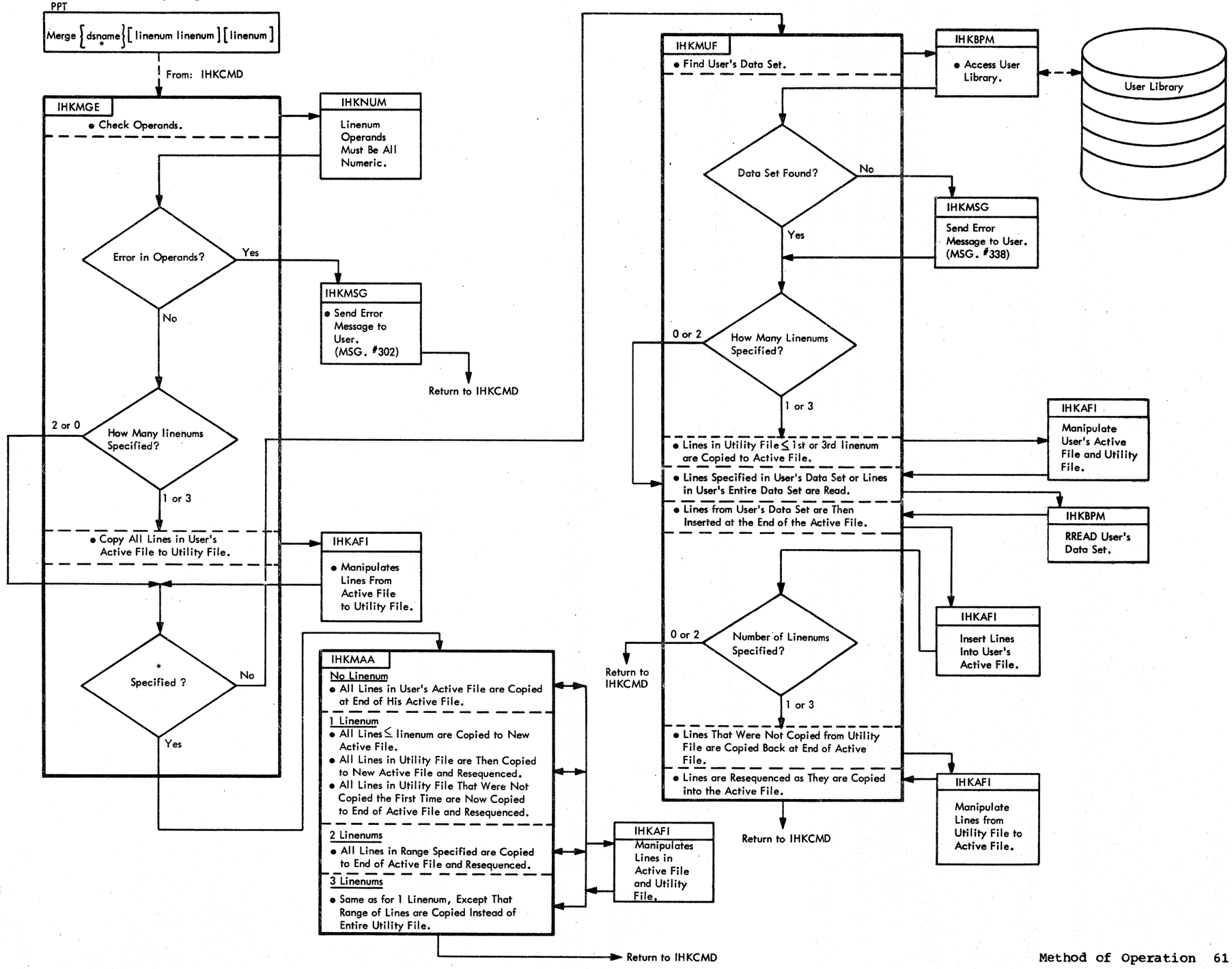
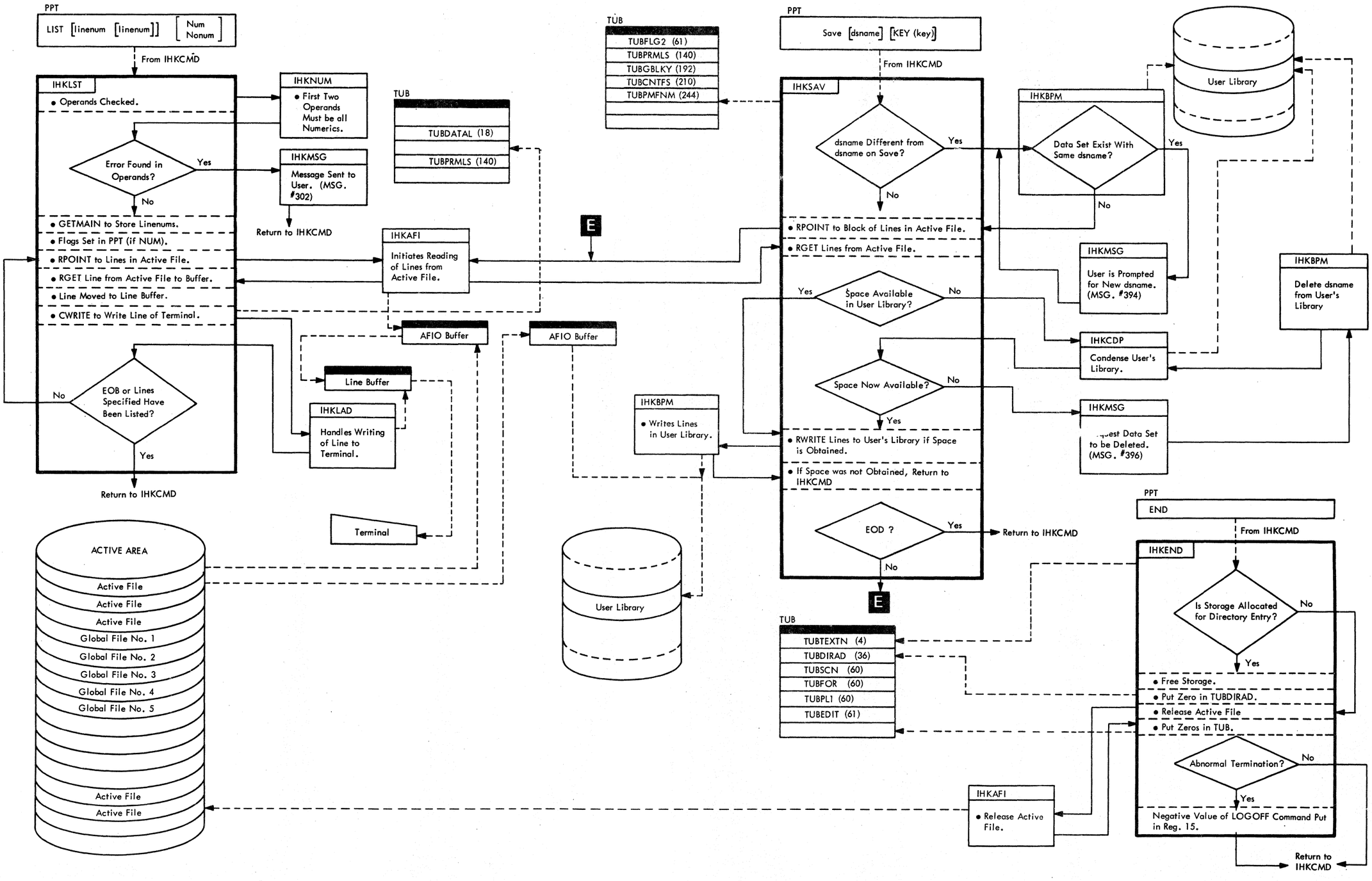
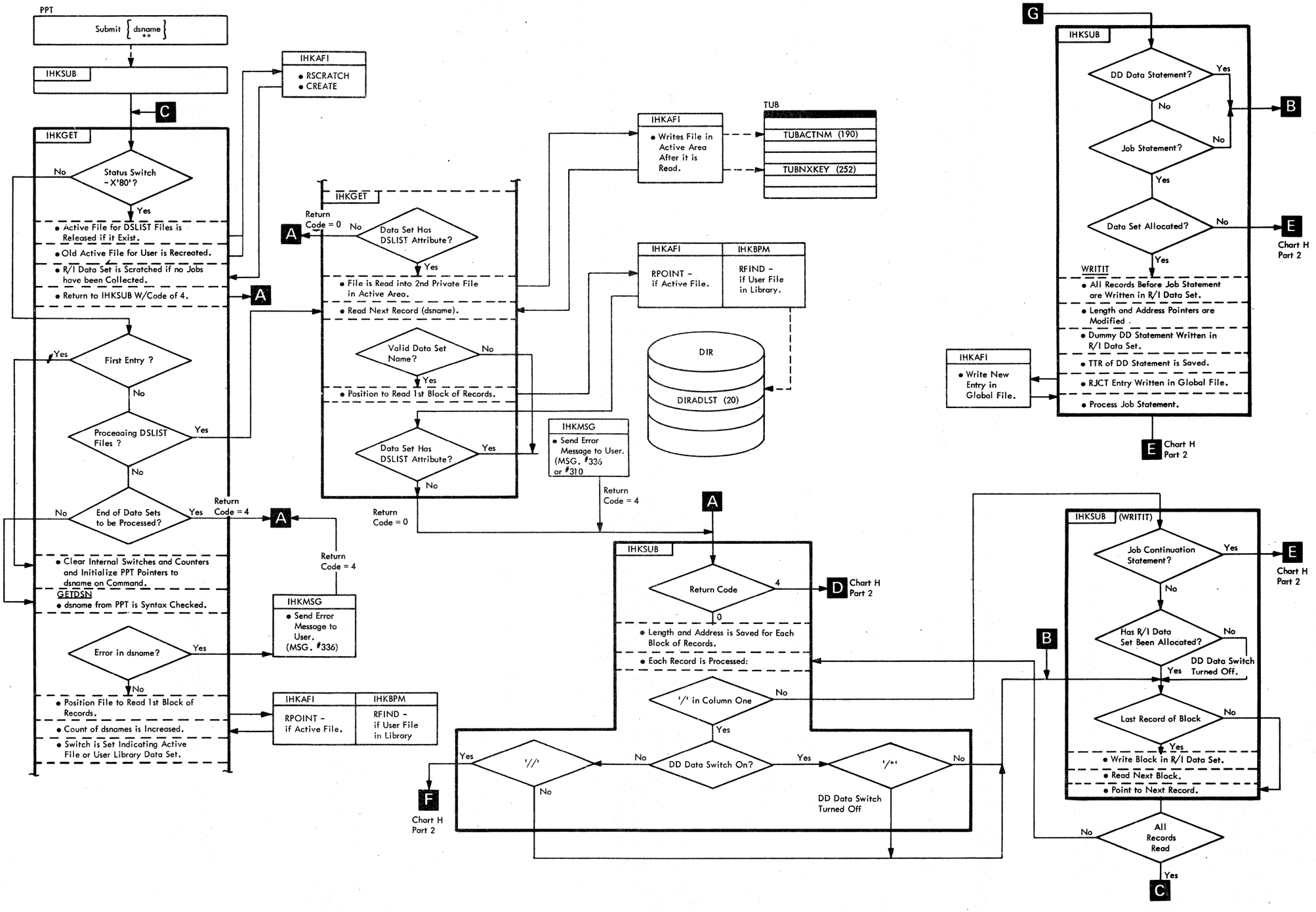


Chart G. LIST, SAVE, and SCRATCH Functions




PPT
Submit { dsname / ** }
IHKSUB
C
IHKGET
No
Status Switch - X'80'?
Yes
• Active File for DSLIST Files is Released if it Exist.
• Old Active File for User is Recreated.
• R/I Data Set is Scratched if no Jobs have been Collected.
• Return to IHKSUB W/Code of 4.
Yes
First Entry ?
No
Proceaaing DSLIST Files ?
Yes
No
No
End of Data Sets to be Processed?
Yes
Return Code = 4
A
• Clear Internal Switches and Counters and Initialize PPT Pointers to dsname on Command.
GETDSN
• dsname from PPT is Syntax Checked.
Error in dsname?
Yes
No
• Position File to Read 1st Block of Records.
• Count of dsnames is Increased.
• Switch is Set Indicating Active File or User Library Data Set.
IHKAFI
• RSCRATCH
• CREATE
Return Code = 4
A
IHKMSG
• Send Error Message to User. (MSG. #336)
IHKAFI
RPOINT - if Active File.
IHKBPM
RFIND - if User File in Library
IHKGET
Return Code = 0
A
No
Data Set Has DSLIST Attribute?
Yes
• File is Read into 2nd Private File in Active Area.
• Read Next Record (dsname).
Valid Data Set Name?
No
Yes
• Position to Read 1st Block of Records.
Data Set Has DSLIST Attribute?
Yes
No
Return Code = 0
IHKAFI
• Writes File in Active Area After it is Read.
TUB
TUBACTNM (190)
TUBNXKEY (252)
IHKAFI
RPOINT - if Active File.
IHKBPM
RFIND - if User File in Library.
DIR
DIRADLST (20)
IHKMSG
• Send Error Message to User. (MSG. #336 or #310
Return Code = 4
A
IHKSUB
Return Code
4
D Chart H Part 2
0
• Length and Address is Saved for Each Block of Records.
• Each Record is Processed:
'/' in Column One
No
Yes
Yes
'//'
No
No
DD Data Switch On?
Yes
'/*'
No
DD Data Switch Turned Off
F Chart H Part 2
G
IHKSUB
DD Data Statement?
Yes
No
B
Job Statement?
No
Yes
Data Set Allocated?
No
E Chart H Part 2
Yes
WRITIT
• All Records Before Job Statement are Written in R/I Data Set.
• Length and Address Pointers are Modified .
• Dummy DD Statement Written in R/I Data Set.
• TTR of DD Statement is Saved.
• RJCT Entry Written in Global File.
• Process Job Statement.
IHKAFI
• Write New Entry in Global File.
E Chart H Part 2
IHKSUB (WRITIT)
Job Continuation Statement?
Yes
E Chart H Part 2
No
Has R/I Data Set Been Allocated?
No
DD Data Switch Turned Off.
Yes
B
Last Record of Block
No
Yes
• Write Block in R/I Data Set.
• Read Next Block.
• Point to Next Record.
No
All Records Read
Yes
C

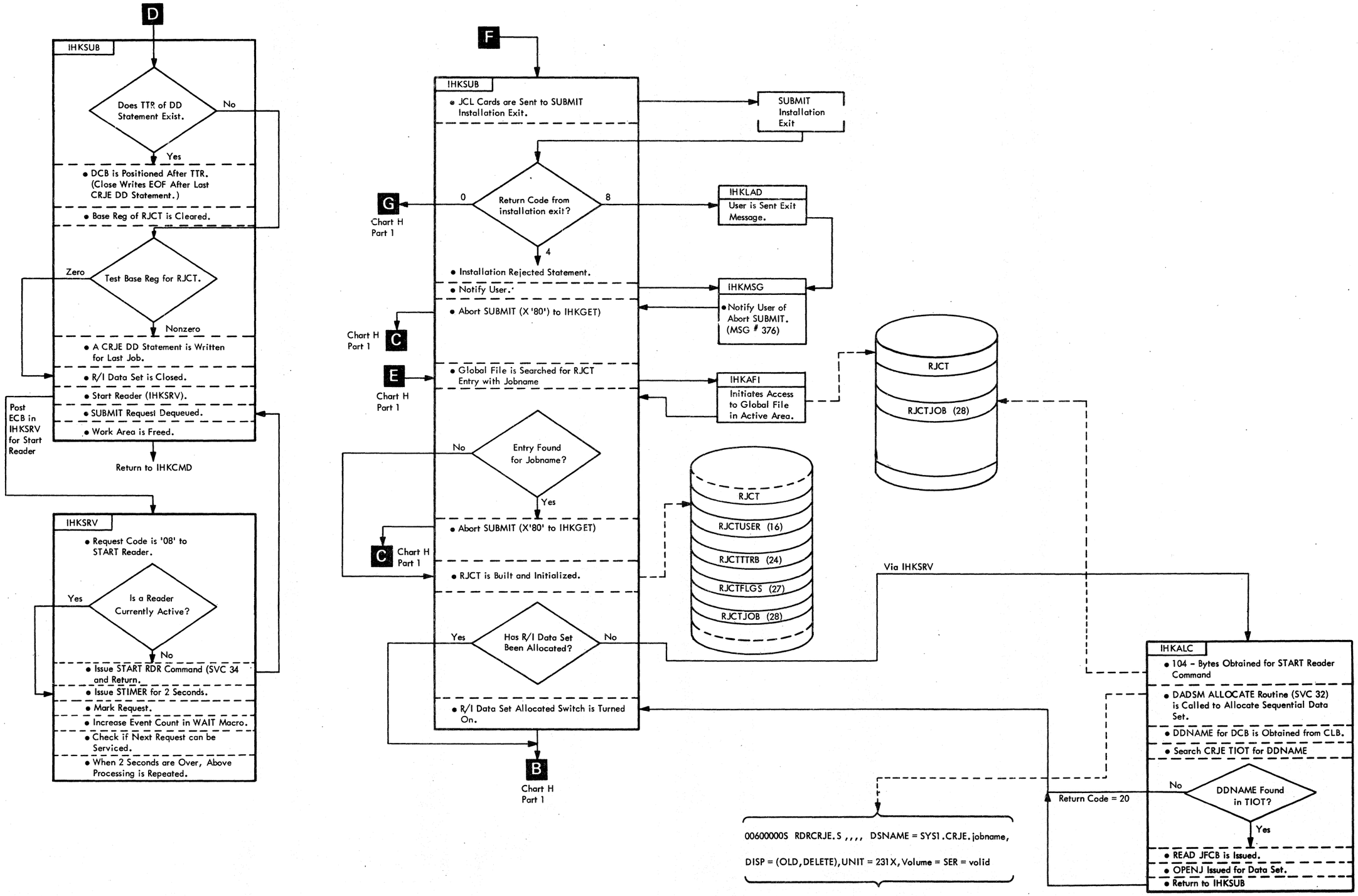

D
IHKSUB
Does TTR of DD Statement Exist.
No
Yes
• DCB is Positioned After TTR. (Close Writes EOF After Last CRJE DD Statement.)
• Base Reg of RJCT is Cleared.
Zero
Test Base Reg for RJCT.
Nonzero
• A CRJE DD Statement is Written for Last Job.
• R/I Data Set is Closed.
• Start Reader (IHKSRV).
• SUBMIT Request Dequeued.
• Work Area is Freed.
Post ECB in IHKSRV for Start Reader
Return to IHKCMD
IHKSRV
• Request Code is '08' to START Reader.
Yes
Is a Reader Currently Active?
No
• Issue START RDR Command (SVC 34 and Return.
• Issue STIMER for 2 Seconds.
• Mark Request.
• Increase Event Count in WAIT Macro.
• Check if Next Request can be Serviced.
• When 2 Seconds are Over, Above Processing is Repeated.
F
IHKSUB
• JCL Cards are Sent to SUBMIT Installation Exit.
SUBMIT Installation Exit
G
Chart H Part 1
0
Return Code from installation exit?
8
IHKLAD
User is Sent Exit Message.
4
• Installation Rejected Statement.
• Notify User.
• Abort SUBMIT (X'80') to IHKGET)
IHKMSG
• Notify User of Abort SUBMIT. (MSG # 376)
Chart H Part 1
C
E
Chart H Part 1
• Global File is Searched for RJCT Entry with Jobname
IHKAFI
Initiates Access to Global File in Active Area.
No
Entry Found for Jobname?
Yes
• Abort SUBMIT (X'80' to IHKGET)
C
Chart H Part 1
• RJCT is Built and Initialized.
RJCT
RJCTUSER (16)
RJCTTTRB (24)
RJCTFLGS (27)
RJCTJOB (28)
Yes
Has R/I Data Set Been Allocated?
No
• R/I Data Set Allocated Switch is Turned On.
B
Chart H Part 1
RJCT
RJCTJOB (28)
Via IHKSRV
IHKALC
• 104 - Bytes Obtained for START Reader Command
• DADSM ALLOCATE Routine (SVC 32) is Called to Allocate Sequential Data Set.
• DDNAME for DCB is Obtained from CLB.
• Search CRJE TIOT for DDNAME
No
DDNAME Found in TIOT?
Yes
Return Code = 20
• READ JFCB is Issued.
• OPENJ Issued for Data Set.
• Return to IHKSUB
006000000S RDRCRJE.S ,,,, DSNAME = SYS1.CRJE.jobname,
DISP = (OLD,DELETE),UNIT = 231X,Volume = SER = volid

Chart I. Notification of Job Output

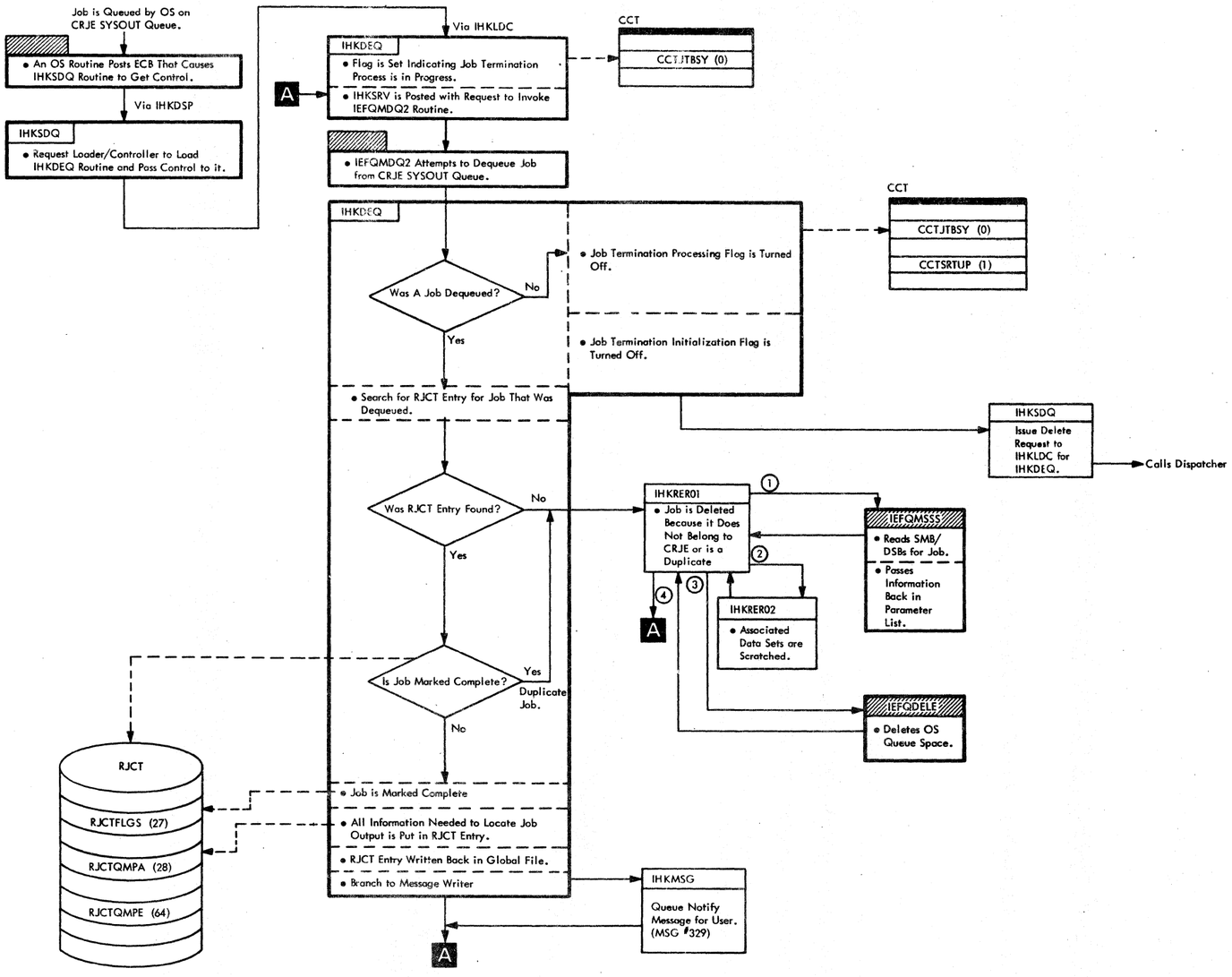
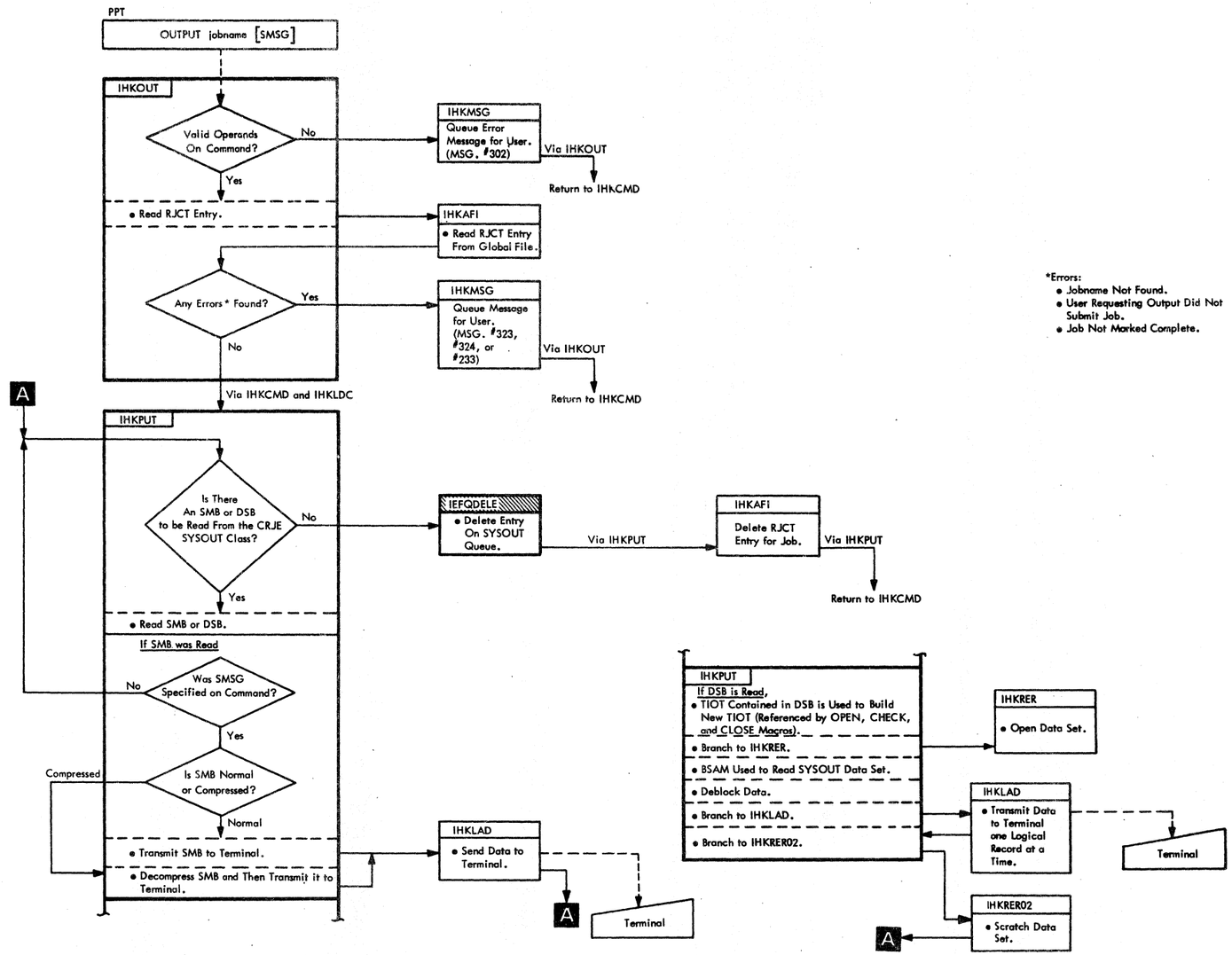
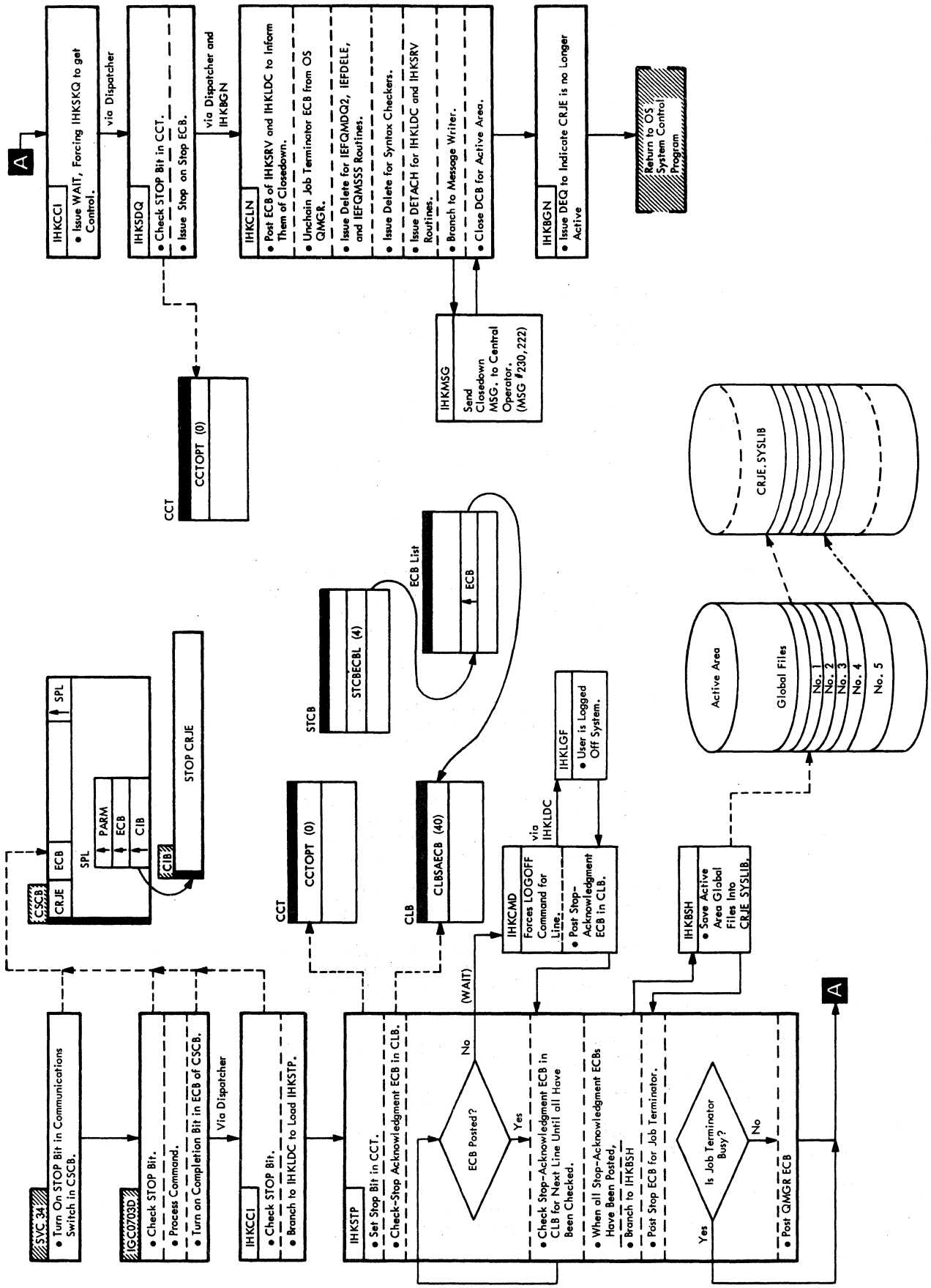


Chart J. Job Output



*Errors:
- Jobname Not Found.
- User Requesting Output Did Not Submit Job.
- Job Not Marked Complete.

Chart K. CLOSEDOWN



This section contains a description of every module in the CRJE program. The modules are grouped together by the function they perform.

START-UP ROUTINES

The following three modules contain the main start-up and initialization routines. Other modules are used in performing initialization, but as that is not their main function, they are described in another section.

CRJE SYSTEM LIBRARY INITIALIZATION UTILITY (IHKINI)

Entry Point

IHKINI - No parameters are passed. Input requirements - IHKSMG (CRJE system messages) and IHKUSR (CRJE users) must be linkage edited with the IHKINI routine, and a DD statement with the DDNAME of the SYSLIB describing CRJE.SYSLIB must be present in the job stream.

Function

This module is a CRJE utility program that must be executed before using the CRJE system. The program resides as a load module (IHKINT) on SYS1.LINKLIB and results from the linkage editing of IHKSMG, IHKUSR, and IHKINI.

The IHKINI module opens the CRJE system library, writes the CRJE system messages in the system library, and issues a STOW macro for the directory entry. If the open fails, an error message is sent to the central operator and control is returned to the operating system.

If the CRJE USERS member is to be initialized, it is written to the CRJE system library and the directory entry is stored.

When the CRJE USERS member has been written (if it was present), the remaining directory entries (user messages, broadcast messages, and job tables) are stored. The system library is closed and control is returned to the operating system.

If any I/O errors are encountered during initialization, a message is sent to the central operator and no further processing is attempted.

The DCB used by this module has the following parameters:

```
DCB BFALN=F,BLKSIZE=880,DDNAME=SYSLIB,DSORG=PO,LRECL=80,                x
    MACRF=WO,RECFM=FB
```

External Routines

BPAM - Invokes the following macros
WRITE, CHECK, NOTE, STOW, OPEN, and CLOSE.

Tables/Work Areas

18-word save area
88-byte DCB
20-byte DECB

Exits

Normal - return to operating system
Error - none

Attributes

Serially reusable

START COMMAND PROCESSOR (IHKBGN)

Entry Point

IHKBGN	-	register 1 contains the address of the start parameter list (SPL). The SPL contains three pointers: PARM field on the EXEC statement in the procedure to start CRJE, an ECB for communication with OS, and the command input buffer (CIB).
IHKBGN01	-	STAE exit routine for the main CRJE task. It expects no input parameters.

Function

The IHKBGN routine is given control by the operating system task control routine at CRJE system start-up time. An ENQ macro is issued to ensure that CRJE is not already an active task within the operating system. If CRJE is already active, the START command is refused. If it is not already active, this routine passes control to the IHKCIP module to initialize the CRJE system. When control returns, the return code is checked:

- 0 - initialization was complete;
- 4 - error encountered, branch to the IHKCLN module for cleanup before exit;
- 8 - bad parameters, issue DEQ macro and exit.

If initialization was successful, a STAE macro is issued to intercept any abend of CRJE and a branch and link is made to the job terminator module (IHKSDQ) to start processing on the lines. When control returns at stop time or a return code of 4 is received during initialization, control is passed to the closedown routine (IHKCLN) to perform cleanup of CRJE before exit. Upon return from the IHKCLN module, a DEQ macro is issued indicating that CRJE is no longer active. Control is then returned to the system control program.

As a result of the STAE macro, when the CRJE main task abends, the CRJE STAE exit routine, IHKBGN01, gains control. IHKBGN01 links to IEFQMUNC to unchain the CRJE job termination ECB and then returns to the operating system so that the abend may be processed.

External Routines

IHKCIP	-	CRJE initialization
IHKSDQ	-	entry point to main body of CRJE
IHKCLN	-	cleanup before exit

IHKMSG - (entry point:IHKMSG01) error messages sent to central operator

IEFQMUNC - unchain the CRJE ECB

Tables/Work Areas

18-word save area

AVT - address of start parameter list is put in IHKSPL.

Exits

Normal - return to system control program

Error - none

Attributes

Resident and serially reusable

CRJE INITIALIZATION ROUTINE (IHKCIP)

Entry Point

IHKCIP - Register 1 contains the address of the CRJE address vector table (AVT).

Function

This routine inspects the parameters on the START command. If an error is detected, the return code is set to 8 and control is returned to the START command processor (IHKBGN). If syntax checkers are specified, the appropriate FORTRAN and PL/1 syntax checkers are loaded and dummy passes are made to them for initialization purposes. If the parameters are valid, an ATTACH macro is issued for the loader/controller (IHKLDC) and the service task (IHKSRV). A LOAD macro is issued for the following OS routines: IEFQMDQ2, IEFQDELE, and IEFQMSSS.

A branch is made to the active area start-up/initialization module (IHKAST is described in the librarian section) to open the active area. If ABNO is specified on the START command, a branch is made to the active area recovery module (IHKAWS is also described in the librarian section). This routine saves the global file in the CRJE system library, saves the user's active file in his library under the name ACTIVE, and releases the allocated space. If NORM is specified, the call to the IHKAWS module is bypassed. The IHKBST module is called to read the CRJE system library into the global files of the active area.

Once the active area has been initialized the CRJE job control table is updated according to the type of START command that was entered: FORM, NFMT, or NONE. This is a two-pass operation.

On the first pass and if FORM was specified, IHKCIP scratches each data set that was built by the SUBMIT processor and that has not been processed by an OS reader.

On the first pass for NORM or NFMT, a START RDR command is issued for each data set that was built by the SUBMIT processor and that has not been processed by an OS reader.

On the second pass each RJCT entry is inspected. All jobs that are not marked complete and that are not found on the SYS1.SYSJOBQE are deleted from CRJE (RJCT entry is deleted).

Depending upon the parameter specified on the START command, the following processing is done for jobs marked complete and found on the SYS1.SYSJOBQE:

FORM start - all RJCT entries are deleted
NFMT start - RJCT entry market not complete
NONE start - IOB built for each complete job

After all RJCT entries are processed the lines are opened. If the open is successful, the dummy ECB in the STCB for that line is changed from a wait (X'80') condition to a post (X'40') condition. If the open failed, the stop acknowledgment ECB in the CLB is posted.

Then the parameters on the EXEC statement are processed. The loader/controller is primed with load requests to fill the transient area. A delete request is issued for all modules that the loader/controller brought into the transient area. This reduces the request count to the correct value (zero) but does not physically remove the modules.

The address of the communications ECB (second word of the SPL) is put in the dispatcher's ECB list for central commands. A message is sent to the central operator notifying him that CRJE is active. Control is then returned to the IHKBGN module.

External Routines

IHKMSG - (entry point:IHKMSG02) to send active message to central operator
IHKAST - to initialize the active area
IHKAWS - to recover the active area if ABNO is specified on START command
IHKBST - to initialize the global files from the system library
IHKAFI - to search global file (RJCT entries) for job recovery
IHKSRV - to get IOB for completed job
IEFLOCDQ - to determine if jobs marked not complete are on job queue, SYSOUT queue, or hold queue. Alias is IHKLOC.

Tables/Work Areas

18-word save area
113-byte START RDR command buffer
80-byte buffer for RJCT entry
CLB - stop acknowledgment ECB (CLBSAECB) is posted with X'FF0000' is the open fails.
STCB - dummy ECB (STCBDUMY) is posted if open is successful.

Exits

Normal - return to the IHKBGN module with return code of 0 in register 15
Error - return to the IHKBGN module with one of the following return codes in register 15:
04 - error in intialization
08 - bad parameters

Attributes

Serially reusable and nonresident

ACTIVE AREA START-UP/INITIALIZATION MODULE (IHKAST)

Entry Point

IHKAST - register 1 must contain the address of the AVT.

Function

The initialization module, IHKCIP, branches to this module during CRJE start-up. The functions of this module are:

- open the active area DCB;
- build and initialize the track allocation table (TAT);
- initialize several track allocation control fields in the IHKNBX (KONBOX).

A DEVTYPE macro is issued to determine the characteristics of the device to which the active area data set has been assigned. This is done to ensure that the device is a 2311, 2314, or 2319. If it is not, error returns are made. The active area DCB is then opened, and the DCB pointer to the DEB is used to determine the DASD extent allocated to the active area. If more than one extent is allocated, an error return is made.

The number of cylinders in the allocated extent and the number of heads per cylinder according to the device (2311, 2314, or 1219) determine the number of bytes necessary for the TAT. A GETMAIN macro is then issued for that number of bytes. The TAT is initialized to reflect all of the tracks in the active area that are available for allocation. After initialization of the TAT, the following fields in the KONBOX are initialized: NUMHEAD, UMAXAVAL, UC1FW, UC12FW, UC3FW, MASTRKFW, MASLGHFW, MIDCYLFW. These fields are used by the IHKAFI and IHKEXC modules during active file track allocation, deallocation, and master index track initialization during a CRJE session.

External Routines

None

Tables/Work Areas

Track Allocation Table

IHKNBX (KONBOX)

Exits

Normal - return to IHKCIP with 0 in register 15

Error - return to IHKCIP with one of the following return codes in register 15:

04	-	GETMAIN failed
08	-	No DD card for active area data set
12	-	Device not DASD
16	-	Device not 2311, 2314, or 2319
20	-	Multiple extents allocated to active area data set

Attributes

Nonresident and serially reusable

ACTIVE AREA RECOVERY MODULE (IHKAWS)

Entry Point

IHKAWS - register 1 contains the address of the CRJE address vector table (ACT).

Function

The initialization module, IHKCIP, branches to this module at CRJE start-up time when the ABNO parameter is specified on the CRJE START command. This module attempts to save the user's active files that existed at the time of a previous abnormal closedown. The user's active file is saved in his user library under the name ACTIVE. If there is not enough space in his library, his active file is lost.

The operation of this module depends upon the ability to gain access to the USERS global file in the active area. If the abnormal closedown from which recovery is being attempted resulted from an inability to gain access to the active area, this module cannot perform its function.

The UVRLNSEQ field in the USERS global file contains the line sequence number that was assigned to the user's TUB when he was active. UVRLNSEQ is used to locate the entry of this user's active file on the master index track. The information contained in this entry allows this module to sufficiently initialize the required AFIO work areas to allow the reading of the user's records from his active file and the saving of them in his user library.

The IHKAST module has initialized the track allocation table, the IHKNBX fields that control active area track allocation, and the master index track location before control is passed to the IHKAWS module. The track allocation table is not used during recovery; the information in the KONBOX concerning the master index track location is used instead.

The IHKAWS module obtains a save area. Then the RPOINT macro is issued to point to the beginning of the USERS global file. An RGET macro is used to retrieve the first record. The UVRLNSEQ field of the record is examined to determine whether this particular user had an active file at the time of closedown. If the user did not have an active file, the next record is read from the USERS global file. If the user did have an active file, a CREATE 0 macro is issued to reinitialize the AFIO work area in the TUB using information from the user's active file entry in the master index track and from the user's own file index track, located through his master index entry.

In a normal CRJE start-up environment, the first CREATE macro that is issued by any command processor causes initialization of the master index track. This function is bypassed during recovery so that information in the master index track existing from the previous CRJE session will not be destroyed.

After the CREATE 0 macro is issued for the user's active file, an RFIND macro is issued to open and point to his user library. An RPOINT macro is then used to point to the beginning of the user's active file. Alternate RGET and RWRITE macros are used to save the user's active file in his library under the name ACTIVE. If the user library has a logical record length of 80 characters (as opposed to a length of 88 characters), IHKAWS checks positions 73-80 for either blanks or the line sequence number. If neither is present, a flag is set so that a warning message will be sent to the user, informing him that the data in columns 73-80 was lost (message IHK410 DATA LOST IN TRUNCATION).

If the save process ends without error, the user library is closed, a saved message is inserted into the message chain, and the next record in the USERS global file is processed in the same manner. If errors are

encountered during the save process, a message is entered into a message chain, built and maintained by this module, and the process continues until EOD is encountered on the USERS global file.

Upon the completion of processing all the records in the USERS global file, a call is made to the IHKBSH module to save all the global files in the system library. This is attempted to save any updates that were made to the global files during the previous CRJE session.

External Routines

IHKAFI - to read USERS global file and to read users' active files
IHKBPM - to write the user's active file to his library
IHKBSH - to save global files into the system library
IHKWTR - to check for completion of RWRITE

Tables/Work Areas

TUB - AFIO related fields
IHKNBX - information concerning the master index track location
AVT - addresses

Exits

Normal - return to IHKCIP with a pointer to the first message of the message chain in TUBPARM4 (zero if no message chain exists) and zero in register 15.

Error - return to IHKCIP with one of the following return codes in register 15:

04 - Active file open failure
08 - I/O error in active file
12 - GETMAIN failure
16 - I/O error in IHKBSH

Attributes

Nonresident and serially reusable

LIBRARY I/O START-UP MODULE (IHKBST)

Entry Point

IHKBST - register 1 must point to a one-word area containing the address of the AVT.

Function

The library I/O start-up module generates the active global files by copying the members of the CRJE system library (CRJE.SYSLIB) into the active area.

After generating the active global files, this module searches the DASD devices for which a DD card was provided with the ddname LIBXXX. These DD cards identify volumes that contain CRJE user libraries having the name CRJE.LIB.userid. When these user libraries are located, the userid is checked for validity. The ddname for the volume on which the library is found is saved in the last eight bytes of that user's entry in the USERS global file.

If a user's library is found for which there is no entry in the USERS global file, the library is ignored. If I/O errors occur while gaining access to the active area or the system library, control is returned to the CRJE initialization routine to initiate closedown.

External Routines

IHKBPM - to gain access to user libraries
IHKAFI - to gain access to the global files in the active area
IHKRNQ - to queue for library I/O

Tables/Work Areas

18-word save area
352-byte DSCB buffer
TUB
KONBOX

Exits

Normal - return to calling routine with a 0 in register 15
Error - return to calling routine with one of the following return codes in register 15:
04 - maximum users exceeded
08 - permanent I/O error in active area
12 - permanent I/O error in system library

Attributes

Nonresident and serially reusable

SHUTDOWN ROUTINES

CRJE STOP MODULE (IHKSTP)

Entry Point

IHKSTP - register 1 must point to a one-word area containing the address of the AVT.

Function

The IHKSTP module first sets the stop bit in the CCT. The CLBs for the lines are then scanned. If the stop acknowledgment ECB in the CLB is posted, the scan moves on to the next CLB. If it is not posted, the stop module issues a RESETPL macro. The RESETPL macro issues a HALT I/O only if an enable or prepare is on the line. This forces the line administrator out of the wait condition. If an enable or prepare is not on the line, the STOP command is recognized by the system administrator when the current command from the line completes processing.

The line administrator recognizes that closedown is in progress and returns to the system administrator. The system administrator forces a LOGOFF for the user, performs general cleanup on the line, and posts the stop acknowledgment ECB.

After performing the previously described actions, the IHKSTP module moves to the next CLB until all CLBs are scanned. When all CLBs have been scanned, IHKSTP makes another pass through them. On this second pass, it waits in the CRJE dispatcher for the stop acknowledgment ECB to be posted for each line.

When all stop acknowledgment ECBs have been posted (indicating that STOP is recognized and completed for each line), control is passed to the IHKBSH module. This module writes the global files of the active area into the CRJE system library. The IHKSTP module then posts the stop ECB for the job terminator, indicating that stop processing is

complete and the IHKSDQ module can now return to the START command processor (IHKBGN). A check is made to see whether the IHKSDQ module is busy. If it is not busy, this module posts the QMGR ECB of the IHKSDQ module, since the IHKSDQ module may be at this time waiting for either the QMGR or stop ECB. Control is then returned to the IHKCCI module, which waits in the dispatcher, thereby giving the IHKSDQ module control.

External Routines

IHKDSP - to wait for the lines to close
IHKBSH - to write the global files of the active area into the system library

Tables/Work Areas

18-word save area

CCT
CCTOPT - (CCTCLS) turned on to indicate closedown in progress

CLB
CLSAECB - (stop acknowledgement ECB) waits for this ECB to be posted indicating line activity is now terminated

Exits

Normal - return to the IHKCCI
Error - none

Attributes

Reentrant and nonresident

CRJE CLOSEDOWN MODULE (IHKCLN)

Entry Point

IHKCLN - register 1 must point to a one-word area containing the address of the AVT.

Function

This closedown module first informs the CRJE attached tasks (IHKLDC and IHKSRV) of the closedown. It then unchains the job terminator's ECB from the OS Queue Manager (IEFQMUNC); issues a DELETE macro for the IEFQMSSS, IEFQMDQ2, and IEFQDELE routines; and issues a CLOSE macro for the lines. If syntax checkers are present, they close down, and a DELETE macro is issued for them. If the attached tasks abended, an error message is sent to the central operator.

The CRJE attached tasks are detached. If a SHOW SESS is outstanding and Multiple Console Support (MCS) has been included in the system, the SHOW SESS bit in the unit control module (UCM) is turned off for each console.

A closedown message (normal or abnormal) is sent to the central operator. Then the DCB for the active area is closed, and control is returned to the IHKBGN module.

External Routines

IEFQMUNC - to unchain queue manager ECB
IHKMSG - (entry point:IHKMSG) to send normal closedown message

- (entry point:IHKMSG02) to send abnormal closedown message

Tables/Work Areas

18-word save area
CCT
CCTOPT - (CCTATERM) set if OS queue manager (IEFQMUNC) has disk error; inspected for type of closedown message to be sent to central operator.
CLB
CLBSAECH - (stop acknowledgment ECB for line) checked for X'FF' at CLBSAECB + 1; if X'FF', line DCB is not opened and no CLOSE macro is issued for that line's DCB.

Exits

Normal - return to the IHKBGN module
Error - none

Attributes

Serially reusable and nonresident

LIBRARY I/O SHUTDOWN MODULE (IHKBSH)

Extry Point

IHKBSH - register 1 must point to a one-word area containing the address of the AVT.

Function

This module saves the active area global files in the CRJE system library (CRJE.SYSLIB) at closedown and at an abnormal start-up.

There is a possibility that an end-of-extent situation may arise. If the copy operation is terminated because of an end-of-volume condition or I/O error, a message is sent to the central operator notifying him of the situation.

If no end-of-volume or I/O error situation arises, all global files are copied into their respective members of the CRJE system library (CRJE.SYSLIB) and control is returned.

External Routines

IHKRNQ - to queue for library I/O
IHKAFI - to gain access to active area
IHKMSG - (entry point:IHKMSG02) to send message to central operator
IHKBPM - to perform library I/O

Tables/Work Areas

KONBOX
TUB (dummy)

Exits

Normal - return to calling routine
Error - none

Attributes

Nonresident and serially reusable

UTILITY TASK

START RDR, ALLOCATE, AND Q MANAGER SERVICE TASK (IHKSRV)

Entry Point

IHKSRV - the address of the AVT and the address of the initialization ECB, for which the IHKCIP module is waiting, are passed via the ATTACH macro.

IHKSRV01 - parameters are passed to the STAE exit entry point.

Function

This service task issues a POST macro for the initialization ECB to inform the IHKCIP module that the ATTACH macro has completed. A STAE macro, with IHKSRV01 as the STAE exit routine, is issued to intercept abends within the service task. This routine then issues a multiple WAIT macro for its ECB list and waits for work. The ECB list, which was generated at CRJE assembly time, contains the following pointers:

- to an ECB in the CCT that is used by the timer exit routine in this task;
- to an ECB in the CLB for each line;
- to an ECB in the CCT for the central commands;
- to an ECB in the CCT for job termination.

This service task gets control again when a requester issues a POST for one of the ECBs in the ECB list. The posted ECB points to the following parameter list:

Word 1 - the address of the return ECB,
Word 2 - the address of a one-byte field containing a code that specifies the function requested,
Word 3 - the address of the parameter list that must be passed to the requested module,
Word 4 - the entry point address of the IHKALC routine (used only for the allocate request).

When control is given back to this module, the START RDR request queue is checked. If the queue is not empty and if no reader is currently active, the first request is taken from the queue and a START RDR (SVC 34) is issued. If a reader is currently active, a STIMER macro is issued. This gives control back to the IHKSRV routine after five seconds, and the remaining START RDRCRJE requests can be satisfied.

On an empty queue or if a reader is running, processing continues. The different requests passed to this routine are analyzed and, according to the codes, the service task takes the following actions:

00 - STOP Return to the control program after all START RDRCRJE requests have completed.

04	-	ALLOCATE	Give control to the IHKALC routine and issue a POST macro for the return ECB (with return code from IHKALC).
08	-	START RDRCRJE	On a START RDRCRJE request a 104-byte parameter list is passed to the IHKSRV routine. The last two words are used by the IHKSRV routine to put this request on a queue (START RDR request queue) and to close the DCB. When the request is queued, the return ECB is posted. If no queue entry is passed to the IHKSRV module, the R/I data set is scratched.
12	-	QMSSS	Give control to the IEFQMSSS routine and issue a POST macro for the return ECB (with return code from IEFQMSSS).
16	-	QMDQ2	Give control to the IEFQMDQ2 routine and issue a POST macro for the return ECB (with return code from IEFQMDQ2).
20	-	QDELE	Give control to the IEFQDELE routine and issue POST macro for return ECB (with return code from IEFQDELE).
24	-	LOC	Give control to the IEFLOCDQ routine after being loaded by loader/controller. Issue a POST macro for the return ECB (with return code from the IEFLOCDQ routine).
28	-	GETCORE	A GETMAIN macro is issued for 72 bytes for an IOB from subpool 253, and a POST macro is issued for a return ECB (with the address of the GETMAIN area as the post code).

IHKSRV01 gains control if an abend occurs within the service task. IHKSRV01 posts the main CRJE task to prevent further accesses to the service task. It then posts each CRJE subtask that is waiting on the service task. The closedown and abnormal closedown flags in the CCT are set and the main CRJE task is posted to initiate closedown of CRJE. It then posts each CRJE subtask that is waiting on the service task and returns control to the operating system.

External Routines

SVC 34	-	START RDR
IEFQMSSS, IEFQMDQ2, IEFLOCDQ and IEFQDELE	-	read or write records on SYS1.SYSJOBQE
IHKALC	-	allocate SYSIN data set for job submission

Tables/Work Areas

18-word save area
AVT
CCT
ECB list

Exits

Normal - return to calling routine
Error - none

Attributes

Resident and serially reusable

LOADER/CONTROLLER TASK

LOADER/CONTROLLER MODULE (IHKLDC)

Entry Points

IHKLDC - is attached by the initialization module (IHCIP) at start-up time and it runs with the same dispatching priority as the originating task. The address of the ECB for the initialization module and the address of the AVT are passed using the ATTACH macro.

IHKLDC01 - the STAE exit routine for the loader/controller task. At entry, no parameters are passed to it.

Function

The purpose of the loader/controller is to load and delete nonresident modules and to serialize requests for serially reusable modules.

The size of the transient area in which the modules are loaded has a minimum of 8K and can be increased at start-up time in multiples of 2K. The modules loaded by the loader/controller are grouped together in 2K load modules. And the modules are always brought into main storage in terms of 2K.

When the IHKLDC module is entered, initialization is performed, the ECB for which the initialization routine (IHKCIP) is waiting is posted, and a STAE macro is issued to intercept abends within the task. Then the loader/controller waits for an ECB list pointing to an ECB in each CLB, to an ECB for central commands, and to an ECB for job termination.

The following conventions must be followed to load or delete a nonresident module:

- ECB (pointed to by loader/controller ECB list) must point to a 7-byte area containing the following:

  Bytes 0-3 - return ECB on which the CRJE dispatcher is waiting while loading or deleting is performed.

  Byte 4 - code of the routine to be deleted or zero if there is no delete request.

  Byte 5 - code of the routine to be loaded or zero if there is no load request.

  Byte 6 - zero (used for marking request as waiting).

- ECB (pointed to by loader/controller ECB list) must be posted.

- Branch is made to dispatcher with register 1 pointing to the return ECB.

By using the ECB and tables, which are described later, the loader/controller can find all the information needed to perform the delete or load function. The requests for deletions or loads are filled by scanning, three times in the following manner, the ECBs for which the IHKLDC module is waiting:

1. The ECBs and the appropriate parameters are scanned first for deletion requests. These requests are satisfied immediately by decreasing the block request count by one. Then the return ECB for this request is posted.

2. The second scan is for load requests asking for modules that are already in the transient area. If the requested module is reentrant, the entry point address is inserted in the return ECB and this ECB is posted. If the module is serially reusable and is already occupied by another user, the event count of the WAIT macro is incremented and the request itself is flagged as waiting for a module. Then the scan is resumed with the next ECB.

3. The third scan is for load requests asking for modules that are not already in the transient area. Before the load module containing the requested module can be loaded, a load module in the transient area must be found for which there are no requests for any of its modules. This load module is then deleted leaving an empty 2K block in the transient area. If no load module is found with a zero request count, the new load request cannot be satisfied. The event count of the WAIT macro is increased and the request is marked as waiting for a module (byte 6 of the area pointed to by the ECB). All requests still pending at this time are treated this way. Then the loader/controller waits for new requests. The next time the loader/controller gets control, all three scans are performed starting with the first request that could not be satisfied.

When a load module is found with a zero request count in the block table and the load request is not for a "trouble module," the load module in the block is deleted and the load module that was requested is loaded. (A "trouble module" is a module that requests the loading of another module before the "trouble module" itself is deleted, or is a module that queues itself for library I/O or uses the IHKUTM module to be queued for library I/O.) If the load request is for a "trouble module," the "trouble module" count (first byte in module table) is checked. The "trouble module" count contains the number of 2K blocks assigned to the transient area minus one. Each time a "trouble module" is loaded, one is subtracted from the count. When a "trouble module" is deleted, one is added to it. So if the count is zero, the load request is marked as waiting. If the count is not zero, the load module is loaded and the count updated.

At closedown time the closedown routine passes binary zeros to the loader/controller in bytes 4 and 5 of the 7-byte area. The loader/controller then deletes all the modules in the transient area and returns.

The loader/controller task STAE exit routine (IHKLDC01) gains control if the task abends. IHKLDC01 posts the completion ECB (IHKLCE) to prevent further accesses to the task. The closedown and abnormal closedown flags in the CCT are set and the main CRJE task is posted to initiate the closedown of CRJE. Each CRJE subtask that is waiting on the loader/controller task is posted. Control is then returned to the operating system to process the abend.

External Routines

None

Tables/Work Areas

AVT

CCT
Module Table (IHKMOD) - contains one entry for each nonresident module. Each entry in the table consists of the following 12 bytes:

Byte 0
Bit 0 - set if last entry in table,
Bit 1 - set if module is not reentrant (i.e., module is serially reusable),

Bit 2 - set if module is a "trouble module,"
Bit 3 - not used,
Bit 4 - set if module is in the transient area,
Bit 5 - set if module is loaded (LOAD macro issued for it),
Bit 6 - set if module is locked,
Bit 7 - not used.

Byte 1 - Offset in block table of 2K block allocated

Byte 2 - number of modules in this load module

Byte 3 - number of entry in module table for the next module in this load module

Bytes 4-8 - five significant characters of module name

Bytes 9-11 - entry point address of module

The first entry in the IHKMOD module is a dummy.
It contains the trouble module count in the first word.

Block Table (IHKMAP) - one entry for each 2K block in the transient area.

Byte 0 - block request count (increased for load requests decreased for delete requests).

Byte 1 - number of entry in IHKMOD of module occupying this 2K block (zero if block is empty).

Translate Table (IHKTRT) - associates the number of the entry in IHKMOD to the command and subcommand codes as they are listed in the major command list (IHKMCL) and the subcommand list (IHKSCL). This is necessary for the loader/controller since one module may process several commands or subcommands.

Exits

Normal - return to calling routine
Error - none

Attributes

Resident and serially reusable

OPEN TASK

OS DATA SET OPEN MODULE (IHKOPN)

Entry Point

IHKOPN - register 1 must point to a five-word parameter list containing the following:

1. address of the TUB
2. address of the AVT
3. address of a 1,376-byte work area
4. address of the end-of-data routine
5. address of the I/O error subroutine in IHKEOS

Function

This module is attached by the IHKEOS routine for opening or scratching an OS data set. If the request is to open, a LOCATE macro is issued to find the data set in the SYSCATLG, and an OBTAIN macro is issued to get the DSCBM. The JFCB is then read and the data set name is inserted. A DCB is set up and an OPEN TYPE=J macro is issued. If the open request is for a member of a PDS, a FIND macro is issued to obtain the desired member. The ECB by which the IHKOPN module received control is then posted with the return code. The IHKOPN module issues a WAIT macro for an ECB to be posted by the IHKEOS routine. When the WAIT is complete, the IHKOPN module closes the data set and returns to the operating system.

If the request is to scratch, the catalog is checked to make sure that the data set exists. If a complete data set is to be scratched and not just a member, the data set must reside on less than 20 volumes. A check is made to be sure that all of the volumes are mounted. If they are not or the data set is on more than 20 volumes, an error message is queued for the user and control is returned. If a complete data set is to be deleted, a SCRATCH macro is issued and the entry is deleted from the catalog. If only a member of a data set is to be deleted, the data set is opened; a STOW DELETE macro is issued and the data set is closed. Control is then returned to the calling routine in the same manner as described in the previous paragraph.

External Routines

None

Tables/Work Areas

Work area passed from IHKEOS, which contains the following:

DCB
DSCB
JFCB

Miscellaneous work areas

TUB
- TUBIRLSA - area to save caller's register 13
- TUBPRMLS - work areas
- TUBUFFAD - contains address of user buffer

User buffer - contains data set name, member name, and ECBs

Exits

Normal - return to calling routine
Error - none

Attributes

Reentrant and nonresident

CENTRAL COMMAND PROCESSORS

RJE/CRJE CENTRAL COMMAND SCHEDULING ROUTINE (IGC1503D)

Entry Point

IGC1503D - when SVC 34 gives control to this entry point, register 2 points to the following 40-byte extended save area:

Bytes 0-3	-	address of XCTL routine used to send error messages to the central oonsole
Bytes 4-7	-	zero
Bytes 8-15	-	IGC1503D
Byte 16	-	error message code
Bytes 17-19	-	address of input buffer (first two bytes of buffer contain the total length; second two bytes are unused; remainder of buffer contains the total input including the command verb.)
Byte 20	-	verb code: 120 - MSG 124 - CENOUT 128 - BRDCST 132 - USERID 136 - SHOW
Bytes 21-23	-	address of operand field of the command (the first nonblank character following the verb); zero if there are no parameters
Bytes 24-31	-	command verb, left-adjusted and padded with blanks.
Bytes 32-39	-	not used

Function

SVC 34 gives control to the IGC1503D routine whenever an RJE or CRJE central command is entered through the central console or the card reader. This routine builds a command input buffer (CIB) and chains it to either the RJE or CRJE command queue. The C required on all CRJE central commands for systems having both RJE and CRJE is used to determine the appropriate command queue.

This routine posts the communications ECB in the CRJE CSCB (Command Scheduling Control Block). This is done so that the interface routine will receive control from the CRJE dispatcher.

The IGC1503D routine handles all RJE and CRJE commands except START, STOP, and MODIFY. When a MODIFY command is entered at a central console, the CSCB marking routine, IGC0703D, recognizes it and identifies the procedure name as being the one for CRJE. The IGC0703D routine sets up the command input buffer (CIB) and then posts the communications ECB for the MODIFY command. The IHKCCI routine then gets control and interfaces with the IHKCC2 routine to process the command.

External Routines

IGC0503D	-	(Message module) to send error messages to the central console

Table/Work Areas

None

Exits

Normal	-	return via register 14
Error	-	IGC0503D is entered via an XCTL, and control is not returned. Register 2 contains the address of the extended save area. An error code in byte 16 requests a particular prepared error message, and bytes 24-31 of the extended save area are inserted in the variable portion of the message.

Attributes

Reentrant and nonresident

CENTRAL COMMAND INTERFACE MODULE (IHKCCI)

Entry Point

IHKCCI

Function

This routine provides an interface between the RJE/CRJE command scheduling module (IGC1503D) and the CRJE central command processors.

This routine receives control from the CRJE dispatcher after the command scheduling routine has posted the communications ECB in the CRJE CSCB (command scheduling control block). The CIB can be found using EXTRACT. The command and its operands are inspected to determine which of the command processors is needed to perform the requested functions. All of the processors are nonresident, so a load request is sent to the loader/controller for the particular processor needed. Control is then passed to this processor with register 1 pointing to a two-word parameter list containing the address of the AVT and the address of the CIB. If the command is a STOP command, the loading and linking of the command processor is not done with the loader/controller but with the LINK macro.

When the processing of the command has been completed and control is returned, the loader/controller is requested to delete the module. This routine dequeues the CIB and returns control to the dispatcher.

Before the loader/controller is requested to load or delete a command processor module, a test is made to determine if the loader/controller task has not abended. A similar test is made after the loader/controller completed the load/delete of the module. If the loader/controller has abended in any case, control is returned to the dispatcher.

External Routines

IHKLDC - to load and delete nonresident central command processors
IHKCC1 - to process the following central commands:

SHOW USERS
SHOW JOBS

IHKCC2 - to process the following central commands:

SHOW LERB
SHOW BRDCST
MODIFY

IHKCC3 - to process the BRDCST central command
IHKCC4 - to process the SHOW MSGS and MSG D=userid central commands
IHKCC5 - to process the CENOUT central command
IHKCC6 - to process the SHOW SESS and SHOW SESSREL central commands
IHKCC7 - to process the USERID central command
IHKCC8 - to process the MSG and SHOW ACTIVE central commands
IHKSTP - to process the STOP command

Tables/Work Areas

CIB
Save area
AVT
CCT
  CCTSLECB - address of ECB

Exits

Normal - return to dispatcher
Error - none

Attributes

Resident and serially reusable

SHOW USERS AND SHOW JOBS CENTRAL COMMAND PROCESSOR (IHKCC1)

Entry Point

IHKCC1 - register 1 must point to a two-word parameter list containing the address of the AVT and the address of the CIB.

Function

Since the functions to be performed depend upon the command verb and its operands, this routine contains a table with valid operands of the command verbs and a branch code for each. This routine first inspects the command verb. Then the address of the operand table is passed to a subroutine. If the operand is not in the table, an error message is sent to the central operator, and control is returned to the interface routine.

The following processing is performed according to the command verb and the operands specified.

SHOW USERS: Each UVR entry is read into the user buffer. If the user is active, his TUB is found. The time that the user has been active is computed and sent to the central operator along with the userid, the line address, and the ACTIVE USER message. If the user is not active, the INACTIVE CRJE USER message with the userid is sent to the central operator.

SHOW JOBS: The IHKAFI module is used to read each RJCT entry into the user buffer. The second bit of the RJCTFLGS field is tested to find out if the job is complete. A message is then sent to the central operator indicating the name of the job and the status.

If a jobname is specified on the command, only that particular RJCT entry is read. A message is then sent to the central operator indicating the status of the job.

External Routines

IHKMSG - (entry point:IHKMSG) to send messages to the central operator
- (entry point:IHKMSG02) to send supplied error messages to the central operator

IHKAFI - to read and write the UVR file, and to read the RJCT entries

Tables/Work Areas

Dummy TUB (contained in IHKCC1)
TUBGBLNM - set to two for UVR and five for RJCT
TUBAFISW - set to cause records to be read into user buffer (contained in IHKCC1)
TUBCLBAD - pointer to CLB to get line address of user
TUBGBLKY - used in IHKAFI

TUBPARM1-TUBPARM5 - contains parameters for IHKMSG
TUBRAFBF - address of user buffer
TUBSIZE - used in allocating storage for TUB
TUBUFFAD - address of user buffer

CIB - contains the command and the operands

CLB
CLBLINE - line address

RJCT
RJCTUSER - userid
RJCTFLGS - job complete bit checked
RJCTJOB - jobname

TUB
TUBTIME - used in computing time since LOGON
TUBNEXT - used to find next TUB

UVR
UVTCNTL1 - (UVRACTVN) active bit checked

Exits

Normal - return to the interface routine with a 0
Error - None

Attributes

Reentrant and nonresident

SHOW LERB, SHOW BRDCST, AND MODIFY CENTRAL COMMAND PROCESSOR (IHKCC2)

Entry Point

IHKCC2 - register 1 must point to a two-word area containing the address of the AVT and the address of the CIB.

Function

This routine checks the command verb and the operands. A branch code is obtained from a table according to the operands specified. If the operand is not found in the table, an error message is sent to the central operator and control is returned to the interface routine.

SHOW LERB: If a line address is specified, it is checked for length and the first character is checked for a zero. If these tests are passed, the relative line number and the DCB address are obtained from the corresponding CLB. If the line address is invalid or is not found, an error message is sent to the central operator. If the specified line is not open, a message is sent. Otherwise, control is passed to the IECTLERP routine, which will send the line error values to the central operator. On a SHOW LERB for all lines, the DCB address and the relative line number of all lines that are open are sent to the IECTLERP routine. When a SHOW LERB command is entered from any console besides the main one, the response is sent to the main console.

SHOW BRDCST: In this case control is passed to the IHKMSG03 entry point of the message writer to send all broadcast messages to the central operator.

MODIFY: When several line address are specified, they must be contained inparentheses. If any errors are encountered, a message is sent to the operator and the next line address is inspected.

For a MODIFY A= command, if the line is open and is not active, and a STOP command has not been issued, this routine posts the ECB, stores the console ID in the CLB, and sets the modify bit in the CLB. If the line is active or is not open, the routine sends a message to the central operator and processes the next line specified.

For a MODIFY D= command, if the line is active, this routine sets the stop bit in the CLB. If the read bit is on, a HALT I/O SVC is issued. The console ID is then stored in the CLB and the modify bit in the CLB is set. If the line is not active, a message is sent to the central operator.

External Routines

IECTLERP - to send line error values to the central operator

IHKMSG - (entry point:IHKMSG) to send messages to the central operator
- (entry point:IHKMSG02) to send supplied error messages to the central operator
- (entry point:IHKMSG03) to send all broadacast messages to the central operator

Tables/Work Areas

CIB - contains command verb and operands

CLB
- CLBSTATS - (CLBACTVN) active bit checked
- CLBSTATS - (CLBMODYN) set for MODIFY
- CLBREQST - (CLBSTOPN) set for MODIFY D=
- CLBLDECB - pointer to DECB
- CLBLINE - physical line address
- CLBMCSCD - console ID stored
- CLBSAECB - checked for X'FF'

DECB of line - read bit is checked

IOB - RESETPL flag is set for MODIFY D=

DCB of line DCBFLGS checked for DCBOPEN
- -offset into line IOB and address of line IOB used
- -address of DEB used

Dummy TUB (contained in IHKCC2)
- TUBPRMLS - used for parameter list

Exits

Normal - return to interface routine with 0 return code in register 15

Error - none

Attributes

Reentrant and nonresident

BRDCST CENTRAL COMMAND PROCESSOR (IHKCC3)

Entry Point

IHKCC3 - register 1 must point to a two-word parameter list containing the address of the AVT and the address of the CIB.

Function

The command verb is inspected first. If the command is anything other than BRDCST, control is returned to the interface module with a code of 0 in register 15.

The possible operands and the functions performed in each case are as follows:

DELETE	-	All broadcast messages are deleted.
nnnn	-	The broadcast message having the specified identifier is deleted.
'text'	-	This text is added to the broadcast message data set as a broadcast message with an identifier that is ten greater than the highest identifier currently being used. If it is not possible to increment the current highest identifier, an error message is sent to the central operator.
nnnn,'text'	-	If a broadcast message exists with the specified identifier, it is replaced with the text given on the command. If a message does not exist with this identifier, the text is inserted at the proper place with the specified identifier.

The broadcast message identifier must be numeric and no more than four characters in length. The text of the message cannot exceed 40 characters in length. If any errors are detected, a message is sent to the central operator. When the maximum number of broadcast messages is reached, the message is added and two messages are sent to the central operator. One message indicates that the broadcast message file is full and the other is an error message.

External Routines

IHKAFI - to read BRDCST message file and to replace, add, or delete records as specified

IHKMSG - (entry point:IHKMSG) to send messages to the central operator

- (entry point:IHKMSG01) to send supplied error messages to central operator

Tables/Work Areas

CCT

CCTBRDNO	-	checked when a message is to be added to file; incremented when a message has been deleted; decremented when a message has been added; when field becomes zero, message is sent to central operator.

CIB - contains command and operands

Dummy TUB (contained in IHKCC3)

TUBGBLNM	-	set to 4 for BRDCST file
TUBAFISW	-	set to read records into user buffer
TUBGBLKY	-	used by IHKAFI
TUBPARM1-TUBPARM5	-	parameters for IHKMSG
TUBRAFBF	-	address of user buffer
TUBSIZE	-	used in allocating storage for TUB
TUBUFFAD	-	address of user buffer

Exits

Normal - return to interface routine with 0 return code in register 15

Error - none

Attributes

Reentrant and nonresident

SHOW MSGS AND MSG D=USERID CENTRAL COMMAND PROCESSOR (IHKCC4)

Entry Points

IHKCC4 - register 1 must point to a two-word parameter list containing the address of the AVT and the address of the CIB.

Function

This routine first checks the command verb and the operands. If any operands are invalid, an error message is sent to the central operator. Once the operands have been validated the following functions are performed according to the command and the operands specified:

SHOW MSGS: The CRJE DELAYED MESSAGES message is sent to the central operator followed by the delayed messages. The userid of the recipient is placed in the first eight bytes of each message. When all messages have been processed and sent to the central operator, the END DELAYED MESSAGES message is also sent. If there are no messages, the NO CRJE MESSAGES message is sent.

SHOW MSGS,USERID: Processing is the same as for the SHOW MSGS command except that only the messages for the specified user are displayed at the central console.

MSG D=USERID: All messages in the delayed message file for this user are deleted and if this is a valid userid, the delayed messages available bit in the UVR is turned off. A message is sent to the central operator indicating that there were no messages for the user or that the delayed messages for the user have been deleted.

External Routines

IHKAFI - to manipulate delayed message and UVR global files
IHKMSG - (entry point:IHKMSG) to send delayed messages to the central operator
- (entry point:IHKMSG02) to send supplied error messages to the central operator

Tables/Work Areas

CIB - contains command verb and operands

UVR

URVCNTL1	- (UVRMSG) turned off when delayed message is deleted.

Dummy TUB (contained in IHKCC4)

TUBGBLNM	- set to 2 for UVR file and 3 for delayed message file
TUBGBLKY	- used by IHKAFI
TUBAFISW	- set to read records into user buffer
TUBPARM1-TUBPARM5	- used for parameters for IHKMSG
TUBRAFBF	- address of buffer to contain records
TUBUFFAD	- address of user buffer
TUBSIZE	- used in allocation of storage for TUB

Exits

Normal - return to interface routine with 0 in register 15
Error - none

Attributes

Reentrant and nonresident

CENOUT CENTRAL COMMAND PROCESSOR (IHKCC5)

Entry Point

IHKCC5 - register 1 must point to a two-word parameter list that contains the address of the AVT and the address of the CIB.

Function

The CENOUT command allows the central operator to have a CRJE user's job output processed by a central installation output writer. This routine first checks the operands specified on the command. The only operands allowed on this command are J=jobname and C=class, where jobname is a one to eight character name and class is one character. Any variation is considered an error, the central operator is notified, and control is returned to the interface routine.

If the operands are valid, the RJCT is searched to determine if the job is in the system. If an entry is not found for the job, the JOB NOT IN SYSTEM message is sent to the central operator and control is returned to the interface routine. If the job is in the system but has not completed processing, the JOB NOT COMPLETE message is sent and control is returned. Otherwise, the TUBs are checked to determine if the job is already queued for delivery. If it is, the JOB WAITING DELIVERY message is sent and control is returned.

If the job is not already queued for delivery, the entry in the RJCT is deleted, and the queue manager is called to read a record for the job. The queue manager assigns queue space for the output records. Each record is processed, depending upon whether it is a data set block or a system message block, and is moved from the old output queue to the new output queue. A zero record is the last record written on the new queue. The CENOUT message is sent to the central operator, the space on the old queue is deleted, and control is returned to the interface routine.

If the service task has abended when the queue manager is to be called or an error causing the service task to abend when calling the queue manager occurred, control is returned to the interface routine, IHKCCI. In the case of an I/O error in calling the queue manager, an error message is sent to the central operator, and control is returned to the interface routine.

External Routines

IHKAFI - to gain access to RJCT
IHKMSG - (entry point:IHKMSG) to send messages to the central operator
IHKDSP - to wait for event completion
IHKCCS - to scan operands for commas
IEFQMSSS - to gain access to output queues
IEFQDELE - to delete job from queue

Tables/Work Areas

RJCT
18-word save area
176-byte buffer for queue manager
44-byte queue manager parameter area
Dummy TUB (contained in IHKCC5)

TUBGBLKY	-	used by IHKAFI
TUBAFISW	-	set to indicate use of optional buffer
TUBGBLNM	-	used by IHKAFI
TUBAFBF	-	address of buffer
TUBPARM1-TUBPARM5	-	used for parameters for IHKMSG

TUB

TUBNEXT	-	used to find next buffer
TUBRJCT	-	checked to see if job output has already been requested

CCT

CCTCSECB	-	used for passing control to IHKSRV routine
CCTJOBS	-	incremented when entry for job in RJCT is deleted
CCTSYSCE	-	interrogated to determine the CRJE SYSOUT class

RJCT

RJCTFLGS	-	tested for job completion
RJCTJOB	-	contains jobname
RJCTQMPA	-	queue manager parameter area
RJCTQMPE	-	extension of QMPA

AVT - addresses of entry points and of CCT and TUB

Exits

Normal - return to the interface routine with 0 in register 15
Error - none

Attributes

Reentrant and nonresident

SHOW SESS AND SHOW SESSREL CENTRAL COMMAND PROCESSOR (IHKCC6)

Entry Point

IHKCC6 - register 1 must point to a two-word parameter list containing the address of the AVT and the address of the CIB.

Function

The command verb is first inspected for SHOW. The address of an operand table containing valid operands of the SHOW verb and a branch code for each is passed to a subroutine. If the operand is not in the table, an error message is sent to the central operator, and control is returned to the interface routine.

The following processing is performed according to the operands specified on the command verb:

SHOW SESS: If MCS is in the system and a general SHOW SESS command is issued, the UCM entry for the requesting console is found. The UCMMSGE (X'0800') bit in the UCMMSG field is set to indicate that SESS is in effect. If it is a composite console, it has two UCM entries and the bit in the second entry is set. The count (CCTSESS) of consoles that have issued SESS commands is incremented so that the central operator will receive notification as CRJE users log on and off the system. The SESS IN EFFECT message is sent to the central operator.

If MCS is not in the system when a general SHOW SESS command is issued, only the CCTSESS count is incremented.

If a userid is specified on the command, the UVR for that user is read. If MCS is not in the system, the UVRSESS bit in the UVRCNTL1 field is set so that the central operator will be notified as this user logs on and off the terminal. If MCS is in the system, the UVR is checked to determine if this console has already requested a SHOW SESS for this user. The UCM entry for this console is checked to see if a general SHOW SESS is in effect for that console. If neither is in effect, the console ID is stored in either the UVRCID1, UVRCID2, or UVRCID3 field. If a SHOW SESS command is in effect for that console, or if the three fields in the UVR are already filled, a message indicating the problem is sent to the central operator. After the console ID is stored in the first available field in the UVR, the SESS IN EFFECT message is then sent to the central operator.

If a general SHOW SESS is in effect, a SHOW SESS for a specific user is not allowed at that console.

SHOW SESSREL: The CCTSESS count in the CCT is decremented so that the central operator will no longer be notified as users log on and off. If MCS is in the system, the UCM entry for that console is found and the UCMMSGE bit (X'0800') in the UCMMSG field is turned off to indicate that a SHOW SESS is no longer in effect. The SESS RELEASED message is sent to the central operator.

If a userid is specified on the command and if MCS is not in the system, the UVRSESS bit in the user's UVR entry is turned off. If MCS is in the system, the console ID in the UVRCD1, UVRCID2, or UVRCID3 field is set to zero. In either case the SESS RELEASED message is sent to the central operator.

External Routines

IHKMSG - (entry point:IHKMSG) to send response messages to the central operator
- (entry point:IHKMSG02) to send supplied error messages to the central operator

IHKAFI - to read and write the UVR entries

Tables/Work Areas

CIB - contains command and operands

CCT

CCTESS	-	checked for SESS in effect; incremented on a SHOW SESS; decremented for a SHOW SESSREL.

CVT

CVTUCM	-	pointer to the UCM base

UCM

UCMID	-	checked for requesting console;
UCMMSG	-	(UCMMSGE) set for SHOW SESS; turned off for SESSREL.
UCMMODE	-	(UCMMCS) checked if MCS is in system.
UCMVEL	-	last entry of UCM
UCMVEZ	-	length of each UCM

UVR

UVRCID1(-3)	-	console ID for MCS in system
UVRCNTL1	-	checked for SESS in effect for this user. When MCS is not in system, set to one for SHOW SESS for this user; turned off for SESSREL.
UVRMXCON	-	maximum number of console IDs for SHOW SESS

Dummy TUB (contained in IHKCC6)

TUBGBLNM	-	set to 2 for UVR
TUBAFISW	-	set to cause records to be read into user buffer (contained in IHKCC6)
TUBGBLKY	-	used by IHKAFI
TUBARM1-TUBPARM5	-	used for parameters for IHKMSG
TUBRAFBF	-	address of user buffer
TUBSIZE	-	used in allocating storage for TUB
TUBUFFAD	-	address of user buffer

Exits

Normal - return to interface with 0 return code in register 15
Error - none

Attributes

Reentrant and nonresident

USERID CENTRAL COMMAND PROCESSOR (IHKCC7)

Entry Point

IHKCC7 - register 1 must point to a two-word parameter list containing the address of the AVT and the address of the CIB.

Function

The command verb is checked for USERID and the address of a table containing valid operands and branch codes is passed to a subroutine. If the operand specified on the command is not in the table, an error message is sent to the central operator, and control is returned to the interface routine.

The following functions are performed according to the operands specified on the command:

```
USERID  A=
USERID  D=
USERID  S(UPPRESS)
USERID  R(ESUME)
```

When a user is to be added, the UVR entries are checked to see if the userid is a duplication. The userid and password are checked for validity. Each must begin with an alphabetic character and the other characters must be alphameric. If the userid has not already been assigned, the userid and password are inserted into a UVR entry. The remainder of the UVR entry is initialized, and the count of users that can be added to the system is decremented. The ADDED TO USER LIST message is sent to the central operator specifying the user who was added. If no more users are allowed on the system, the USER LIST FULL message is sent to the central operator.

If the password entered at the terminal matches the password in the UVR entry for the user who is to be deleted, the UVR entry is deleted. When a user is deleted, the count of users allowed on the system is incremented. The DELETED FROM USER LIST message is sent to the central operator indicating the user who has been deleted. If the user is active at the time, he cannot be deleted.

USERID SUPPRESS: The CCTSUP bit is set; no users can log on as long as this bit is set. The LOGONS SUPPRESSED message is sent to the central operator.

USERID RESUME: The CCTSUP bit is turned off; this makes it possible for users to log on and off the system. The LOGONS RESUMED message is sent to the central operator.

External Routines

IHKMSG - (entry point:IHKMSG) to send message to central operator
IHKAFI - to read, write, or delete the UVR entries

Tables/Work Areas

CCT

CCTOPT1	-	(CCTSUP) set for USERID SUPPRESS: turned off for USERID RESUME.
CCTUSERS	-	decremented when a user is added; incremented when a user is deleted.

CIB - contains command and operands

Dummy TUB (contained in IHKCC7)

TUBGBLNM	-	set to 2 for UVR
TUBAFISW	-	set to cause UVR entry to be read into user buffer (contained in IHKCC7)
TUBGBLKY	-	used in reading and writing UVR entries
TUBPARM1-TUBPARM5	-	used for work area and for parameters for IHKMSG
TUBRAFBF	-	address of user buffer
TUBSIZE	-	used in allocation of storage for TUB
TUBUFFAD	-	address of user buffer

UVR

UVRCNTL1	-	(UVRACTUN) active bit checked

Exits

Normal - return to interface routine with a 0 return code in register 15
Error - none

Attributes

Reentrant and nonresident

MSG AND SHOW ACTIVE CENTRAL COMMAND PROCESSOR (IHKCC8)

Entry Point

IHKCC8 - register 1 must point to a two-word parameter list containing the address of the AVT and the address of the CIB.

Function

The command verb is inspected for MSG or SHOW, and the address of a table containing valid operands and branch codes is passed to a subroutine.

If the operand specified on the command is not found in the table, an error message is sent to the central operator, and control is returned to the interface routine.

MSG: The possible operands and the functions performed in each case are as follows:

M='text'	-	The message text is sent to all active users.
M='text',U=userid	-	The message text is sent to the specified user if he is active
M='text',U=userid,Q	-	The message text is sent to the specified user if he is active. Otherwise, the message is queued and will be sent to the user when he logs on.

The operands of the MSG command are keyword operands and are not positional. The userid is verified by the message writer; if an invalid userid is found, the message is not sent. A message may be sent to the central operator indicating that the message was not sent. As with broadcast messages, the message text cannot exceed 40 characters.

SHOW ACTIVE: The TUBs are searched and the time since LOGON is computed for each user having a TUB. The ACTIVE USER message together with the userid, the line address, and the session time for each active user is sent to the central operator.

If the command is SHOW ACTIVE, NUMBER, the only message sent to the central operator is the ... ACTIVE USERS message with the correct number inserted.

When a SHOW ACTIVE command or a MSG command for all active users is entered, the TUBs are searched initially and the TUBCMSG bit is set. The TUBs are then searched again, and when the bit is found on, it is turned off. If the command was SHOW ACTIVE, the active message is sent to the central console for that user. If the command was MSG, the delayed message is queued. The search is repeated until no more TUBs are found with the bit on.

External Routines

IHKMSG
- (entry point:IHKMSG) to send response messages to central operator
- (entry point:IHKMSG01) to queue messages for user
- (entry point:IHKMSG02) to send supplied error messages to the central operator

Tables/Work Areas

CIB - contains command and operands

TUB

TUBNEXT	-	used to find next TUB
TUBFLG1	-	(TUBCMSG) initially set, then turned off when message is sent.

Dummy TUB (contained in IHKCC8)

TUBPARM1-TUBPARM5	-	used for parameters for IHKMSG
TUBRAFBF	-	address of user buffer
TUBSIZE	-	used for allocation of storage for TUB
TUBUFFAD	-	address of user buffer

Exits

Normal - return to interface routine with a 0 return code in register 15
Error - none

Attributes

Reentrant and nonresident

JOB TERMINATION SUBTASK

JOB TERMINATION HANDLING MODULE (IHKSDQ)

Entry Point

IHKSDQ - (The IHKBGN module passes control to this when initialization is complete.)

Function

The job termination handling routine first passes control to the dequeue module (IHKDEQ) to perform job end processing. When control returns, this module determines whether a STOP (either normal or abnormal) is pending. It then sets up the appropriate ECB address and return address for waiting in the dispatcher.

If a STOP is not pending, this IHKSDQ module waits on the ECB given to the OS queue manager (IEFQMDQ2) by the job end processor (IHKDEQ). This ECB will be posted by OS when a job is queued on the CRJE SYSOUT queue. Return is set to the point where the IHKDEQ module is invoked.

If a STOP is pending, or if the module IHKDEQ cannot be loaded (abend of the loader/controller task), the IHKSDQ module waits on its STOP ECB. When control is returned from the dispatcher, this module returns to the IHKBGN module.

External Routines

IHKDEQ - to perform job end processing

Tables/Work Areas

18-word save area
CCT
CCTOPT - (CCTCLS) tested for STOP CRJE condition

Exits

Normal- return to the IHKBGN module
Error - none

Attributes

Resident and serially reusable

Entry Point

IHKDEQ- register 1 must contain the address of the AVT.

Function

The job end processor provides CRJE with a means of recognizing when CRJE jobs (entered by the SUBMIT command) are complete and of notifying the CRJE user that his job is complete.

When the job end processor gets control, a flag is set indicating that it is processing job output. This is done so that at STOP time, the STOP processor (IHKSTP) will know whether or not job termination is in process. If this routine is processing a job at STOP time, it returns control to the job termination handling routine (IHKSDQ), which recognizes the STOP command. However, if this routine is not processing a job when the STOP command is received, the IHKSDQ module will be waiting in the CRJE dispatcher for work. In this case the STOP processor (IHKSTP) must enter the IHKSDQ module to force it out of the wait so that closedown can proceed.

Before control is returned to the IHKSDQ module the in-process flag and the job termination initialization bit are turned off. The job termination initialization bit is checked to insure that all jobs that are complete are dequeued from OS before any line I/O is attempted by CRJE. The bit is checked by the CRJE system administrator, and no I/O is initiated until it is turned off.

The job end processor then branches to the OS queue manager dequeue routine (IEFQMDQ2) to look for jobs on the CRJE SYSOUT queue. If the IEFQMDQ2 routine encountered an I/O error, an error message would be sent to the central operator, and control would be returned to the job termination-handling routine (IHKSDQ).

If a job is found on the output queue, a search is made for an RJCT entry having the jobname of the job that was dequeued. If an RJCT entry is not found, or if the jobname is JOBFAIL but the original name cannot be determined from SMBs, control is passed to the IHKRER01 routine to delete the job. Then the next job on the SYSOUT queue is processed.

If the RJCT entry was found, a check is made to see if the RJCT entry is marked job completed. If it is marked job completed, this means that a job with a duplicate jobname has been entered at the central system. Control is passed to the IHKRER01 routine to delete the duplicate job. Then the next job on the SYSOUT queue is processed. If the RJCT entry is not already marked completed, it is now marked complete, updated, and written in the global file. A notification message is queued for the user, and the next job on the SYSOUT queue is processed.

If the IHKAFI routine encountered an error, the CRJE abnormal termination bit in the CCT is set, the central command subtask is posted to stop, and control is returned to the IHKSDQ routine.

If IHKDEQ cannot perform its function because of an abend of a CRJE task (service or loader/controller), control is returned to IHKSDQ to wait on the closedown.

External Routines

IEFQMDQ2	-	(OS queue manager dequeue) to dequeue jobs from CRJE SYSOUT queue
IHKRER	-	(entry point:IHKRER01) to delete OS queue entry and if indicated, RJCT entry
IHKDSP	-	CRJE dispatcher for dummy wait

IHKMSG - (entry point:IHKMSG02) to send or queue messages for terminal user
IHKAFI - to access JBTBLS global file

Tables/Work Areas

18-word save area
80-byte RJCT buffer
176-byte SMB-DSB buffer
44-byte QMPA and QMPE buffer

Exits

Normal - return to IHKSDQ
Error - none

Attributes

Reentrant and nonresident

SYSTEM ADMINISTRATOR

The three principal functions of the system administrator are reflected in its three modules: command analyzer (IHKCMD), error recovery routine (IHKERR), and dispatcher (IHKDSP).

The system administrator makes available the CRJE function that the user requested when he typed in a command at his terminal. The command analyzer first examines the command for validity. If the command is invalid, it is rejected and an error message is sent to the user. The parameters of a valid command are taken out of the line buffer and put in a PPT. The routine that is responsible for performing the specified function is loaded into the transient area, if it is nonresident, and then control is passed to it. When the command processor has completed its function, the command analyzer again gets control. A request to delete the command processor, if it is nonresident, is passed to the loader/controller. Then the user is asked by the line administrator to enter a new command.

The message writer and command processors use the same set of return codes. The line administrator and the command processors pass different return codes back to the command analyzer, indicating the occurrence of abnormal conditions. The command analyzer inspects these return codes and takes the appropriate action. Recovery from line errors and active area I/O errors is handled by the nonresident routine IHKERR.

The dispatcher is responsible for making the CRJE services available to all users. When a routine cannot continue (waiting for completion of I/O operation), control is returned to the dispatcher. The dispatcher saves the registers and establishes a connection between the ECB that was returned to the dispatcher and the STCB representing the subtask that yielded control. The dispatcher scans the STCBs for a posted ECB. When one is found the registers are restored, and return is made to the module that had entered the dispatcher to wait for an event.

COMMAND ANALYZER MODULE (IHKCMD)

Entry Point

IHKCMD - register 1 must point to a CLB.

Function

A CREAD I macro is issued at the very beginning for every connected line. In response to this macro, the line administrator returns control to the command analyzer with a zero-length record in the user buffer. The command analyzer then prompts the user for his first command. The user buffer contains the command as it was typed in at the terminal. The command is checked for validity, and a PPT is built. If the command is continued on a succeeding line, another PPT is allocated and chained to the previous one for as many lines and PPTs as necessary.

Before control is passed to the command processor (determined by command verb), the command analyzer takes the following actions:

- Before the first command is analyzed, the LOGON suppression flag is checked. If it is on, a message is sent to the user and a CREAD I macro is issued.
- The first command entered must be a LOGON command; any other command is rejected. The user is prompted once more to enter a LOGON command. After the second non-LOGON command is received, a CREAD I macro is issued.
- A zero-length record in major command mode is ignored, and the user is prompted for another command.
- For a zero-length record in edit mode, the command code for the IHKIRL01 routine is inserted into the PPT.
- For a zero-length record while the user is making corrections, the command code for the IHKIRL02 routine is inserted into the PPT.
- If an active file has the TEXT attribute, the command analyzer sets the TUBTEXTN flag before the CREAD macro is issued. This tells the line administrator not to translate the incoming command to upper case. For verification, the command analyzer translates the command verb and the operands (with the exception of the two text operands in the CHANGE command and the entire Implicit command) to upper case.
- If a CLIST is being processed, the LOGON, LOGOFF and EXEC commands are rejected.
- A command with unpaired parentheses in its operands is rejected, and a message is sent to the user.
- A command with unpaired quotes in its operands is rejected, and a message is sent to the user.
- Double quotes are resolved to single quotes.
- A command containing a zero-length operand, e.g. (), is rejected.
- A command containing an operand longer than 56 characters is rejected (except on the Implicit command), and a message is sent to the user.
- The two text operands in the CHANGE command cannot exceed 40 characters each.
- If operands on a command are separated by multiple commas, a warning is sent to the user. The command is accepted, however.
- If the command verb and the first operand are separated by one or more commas, a warning message is sent to the user; the command is accepted, however.

- For a LISTBC command the broadcast messages required flag (TUBBRD) is set, and control is given to the point in the command analyzer where all the command processors return.

- If a LOGON command is entered while a user is already logged on, control is given first to the LOGOFF command processor. Upon returning, the LOGOFF processor requests that control be given to the LOGON processor.

- If the command processor specified by the command code in the PPT is nonresident, a load request for the appropriate module is passed to the loader/controller, and control is given back to the dispatcher until the requested module is loaded. If it is determined that IHKLDC abended, control is passed to IHKERR with a LINK macro instruction.

After all of the preceding functions are checked or performed, control is passed to the requested command processor with register 1 pointing to a two-word parameter list containing the address of the TUB and the address of the AVT. For installation-provided commands, control is passed to the entry point specified in the CRJETABL macro with register 1 pointing to the UCCT (User Command Control Table). Upon return from the installation-provided command, the return code is checked. If the return code is zero, the next command is read from the terminal. If it is not zero, the 120-byte message in the user buffer is sent to the terminal before the user is prompted for another command.

When control is returned to the command analyzer, the command code in the PPT is checked to determine whether the returning processor is resident or nonresident. Nonresident processors must be deleted. Before the deletion request is passed to the loader/controller, register 15 is checked for a negative value. If it is a negative value, it is the command code of the processor that is to receive control next. If the next command processor is nonresident and the returning processor is nonresident also, a deletion-load request is passed to the loader/controller. If only the returning processor is nonresident, a deletion request is passed to the loader/controller. During the time the loader/controller performs the deletion or the deletion/load function, the command analyzer waits in the dispatcher. If register 15 is not negative, it contains the return code used to select the action to be performed.

If the return code is 0 (good return) and an EXEC command is not being processed, the following actions are performed:

- All queued messages are sent.

- If the returning processor is the LOGOFF processor, the PPTs are deallocated, the save area is freed, and a CREAD I macro is issued.

- If the returning processor is the LOGON processor and the LOGON process was not successful, the action taken is the same as returning from the LOGOFF processor.

- For all other command processors, the PPTs are deallocated. If the user is in edit mode, the directory entry is checked. If the directory entry contains the TEXT attribute, the lower case preservation flag (TUBTEXTN) is set for the line administrator. Then the user is prompted to enter a new command.

If the return code is 0 and the EXEC command is being processed, the following actions are performed:

- If control was not returned from the EXEC processor (IHKED1), all queued messages are sent, the PPTs are deallocated, and a dummy PPT is used to go to the EXEC processor to get the next command from a CLIST file.

- If control was returned from the EXEC processor (IHKED1), then the user buffer contains the next command that must be analyzed by the command analyzer. If the TUBEXLST flag is set, the command is written at the terminal before it is analyzed.

If the return code is 4 (interrupt, line error, or closedown) and no TUB is allocated for the line, a dummy PPT is used to branch to the error recovery routine (IHKERR) in the same way that control is given to the command processors. The parameter list passed to the IHKERR routine contains a pointer to the AVT and a pointer to the CLB.

If the return code is 4 and a TUB is allocated, the following actions are taken:

- If the TUBRAKEN flag is not set and if no user is logged on, the PPTs are deallocated, and a dummy PPT is used to branch to the IHKERR routine. If a user is logged on, a PPT is allocated, if necessary, and the command code for the IHKERR routine is inserted into the PPT, and control is given to the IHKERR routine.

- If the TUBRAKEN flag is on, this means that a break has occurred, either real or simulated. The command entered after the break is already in the user buffer. If an EXEC command is not in progress, the command is analyzed and processing continues. If an EXEC command is in progress and the command in the user buffer is not END, the EXEC process is terminated and the new command is analyzed. If the new command is END, only the currently active command is terminated, and control is given to the EXEC processor to get the next command from the CLIST.

If the return code from the command processor or the message writer is 8 (GETMAIN failure), an OUT OF SPACE message is sent to the terminal. If the processor that returned the 8 return code was IHKLGN, then IHKLGF is loaded to call the logoff exit, if it is required. In the case where the processor was IHKLGF, the TUBLGNAB flag is set, and the TUBUSRID field is not cleared, the IHKLGF processor is reloaded. If the EXEC command was in progress, it is terminated. The user is then prompted for the next command. An OUT OF SPACE message is also sent to the central operator.

If a return code of 12 (active area I/O error or subtask abend) is received from the command processor or from the message writer, the TUBAARRN flag is turned on, and control is given to the error recovery routine (IHKERR). This is done by using a dummy PPT if no user is logged on, or by using a real PPT if a user is logged on.

If a return code of 16 (error message lost) is received from the command processor or from the message writer, and the EXEC command is in progress, the EXEC process is terminated. The user is then prompted to enter his next command.

If a return code of 20 (active area out of space) is received from the command processor or the message writer, a message is sent to the central operator, and the user is prompted for his next command.

A return code of 24 is received from the EXEC command processor when an EOF is encountered in a CLIST file. In this case the command analyzer frees the EXEC buffer and prompts the user for the next command.

A return code of 28 is only received from the error recovery routine. The CLBSAECB field is posted in this case and control is given to the dispatcher to wait for the line ECB.

A 32 return code is received exclusively from the IHKERR routine. In this case the command analyzer (IHKCMD) issues a CREAD I macro instruction.

External Routines

IHKCCS - to scan a character string for one or more given characters
IHKMSG - (entry points:IHKMSG, IHKMSG01, and IHKMSG02) to build, queue, and send error messages to the terminal and to the central operator
IHKDSP - to wait for various actions to be completed to perform operations on the lines when CREAD I, CWRITE, CWRITE R, and CREAD R macros are issued
IHKERR - to recover from line errors and from active area I/O errors

All command processors - to process the command contained in the PPT

Tables/Work Areas

GETMAIN save area
CCT
- CCTSUP - indicates that LOGONs are to be suppressed
- CCTUSR - address of the user-supplied command exit

CLB
- CLBSAECB - stop acknowledgment ECB

TUB
- TUBAARRN - indicates active area I/O error
- TUBBRD - indicates broadcast messages are to be sent
- TUBCLBAD - contains the address of the CLB
- TUBCNTEN - indicates a continued command
- TUBCNTSN - indicates that this command is to be continued
- TUBCOMAN - indicates comma found in command scan
- TUBCORRN - indicates syntax error corrections may be made
- TUBDATAL - length of data in user buffer
- TUBDIRAD - contains address of directory entry for active file
- TUBEDIT - indicates edit mode
- TUBEXCAD - contains address of EXEC work area
- TUBEXCLG - length of EXEC work area
- TUBEXLST - indicates the command is to be listed before it is processed on the EXEC command
- TUBIDENT - identifies this control block as a TUB
- TUBLGNRN - indicates user has been prompted for a LOGON
- TUBNOCRN - no carriage return after a line
- TUBNULPN - null parameter is indicated
- TUBPRMLS - parameter list area
- TUBPPTAD - address of the PPT
- TUBRAKEN - break has occurred on the line
- TUBREAKN - break has occurred on the line
- TUBSTOP - closedown is in progress
- TUBTEXTN - active file has text attribute
- TUBUFFAD - address of the user buffer
- TUBWARNN - comma warning message must be sent

User buffer contains command
PPT - used to pass command to command processor
IHKMCL - list of major commands
IHKSCL - list of edit subcommands

DIR
- DIRATEXT - indicates active file has text attribute

Exits

Normal - return to dispatcher
Error - none

Attributes

Reentrant and resident

CRJE DISPATCHER (IHKDSP)

Entry Point

IHKDSP

Function

A subtask is represented by a subtask control block (STCB). For each priority the STCBs are chained together in a circle. The priorities of the STCBs are as follows:

Highest priority: one STCB for each line;

Medium priority: one STCB for job termination handling;

Lowest priority: one STCB for the central command processor.

Each CRJE routine returns to the dispatcher instead of issuing a WAIT macro.

For each priority the AVT contains a pointer to the STCB that last received service. The current priority is saved in the dispatcher. When the dispatcher gets control, it saves the registers (unless register 13 contains zero upon entry, which effectively frees the STCB), places the pointer to the ECB (passed in register 1 by the returning routine) in the ECB list of the multiple wait, and inserts a pointer to this ECB list entry in the corresponding STCB. (See Figure 6.)

The dispatcher now looks for a posted ECB by scanning the STCB circle of the highest priority one time. When a posted ECB is found, the current priority and the pointer to the STCB are saved, the registers are restored, and return is made via register 14. If no ECB is posted, the next lower STCB circle is scanned. Only after an unsuccessful scan through all three circles of STCBs does the dispatcher issue a WAIT macro. When one event that the dispatcher is waiting for occurs, control is returned to the dispatcher, and the scan starts with the highest priority.

External Routines

None

Tables/Work Areas

STCBs
AVT

Exits

Normal - to OS supervisor to wait for completion of an event
Error - none

Attributes

Resident and reentrant

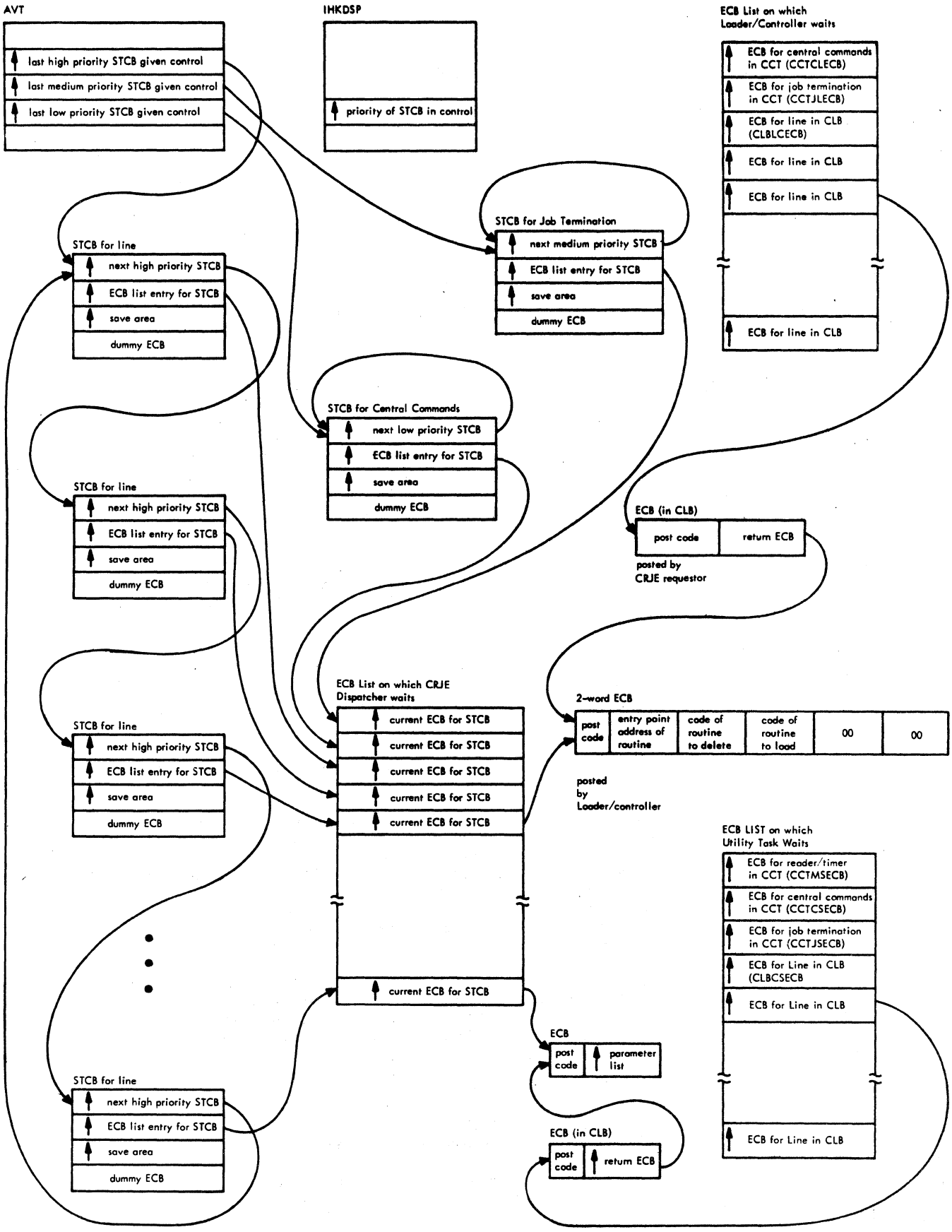


Figure 6. Dispatcher

LINE ERROR AND ACTIVE AREA I/O ERROR RECOVERY MODULE (IHKERR)

Entry Point

IHKERR - The command analyzer (IHKCMD) is the only module that calls this routine. A two-word parameter list is passed. The first word points to the AVT and the second word points to the TUB or the CLB.

Function

If a pointer to the CLB is received in the parameter list, the message OUT OF MAIN STORAGE is sent to the user and to the central operator. A return code of 32 is sent to the command analyzer.

If a TUB is allocated, the following actions are taken:

- If the CCTCLS flag and the CLBSTOPN flag are not set, something must be wrong with the line. The user's session is terminated in the following way: If the EXEC command is in progress, it is terminated and the buffer is freed. If the user is in EDIT mode, the TUBABEND flag is set and control is given to the SAVE processor; If the user is not in EDIT mode he is logged off. The message TP LINE ERROR is queued.

- If the CLBSTOPN flag is set, the TUBSTOP flag and the TUBABEND flag are turned on by the command analyzer. Processing then continues as if the flag was not set. The message LINE DEACTIVATED BY OPERATOR is queued for the user.

- If the CCTCLS flag is set, the session of the user is terminated in the same way as if the flag was not set. The message CRJE CLOSEDOWN is sent to the user, a CREAD R macro is issued, and control is returned to the command analyzer with a return code of 28.

If the TUBAARRN flag is set, the error recovery routine has been called because of an active area I/O error. In this case the following functions are performed:

- If the CCTATERM flag is not yet set, the IHKERR routine sets the CCTCLS, CCTSUP, and CCTATERM flags. The communications ECB is posted with the stop code to stop CRJE, and the CRJE ABNORMAL CLOSEDOWN message is sent to the central operator. The same message is sent to the terminal, and if a user is logged on, the LOGOFF processor is given control after the TUBUTMN flag is set. Then a CREAD R macro is issued, and control is returned to the command analyzer with a return code of 28.

- If the CCTATERM flag is set, the CRJE ABNORMAL CLOSEDOWN message is sent to the user, the TUBUTMN flag is set, and control is then passed to the LOGOFF processor. A CREAD R macro is issued, and control is returned to the command analyzer with a return code of 28.

External Routines

IHKMSG - (Entry point:IHKMSG) to build and send error messages to the terminal user and to the central operator

IHKLAD - issue CREAD R macro and send messages to the user

Tables/Work Areas

GETMAIN for save area
CCT
CLB

TUB

TUBUFFAD	-	indicates user buffer address
TUBUSRID		indicates user's identification sequence
TUBUTMN	-	indicates not going to IHKUTM to mark the user inactive
TUBXLSTF	-	turns off list-commands-before-processing flag for EXEC

AVT

Exits

Normal - return to command analyzer (IHKCMD)
Error - none

Attributes

Reentrant, refreshable, and nonresident

LINE ADMINISTRATOR

MACROS

There are two macros by which routines in CRJE request reading and writing on the telecommunications lines. The CREAD macro specifies the read operation. Two operands are provided: the I operand specifies a read initial operation; the R operand indicates a resetting of a line that has become inoperative. The CWRITE macro indicates writing to the line. The R operand on the CWRITE macro means a combined write/read operation is performed on the line.

CREAD

Register 15 must contain the address of the TUB.
Register 13 must contain the address of the save area.

Function

This macro sets the request flag for the read continue operation, provides register 1 with the address of a one-word parameter list containing the address of the TUB, and generates a branch and link to the IHKLAD routine. The IHKLAD routine performs the read continue operation.

CREAD I

Register 15 must contain the address of the TUB. (Except for the very first time on a line, register 15 must contain the address of the CLB.)

Register 13 must contain the address of the save area.

Function

This macro allows a user to log on at the terminal. It sets the request flag for a read initial operation, provides register 1 with the address of a one-word parameter list containing the address of the TUB

or the CLB, and generates a branch and link to the IHKLAD routine, which will perform the operation.

CREAD R

Register 15 must contain the address of the TUB.
Register 13 must contain the address of the save area.

Function

This macro is used when a line has become inoperative or the central operator has requested that the line be disconnected again. The CREAD R macro causes a FREEMAIN macro to be issued for the line buffer, line save area, and the TUB; the CLBACTVN flag is turned off.

CWRITE

Register 15 must contain the address of the TUB.
Register 13 must contain the address of the save area.
The length of the data to be written must be specified in the half-word field TUBDATAL. The data to be written must be in the user buffer (pointer to user buffer in TUBUFFAD).

Function

This macro sets the request flag for the write continue operation, provides register 1 with the address of a one-word parameter list containing the address of the TUB, and generates a branch and link to the IHKLAD routine. The routine requesting the write operation must set the TUBNOCRN flag in the TUBFLG3 field or the line administrator will remove all trailing blanks that appear on the text to be sent and return the carrier.

CWRITE R

Register 15 must contain the address of the TUB.
Register 13 must contain the address of the save area.
The length of the data to be written must be specified in the half-word field TUBDATAL. The data to be written must be in the user buffer (pointer to user buffer in TUBUFFAD field). If the response is expected to appear on the same line as the text that was sent, the TUBNOCRN flag in the TUBFLG3 field should be turned on.

Note: When specifying the length of data to be written, the number of trailing blanks necessary to provide enough space between the text to be sent and the response expected should be included.

Function

This macro is used to request a combined write/read operation on the line. It is used when a text response to the data being sent is expected. The macro sets the request flag for a write operation followed by a read operation, provides register 1 with the address of a one-word parameter list containing the address of the TUB, and generates a branch and link to the IHKLAD routine. Unless the routine requesting the operation sets the TUBNOCRN flag in the TUBFLG3 field, the line administrator removes all trailing blanks appearing on the text to be sent and returns the carrier.

GENERAL DESCRIPTION

Upon request from a CRJE module, the line administrator performs read and write operations on the communications line using BTAM device-dependent code. The line administrator provides translation of the characters being sent or received from EBCDIC to another transmission code or from the transmission code to EBCDIC.

The line administrator is composed of six modules: IHKLAD, IHKLAB, IHKLAP, IHKLEW, IHKLAT, and IHKLAY. IHKLAD is always used as the entry point when a routine issues one of the line administrator macros. The IHKLAB, IHKLAP, IHKLEW, IHKLAT, and IHKLAY modules are accessible only to the IHKLAD module.

The following functions are performed when the IHKLAD module is entered following the issuance of a CREAD I macro, with register 1 pointing to a parameter list containing the address of the CLB:

- a 72-byte line save area is acquired,
- the address of the save area for the routine issuing the macro and the address of the line save area are saved,
- the CLB is updated,
- the IHKLAP module is called by a branch and link.

The IHKLAP module initiates a read initial operation allowing the user to start his session at the terminal. The user starts his session by entering a carrier return (CR) and an EOB (1050/2740 terminal) or an EOT (2741 terminal). Control is returned to the IHKLAD module. CRJE recognizes that the user wants to log on. The command analyzer then requests that the user be prompted with an underscore.

The IHKLAD module checks the BTAM return code. All nonzero return codes indicate that the I/O operation was not initiated. Errors are flagged and control is returned to the routine that issued the macro. If the return code is 0, the DECB address is put in register 1, and this module waits in the dispatcher for the completion of the operation.

When the operation is completed, the IHKLAD module again gets control. The BTAM completion code is checked and if it is not X'7F', only the lost-data condition is accepted. (A negative response to polling for the 1050 nonswitched is retried by BTAM; a negative response to polling for the 1050 switched is retried by the line administrator.) Otherwise, a line error is flagged, and control is returned to the calling routine with a nonzero return code in register 15.

If a X'7F' return code is received from BTAM and the DECFLAGS field is zero, a GETMAIN macro is issued for a line buffer and a TUB. If the GETMAIN macro is successful, the address of the line buffer + 16 (user buffer) is put in the TUB and the address of the line buffer is put in the CLB. The address of the TUB is added to the chain. The address of the line buffer is put in the TUB. Control is returned to the requesting routine with a length of zero in the TUBDATAL field and a 0 return code in register 15. If the GETMAIN macro fails, the NO SPACE message is sent to the terminal, and control is returned to the requesting routine with a nonzero return code.

The functions performed depend upon the macro that was issued.

CREAD R	TUB is deallocated; TUB chaining is updated; FREEMAIN macro is issued for TUB, line buffer, and 72-byte line save area; active flag in CLB is reset; and control is returned with 0 return code.

CREAD I — TUB is deallocated; TUB chaining is updated; FREEMAIN macro is issued for line buffer and TUB; request flag in CLB is set; and control is passed to the IHKLAP routine, which initiates a read initial operation on the line.

CREAD — Control is passed to the IHKLAB routine, which recognizes read request and returns control to the IHKLAD routine. Control is then passed to the IHKLAP routine, which initiates the read operation. The IHKLAD routine then waits in the dispatcher for the completion of the operation.

CWRITE — The IHKLAB routine recognizes the write request, moves the output text from the user buffer to the line buffer, inserts the necessary line control characters and idle characters to allow the carrier to return, before returning control to the IHKLAD routine. The IHKLAP routine translates the text to be written to the appropriate code for that line and initiates the write operation. The IHKLAD routine waits in the dispatcher for the completion of the operation.

CWRITE R — The sequence of functions performed is the same as for the CWRITE macro, except that the IHKLAP routine recognizes the request for a response and initiates a combined write/read operation before returning control to the IHKLAD routine.

When the operation is completed and the dispatcher gives control back to the IHKLAD module, the BTAM completion code is checked. If a good BTAM completion code is posted in the DECB and the operation was a read, the text that was read is translated from transmission code to internal EBCDIC code. If the character sequence at the beginning of the text contains the ending sequence, the TUBDATAL field is set to zero and control is returned to the requesting routine. If the character sequence at the end of the text contains the ending sequence preceded by a backspace character, the line of input is canceled and the line administrator prompts the user with another read without returning control to the requesting routine.

Otherwise, the text is properly edited in the user buffer, nonsignificant characters such as line feed and backspace are eliminated, the length of the data is computed, and the exact value is put in the TUBDATAL field. Flags are set if no CR was in the text and if EOT was the ending character (1050/2740). Then control is returned to the requesting routine with a 0 return code.

If a bad completion code from BTAM is posted in the DECB, the line administrator tries to recover from those error conditions that are recoverable, as follows:

Lost data - When the user enters too many characters and the input exceeds the line buffer, the IHKLAD routine sends a message to the terminal and prompts the user to reenter the line.

Time-out - On a 1050 terminal without time-out suppression, the user did not enter a character before time elapsed. The IHKLAD routine sends a message to the user and prompts him to reenter the line of input.

Negative response to polling or addressing when a 1050 switched terminal does not respond to polling or addressing (BTAM retries a nonswitched line). The IHKLAD routine repolls the terminal without returning control to the requesting routine. The IHKLAD routine sends

messages and retries when a data check is received on a read from a 2741.

For other error conditions, the IHKLAD module flags the error in the TUB and returns control to the requesting routine with a nonzero return code.

COMMUNICATION LINE ADMINISTRATOR MODULE (IHKLAD)

Entry Points

IHKLAD - control is passed to this entry point whenever one of the line administrator macros is issued. Register 1 must point to a one-word area containing the address of the TUB or the address of the CLB.

Function

If register 1 points to a one-word area containing the address of the CLB, a GETMAIN macro is issued for a 72-byte line save area. If the GETMAIN macro is successful, the save area address is saved, the active flag in the CLB is turned on and the IHKLAP routine is called to issue the read initial on the line. If the GETMAIN macro fails, control is returned to the requesting routine with a nonzero return code in register 15.

If the return code from the IHKLAP routine is bad when line operation is initiated, the inoperative flag in the CLB is set, the line save area is freed, and control is returned to the requesting routine with a nonzero return code. If the return code is good, the line administrator waits in the dispatcher for the completion of the line operation.

When the line operation is complete and control is returned from the dispatcher, a check is made to see if a TUB exists. If a TUB has not been allocated yet and a bad completion code is received from BTAM indicating that the error is a lost data condition, the error is ignored and processing continues normally. If the error is anything besides lost data, the inoperative flag in the CLB is set and control is returned to the requesting routine with a nonzero return code. If a TUB has not been allocated and a good completion code is received from BTAM, a GETMAIN macro is issued for a 156-byte line buffer and a TUB. If the GETMAIN fails, the OUT OF SPACE message is sent to the terminal, if possible; the line save area is freed; and control is returned to the requesting routine with a nonzero return code.

If the GETMAIN is successful, the address of the line buffer and the address of the CLB are stored in the TUB. The entire TUB is zeroed, and the TUB chain is then updated, the address of the last TUB in the chain is put in the AVT, and the address of the user buffer (line buffer +16) is stored in the TUB. The TUB identifier flag (TUBIDENT) is set, and the sequence number from the CLB (CLBLNSEQ) is stored in the TUB. The request flag in the CLB (CLBRDTIF) is turned off, and control is returned to the requesting routine with a zero length in the TUBDATAL field.

Note: Since the request was a CREAD I with the CLB as a parameter, the command anlayzer is assumed to be the requesting routine, and it is assumed that the next request will be a CWRITE R (to prompt the user).

If the request is for reset (CREAD R), the TUB is deallocated, the TUB chain is updated, and if the TUB is the first or last in the chain, the addresses in the AVT are updated. A FREEMAIN macro is issued for the 156-byte line buffer, the 72-byte line save area, and the TUB. Control is then returned to the requesting routine with a 0 return code.

If the request is for a read initial (CREAD I) for a switched line connected to a 1050 or 2740 terminal, a WRITE TTR macro is issued to reset the terminal and prevent the printing of unwanted data at the terminal. A FREEMAIN macro is issued for the line buffer and the TUB. The TUB chain and the TUB addresses in the AVT are updated. The read initial request flag in the CLB is set and control is passed to the IHKLAP routine to initiate the line operation.

For a CWRITE or CWRITE R request the IHKLAB routine is called to check the length of the data to be written and to eliminate any trailing blanks. When control is returned, the TUBOUTAD field is checked to determine whether TABSET is specified. If it is not specified, the IHKLAB routine is called a second time to insert idles, other control characters, and to format the text. If TABSET is specified and the return code from the IHKLAB routine is 0, the IHKLEW routine is called to insert the necessary tab and idle characters. After control is returned from the IHKLEW routine, the IHKLAB routine is called the second time. If TABSET is specified and the return code from the IHKLAB routine is nonzero, the IHKLAB routine is called the second time. When control is received from the IHKLAB routine the second time, the IHKLAP routine is called to initiate the write operation.

For the CREAD request, the n-lines override flag (TUBOVERN) is reset and the IHKLAP routine is called to initiate the read operation. When control is returned from the IHKLAP routine, the BTAM return code is checked. If nonzero, the error flag in the TUB is set and control is returned to the requesting routine with a nonzero return code. If the BTAM return code is 0, the line administrator waits in the dispatcher for the completion of the line I/O operation.

When the operation is complete and control is returned from the dispatcher, the BTAM completion code is checked. If the code is good and the operation was a write, the request flag in the TUB for write is reset, and control is returned to the requesting routine with a 0 return code. If the completion was good and the operation was a read, then the text that was read is translated from the transmission code to EBCDIC code. The following tests are performed and the necessary actions taken:

- If the text starts with any of the ending sequences and interrupt flag in the TUB is not set, control is returned with a length of zero specified in the TUBDATAL field. If the interrupt flag is set, then before returning with a length of zero the flag is reset.

- If the text ends with a backspace character followed by an ending sequence, the line of text is canceled, and another read operation is initiated by the IHKLAP routine.

- If the interrupt flag is set and the line of text starts a space character followed by the ending sequence, the interrupt override flag in the TUB is set, the interrupt flag is turned off, and control is returned.

If all of these tests are passed, the line of text is scanned for all nonsignificant characters such as line feed and backspace characters. These are eliminated, and the valid text is provided in the user buffer. If the horizontal tab character is found and the TABSET request is specified in the TUBTABAD field, control is passed to the IHKLAT routine to edit the input data. When control is returned, the scan is continued until the ending sequence is found. The length of the edited text is stored in the TUBDATAL field and in the DECB. The request flags in the TUB are reset, the requesting routine's registers are restored, and control is returned with a 0 return code.

If a bad completion code is received when control is returned from the dispatcher, the following investigations are made and the appropriate action taken:

- When the terminal is a 2741 and the interrupt flag in the DECB is set, the simulated interrupt flag and the read request flag in the TUB are set, the write request flag in the TUB is reset, and control is passed to the IHKLAP routine.

- When a negative response to polling is posted by BTAM and the simulated interrupt flag in the TUB is set, then the interrupt flag is reset, and control is returned with a zero length specified.

- When a negative response to polling is indicated, a switched line is repolled by issuing another read request.

- When the lost data condition is posted on a read operation, the message LINE TOO LONG, REENTER LINE is moved to the user buffer, the request for write and read is set in the TUB, and control is passed to the IHKLAB routine.

- When a time-out occurs on a text operation, the TIMEOUT message is moved to the user buffer and the remainder of the procedure is the same as for lost data.

- When a time-out occurs on a response to polling, the time-out error flag in the TUB is set, and control is returned with a nonzero return code.

- When nontext time-out occurs the terminal is considered off line/on hook, the appropriate error flag in the TUB is set, and control is returned with a nonzero return code.

For any other error condition, the TUBTIMEN flag is set and control is returned. The TUBTIMEN flag is recognized by the command analyzer to issue a request to reset the line (CREAD R).

External Routines

```
IHKLAT  -  to edit input when horizontal tabs are encountered
IHKLAB  -  to format output text
IHKLAP  -  to initiate the input/output operations on the line
IHKDSP  -  to wait for completion of line I/O operations
IHKLEW  -  to insert tab and idle characters
IHKLAY  -  to handle 1050X programmed time-out situations
```

Tables/Work Areas

```
DECB
User buffer
CLB
     CLBACTVN  -  active flag is set
     CLBINOPN  -  line inoperative flag is set
     CLBRDTIF  -  request flags are turned off
     CLBSTOPN  -  stop flag is checked
     CLBLNSEQ  -  sequence number is stored in TUB

TUB
     TUBIDENT  -  set to identify TUB
     TUBDATAL  -  set to length of data
     TUBTIMEN  -  set to indicate time-out error
     TUBWRITF  -  reset to indicate write request
     TUBREAKN  -  checked for interrupt
     TUBOVERN  -  set for interrupt override
     TUBREQST  -  reset for certain requests
     TUBREADN  -  set for read request
     TUBWRITN  -  set for write and read
     TUBOFFLN  -  set for off line/on hook error
     TUBTABAD  -  checked for input tabs
     TUBOUTAD  -  checked for output tabs
```

Exits

Normal - return to requesting routine with a 0 return code in register 15
Error - return to requesting routine with a nonzero return code in register 15

Attributes

Reentrant and resident

INPUT/OUTPUT OPERATION INITIATION MODULE (IHKLAP)

Entry Point

IHKLAP - register 1 must point to a one-word area containing the address of the CLB or the TUB.

Function

This is the line input/output operation initiation module of the line administrator. BTAM READ/WRITE macros are used to initiate the operation on the communication line.

If the address of a CLB is the parameter that was passed to this module, the request is for a read initial the very first time. Consequently, a read initial is issued for only four characters, permitting the user to start the session at the terminal by entering the ending sequence. A 72-byte save area is provided. The BTAM READ macro is issued, and the return code is stored in the CLB. If a return code of 20 is received, an LOPEN macro is issued to try to recover from a line error at open time. If the LOPEN macro succeeds, then the read initial operation is restarted.

If the address of a TUB was passed as a parameter, the request field in the TUB must be checked. If the request is for a write, the text in the user buffer is translated into the proper terminal transmission code. In the case of continuous output to those terminals without the automatic interrupt feature, an interrupt is simulated by setting the interrupt flag in the TUB when the number of lines specified in the CCT is reached.

If the request is for a write with response, a combined write/read operation is initiated. After either a single or combined operation is started, the BTAM return code is stored in the CLB, and control is returned to the IHKLAD routine.

If the request is for a read, depending on the previous operation performed on the line, the proper type of BTAM READ macro is used. After the operation is started, the BTAM return code is stored in the CLB, and control is returned to the IHKLAD routine.

External Routines

BTAM - whenever BTAM READ/WRITE macros are used
IECTLOPN- used if OPEN failed

Tables/Work Areas

DECB
CCT
User buffer/line buffer
CLB
CLBRTNCD - set with BTAM return code

CLBDIAL - checked for type of terminal
CLB1050 - checked for type of terminal
CLB2741D - checked for type of terminal
CLBREAD - checked for interrupt feature on terminal
CLBRDTIN - checked for read initial request

TUB

TUBREADN - set to simulate interrupt feature
TUBREADN - checked for read request
TUBIDENT - checked for identifier
TUBWRITN - checked for write request
TUBOVERN - checked for line override

Exits

Normal - return to the IHKLAD routine
Error - return to the IHKLAD routine with BTAM return code stored in the CLBRTNCD field

Attributes

Serially reusable and resident

OUTPUT TEXT FORMATTING MODULE (IHKLAB)

Entry Point

IHKLAB - register 1 must point to a one-word parameter list containing the address of the TUB.

Function

The basic functions of the IHKLAB module are formatting outgoing text, inserting idles when printing on "fly back" is to be avoided, appending necessary line control characters, and moving the text from the user buffer to the line buffer.

The IHKLAB module performs its functions in a two-pass operation. This is necessary because of the IHKLEW routine, which speeds up the output to the terminal by inserting tabs. The IHKLAB module is entered twice from the IHKLAD module before the IHKLAP module is called to initiate the line operation.

The IHKLAB module recognizes an entry as the first one if the TUBIDAPN flag in TUBFLG4 is set. The length of the data to be written is tested for zero on the first pass of a two-pass operation. If the length is zero, control is returned immediately to the IHKLAD module. If the TUBNOCRN flag is set, no CR (carrier return) is appended, and the trailing blanks are not removed. Any trailing blanks are eliminated if they are not required, and the new length is stored in the TUBDATAL field. The length is also saved in the TUBPARM2+2 field to be used on the second pass. Control is then returned to the IHKLAD module.

On the second pass to the IHKLAB module, the TUBNOCRN flag is checked to determine whether a carrier return should be appended at the end of the text. If the flag is set, the CR is appended at the end of the text. For the 1050 and 2740 terminals an EOB is also inserted at the end of the text. When the previous read operation has terminated without a CR or when the previous text written to the terminal contained the correct number of characters to cause printing on the fly back, then the necessary number of idle characters are inserted in front of the text. After the number of preceding idles is defined, the text is moved to the line buffer, the exact length is stored in the TUBDATAL field,

the length for the next idle computation is saved in the TUBIDLEN field, and control is returned to the IHKLAD module.

External Routines

None

Tables/Work Areas

DECB
CLB
CLBDEVTP - checked for device and line type
TUB
TUBIDAPN - checked for first or second entry
TUBIDLEN - set for preceding idle characters
TUBNOCRN - checked for carrier return character at end of text

Exits

Normal - return to the IHKLAD module
Error - none

Attributes

Reentrant and resident

TABSET EDIT MODULE (IHKLAT)

Entry Point

IHKLAT - register 1 must point to a 3-word parameter list containing the following:

- address of TUB,
- address of position in user buffer where horizontal tab character is encountered by the IHKLAD routine,
- address of the position in the user buffer following the last valid character.

Function

When the IHKLAD routine is called as a result of a CREAD macro and the TUBTABAD field is nonzero, the TABSET edit module is called each time a horizontal tab character is encountered. This module performs all necessary editing according to the tab settings specified in the TABSET command.

If editing the line would cause the number of characters to exceed 128, control is returned to the IHKLAD module with a four in register 15.

If nonvalid characters such as backspace or line feed immediately precede the horizontal tab character, then the horizontal tab character and the text following are shifted to the left within the user buffer so that the horizontal character follows the last valid character.

The text preceding the horizontal tab character is moved to a work area and padded with blanks if necessary. When the horizontal tab character is encountered past the logical location of the last tab setting entered by the TABSET command, then the horizontal tab character is replaced by one blank. The text remaining in the user buffer is then moved to the work area. When the complete line is edited, it is moved back to the user buffer, and control is returned to the IHKLAD routine.

The following 2-word parameter list is returned to the IHKLAD routine with a pointer in register 1:

- address of TUB,
- address of position where the scan is to be continued.

The IHKLAD routine stores the length of the unedited data (the text following the horizontal tab character) in the TUBDATAL field.

External Routines

None

Tables/Work Areas

128-byte work area
TUB

TUBUFFAD	-	contains address of user buffer
TUBTABAD	-	contains address of the tab buffer

User buffer - contains line to be edited
Tab buffer - contains tab settings

Exits

Normal	-	return to IHKLAD with 0 in register 15
Error	-	return to IHKLAD with 4 in register 15

Attributes

Serially reusable and resident

LINE EDIT WRITE MODULE (IHKLEW)

Entry Point

IHKLEW	-	used only by the IHKLAD module. Register 1 points to a one-word parameter list containing the address of the TUB.

Function

This routine is used by the IHKLAD module to insert tab characters (when TABSET has been specified) in lines of output to avoid sending a string of blanks over the line to the terminal. The horizontal tab (HT) character and idle characters are inserted into the output line whenever possible. Sometimes it is necessary to use the backspace character (BS) along with the horizontal tab and idle characters.

The line of output is contained in the user buffer. The length of the data is in the TUBDATAL field. The count of tab settings and the logical carrier position of each tab is contained in the tab table, the address of which is in the TUBOUTAD field. The complete data string that has been edited with tab, idle, and backspace characters is passed to the IHKLAD routine in the user buffer. The length of the string is put in the TUBPARMS field.

There are two situations in which the data string will be modified. The first situation is when a blank string precedes a user specified tab setting. No change is made to the data string unless the blank string contains more than three blanks. A horizontal tab (HT) character and idle characters are substituted in the data string in place of the blank string. The number of idle characters inserted is computed by the following algorithm:

$$\text{Idle characters} = \left\lceil \frac{24+P}{10} \right\rceil$$

where P is the number of positions to the tab, using the first blank position as the origin.

The second situation in which the data string will be modified is when a tab is preceded by a character string which is preceded by a blank string. The following algorithm must be satisfied:

$$2C + 3 \leq B$$

where C is the length of the character string preceding the tab position and B is the length of the blank string preceding the character string. If the algorithm is satisfied, the tab and idle characters are inserted in the data string. Backspace characters are used to return the carrier to the proper position for printing the character string. The blank string is ignored and the character string is repositioned as part of the data string.

All tabs in the output data are examined. Control is then returned to the IHKLAD module for the output to be sent to the terminal.

External Routines

None

Tables/Work Areas

TUB

TUBUFFAD	-	user buffer address
TUBOUTAD	-	address of area containing output tabs
TUBDATAL	-	user buffer data length
TUBPARM5	-	user buffer edited data length

User buffer - at entry contains output data; at exit contains edited data string

Dummy three-word save area

Tab Table - count of tab settings specified and logical carrier position of each tab

Exits

Normal - return to IHKLAD module
Error - none

Attributes

Resident and reentrant

1050X PROGRAMMED TIME-OUT MODULE (IHKLAY)

Entry Point

IHKLAY - register 1 must point to a one-word area containing the address of the CLB.

Function

The function of this module is to prevent a permanent wait from occurring on a read response to polling (INHIBIT) for the 1050X terminal on a leased line. The module is loaded at start-up time if such a

terminal is found. The IHKCIP module passes the address of the AVT. This module then includes its STCB in the first priority circle and adds its ECB to the end of the ECB list. After initialization the module is called by either the IHKLAD module or the IHKLAP module.

When control is from the IHKLAP module, the completion time is computed and stored in the CLB, and the CLB is then queued. If this is the first CLB on the queue, an STIMER macro is issued specifying the laytimer subroutine of this time-out module as the exit. After the queued flag is set (CLBTQUEN), control is returned to the IHKLAP module.

Control is from the IHKLAD module when there is an I/O completion on a line with the queued flag set. The dequeue subroutine removes the CLB from the queue and, if necessary, issues a STIMER macro for the first CLB on the queue. The dequeue subroutine also scans the queue to determine if any CLBs on the queue can be removed because of a positive response to polling, or if the completion time is within a certain predefined limit. The module then checks for an IOHALT macro. If one has not been issued, control is returned to the IHKLAD module to allow normal processing to continue. If an IOHALT macro was issued, a check is made to see if a MODIFY command or a STOP command is in progress, since they also issue IOHALT macros. If one of these commands is in progress, control is returned to the IHKLAD module. Otherwise, it is assumed that the IOHALT macro was issued elsewhere in the module, and the DECB is posted as a should-not-occur error. Control is then returned to the IHKLAD module.

The laytimer subroutine gets control asynchronously from OS when the outstanding time interval expires. A check is first made for line I/O completion, for a STOP command, or for a MODIFY command in progress. Any of these conditions result in an immediate return to OS. If none of these conditions are found, a check is made for a READ TI and for zeros in the input buffer. If this condition is found, an IOHALT macro is issued. If a READ TI is not found, a check is made for a positive response to text before checking for zeros in the input buffer. If an IOHALT macro is not issued, the ECB for the IHKLAY module is posted. Control is then returned to the IHKLAD module.

When this module (laypost subroutine) gets control from the CRJE dispatcher, the dequeue subroutine checks for a negative response. If it is not found, control is returned to the dispatcher. If a negative response is found, the queue subroutine computes the completion time and stores it in the CLB. The CLB is queued, and, if it is the first one on the queue, a STIMER macro is issued with the laytimer subroutine specified as the exit. The queued flag (CLBTQUEN) is set before control is returned.

External Routines

None

Tables/Work Areas

DECB
UCB
IOB
DEB
CLB

CLBTQUEN	-	set if CLB has been queued
CLBSTOP	-	checked for STOP or MODIFY command
CLBQUEUE	-	checked for CLB pointer for 1050 terminals on leased lines

Exits

Normal - return to calling routine (IHKCIP, IHKLAP, or IHKLAD)
Error - none

Attributes

Nonresident and reentrant

TERMINAL COMMAND AND SUBCOMMAND PROCESSORS

CHANGE SUBCOMMAND PROCESSOR (IHKCGN)

Entry Point

IHKCGN - register 1 must point to a two-word parameter list containing the address of the TUB and the address of the AVT.

Function

All operands of the CHANGE subcommand are checked. If any errors are found, an error message is sent to the user, and control is returned to the calling routine.

If the TUBCORRN field is set, indicating that syntax analysis errors are to be corrected, the line numbers specified on the subcommand must be within the range of line numbers pointed to by the TUBIRLSA field.

If no errors are found and if two line numbers are specified, the lines starting with the first line number are scanned for a match on text1. If no match is found, the next line is retrieved and the process is repeated. When a match is made, text1 is replaced by text2. If text2 is shorter than text1, the line is condensed and padded with blanks. If text2 is longer than text1, the line is expanded. If the insertion of text2 causes the line to exceed the maximum length, then the line is truncated and an error message is sent to the user. When the ALL operand is specified, the process is repeated for each instance of text1 within each line. If ALL is not specified, only the first text1 in each line is replaced.

After replacement the next line is retrieved, and the process is repeated until the specified last line is reached or EOD is detected. If only one line is specified, the same process is performed for only that one line. If the specified line does not exist, or if there are no lines within the specified range of line numbers, an error message is sent to the user and control is returned.

If a match on text1 is not found within the specified line or lines, an error message is sent to the user, and control is returned to the calling routine.

External Routines

IHKAFI - to manipulate active area
IHKMSG - (entry point:IHKMSG01) to queue error messages for user
IHKCCS - to scan lines for text1
IHKNUM - to check line number operands for all numerics

Tables/Work Areas

18-word save area
88-byte work area
PPT - contains subcommand and operands
TUB
TUBPPTAD - contains address of PPT
TUBUFFAD - contains address of user buffer
TUBPRMLS - contains parameter lists for IHKMSG, IHKCCS, and

IHKNUM

TUBIRLSA - contains address of line numbers scanned
TUBAFISW - buffer control switch turned on and off
TUBAFISW - buffer control switch turned on and off
TUBCORCN - turned on to indicate correction made
TUBCORRN - checked to find out if syntax error corrections are to be made
TUBLNUMF - checked for line number in data set
TUBNXXKEY - set and checked for key for RPOINT macro
TUBRAFBF - set and checked for AFIO buffer address
TUBUSRID - userid that is passed to message writer

User buffer - last 40 bytes used as work area.

Exits

Normal - return to calling routine with 0 in register 15.
Error - return to calling routine with one of the following return codes in register 15:

04 - line error
08 - GETMAIN failure
12 - active area I/O error
16 - error message lost
20 - active area out of space

Attributes

Reentrant and nonresident

EDIT, DELETE, AND EXEC COMMAND PROCESSOR (IHKEDT)

Entry Point

IHKEDT - register 1 must point to a two-word area containing the address of the TUB and the address of the AVT.

Function

This module checks the validity of the operands for the EDIT, DELETE (major), and EXEC commands. All of these commands must have at least one operand, the dsname, which is checked for length and first alphabetic character. OS data sets are indicated by quotes around the dsname. For deletion of an OS data set, the dsname must begin with the userid. If an OS data set is to be edited, the name must not begin with CRJE.

All operands are checked, and error messages, if any, are queued. If any errors occurred, any bits that were turned on in the TUB are turned off, the entire GETMAIN area is freed, and control is returned to the calling routine.

For a valid EDIT command this module turns on the bits in the TUB as specified by the operands, turns on the edit bit, and frees the entire GETMAIN area. For an EDIT NEW command, control is passed to the IHKIRL module. For an EDIT OS command, control is passed to the IHKEOS module. For an EDIT OLD (CRJE data set) command, control is passed to the IHKED1 module.

For a valid EXEC command, this module initializes the GETMAIN area to contain the dsname, key, userid, and record and block numbers. The address and length of this area are then stored in the TUB, the remaining GETMAIN area is freed, and control is passed to the IHKED1 module.

For a valid DELETE command, the dsname is put in the TUBPMFNM field, and control is passed to the IHKED1 module or the IHKEOS module, depending upon whether the data set is a CRJE data set or an OS data set.

External Routines

IHKED1 - for EXEC, DELETE, or EDIT (CRJE data set)
IHKEOS - for DELETE or EDIT (OS data set)
IHKIRL - for EDIT NEW
IHKMSG - (entry point:IHKMSG01) to queue error messages

Tables/Work Areas

GETMAIN - save area, switches, EXEC area
PPT - switches in PPTFLG and PPTPARS to be passed to other modules

DEF

DEFDEL	-	DELETE default
DEFEDIT	-	EDIT default
DEFEATTR	-	EDIT default
DEFPLBSM	-	EDIT default
DEFPLESM	-	EDIT default
DEFEXEC	-	EXEC default

TUB

TUBPARM1	-	for GETMAIN
TUBPRMLS	-	for IHKMSG
TUBEXCAD	-	set to address of GETMAIN area for EXEC
TUBEXCLG	-	set to length of GETMAIN area for EXEC
TUBFLG1	-	(TUBSCN, TUBFOR, TUBPL1) may be set
TUBFLG2	-	(TUBLNPMT) may be set
TUBPMFNM	-	dsname for CRJE data set, stored for EDIT, DELETE, and EXEC
TUBUSRNM	-	userid of user or userid specified
User Buffer	-	dsname, key, or member name, stored to be passed to IHKED1 or IHKEOS

Exits

Normal - return to calling routine with the negative code of the module to be loaded in register 15
Error - return to calling routine with one of the following return codes in register 15:

0	-	error(s) found and message(s) queued
8	-	GETMAIN failure
12	-	active area I/O error
16	-	message lost
20	-	active area out of space

Attributes

Reentrant and nonresident

EDIT, DELETE, AND EXEC COMMAND PROCESSOR (IHKED1)

Entry Point

IHKED1 - register 1 must point to a two-word area containing the address of the TUB and the address of the AVT.

Function

This module is called to delete or edit a data set in a user library, or to execute the next line from a CLIST data set. If there is no PPT,

the command is assumed to be EXEC. If there is a PPT, this module checks for the EXEC command code.

If EXEC is the command, the IHKUTM module verifies the userid and queues the line for library I/O. The data set whose dsname, userid, and key are in the GETMAIN area indicated by the TUBEXCAD field is found, and the directory entry is checked for a CRJE data set with the CLIST attribute. The block and record number of the record to be read is kept in the GETMAIN area and is updated each time a record is read. The indicated record is read, and the block and record numbers are updated. The record is stored in the user buffer, and the block area is freed. The data set is then closed, and control is returned with a 0 return code. If the data set is not found, or does not have the CLIST attribute, or if end-of-data is reached, this module returns with a 24 return code. If the EXEC data set is increased while being executed, a message is queued for the user and a 24 return code is used.

For the DELETE command, the IHKUTM module verifies the userid, queues for library I/O, then deletes the data set. Return is then made to the calling routine.

For the EDIT command, this module creates the active file and points to the beginning. The IHKUTM module is used to verify the userid and queue for library I/O. The data set is then found, and a GETMAIN macro issued for the directory entry. The directory entry is moved to the GETMAIN area and is inspected to ensure that the user data portion is in the CRJE format. If the data set is key protected, the user-specified key must match. If they do not match, an error message is queued for the user, the edit bit in the TUB is turned off, and return is to the calling routine with a 0 in register 15. Otherwise, changes are made in the user data portion of the directory entry according to the operands specified on the EDIT command. The count of the number of times accessed is updated.

If the directory entry format is for CRBE, the entry is expanded to the CRJE format using the information in the directory entry and the operands of the EDIT command. A directory entry that does not conform to either format is expanded using the operands from the EDIT command and the EDIT default options in the default table (DEF).

As the data set is read in, it is tested for Xs in the first 8 spaces, which indicate a data set created by a utility program with no line numbers. When this situation occurs, the keys are sequenced using the default starting line number and the default increment. If the SEQ attribute has been specified (by NUM or by default), the sequence number is stored in positions 73-80. If Xs are not found, the first 8 spaces are checked for numerics. If any non-numeric characters are found, the 8 characters are changed to zeros and the edit continues. An error message is sent for the first incorrect line number; all subsequent incorrect line numbers are changed to zeros. After the data set has been read into the active file, control is returned to the calling routine.

External Routines

IHKMSG	-	(entry point:IHKMSG01) to queue message for user (entry point:IHKMSG02) to queue message for central operator
IHKUTM	-	to verify userid and queue for library I/O; to declare library inoperative on I/O error
IHKAFI	-	to manipulate active file
IHKBPM	-	to manipulate data set in user library

Tables/Work Areas

GETMAIN	-	for save area and for directory entry
PPT	-	contains switches for EDIT and DELETE
DEF	-	line number increment used to initialize directory entry

for non-CRJE or non-CRBE data sets; line number increment and starting line number used to sequence data sets created by utility programs.

TUB

- TUBPRMLS - for IHKMSG parameter list
- TUBEXCAD - used for EXEC
- TUBDATAL - length of line stored in user buffer set by EXEC
- TUBCNTFS - number of lines read
- TUBDIRAD - address of directory entry stored
- TUBFLG1
  - TUBPL1 - set as specified in EDIT
  - TUBFOR - command or default for
  - TUBSCN - non-CRJE or non-CRBE data set
- TUBFLG2
  - TUBLNUMN - set as specified in EDIT or if directory entry indicates line-numbered data set
- TUBGBLKY - userid of file to be accessed stored here for UTM
- TUBNXXKEY - starting line number stored for non-CRJE and non-CRBE data sets
- TUBPMFNM - data set name for data set in library
- TUBUSRNM - userid name
- TUBPPTAD - pointer to PPT
- TUBRAFBF - address of buffer
- TUBUFFAD - pointer to user buffer
- TUBUSRID - userid

User buffer - dsname and key, if any, stored for EDIT and DELETE; for EXEC, line to be executed is stored here.

Exits

Normal - return to command analyzer with a 0 in register 15
Error - return to command analyzer with one of the following return codes in register 15:

- 8- GETMAIN failure
- 12- active area I/O error
- 16- message lost
- 20- active area out of space
- 24- EXEC is to be terminated

Attributes

Reentrant and nonresident

EDIT COMMAND PROCESSOR (IHKEOS)

Entry Point

IHKEOS - register 1 must point to a two-word parameter list containing the address of the TUB and the address of the AVT.

Function

This module receives control from the IHKEDT module to read an OS data set into a user's active file or to delete an OS data set. If an OS data set is to be read into the active area, the IHKIRL routine is called to create the directory entry and the active file. The loader/controller is then called to load the IHKOPN module. If the loader/controller routine (IHKLDC) abended, control is returned to the caller with a return code of 12 in register 15. After the module is loaded it is attached as a task, and control is passed to it to open or

scratch the OS data set. If the data set was scratched, the IHKEOS routine detaches the IHKOPN module and returns to the calling routine. If the data set was opened, the block size of the data set is found, and a GETMAIN macro is issued for a read buffer. The TUBLNUMN field is checked to see if it is on; if it is on, a maximum line length of 72 is set. If it is not on, a maximum line length of 80 is set. A block of records is read into the buffer, and the record size and blocking factor are checked. For fixed blocked or unblocked records with a record size equal to the maximum line length or less, each record is put in a line and padded as necessary. For fixed blocked or unblocked records with a record size greater than the maximum line length, the records are put into as many lines as needed to contain each line. If the record size is between the maximum line length and 120, the TUBLONGN field is set to indicate that the data set is to be listed on a single line per record. For variable blocked and unblocked records, each record is placed into as many lines as needed to contain the record, depending on the record length. When undefined records are encountered, each block is placed in as many lines as needed to contain the block. As the lines are built, they are assigned line numbers using the default starting line number and increment. If the TUBLNUMN flag is set, these line numbers are also put in bytes 73-80 of the line. When end-of-data is encountered on the data set, the IHKOPN module is detached, and control is returned to the calling routine.

External Routines

```
IHKDSP  -  to wait for completion
IHKAFI  -  to manipulate active file
IHKMSG  -  (entry point:IHKMSG01) to queue error messages
BSAM    -  to read data set
IHKOPN  -  to open or scratch OS data sets
IHKLDC  -  to load IHKOPN module
```

Tables/Work Areas

```
18-word save area
PPT
DCB
DSCB
JFCB
DIR  -  directory
TUB
     TUBUFFAD  -  buffer contains parameters
     TUBLNUMN  -  indicates line numbers contained in line
     TUBLONGN  -  set to indicate record length between 80-120
     TUBAFISW  -  indicates buffer control used
     TUBPPTAD  -  indicates buffer control used
     TUBPPTAD  -  PPT contains processing switches
     TUBCNTFS  -  number of lines to be inserted
     TUBDIRAD  -  contains directory address
     TUBNXXKEY -  contains line number for IHKAFI
     TUBPRMLS  -  parameter lists for calling routines
     TUBRAFBF  -  IHKAFI buffer address
User buffer    -  contains OS data set name, parameter list, and
                  ECBs for IHKOPN

Variable length Read Buffer  -  for reading data set
888-byte IHKAFI buffer
```

Exits

```
Normal  -  return to calling routine with a zero return code in
           register 15
Error   -  return to calling routine with one of the
           following return codes in register 15:
                08  -   GETMAIN failure
                12  -   active area I/O error or IHKLDC abend
```

16 - error message lost
20 - active area out of space

Attributes

Reentrant and nonresident

END SUBCOMMAND PROCESSOR (IHKEND)

Entry Point

IHKEND - register 1 must point to a two-word area containing the address of the TUB and the address of the AVT.

Function

The TUBDIRAD field in the TUB is checked first to determine if main storage has been allocated for a directory entry. If main storage has been allocated, it is freed, and the TUBDIRAD field is cleared to zeros. A RELEASE macro is issued to free the user's active file. The following switches in the TUB are reset to zero: TUBEXIT, TUBSCN, TUBFOR, TUBPL1, TUBLONGN, and TUBLNUMN.

If a disk error occurred on the RELEASE macro, control is returned to the calling routine with a return code of 12.

If this processor was called because of abnormal termination, then the negative of the LOGOFF command verb is put in register 15, and control is returned. Otherwise, control is returned to the calling routine with 0 in register 15.

External Routines

18-word save area
AVT
PPT - contains any operands entered
TUB
TUBPPTAD - contains address of PPT
TUBDIRAD - contains address of directory entry; cleared to zeros
TUBEDIT - set to zero
TUBSCN - set to zero
TUBFOR - set to zero
TUBPL1 - set to zero
TUBLONGN - set to zero
TUBLNUMN - set to zero
TUBABEND - checked to determine abnormal termination
TUBPRMLS - used for parmaeter lists
TUBUSRID - contains userid

Exits

Normal- return to calling routine with 0 in register 15; or if abnormal termination, the negative of LOGOFF command verb code in register 15.
Error - return to calling routine with one of the following return codes in register 15:
08 - GETMAIN failure
12 - active area I/O error

Attributes

Reentrant and nonresident

INPUT SUBCOMMAND PROCESSOR (IHKIPT)

Entry Point

IHKIPT- register 1 must point to a two-word parameter list containing the following: the address of the TUB and the address of the AVT.

Function

This module first checks each of the operands entered with the INPUT subcommand. The IHKNUM module is called to check the operands for eight or fewer numeric characters. The first two such numeric operands are saved as the line number and the increment. If the I operand is specified, the appropriate switch in the PPT is set. The operand I must follow a line number, and, if no increment is specified before the I, then the default increment of 1 is saved. If a line number is specified without the operand I, then the increment value is obtained from the user's directory entry. If the R operand is specified, the appropriate switch is set in the PPT. The R operand must follow a line number and cannot be preceded by an increment. The PROMPT operand causes the TUBLNPMT switch to be set, and the NOPROMPT operand causes the TUBLNPMT switch to be turned off.

The following conditions cause the excessive operands message to be queued for the user: more than four operands; any operands after a P[ROMPT] or NOP[ROMPT] operand. Testing of operands ceases after this message is queued, and control is returned to the calling routine.

The following conditions cause the invalid operand message to be queued: R or I operand before line number operand; increment before or after R operand; increment after I operand; unrecognizable operand. The multiple keywords message is queued whenever two or more I or R operands are specified or whenever both the I and R operands are specified. When a condition for the invalid operand or multiple keywords message is detected, the remainder of the operands are checked, and control is returned to the calling routine.

If no numeric operands are specified, control is passed back to the calling routine with the call code for the IHKIRL module (entry point: IHKIRL01) in register 15. Otherwise, an RPOINT macro is issued to gain access to the active area. If the I operand is specified, the RPOINT macro is for the specific line number. If the line number already exists in the active file, then an error message is queued and control is returned. Otherwise, the RPOINT macro is for after the next lower-numbered line. Parameters are set up for the IHKIRL module, and control is returned to the calling routine with the call code for the IHKIRL module in register 15.

If the R operand is entered, then an RPOINT macro is issued to point to the specific line number. If the line number does not exist, an error message is queued, and control is returned to the calling routine. Otherwise, parameters are set up for the IHKIRL module, and control is returned to the calling routine with the call code for the IHKIRL module in register 15.

If neither the I or R operand is specified, the RPOINT macro is issued to point to the specific line number. If the line number exists, then the parameters are set up for the IHKIRL module, and control is returned to the calling routine with the call code for the IHKIRL module in register 15. If the line number does not exist, the RPOINT macro is for after the next lower-numbered line. Then parameters are set up for the IHKIRL module, and control is returned with the call code for the IHKIRL module in register 15.

The parameters that are set up for the IHKIRL routine depend upon the operands specified on the INPUT command. If no line number is specified, the TUBDATAL field is cleared to zeros, and control is returned to the calling routine with the code for the IHKIRL01 routine in register 15. If a line number is specified on the INPUT command, then the following flags are set according to the operands:

INPUT linenum [increment]

If the record exists, the X'02' flag is set in the flag byte. Otherwise, the flag is not set. The increment is set to the value entered on the command, or to the value in the directory entry if no increment is given on the command.

INPUT linenum [increment] I

The X'40' flag is set in the flag byte. The increment is set to the value specified, or to one if no value is specified.

INPUT linenum R

The X'20' flag is set in the flag byte.

In all three cases the X'80' flag is set to indicate to the IHKIRL routine that control came from the IHKIPT routine. The parameters overlay the PPT and have the following format:

```
FLAGS     DS   C    FLAGS
          DS   CL5  UNUSED
LN        DS   CL8  LINE NUMBER
INCRE     DS   F    INCREMENT
```

External Routines

IHKMSG - (entry point:IHKMSG01) to queue messages for the user
IHKAFI - to manipulate the active file
IHKNUM - to check operands for numerics

Tables/Work Areas

18-word save area
PPT - contains command operands and processing switches
AVT
DIR
    DIRINC - default increment
TUB
    TUBPPTAD - contains address of PPT
    TUBDIRAD - contains address of directory entry
    TUBLNPMT - line number prompts switch, which is set or turned off as requested
    TUBDATAL - set to 0 for IHKIRL01
    TUBNXXKEY - key set for IHKAFI
    TUBUSRID - userid for IHKMSG
    TUBUFFAD - address of user buffer

Exits

Normal - return to calling routine with negative of command verb code in register 15 as directed by operands specified for either IHKIRL or IHKIRL01.
Error - return to the calling routine with one of the following return codes in register 15:
    08 - GETMAIN failure
    16 - error message lost.

Attributes

Reentrant and nonresident

Entry Points

IHKIRL - to insert lines into the active file, to create a new active file, to delete lines in the active file, or to create a directory entry. Register 1 must point to a two-word area containing the address of the TUB and the address of the AVT.

IHKIRL01 - to process the Implicit subcommand, a null line of input in edit mode, or an INPUT subcommand with no operands. Register 1 must point to a two-word area containing the address of the TUB and the address of the AVT.

IHKIRL02 - to process a null line when corrections are being made (control passed from command analyzer). Register 1 must point to a two-word area containing the address of the TUB and the address of the AVT.

Function

This routine services requests to create an active file, to modify an existing file, and to create a directory entry. It also processes the Implicit subcommand and the DELETE subcommand.

When control is passed to the IHKIRL entry point from an EDIT command, a new active file is created, a directory entry is created, and if NEW is specified on the command, lines are inserted into the active file as described for an insert-only request. Otherwise, a return is made to the calling routine.

When a DELETE subcommand is entered the operands are checked, and if an error is detected, an error message is queued for the user. When the command is accepted, and if one line has been specified, that line is deleted. If that line does not exist, an error message is queued for the user and return is made to the calling routine. When a range of lines is specified, those lines with line numbers greater than or equal to the first line number specified and less than or equal to the second line number are deleted. If there are no lines existing in the specified range, a message is queued for the user.

If the request is for insert only, the user is prompted with a line number, the reply text is inserted into the active file, the next line number is calculated, and the user is again prompted. This continues until the user enters a null line or until the calculated line number is greater than or equal to the next line number already existing in the active file.

If the request is for insertion or replacement, the user is prompted, and the line is inserted into or replaced in the active file at that place. The next line number is calculated and the user is prompted again. This continues until the user enters a null line.

When the request is for replacement, the line number in the subcommand is used to prompt the user for a line of text. When the user enters the line, it is put in place of the old line. The next line number is taken from the TUBNXKEY field, and the user is prompted again. This continues until the user enters a null line, or until the end of the active file is reached.

If the user has requested that input lines be syntax scanned, each input line, except those entered with the Implicit subcommand, are checked for a continuation character (-) at the end of the line. If there is a continuation character, it is removed and the line is put

into the active file in the normal manner. When a line without a continuation character is detected, it is put into the active area, and the line number range since the last noncontinuation line is passed to the syntax checker interface routine (IHKSYN). The return code from the interface is checked. If the code indicates an error, the appropriate error message is queued for the user. If the code is good, the user is prompted with the next line number and normal processing continues. If the return code from the interface indicates a syntax error, return is made to the command analyzer. This module is reentered at the IHKIRL02 entry point upon detection by the command analyzer of a null line of input. If the TUBCORCN switch is set, the lines are again passed to the interface routine and the cycle repeats.

External Routines

```
IHKLAD  -  to prompt user for input lines
IHKSYN  -  to pass lines to syntax checker
IHKAFI  -  to manipulate active file
IHKMSG  -  (entry point:IHKMSG and IHKMSG01) to queue and send
           messages for user
```

Tables/Work Areas

```
AVT
18-word save area
TUB
     TUBDIRAD  -  directory entry address
     TUBSCN    -  tested for syntax scan requested
     TUBFOR    -  FORTRAN scan requested
     TUBPL1    -  PL/1 scan requested
     TUBCORCN  -  tested for syntax errors corrected
     TUBCORRN  -  tested for syntax errors
     TUBIRLSA  -  address of parameter list for IHKSYN
     TUBTEXTN  -  set for upper/lower case preservation
     TUBDATAL  -  contains line length
     TUBLNPMT  -  tested for line number prompts
     TUBLNUMN  -  tested for line numbers in line
     TUBNXXKEY -  line number
     TUBPRMLS  -  parameter area
     TUBRAFBF  -  IHKAFI buffer address
     TUBUFFAD  -  buffer address
User buffer    -  buffer for IHKAFI
PPT            -  contains DELETE and Implicit subcommands
DIR            -  directory entry
CCT
     CCTPL1    -  PL/1 syntax checker in system
     CCTFOR    -  FORTRAN syntax checker in system
```

Exits

```
Normal  -  return to calling routine with 0 in
           register 15
Error   -  return to calling routine with one of the
           following return codes in register 15:
                4  -  line error
                8  -  GETMAIN failure
               12  -  active area I/O error
               16  -  error message lost
               20  -  active area out of space
```

Attributes

Reentrant and resident

LIST SUBCOMMAND PROCESSOR (IHKLST)

Entry Point

IHKLST - register 1 must point to a two-word parameter list that contains the address of the TUB in the first word and the address of the AVT in the second word.

Function

The numeric verification module (IHKNUM) checks the linenum operands, if specified, for all numerics. A GETMAIN macro is issued for two double words to contain the line numbers. If there are any errors in the operands, an error message is sent to the user, and control is returned to the calling routine.

If no line numbers are specified, the entire data set is listed. If a range of line numbers is specified, the listing starts with the first specified line or, if that line does not exist, with the next higher numbered line. The listing continues to give all successive lines until EOD is detected or until the next line number in the data set is greater than the specified last number. If only one line number is specified, that line alone is listed. If the line cannot be found, a message informs the user that the line does not exist.

If the NUM operand is specified, each line listed is preceded by its line number. If the NONUM operand was given, only the line is listed. However, even if the data set has the SEQ attribute, the last 8 positions of the line are not printed for either NUM or NONUM conditions.

The TUBLONGN switch of the TUBFLG3 field is checked to determine the presence of records with a length of up to 120 characters. If so, the extra forty characters are listed following the standard 80-character line, if line numbers are not contained in the line. If line numbers exist, the first 72 characters of the first line are listed, followed by the first 48 characters of the next line.

The RPOINT and RGET macros of AFIO are used to read the active file one line at a time. An 80-character line is put in the user buffer. If a 120-character line is indicated, a second line is put in an allocated AFIO buffer, and the first 40 or 48 characters are moved into the user buffer following the first full line. The AFIO buffer is then freed. Both 80- and 120- character lines are written from the user buffer to the user at a terminal by use of the CWRITE macro.

The routine is also used by the LISTDS/LISTLIB processor to list the requested information pertaining to a data set or sets from a user library. The IHKLDS module builds the lines in the active area and branches to this module to write the lines. Upon completion of this IHKLDS listing, the active area created by the IHKLDS routine is released.

External Routines

IHKNUM - to check linenum operands for all numerics
IHKAFI - to read active file (or certain lines)
IHKLAD - to write active file (or certain lines) at the terminal
IHKMSG - (entry point:IHKMSG01) to queue error messages for user

Tables/Work Areas

18-word save area
96-byte buffer used when listing line of a length
greater than 80 characters.
2 doublewords to store line numbers

User buffer to pass line to IHKLAD module
PPT - flag byte
TUB
TUBLNO - indicate whether line numbers are requested
TUBNXKEY - indicates line number for FIO operations
TUBPRMLS - used for parameter lists for the IHKNUM and IHKMSG modules
TUBDATAL - used to specify data length for CWRITE operation
TUBAFISW - set and turned off for AFIO buffer control
TUBLNUMN - checked for sequential number in line
TUBLONGN - checked for 80- or 120-character line
TUBPPTAD - checked for PPT address
TUBRAFBF - checked and set for AFIO buffer address
TUBUFFAD - checked for user buffer address
TUBUSRID - userid passed to message writer
DEF
DEFLST - indicates default for line numbers

Exits

Normal - return to calling routine with 0 in register 15
Error - return to calling routine with register 15 set to one of the following:
4 - line error
8 - GETMAIN failure
12 - active area or library I/O error
16 - message lost
20 - active area out of space

Attributes

Resident and reentrant

LISTDS AND LISTLIB COMMAND PROCESSOR (IHKLDS)

Entry Point

IHKLDS - register 1 must point to a two-word parameter list containing the address of the TUB in the first word and the address of the AVT in the second word.

Function

The operands of the LISTDS or LISTLIB command are checked. If an error is found, an error message is sent to the user and control is returned to the calling routine with a 0 in register 15. The command code in the PPT is checked to determine which command was entered. If a data set name is specified on the LISTLIB command, it is flagged as an error. A dsname must be specified on the LISTDS command and is put in the TUBPMFNM field. If STATUS and/or HISTORY is specified, corresponding flags are set in the PPTFLG byte.

The IHKUTM module is used to queue for library I/O. The I form (LISTDS) or the F form (LISTLIB) of the RFIND macro is then issued to find data sets in the user library. If the requested data set (LISTDS) or the user library (LISTLIB) is not found, an appropriate message is sent to the user. If the command is a LISTDS, the directory for the data set is in the KONBOX at BPBLDLST + 4. If the command is a LISTLIB, a block of directory entries is read into the area pointed to by the SFBUFF1 field. If the RREAD macro indicates no data sets in the user library, a message is sent to the user to indicate that the library is empty. The IHKLDS module terminates after sending these messages to the user.

CREATE and RPOINT macros are issued to acquire an active area in which to build the information to be sent back to the terminal. A header message is obtained from the message writer and inserted into the active area. The information requested will be listed under the appropriate labels of the header.

The requested information contained in a directory is interpreted. The necessary information is assembled in the user buffer in an alignment that corresponds to the labels of the header. The flags indicating what operands were specified in the command are checked. Any unnecessary information is blanked out and the line is compressed. The line of information is then inserted into the active area.

When all the information requested is gathered in the active file, the processor dequeues itself from the library I/O queue and control is passed to the LIST processor (IHKLST). The LIST processor requests that the active file be listed at the terminal. After all the lines are listed, the active file is released by the LIST processor.

External Routines

IHKBPM - to manipulate a user library
IHKAFI - to create and manipulate an active area
IHKMSG - to send error messages to the terminal user and to obtain the message header
IHKUTM - to queue for library I/O
IHKLST - to list the header and the data set information requested at the terminal

Tables/Work Areas

18-word save area
TUB
- TUBUSRID - used to obtain specified data set name directory information for LIST processor
- TUBPRMLS - contains parameter list for external routines
- TUBAFISW - to indicate to AFIO that the user buffer is not to be released
- TUBGBLKY - used to pass userid to the IHKUTM routine
- TUBMID - checked for message IDs
- TUBPMFNM - set to BPAM file name
- TUBPPTAD - checked for address of PPT
- TUBRAFBF - used for buffering control when inserting lines
- TUBUFFAD - contains address of user buffer
- TUBUSRNM - set to user name for RFIND1 for LISTDS

User buffer - used to assemble the data set information to be inserted into the active area

Active area - used to retain the assembled line of information from the user buffer and pass the line or lines to the IHKLST processor

PPT
- PPTFLG - indicates operands specified on the command
- PPTPARS - used to pass a code to the IHLST module that identifies this listing as an IHKLDS request

Exits

Normal - return to command analyzer with the negative value of the LIST command in register 15 or a 0 to indicate good completion, without going to the IHKLST module.

Error - return to command analyzer with one of the following return codes in register 15:

- 4 - line error
- 8 - GETMAIN failure
- 12 - active area I/O error

16 - message lost
20 - active area out of space

Attributes

Reentrant and nonresident

LOGOFF COMMAND PROCESSOR (IHKLGF)

Entry Point

IHKLGF - register 1 must point to a two-word parameter list containing the address of the TUB and the address of the AVT.

Function

This routine is responsible for terminating a user's session at the terminal. Upon entry, the time of LOGOFF is obtained and saved. If any operands are entered with the command, the excessive operands message is sent to the user and control is returned to the calling routine. The IHKUTM routine is used to verify the userid and to turn off the active bit in the user verification record (UVR). The CCT is then checked to determine whether or not any console has requested that all LOGOFFs be shown. If a request has been made, then a show session message is sent to the central console. If a nonzero return code is received from the IHKUTM module, then the console IDs, which were returned by the IHKUTM module, are checked and a show session message is sent to each of the requesting consoles.

Using the LOGOFF time and the LOGON time, which was saved in the TUB, the total session time is calculated. The LOGOFF message is sent to the user, and the LOGOFF exit routine is called. If storage has been allocated for the IHKTAB routine, it is freed. The eight-byte field in the TUB containing the userid (TUBUSRID) is cleared to zeros. If the return code for the IHKLGF routine is not 0, or if the command verb in the PPT is the command code for LOGOFF, then control is returned to the calling routine. Otherwise, the TUB is cleared to zeros with the exception of the first sixty bytes. If the CCTSUP bit is set, the LOGONs suppressed message is sent to the user, and control is returned to the calling routine. Otherwise, the negative of the command verb for LOGON is placed in register 15 and control is returned.

If the TUBUTMN switch is set, it means that the user is being logged off because of an I/O error in the active area. In this case the following procedures are not executed:

- No call to the IHKUTM routine;
- No session messages are sent; and
- No LOGOFF message is sent.

If the TUBLGNAB flag is set, IHKLGF was entered because of incomplete logon processing. The following procedures are not executed:

- No operand check is made;
- No LOGOFF message is sent;
- The LOGOFF exit is not entered if logon processing did not get to the point where the logon exit would be entered if there was one;

- No session messages are sent.

The return code for the LOGOFF routine is set to 12.

If the TUBABEND switch is set, this means that the user is being logged off because of abnormal termination. In this case the abnormal termination switch in the UVR is set when the IHKUTM routine is called and the return code for the LOGOFF routine is set to 4. Also, if the TUBSTOP switch is not set, then no session messages and no LOGOFF messages are sent. Both the TUBUTMN switch and the TUBABEND switch are checked before the IHKUTM routine is called.

External Routines

IHKUTM - to verify userid, turn off active bit, and if requested, turn on abnormal termination bit in UVR.
IHKMSG - (entry point:IHKMSG) to send error message or LOGOFF message to user.

Tables/Work Areas

18-word save area
AVT
TUB
- TUBUTMN - indicates I/O error in active area
- TUBLGNAB - indicates incomplete LOGON processing
- TUBTIME - contains LOGON time
- TUBGBLKY - key for IHKUTM (set by IHKLGF)
- TUBABEND - checked for abnormal termination
- TUBPPTAD - address of PPT
- TUBUSRID - userid
- TUBTABAD - address of main storage used in IHKTAB
- TUBPRMLS - used for parameters
- TUBCLBAD - address of CLB
- TUBSTOP - modify off
- TUBUFFAD - user buffer

PPT
- PPTCMD - command verb code

CCT
- CCTSESS - show session
- CCTSUP - suppress mode

Exits

Normal - return to calling routine with 0 in register 15
Error - return to calling routine with one of the following in register 15:

- 4 - TUBABEND is on
- 8 - GETMAIN failure
- 12 - TUBUTMN is set or active area I/O error
- 16 - error message lost

Attributes

Reentrant and nonresident

LOGON COMMAND PROCESSOR (IHKLGN)

Entry Point

IHKLGN - register 1 must point to a two-word parameter list containing the address of the TUB and the address of the AVT.

Function

The presence of operands is checked first. The first operand detected is considered the userid and password. If a slash is found, it is removed and the operand is separated. If no slash is found, the operand is considered to be only the userid. If the first character of the operand is a slash it is removed, and the operand is considered the password. If the last character is a slash then the slash is removed, and the operand is considered a userid. The remaining operands are checked, and appropriate switches in the TUB are set. If any errors (such as duplication of operands or contradictory operands) occur, then an error message is sent to the user and control is returned to the calling routine.

When all operands have been checked, the IHKUTM module is called to validate the userid and password. If no userid was specified, the message writer prompts the user for a userid. The userid is used as the key in TUBGBLKY for the IHKUTM routine. If the userid is not found or is already in use, the user is prompted for another one. If the userid is valid, the IHKUTM module returns the password in the TUBPARM4 field. If no password was entered with the command, the user is prompted for a password. The passwords are compared; and if they are not the same, the user is prompted for another one. If the passwords are the same, then the IHKUTM module is called again to set the active bit in the user verification record (UVR). The user is prompted only once for a userid and only once for a password. If the reply to a prompt is invalid the user is notified, and control is returned to the calling routine.

If a LOGON exit routine exists, the parameter list is specified, and a branch is made to the exit routine. If a nonzero return code is received from the user exit, indicating that the user is not to be allowed on the system, a message is sent to the user, and control is returned to the calling routine. Otherwise, the TIME macro is issued and the value is stored in the TUB. The message writer is called to send the LOGON message to the user. If the CCTSESS field is nonzero, indicating all LOGONs are to be shown, the show session message is sent to the central console. If the CCTSESS field is zero and the UVRSESS field is nonzero (indicating LOGON is to be shown for a particular user), then the show session message is sent to the central console.

If a nonzero return code is to be returned, the TUBLGNAB, flag is set to indicate that IHKLGF is to be entered in order to call the LOGOFF exit if the LOGON processor actually logged the user on the system.

External Routines

IHKMSG	-	(entry point:IHKMSG) to send messages to terminal user and central operator
IHKUTM	-	to verify userid, set active bit in UVR, get password, and put console IDs in TUBPARM3
LOGON exit	-	to process accounting information

Tables/Work Areas

18-word save area
TUB

TUBMID	-	set according to operands on command
TUBUSRID	-	userid inserted
TUBTIME	-	time of LOGON inserted
TUBBRD	-	set or turned off according to operands on command
TUBLGNAB	-	set to indicate incomplete LOGON processing
TUBGBLKY	-	key for IHKUTM
TUBPPTAD	-	address of PPT
TUBPRMLS	-	used for parameter lists
TUBDATAL	-	contains length of data in user buffer
TUBUFFAD	-	address of user buffer
TUBCLBAD	-	address of CLB
TUBDIRAD	-	used as temporary work area

TUBIRLSA - used as temporary work area
TUBRJCT - used as temporary work area
PPT - contains command and operands
CCT
CCTSESS - tested for show session
CLB
CLBLINE - contains physical line address
UVR
UVRMXCON - number of console IDs
AVT

Exits

Normal - return to calling routine with 0 in register 15
Error - return to calling routine with one of the following return codes in register 15:
4 - line error in IHKMSG
8 - GETMAIN failure
12 - active area I/O error
16 - error message lost

Attributes

Reentrant and nonresident

MERGE SUBCOMMAND PROCESSOR (IHKMGE)

Entry Point

IHKMGE - register 1 must point to a two-word area containing the address of the TUB and the address of the AVT.

Function

The first operand is checked and if it is an asterisk (*), a switch is set to indicate that an active area merger is to be performed. Otherwise, the dsname is stored. The next operands, if any, are checked for all numerics and stored. Any error causes an error message to be sent to the user and control returned.

If there are none or two numeric operands, a test is made to find if an active area merger is to be performed or if lines from a user library are to be merged. If the merger is within the active area, the IHKMAA module is loaded and control is passed to it. If the merger is from a user library, the IHKMUF module is loaded and control is passed to it.

When there are one or three numeric operands, the entire active file is moved to a utility file, the active file is released and a new one created. Control is then passed to the IHKMUF or IHKMAA routine.

External Routines

IHKAFI - to move lines to utility file
IHKMSG - (entry point:IHKMSG01) to queue error messages
IHKNUM - to verify all numerics in operand

Tables/Work Areas

18-word save area
28-byte work area
PPT-contains subcommand and operands
User buffer - used as work area
888-byte buffer
TUB
TUBAFISW - buffer control for IHKAFI
TUBCNTFS - number of lines (IHKAFI)

TUBGBLKY - utility file pointer
TUBNXKEY - active file pointer
TUBPARMLS - parameter area
TUBRAFBF - IHKAFI buffer address
TUBUFFAD - user buffer address
TUBUSRID - userid of user
TUBUSRNM - userid of owner of library

Exits

Normal - return to calling routine with the negative of the code for IHKMAA or IHKMUF in register 15
Error - return to the calling routine with one of the following return codes in register 15:

8 - GETMAIN failure
12 - active area I/O error
16 - error message lost
20 - active area out of space

Attributes

Reentrant and nonresident

MERGE SUBCOMMAND PROCESSOR (IHKMAA)

Entry Point

IHKMAA - register 1 must point to a two-word area containing the address of the TUB and the address of the AVT.

Function

This module handles the merging of active area lines. If a utility file exists, the lines up to and including the line after which the merger is to be made are moved to the new active file. If a range of lines has been specified, the lines within the range are copied to the end of the user's active file and resequenced. If no range is specified, the entire active file is copied to the end of the active file.

If a MERGE after a line number is specified, the rest of the lines in the utility file are copied at the end of the merged active file and the lines are resequenced. Control is then returned to the calling routine.

External Routines

IHKAFI - to manipulate active file and utility file
IHKMSG - (entry point:IHKMSG01) to queue error message for user

Tables/Work Areas

18-word save area
TUB
TUBLNUMN - indicates line numbers are contained in line
TUBPPTAD - address of PPT
TUBUFFAD - buffer has parameters from IHKMGE module
TUBLNUMN - checked for line numbers to be contained in line
TUBNXKEY - active file pointer
TUBCNTFS - number of lines (IHKAFI)
TUBGBLKY - utility file pointer
TUBPRMLS - parameter area
TUBRAFBF - IHKAFI buffer address

PPT
User buffer
888-byte buffer for IHKAFI

Exits

Normal - return to the calling routine with a 0 in register 15
Error - return to the calling routine with one of the following return codes in register 15:

8	-	GETMAIN failure
12	-	active area I/O error
16	-	error message lost
20	-	active area out of space

Attributes

Reentrant and nonresident

MERGE SUBCOMMAND PROCESSOR (IHKMUF)

Entry Point

IHKMUF - register 1 must point to a two-word area containing the address of the TUB and the address of the AVT.

Function

This module merges lines from a data set in a user library to the active file. If a utility file exists, the lines up to and including the line after which the merger is to be made are moved to the new active file. The data set is opened and the member found. If the data set or member is not found, an error message is queued for the user, and control is returned to the caller. The remainder of the utility file, if it exists, is moved to the active file.

If a range of line numbers is specified, the data set is read until a line number equal to or higher than the beginning line number is found. All lines in the range specified are inserted at the end of the active file and resequenced using the last line number and increment of the active file. If no range of lines is specified, the entire data set is copied.

If a utility file exists, the remaining lines are inserted at the end of the active file and resequenced. Control is then returned to the calling routine.

External Routines

IHKBPM - to read from a user library
IHKAFI - to manipulate active file and utility file
IHKMSG - (entry point:IHKMSG01) to queue messages for user and to send messages to central operator
IHKUTM - to verify userid, to put ddname in UVR, and to queue for library I/O

Tables/Work Areas

18-word save area
TUB
TUBLNUMN - indicator for line numbers contained in line
TUBPPTAD - PPT contains switches passed from the IHKMGE module
TUBUFFAD - buffer contains parameters from the IHKMGE module

TUBNXKEY - active file pointer
TUBCNTFS - number of lines
TUBGBLKY - utility file pointer
TUBDIRAD - directory entry address
TUBGBLNM - utility file number
TUBPRMLS - parameter area
TUBRAFBF - IHKAFI and IHKBPM buffer address
TUBUSRNM - owner of library

PPT
User buffer
888-byte buffer
Directory entry

Exits

Normal - return to the calling routine with a 0 in register 15
Error - return to the calling routine with one of the following return codes in register 15:
8 - GETMAIN failure
12 - active area I/O error
16 - error message lost
20 - active area out of space

Attributes

Reentrant and nonresident

OUTPUT AND CONTINUE COMMAND PROCESSOR (IHKOUT)

Entry Point

IHKOUT - register 1 must contain the address of a two-word parameter list containing the address of the TUB and the address of the AVT.

Function

The OUTPUT command allows the user to obtain the output of jobs he has submitted through CRJE. Only OS data sets in the CRJE SYSOUT class and job management messages, if MSGCLASS specified the CRJE SYSOUT class, are available. This module also processes the CONTINUE command for discontinued output.

This module first inspects the operands of the OUTPUT and CONTINUE commands. If an invalid operand is found, an error message is sent to the user. If a CONTINUE command is entered when the user is not in a discontinued state, the command is rejected and a message is sent to the user. If the command is OUTPUT and any of the following errors are detected, a message is sent to the user: jobname not found, user requesting output did not submit the job, or the job is not complete.

If the operands are verified, the IHKAFI module is called to get the RJCT entry. Control is then returned to the command analyzer with a code in register 15 requesting that the transmit output routine (IHKPUT) be loaded and control passed to it.

External Routines

IHKAFI - to read RJCT entry from global file
IHKMSG - (entry point:IHKMSG01) to queue messages for user

Tables/Work Areas

504-byte GETMAIN work area for IHKPUT routine
PPT,
- PPTCMD - checked for OUTPUT or CONTINUE command
- PPTCHAD - checked for operands in another PPT
- PPTPARS - contains first operand on command

RJCT
- RJCTFLGS - (bit 1) test for job completion
- RJCTUSER - userid of user who submitted job; checked against TUBUSRID who issued OUTPUT command.

TUB
- TUBFLG2 - (TUBDISOP) checked for discontinued output
- (TUBSMSG) set to 1 if SMSG is specified
- TUBRJCT - jobname inserted to process OUTPUT command; used as jobname for CONTINUE command
- TUBUSRID - userid who entered OUTPUT command

DEF
- DEFCON - checked for default CONTINUE operand if none is specified

Exits

Normal - return to command analyzer with negative value of code for IHKPUT in register 15

Error - return to command analyzer with one of the following return codes in register 15:
- 0 - error message queued for the user
- 8 - GETMAIN failure
- 12 - Active area I/O error

Attributes

Reentrant and nonresident

TRANSMIT OUTPUT MODULE (IHKPUT)

Entry Point

IHKPUT - register 1 must contain the address of a two-word parameter list containing the address of the TUB and the address of the AVT.. TUBPARM5 must contain the address of the save and work area for IHKPUT.

Function

A check is made to determine if there is an SMB or DSB to be read from the CRJE SYSOUT class. If there is not one, the IEFQDELE routine is called to delete the entry on the SYSOUT queue, the IHKAFI module is called to delete the RJCT entry, and control is returned to the calling routine. If there is an SMB or DSB, it is read. If the record is a scratched data set, this routine reinitializes the TTR of the last block read from the data set, and the next SMB or DSB is read if it exists. If the record is not a scratched data set and if an SMB was read instead of a DSB, a check is made to see if SMSG was specified on the command. If it was not, the next SMB or DSB is read and the process repeated.

If SMSG was specified, this indicates that the user wants the job management messages. This module then determines if the SMB is a compressed or normal SMB. Normal SMBs are transmitted directly to the remote terminal. Compressed SMB text is decompressed before being transmitted to the user. When the end of the SMB is reached, the next SMB or DSB is read and processed if it exists.

If a DSB is read instead of an SMB, IHKPUT determines whether it needs an SMF buffer to write the SMF type-6 record. If an SMF buffer is needed, it is gotten and initialized from the DSB. If an SMF buffer already exists, the count of SYSOUT data sets is incremented by one. The DSB contains the TIOT for a SYSOUT data set entry and is used by this routine to create a new TIOT referred to by the OPEN, CHECK, and CLOSE macros to provide access to the SYSOUT data set. BSAM is used to read the SYSOUT data. This module deblocks the data and transmits it to the remote terminal one logical record at a time. For each line transmitted, the SMF logical record count is incremented by one. Once transmitted, the data set is scratched and the next SMB or DSB is read. When all SYSOUT data sets have been transmitted, the SMF record is written to SYS1.MAN.

The TTR of the current block being transmitted is saved by this module when an unsuccessful write to the line is detected. This permits resumption of processing from that point after a discontinue/continue sequence.

IHKPUT makes use of the CRJE loader/controller task to load IHKRER and the CRJE service task to interface with IEFQDELE and IEFQMSSS. If either task is not active when IHKPUT needs to use them, IHKPUT returns to the system administrator with a return code of 12.

External Routines

IHKAFI - to delete or replace the RJCT entry in the active area
IHKMSG - (entry point:IHKMSG01) queue message for user
- (entry point:IHKMSG) send message to the central operator
IHKDSP - CRJE dispatcher to wait on I/O
IHKLAD - CWRITE macro to write to the line
IHKRER - (entry point:IHKRER) to read DSCB of SYSOUT data set from the VTOC to determine if there is a DSCB and if so, if the data set was accessed. If not, the data set is not opened.
- (entry point:IHKERR01) to scratch SYSOUT data sets
IEFQDELE- to return chain of SMB/DSBs to queue manager disk free space
IEFQMSSS- to read SMB/DSBs into main storage
BSAM - to read SYSOUT data sets

Tables/Work Areas

20-byte model DECB
88-byte model DCB for SYSOUT data sets
504-byte work area for parameter lists for subroutines, save area, DCB, DECB, SMB/DSBs, RJCT entry, and temporary constants.
48 bytes for a new TIOT entry
57-byte SMF buffer
BSAM input buffer (size is defined in DCBLKSI of DCB for SYSOUT data set

RJCT
RJCTTTRB - TTR of last SYSOUT block read, set after every read.
RJCTQMPA - used to read SMB/DSBs from SYS1.SYSJOBQE.

TUB
TUBDATAL - set to length of data for CWRITE macro
TUBFLG2 - (TUBDISOP) set on I/O error or discontinue; (TUBSMSG) set to 1 if SMSG specified, otherwise set to 0.
TUBRJCT - set to jobname to process for OUTPUT command; used as jobname for CONTINUE command.

Exits

Normal - return to command analyzer with 0 in register
Error - return to command analyzer with one of the following return codes in register 15:
4 - unsuccessful write to line
8 - GETMAIN failure
12 - CRJE task failure

Attributes

Reentrant and nonresident

SYSOUT OPEN, JOB DELETE, DATA SET SCRATCH, AND CANCEL MODULE (IHKRER)

The load module IHKRER is composed of four control sections, each of which performs an independent function. Linkage to a control section in the IHKRER module is made through a branch and link if from another CSECT in the IHKRER module; otherwise linkage is through the loader/controller (IHKLDC). The IHKRER module resides on SYS1.LINKLIB under the name IHKRER with aliases of IHKRER01, IHKRER02, and IHKRER03. The control section IHKRER is invoked by the IHKOUT module to open SYSOUT data sets of jobs being processed. The IHKRER01 section is invoked by the IHKDEQ module and the IHKRER03 section to delete a job from the CRJE and OS systems. The IHKRER02 section is invoked by the IHKOUT module and the IHKRER01 section to scratch a SYSOUT data set. The IHKRER03 section is invoked by the command analyzer to cancel a CRJE submitted job.

Entry Points

IHKRER - to open SYSOUT data sets
IHKRER01 - to delete jobs
IHKRER02 - to scratch SYSOUT data sets
IHKRER03 - to cancel jobs

Attributes

Reentrant and nonresident

First Entry Point

IHKRER - (from IHKOUT) register 1 must point to a two-word parameter list containing the following:

1. pointer to a six-character string of volume serial numbers,
2. pointer to SYSOUT DCB.

Function

When an OUTPUT command is processed, the IHKOUT module is called to send the output to a remote terminal. The IHKOUT module links to the IHKRER section to determine whether the SYSOUT data sets can be opened, and if so, opens them.

When this section gets control it reads the job file control block (JFCB) into main storage and changes the BUFNO field to zero. Using the DSNAME of the JFCB, this section tries to read the corresponding data set control block (DSCB) from the volume table of contents (VTOC). If no DSCB is available, no attempt is made to open the data set. If the DSCB is available, it is read into main storage, and the TTR of the last block written is checked for zero. If it is zero, the data set was not accessed, and this section does not try to open it. If it is not zero, this IHKRER section issues the OPEN macro (TYPE=J) to open the SYSOUT data set. Control is then returned to the IHKOUT module.

External Routines

SVC 64 - RDJFCB macro reads JFCB into main storage.
SVC 27 - OBTAIN macro reads DSCB into main storage.
SVC 22 - OPEN macro, TYPE=J, opens SYSOUT data set.

Tables/Work Areas

JFCB - contains DSNAME
DSCB - contains TTR pointer

Exits

Normal - return to IHKOUT with a 0 in register 15 if the OPEN J macro was issued; return to IHKOUT with a nonzero code in register 15 if the OPEN J macro was not issued.
Error - none

Second Entry Point

IHKRER01- register 1 points to a four-word parameter list containing:
1. pointer to RJCT entry
2. not used
3. pointer to the TUB
4. pointer to the AVT

Function

This section removes all references to a job from the CRJE system (except named data sets, which remain under OS control).

As a result of the OS CANCEL command, the job is enqueued on the output queue. When the job is enqueued, the CRJE job termination subtask (IHKDEQ) gets control and inspects the job-delete bit. Control is then passed back to this section to delete the job, which is then marked complete. When this section returns to the IHKDEQ module, the job cancel message is sent to the user.

If the job is complete, three routines are called to remove it from the system:

- The OS queue manager read routine reads the SMB/DSBs for the job.
- The data set scratch routine (IHKRER02) scratches the associated data sets.
- The OS queue manager delete routine deletes the OS queue space.

If access cannot be gained to the OS queue manager read or delete routines because of an abend in the CRJE service task, control is returned to the caller so that CRJE may be closed down.

External Routines

IHKRER02 - to scratch given data set
IEFQDELE - to delete OS queue space
IEFQMSSS - to read SMB/DSBs from disk

Tables/Work Areas

19-byte buffer to build OS CANCEL command
176-byte buffer for SMB/DSB

RJCT
RJCTFLGS - (bit 3) checked for job completion
- (bit 4) set for CANCEL command

RJCTQMPA - Q manager parameter area to interface with OS queue manager.

Exits

Normal - return to calling routine
Error - none

Third Entry Point

IHKRER02- register 1 must contain the address of a two-word parameter list:
1. address of queue manager parameter area (QMPA)
2. address of data set block (DSB)

Function

This section is used by the IHKOUT module and by the IHKRER01 section to scratch output data sets. The Job File Control Block (JFCB) is read by IEFQMSSS to determine the number of volumes on which the data set resides. A scratch list is built for all volumes (maximum of five) occupied by the data set, and the SCRATCH macro is issued to scratch the data set. If more than five volumes are involved, all over five remain unscratched.

If access cannot be gained to the OS queue manager read or delete routines because of an abend in the CRJE service task, control is returned to the caller so that CRJE may be closed down.

External Routines

IEFQMSSS- to read SMBs/DSBs from disk.

Tables/Work Areas

None

Exits

Normal - return to calling routine with a 0 in register 15 to indicate the data set has been scratched successfully.
Error - return to calling routine with a nonzero value in register 15 to indicate the scratch was unsuccessful.

Fourth Entry Point

IHKRER03 -register 1 must point to a two-word parameter list containing the address of the TUB and the address of the AVT.

Function

This section verifies the existence of the job for which the CANCEL command was issued. It also verifies that the user who issued the CANCEL command is the user who submitted the job. The IHKAFI routine is called to remove the RJCT entry and the IHKRER01 routine is called to remove the job if it is complete. If the job is not complete, an OS CANCEL command is issued for the job, the job-delete bit in the RJCT is turned on, and the RJCT entry is written in the global file. If the job is not found, a message is sent to the user.

External Routines

IHKAFI - to search for job entry in RJCT
IHKRER - (entry point:IHKRER01) to cancel the job
IHKMSG - (entry point:IHKMSG01) to send error messages to user

Tables/Work Areas

18-word save area
80-byte buffer for RJCT entry

TUB
TUBUSRID - userid of user who submitted the job
RJCT
RJCTFLGS - (bit 4) delete bit is checked on return from IHKERR01

Exits
Normal - return to command analyzer
Error - none

RENUMBER SUBCOMMAND PROCESSOR (IHKRNR)

Entry Point

IHKRNR - register 1 must point to a two-word parameter list containing the address of the TUB and the address of the AVT.

Function

The operands, if any, of the subcommand are checked. If either operand contains a non-numeric character, an error message is queued for the user, and return is made to the calling routine. The first operand is assumed to be the starting line number and the second, the increment. A default value of ten is assumed for any missing operand.

After the operand values are stored in a work area, a utility global file is accessed. A block of records is read from the active file, and the keys are resequenced. If line numbers are contained in the last eight characters of the line, they are also changed. When all lines in the block have been resequenced, the block is written to the utility file. This process continues until EOD is encountered in the active file. The utility file is then read, and the updated lines written back to the active file. The utility file is released when the entire file has been written back to the active file, and control is returned to the calling routine.

The increment used by the RENUMBER subcommand processor is inserted into the directory entry and becomes the increment for the data set. If a RENUMBER subcommand is entered for a null data set, the increment attribute in the directory entry is updated.

If in resequencing the line numbers the maximum allowable line number is reached before EOD, the utility file is released, the active file line numbers are not changed, and an error message is sent to the user.

External Routines

IHKAFI - to manipulate active file and utility file
IHKMSG - (entry point:IHKMSG01) to queue error messages
IHKNUM - to verify numeric operands

Tables/Work Areas

18-word save area
888-byte buffer used by AFIO
TUB
TUBUFFAD - contains address of user buffer
TUBPPTAD - contains address of PPT
TUBPRMLS - contains parameter lists for IHKMSG and IHKNUM

TUBDIRAD - contains address of directory entry
TUBAFISW - set and turned off for buffer control
TUBCNTFS - set and checked for block count for RGET macro
TUBLNUMN - checked for line number
TUBRAFBF - set for buffer control
TUBUSRID - userid passed to message writer
PPT - contains subcommand and operands
User buffer - last 40 bytes used as work area
DIR - directory entry

Exits

Normal - return to calling routine with 0 in register 15.
Error - return to calling routine with one of the following return codes in register 15:
4 - line error
8 - GETMAIN failure
12 - active area Ie/O error
16 - error message lost
20 - active area out of space

Attributes

Reentrant and nonresident

SAVE SUBCOMMAND PROCESSOR (IHKSAV)

Entry Point

IHKSAV - register 1 must point to a two-word area containing the address of the TUB in the first word and the address of the AVT in the second word.

Function

The operands, if any, are checked; if an error is found, an error message is sent to the user, and control is returned with a 0 in register 15. If the OLD keyword was specified on the EDIT command, and either the dsname is not specified on the SAVE subcommand, or the dsname on the SAVE subcommand is the same as the dsname on the EDIT command, the active file replaces the CRJE data set that has the same name. If the NEW keyword was specified on the EDIT command, or if the dsname on the SAVE subcommand is different from the dsname on the EDIT command, CRJE checks for an existing data set with the same name. If a duplicate data set name is found, the user is prompted for a new dsname. If the user responds with a null input line, then the active data set replaces the data set in the user library, and a new creation date is entered in the directory. In all cases the last modified date is updated.

If there is not enough space available in the user library, the library is automatically condensed to free unused space. If sufficient space is not available after condensing, the user is notified that his library is full and is asked for the name of a data set that can be deleted. If the dsname the user sends in response is valid, it is deleted. The user is prompted for data set names until he enters a null line. Then the library is again condensed. If there is still not enough space, the process is repeated beginning with the deletion prompts. If the user is unable to give a data set name for deletion the first time that he is prompted to do so subsequent to each condensing of the library, he may terminate the save attempt by entering a null line.

If enough space becomes available, the active file is saved in his library. After the active file is saved, or after the user terminates the process of condensing, control is returned to the calling routine with a 0 in register 15.

If the user enters a valid protection key operand without a data set name, the key is positioned in the key field of the directory belonging to the data set named in the EDIT command. If a valid protection key is given along with a data set name, the directory for that data set receives the new key. In both cases the key replaces any existing key without notification to the user.

If neither the dsname nor a key is specified with the SAVE command, the key field of the directory belonging to the data set named in the EDIT command remains unchanged. If a dsname is given with the SAVE command but no key is specified, the key field is blanked in the directory belonging to the named data set. This is the means that CRJE provides for dropping a key already assigned to a data set. However, no check of the key field is made prior to the blanking out and if a key is destroyed, no warning notification is issued to the user.

If the user has an 80-character library (as opposed to an 88-character library), a message will be sent warning him that data in positions 73-80 of a line or lines has been destroyed during the save, if the line sequence field is not blanks or does not equal the key field.

External Routines

IHKAFI - to manipulate active area
IHKBPM - to manipulate user library
IHKWTR - to wait for completion of RWRITE macro
IHKUTM - to queue for library I/O operations
IHKMSG - (entry point:IHKMSG) to prompt user for data sets to be deleted and retrieve the response
IHKMSG - (entry point:IHKMSG01) to inform the user of existing conditions and/or errors
IHKRNQ - to queue for library I/O after the initial use of IHKUTM
IHKCDP - to condense user library

Tables/Work Areas

18-word save area
PPT-flag byte used for SAVE indicators
KONBOX
    BPBLDST + 4 and succeeding bytes equaling the length of the DIR are changed to reflect the new dsname and/or keyword, if specified as operands.
    BPPARMFS - used to retrieve DDNAME from IHKUTM
    BPQCTLFW - used for BPAM queue control

TUB
    TUBCNTFS - set for RGET; altered by RGET for use by RWRITE
    TUBFILNM - set with new dsname, if specified as an operand
    TUBFLG2 - tested for ABEND call to IHKSAV
    TUBPRMLS - used for parameters to external routines
    TUBGBLKY - used for userid for IHKUTM
    TUBDATAL - used to retrieve user response to prompt
    TUBBPQEL - checked for BPAM queue element for IHKRNQ
    TUBDIRAD - checked for directory entry address
    TUBPPTAD - checked for address of PPT
    TUBRAFBF - used for retrieval of line by RPOINT macro, referred to in order to pass record from RGET AFIO buffer to SEBUFF1 address for RWRITE into library operation
    TUBUFFAD - referred to when returned from message writer prompts
    TUBUSRID - referred to for IHKUTM and for message writer
DIR
    DIRNAME - used to retain file name as edited

Exits

Normal - return to the calling routine with 0 in register 15 or the negative value of the END command verb, which signals the normal completion of SAVE in response to an ABEND condition.

Error - return to the calling routine with one of the following return codes in register 15:

4 - line error
8 - GETMAIN failure
12 - active area I/O error
16 - message lost
20 - active area out of space
Negative value of X'4A' for ABEND condition

Attributes

Reentrant and nonresident

SCAN SUBCOMMAND PROCESSOR (IHKSCN)

Entry Point

IHKSCN - register 1 must point to a one-word area containing the address of the TUB.

Function

The SCAN processor checks the presence of the PL/1 syntax checker or a FORTRAN syntax checker depending upon the attribute of the data set. The FORTRAN syntax checker checks for the specific FORTRAN checker since there are several possibilities. It returns with an error code if the specified FORTRAN checker is not present. The numeric verification module (IHKNUM) checks the linenum operands for all numerics. If an error is found, a message is queued for the user, and control is returned. If no errors are found in the operands, this routine passes the lines to be scanned to the syntax checker interface (IHKSYN).

When a range of lines is specified, these lines plus continuation lines are scanned. A SCAN subcommand with one line number specified causes a scan of that one line plus continuation lines. A subcommand with no operands causes the entire data set to be scanned. If ON or OFF is specified on the subcommand, it must be the last operand. If ON or OFF is the only operand specified, the SCAN flag in the TUB is turned on or off, and no scan is performed.

This module uses the IHKRNQ module to be queued for the syntax checker interface, since the module is serially reusable. It dequeues itself immediately after return from the interface. The interface returns to this module when error messages have been queued, or when all lines have been scanned. If the scan is not finished, this module sends the messages and calls the interface (IHKSYN) again. When the scan is finished, control is returned to the calling routine.

External Routines

IHKNUM - to verify the linenum operands as all numerics
IHKSYN - to send line numbers to be scanned to interface
IHKMSG - (entry point:IHKMSG) to send error messages to the user
- (entry point:IHKMSG01) to queue operand error messages for user
IHKRNQ - to queue the SCAN processor for the syntax checker interface

Tables/Work Areas

GETMAIN area for save area and parameters for the interface
PPT - contains subcommand and operands; Q element and line numbers are stored while lines are being scanned.

CCT
- CCTOPT1 - (CCTPL1) checked for PL/I checker
- CCTOPT2 - (CCTFORT) checked for FORTRAN checker

TUB
- TUBFLG1 - TUBFOR or TUBPL1 checked; TUBSCN turned on or off if specified.
- TUBFLG3 - TUBDLAYN set on while scanning
- TUBIRLSA - pointer to address of line number range to be scanned
- TUBPARM1 - used for GETMAIN
- TUBPRMLS - used for parameter list for IHKMSG and IHKRNQ
- TUBPPTAD - pointer to PPT

User Buffer - used while operands are syntax checked

Exits

Normal - return to calling routine with 0 in register 15
Error - return to calling routine with one of the following return codes in register 15:
- 4 - line error
- 8 - GETMAIN failure
- 12 - active area I/O error
- 16 - error message lost

Attributes

Reentrant and resident

SEND COMMAND PROCESSOR (IHKSND)

Entry Point

IHKSND - register 1 must point to a two-word parameter list containing the address of the TUB and the address of the AVT.

Function

All operands are checked for validity except the first one (which is the message to be sent). An error message is sent to the user for each invalid operand.

If the message is to be sent to a terminal user, the IHKUTM module is called to verify the userid and to determine whether the user is active. If the userid is invalid, an error message is sent to the user who entered the SEND command, and control is returned to the calling routine.

If the user is active, the message writer is called to send the message to the terminal user. If the user is not active and if LOGON is specified on the command, the message writer is called to queue the message. When LOGON is not specified, a message is sent to the user who entered the command to indicate that the message was not sent.

If the message is the only operand, or if OPERATOR is specified, the message writer is called to send the message to the central operator. NOW and LOGON are ignored when used with OPERATOR. If there is more

than one central console, the user can specify where the message is to be sent by including the routing code (following OPERATOR). A routing code greater than 16 results in an error message. A code of 0 has the same meaning as a code of 1.

External Routines

IHKMSG - (entry point:IHKMSG) to send the message to the central operator;
- (entry point:IHKMSG01) to queue the message for a terminal user and to queue error messages for the sender.
IHKUTM - to verify the recipient userid and to dertermine if the recipient user is active.

Tables/Work Areas

18-word save area
TUB
TUBPPTAD - contains address of PPT
TUBUFFAD - contains address of user buffer
TUBPRMLS - contains parameter lists for IHKMSG and IHKUTM
TUBGBLKY - set with userid for IHKUTM
TUBUSRID - used in calling message writer
User Buffer - last 40 bytes used as work area
PPT - contains command and operands

Exits

Normal - return to calling routine with 0 in register 15.
Error - return to calling routine with one of the following return codes in register 15:
4 - line error
8 - GETMAIN failure
12 - active area I/O error
16 - error message lost
20 - active area out of space

Attributes

Reentrant and nonresident

STATUS COMMAND PROCESSOR (IHKSTS)

Entry Point

IHKSTS - register 1 must point to a two-word parameter list containing the address of the TUB and the address of the AVT.

Function

The operand, if specified, is checked for validity. If more than one operand is specified and if the length of the operand is greater than eight, or if the operand is in parentheses, an error message is queued for the user, and control is returned to the calling routine. If one jobname operand is present, an RPOINT macro is issued for the RJCT entry to determine whether the job is in the system. If the RJCT entry is not found, or if the job was not submitted by the user that issued the STATUS command, a message is queued for the user, and control is returned to the calling routine. If the job is found but was submitted by another user, then the invalid recipient message is sent to the user, and control is returned to the calling routine. Otherwise, the CHKSTS subroutine in the IHKSTS module is used to get the status of the job.

When the jobname operand is not present, an RPOINT macro is issued to search the RJCT entries for a job submitted by the user. If none is found, a message is queued for the user, and control is returned to the calling routine. If a job is found, the CHKSTS subroutine is used to obtain the status of the job. Then search of the RJCT entries is continued to get the next job belonging to the user. After the status of all the jobs submitted by the user is determined, control is returned to the calling routine.

The CHKSTS subroutine is used to test the information returned when the RJCT entry was read to determine whether the job was complete. If the job is complete, a job complete message with an indication of normal or abnormal completion is queued for the user. Otherwise, the loader/controller is called to load the IHKLOC routine and the IHKSRV module is called to pass control to the IHKLOC routine. This routine searches the job queue. If the job is queued, then a message indicating the position on the queue is queued for the user. Otherwise, the TIOTs are searched to determine whether the job is executing. If it is, the executing message is queued for the user. Otherwise, the RJCT entry is checked to determine whether or not the job is complete. If it is, then a job complete message with an indication of normal or abnormal completion is queued for the user. Otherwise, the not-queued message is queued for the user. Control is then returned to the calling routine. In case of an error in calling the IHKLOC routine, a disk error message is queued for the user. If either the loader/controller task or the service task abends, control is returned to the calling program with a return code of 12.

External Routines

IHKDSP - to wait for event completion
IHKMSG - (entry point:IHKMSG01) to queue messages for the terminal user
IHKLOC - to search the OS job queue
IHKAFI - to search and read RJCT entries

Tables/Work Areas

18-word save area
PPT - contains command and operands
TUB
- TUBGBLKY - key for AFIO
- TUBRAFBF - load address of area into which AFIO inserts information
- TUBAFISW - indicates that an optional buffer is provided
- TUBPRMLS - used as parameter list
- TUBPPTAD - contains address of PPT
- TUBUSRID - contains userid
- TUBUFFAD - contains address of user buffer
- TUBGBLNM - key for global file

User buffer - used as work area

CLB
- CLBLCECB - ECB for IHKLDC
- CLBUTECB - ECB for IHKRSV

RJCT
- RJCTFLGS - check for job completion
- RJCTJOB - contains jobname
- RJCTUSER - user identification

Exits

Normal - return to calling routine with 0 in register 15
Error - return to calling routine with one of the following return codes in register 15:
- 4 - TP line error
- 8 - GETMAIN failure

12 - active area I/O error or CRJE task abend
16 - error message lost

Attributes

Reentrant, refreshable, and nonresident

SUBMIT COMMAND PROCESSOR (IHKSUB)

Entry Point

IHKSUB - register 1 must point to a two-word parameter list containing the address of the TUB in the first word and the address of the AVT in the second word.

Function

The SUBMIT command processor allows up to ten CRJE data set names (plus dsnames found in DSLIST files), combined into a sequential data set, to be entered into the OS job input stream.

The following two modules perform special services for the SUBMIT processor. The IHKGET module checks the syntax of the dsnames submitted and points to blocks of input records (usually ten per block) from which the reader/interpreter data set is built. The IHKALC routine allocates the sequential, reader/interpreter data set and builds the S RDRCRJE command.

For each block of records, the IHKSUB module saves the length and address of the block. Each record of the block is inspected. The first check is for a slash in column one. If there is not a slash in column one, the record is passed to the reader/interpreter data set write subroutine (WRITIT). The WRITIT subroutine checks to see whether the reader/interpreter data set has been allocated. If it has not, the DD DATA switch is turned off, and the next record is inspected. If the reader/interpreter data set has been allocated, the WRITIT subroutine determines whether the record is the last one of the block. If it is not, the next record is inspected. If it is the last record of the block, the entire block is written in the reader/interpreter data set, and the next block is read.

If there is a slash in column one, the DD DATA switch is tested. If it is on, the record is inspected for a slash asterisk (/*). If there is a '/*' in columns 1 and 2, the DD DATA switch is turned off, and the WRITIT subroutine writes the record in the reader/interpreter data set. If there is not a '/*' in columns 1 and 2, the WRITIT subroutine writes the record without any switches being turned on or off. If the DD DATA switch is not on, columns one and two are inspected for slashes (//). If double slashes are not present, the WRITIT subroutine is called to write the record in the data set.

If the installation JCL exit is present, it is called, and depending on the return code, the SUBMIT processor either sends the user an error message and aborts the SUBMIT process, aborts the SUBMIT process with no exit message, or continues processing.

If there is a slash in columns 1 and 2 and the record is a DD DATA statement, the DD DATA switch is set, and the WRITIT subroutine writes the record in the data set.

If the record is not a JOB statement, the WRITIT subroutine writes the record in the data set. If the record is a JOB statement, the job card and the DD DATA switches are cleared.

If the statement is a job statement, and the return code from the installation exit is 0, the RJCT is searched for an entry with the specified job name. If an entry is found or any other error occurs, the SUBMIT is aborted. If an entry is not found, the RJCT entry with the job name is built in main storage and initialized.

If the input data set for the OS reader/interpreter has not been allocated, the IHKALC module is called to perform the allocation, and the reader/interpreter data set-allocated switch is turned on.

Each JOB card is written as a separate block in the reader/interpreter data set. This prevents loss of data if the reader RDRCRJE is rolled out in MFT. After the JOB card has been written, the TTR of the statement is saved for abnormal processing.

When a complete job has been collected, the RJCT entry for the job is written in the JBTBLS global file.

If the SUBMIT is to be aborted because of a bad return from the IHKGET routine, the error messages returned by the IHKGET routine are sent to the user, and the IHKGET module is deleted. If the abort is for any other reason, the IHKGET routine is called with a special flag to force end of input. If the IHKGET module finds that no jobs were collected, it builds the parameter list to scratch the reader/interpreter data set (scratched by IHKSRV) and frees the START RDRCRJE command area. For all SUBMIT aborts, the TTR, which was saved after writing the last JOB card, is inspected. If the TTR does not exist, no jobs have been collected. A message is sent to the user informing him that no jobs were found; the save and work area is freed; and control is returned to the calling routine. If the TTR is present, the DCB is positioned before the TTR forcing the CLOSE macro to write EOF after the last completely collected job. The base register of the RJCT is cleared for normal SUBMIT termination.

In normal SUBMIT termination the base register for the RJCT is tested. If it is nonzero, the RJCT entry is written for the last job. The reader/interpreter data set is closed, if it exists. The CRJE service routine, IHKSRV, is called to close the DCB and to start the reader processor (RDRCRJE) on the data set; the save and work area is freed; and control is returned to the routine that called the SUBMIT processor.

If either the loader/controller task or the service task has abended, control is returned to the system administrator with a return code of 12.

External Routines

IHKALC	-	to allocate the reader/interpreter data set
IHKGET	-	to check the dsname and position to the next input record
IHKAFI	-	to write the RJCT entry in the global file
IHKMSG	-	(entry point:IHKMSG01) to send error messages to the user
IHKLAD	-	to write JCL exit message to user
IHKUTM	-	to mark library inoperative on library I/O error

Tables/Work Areas

432-bytes for save area, RJCT, and work area

Exits

Normal	-	return to calling routine with 0 in register 15
Error	-	return to calling routine with a 12 (active area I/O error or CRJE task abend) or 8 (GETMAIN failure) in register 15.

Attributes

Nonresident and reentrant (marked reusable in IHKMOD table)

SUBMIT INPUT RECORD PROCESSOR (IHKGET)

Entry Point

IHKGET - register 1 must contain the address of a four-word parameter list. The first word is not used; the second word points to the SUBMIT work area; the third word contains the address of the TUB; and the fourth word is the address of the AVT.

Function

The function of the SUBMIT input record processor is to set up pointers to blocks of input records from which the input job stream is built for the OS reader/interpreter.

If this is the first entry of the SUBMIT processor, the IHKGET module clears its internal switches and counters and initializes its internal PPT pointers to the first dsname in the PPT. If this is not the first entry, the IHKGET routine attempts to read the next data set name. The internal subroutine GETDSN gets the data set name from the PPT, checks the syntax of it, and if valid, the file is positioned to read the first block of records. If an error is found in the data set name, an error message is sent to the user, and an exit is made to the SUBMIT processor with an end-of-data condition indicated.

After the file has been positioned, the count of the data set names processed is increased by one, and a switch is set indicating whether the file is an active file or a data set from a user library. The attribute of the file is checked and if it is not a DSLIST file, control is returned to the SUBMIT processor.

If it is a DSLIST file, the entire file is read into a second private file in the active area. All subsequent calls to this module result in processing data set names from the private file instead of the PPT. When end-of-file is reached, control is given to the GETDSN subroutine to inspect the next data set name in the PPT. Nested DSLIST files are not permitted.

The SUBMIT status switch is inspected upon entry to this module. If bit 0 (X'80') is set, the entry is for closedown. If a second private file for DSLIST files exist, it is released, and the old active file for the user is recreated. If the reader/interpreter data set has been allocated but no jobs have been collected, the scratch parameter list is built.

External Routines

IHKAFI - to manipulate active files
IHKBPM - to read a data set from a user library
IHKUTM - to get ddname for user library and to get on library I/O queue

Tables/Work Areas

Work area is provided by the IHKSUB module.

Exits

Normal - return to IHKSUB with a completion code of 0 in register 15, except if end-of-input is encountered, when a return code of 4 is used.

Error - return to IHKSUB with completion code of 4 in register 15 and the address of the error message in the TUBPARMS field.

Attributes

Nonresident and reentrant. (The module is actually used serially by IHKSUB.)

ALLOCATE ROUTINE (IHKALC)

Entry Point

IHKALC - register 1 must contain a pointer to a five-word parameter list containing the following:

1. address of the DECB
2. address of the RJCT
3. address of the DCB
4. address of the TUB
5. address of the AVT

Function

This routine allocates the data set used by the SUBMIT processor to build the OS job stream and opens the data set.

Before the data set is allocated, 104 bytes of main storage are obtained for the START RDRCRJE command. The DCB and the DECB for the data set are initialized. The count, constant parameters, jobname (from RJCT), and unit and volume (from UCB) fields are moved to the 104-byte area. The address of the START RDRCRJE command is placed in the TUBIRLSA field for the job card processor. If any errors are found in this routine, the 104 bytes are freed before returning.

The format of the START command is as follows:

```
RDRCRJE.S,,,,DSNAME=SYS1.CRJE.jobname,DISP=(OLD,DELETE),UNIT23x,
VOLUME=SER=volid
```

A sequential data set is allocated for the job stream using the DADSM ALLOCATE routine (SVC 32). There is a DD card for each line in the CRJE procedure. The DD card specifies volume, unit, and primary and secondary allocation for the data set. The ddname for the DCB is obtained from the CLB. A search of CRJE's TIOT (task input/output table) is made for the ddname. If the ddname is not found, control is returned to the SUBMIT processor. If the ddname is found, a READ JFCB (job file control block) macro is issued. Block size and logical record length are fixed at 800 and 80, regardless of what is specified on the DD card. Once the data set is allocated, an OPENJ macro is issued for it and control is returned.

External Routines

RDJFCB - Read JFCB into storage for DADSM allocate
IGC0003B- (DADSM Allocate) - to allocate space for the job stream data set
OPENJ - to open the data set

Tables/Work Areas

288-byte buffer gotten by IHKALC is used as a save area for the JFCB and for the START RDR command

TUB

TUBIRLSA - contains address of START RDRCRJE command

RJCT

RJCTUNIT - set to 1 for 2311 and 4 for 2314
RJCTVOL - set to volume serial of pack on which the reader/interpreter data set resides

Normal - return to IHKSUB with return code of 0 in register 15
Error - return to IHKSUB with one of the following return codes in register 15:

- 4 - duplicate DSNAME on volume
- 8 - out of space
- 12 - I/O error
- 16 - GETMAIN failure
- 20 - no DD statement in procedure
- 24 - reader/interpreter data set not on 2311 or 2314

Attributes

Reentrant and nonresident

TABSET COMMAND PROCESSOR (IHKTAB)

Entry Point

IHKTAB - register 1 must point to a two-word area containing the address of the TUB and the address of the AVT

Function

When no operands are specified, or when IN or INPUT and/or OFF are specified, the TUBTABAD field is tested. If it contains an address, the area to which it points is deallocated, and the TUBTABAD field is set to zero. If the TUBTABAD field is already zero, no action is taken.

When OUT or OUTPUT is specified with or without OFF, the TUBOUTAB field is tested. If it contains an address, the area to which it points is deallocated, and the TUBOUTAB field is set to zero.

When tab settings are specified, they are checked for length, ascending order, and numerics. If an error is found, an error message is sent to the user, and the settings are rejected. Settings of 0 and 1 are rejected as being without meaning; in this case, the user is notified and the settings are ignored. A setting greater than 80 for input or 120 for output is rejected; the user is notified; the acceptable settings are retained, but the checking of remaining settings is terminated. When more than 10 settings are specified, an excessive operand message is sent to the user and the first 10 settings are retained. As the settings are checked they are temporarily stored in the user buffer.

When tab settings are specified (with or without IN or INPUT), the TUBTABAD field is tested. If it contains an address, the area to which it points is used to store the input tab settings. If it contains zero, an area is dynamically allocated for the input tab settings, and the address is stored in the TUBTABAD field.

When OUT or OUTPUT is specified with the tab settings, the TUBOUTAB field is tested. If it contains an address, the area to which it points is used to store the output tab settings. If it contains zero, an area is dynamically allocated for the output tab settings, and the address is stored in the TUBOUTAB field.

After all the operands have been checked and no terminal errors detected, the number of settings is placed in the first halfword of the area followed by the settings.

External Routines

IHKMSG - (entry point:IHKMSG01) to build and send error messages

Tables/Work Areas

16-byte tab area
18-word save area
TUB
- TUBOUTAB - contains address of output tab table
- TUBPPTAD - contains address of PPT
- TUBUFFAD - contains address of user buffer
- TUBPRMLS - contains parameter lists for IHKMSG
- TUBTABAD - contains address of input tab table
- TUBUSRID - used in calling message writer
- TUBCLBAD - contains the address of the CLB

PPT - contains command and operands
User buffer - last 40 bytes used as work area
Input tab table - contains input tab settings
Output tab table - contains output tab settings
CLB
- CLBDEVTP - contains the terminal and line type

Exits

Normal - return to calling routine with 0 in register 15
Error - return to calling routine with one of the following return codes in register 15:
- 4 - line error
- 8 - GETMAIN failure
- 12 - active area I/O error
- 16 - error message lost
- 20 - active area out of space

Attributes

Reentrant and nonresident

MESSAGE WRITER (IHKMSG)

Entry Points

IHKMSG
- to build and send a message to the central operator;
- to build and send a message to a terminal user with or without a response;
- to send all messages (broadcast, delayed, and queued) that the user has requested either explicitly or implicitly;
- to build a message.

Register 1 must point to a variable parameter list:

Word 1 - address of TUB for requesting CRJE subtask;
Word 2 - address of TUB for terminal user or zero if message is

for central operator; if message is for specific console, console ID is in the high-order byte of the word of zeros;

Word 3 - address of a four-byte message area (on a word boundary).

The high-order byte of the message area contains:

X'10' - if message is to be built but not sent;
X'01' - if a response is required from the terminal user;
X'02' - if all available messages are to be sent to the terminal user;
X'08' - if message is a general response to a SHOW SESS request;
X'20' - if message is to go to a specific console.

The low-order three bytes of the message area contain:

- serial number, in EBCDIC, of prepared message to be built;
- X'F0F0F0' if no message is to be built;
- X'000000' if message is from another user or operator, and therefore CRJE MSG FR USERID or CRJE MSG FR CENTRAL is to be placed in front of supplied text.

Word 4 - address of an optional parameter area (on a word boundary) containing a one-byte text length followed by data to be inserted in the prepared text;

Word 5 - address of a second optional parameter area with the same format as for word 3.

IHKMSG01- to build and queue a message for later delivery to a terminal user. If the user is inactive, the message is written to the delayed message data set.

Register 1 must contain the address of a variable length parameter list:

Word 1 - address of TUB for requesting CRJE subtask;
Word 2 - address of userid of terminal user who is to receive the message;
Word 3 - address of a 4-byte message area (on a word boundary). The high-order byte of the message area is ignored. The low-order three bytes of the area contain the serial number, in EBCDIC, of the prepared message to be used.
Word 4 - address of an optional parameter area (on a word boundary) containing a one-byte text length followed by text to be inserted in the prepared text.

Word 5 - address of a second optional parameter area with the same format as for word 3.

IHKMSG02- to queue a message that is already built; to send a supplied message to the central operator.

Register 1 contains the address of a three-word parameter list (with high-order bit of the third word turned on):

Word 1 - address of TUB for the requesting CRJE subtask;
Word 2 - address of TUB for an active terminal user or binary zero if the message is for the operator;
Word 3 - address of a 72-byte message (including the message ID) for a supplied message for a terminal user or the address of a halfword message length field followed by a halfword of binary zeros and the message text for an operator message.

IHKMSG03- to list broadcast messages for a central operator. Register 1 contains the address of a parameter list:

Word 1 - address of TUB for the requesting CRJE subtask;
Word 2 - address of a one-byte area containing the console ID for the console to receive the messages.

Function

All messages, whether for the central operator or for the terminal user, are built in the same manner. The prepared section of the message text is read from the SYSMSGS global file and is put in the first 88 bytes of the user buffer. The first two words contain AFIO control information. The key for the record is IHK3xx or IHK4xx for messages for a terminal user. The key is IHK2XXI for messages for a central operator.

If bytes 72 and 73 of the prepared text contain X'0000', no text can be inserted. If text can be inserted, these bytes contain the two offsets at which the supplied text is to be added. If text insertion is permissible and text has been provided, it is moved into the proper place in the prepared message.

If the request is for build only, all switches in the TUB are set exactly as required for a CWRITE macro. The message is placed in the user buffer.

If the message is to be queued for later delivery, a check is made to see if the CRJE subtask that requested the queuing is the subtask for the user who is to receive the message. If this is the case and no message is currently queued for that user (TUBFQEB = X'00000000'), a conditional GETMAIN macro is issued for a 72-byte area to contain the message. The entire text of the message is moved into this area, and the address of the area is placed in the TUBFQEB field. If a message is currently queued for the user and the subtasks are the same, the message is written in the USRMSGS global file. If the user is active, the TUBMSG bit in the user's TUBFLG1 field is turned on. If the user is inactive, the UVRDMSG bit in the user's verification record is turned on.

If the CRJE subtask that requested the queuing is not the subtask for the user who is to receive the message (whether or not a message is currently queued for that user), the entire message is written in the USRMSGS global file. The key used for the message is the seven-byte userid of the recipient user plus a one-byte sequence number to denote the relative position of the message in the list of messages for that user. If the recipient user is active, the TUBMSG bit in the TUBFLG1 field of the user's TUB is turned on. Otherwise, the UVRDMSG bit in the UVRCNTL1 field of the recipient's user verification record is turned on.

If the message is to be sent immediately to a terminal user, a check is made to see if message IDs have been requested. If the TUBMID bit in the TUBFLG1 field is not on, message IDs are not requested, and the message text is moved to the left to remove the message ID from any message whose first four characters are neither CRJE nor BRDb. The TUBDATAL field is set to X'40' if the message ID is not included or is set to X'48' if the message ID is included. If a response to the message is requested, the TUBNOCRN flag is turned on to suppress the carrier return on the terminal after the message is printed. Also, when a response is expected, all switches are left exactly as set by the line administrator, and the input line in the user buffer is not changed. A CWRITE macro is then issued to print the message at the terminal, or a CWRITE R macro is issued to print the message and read a response.

In sending a message to the central operator, the last nonblank character in the message is located, and the length of the text + 4, and minus any trailing zeros, is placed in the halfword located in the user buffer + 4. The correct MCS flags are set in the next halfword. The MCS descriptor codes and routing codes are moved from bytes 75-77 of the prepared text into the area immediately following the text, minus any trailing blanks. For a requesting console routing code, the console ID

is right justified in register 0. For a SHOW SESS command response, the MCS type flags occupy the two bytes immediately after the descriptor codes. The execute form of the WTO macro is issued to print the message at the operator console. Register 1 in this case contains the address of the user buffer + 4.

When the message writer is called to send all available messages, the LOST MESSAGE message is sent if the lost message bit in the TUB is on. The queued message in main storage is sent, and all messages in the USRMSGS global file for this user are sent. The broadcast messages are also sent if the broadcast message bit in the TUB is on.

External Routines

IHKAFI - to obtain messages from the USRMSGS global file and the SYSMSGS global file and to write messages in the USRMSGS global file; also to read from and write to the USERS global file.
IHKLAD - to write messages to a terminal user and read responses.

Tables/Work Areas

TUB

- TUBSTATS - status flags
- TUBREAKN - checked for real or simulated interrupt
- TUBUFFAD - address of 136-byte user buffer
- TUBDATAL - two-byte length of text to be written to a terminal or text read from a terminal (set to 72 bytes for messages containing a message ID or beginning with CRJE or BRDb; set to 64 bytes for messages, whose ID has been deleted)
- TUBUSERID - ID of user active on this terminal (checked to locate TUB for a given user)
- TUBNEXT - address of next TUB in chain (used in search for TUB for a given user to set message lost bit)
- TUBFLG1 - status flags
  - TUBMID - checked to see if terminal user has requested message identifiers on his system messages
  - TUBMSG - checked to see if any messages exist in the USRMSGS global file for a given active user
  - TUBBRD - checked to see if terminal user has requested a listing of broadcast messages.
- TUBFLG2 - status flags
  - TUBLMSGN - checked to see if a message for the user has been lost (if bit is on, LOST MESSAGE is sent to the user)
- TUBFLG3 - status flags
  - TUBNOCRN - set on when the line administrator is called to send a prompt and read a reply (causes no carrier return at the terminal after the prompt)
- TUBFQEB - four-byte address of a 72-byte message located in main storage for the user or binary zero if no message exists
- TUBRAFBF - four-byte address of the area from which or into which the prepared text of a system message, delayed message, user record, or broadcast message is to be written or read (this module always stores the address of the user buffer in this area)
- TUBGBLNM - one-byte field containing number of the global file being referred to:
  - X'01' - prepared text of system messages - SYSMSGS
  - X'02' - user verification records - USERS
  - X'03' - delayed messages - USRMSGS

X'04' - broadcast messages - BRDCST

TUBGBLKY - eight-byte field containing key of record in global file

TUBPARM1 - four-byte area used for parameter list for IHKAFI

TUBAFISW - set to X'80' to request optional AFIO buffer control

AVT

IHKYYCCT - address of CRJE control table (CCT)

IHKYYTUB - address of first TUB in chain

CCT

CCTMSGNO - maximum number of delayed messages (set and checked)

UVR

UVRCNTL1 - flag byte

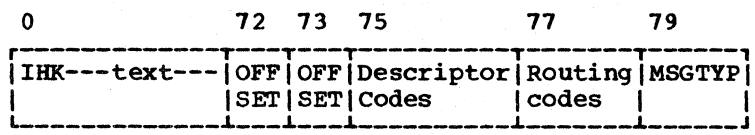
UVRACTVN - checked to see if user is active

UVRDMSG - turned on when a message for an inactive user is added to the delayed message data set

UVRLMSG - turned on when a delayed message cannot be added for an inactive user.

User buffer - first 88 bytes are used as work area. All AFIO reading is into this buffer and all AFIO writing is from this area. All messages are built at TUBUFFAD + 8 and all CWRITE macros are issued from the user buffer. After a CWRITE R, the response is returned in the user buffer.

Format of prepared message text:



Exits

Normal - return to calling routine with one of the following return codes in register 15:

04 - line error

08 - GETMAIN failure

12 - active area I/O error

16 - message lost

20 - active area out of space

24 - invalid userid

Attributes

Resident and reentrant

CRJE LIBRARIAN

The CRJE librarian performs all disk I/O for the active area and for the system and user libraries. The librarian functions are performed by the following modules:

ACTIVE AREA I/O (AFIO)

IHKAST - Active Area Start-up

IHKAWS - Active Area Recovery

IHKEXF - Working Storage Areas for AFIO

IHKIRP - AFIO/LIB I/O Constants, Control Fields and work areas
IHKNBX - AFIO/LIB I/O Constants, Control Fields, and work areas
IHKAFI - Active Area I/O Control/Command Interpreter
IHKEXC - Channel Program Initializer/Requester
IHKGCW - Channel Command Word List Generator

LIBRARY I/O (LIBIO)

IHKBST - Library I/O Start-up
IHKBPM - Library I/O Executor
IHKBSH - Library I/O Shutdown
IHKCDP - Library Condense
IHKWTR - Library I/O Wait
IHKRNQ - Librarian Queue

The IHKEXF, IHKIRP, and IHKNBX modules contain no executable code. There are module descriptions for all of the modules of the librarian except the three main AFIO modules IHKAFI, IHKEXC, and IHKGCW. These three modules are discussed as a group in the last five sections of Active File Input/Output (AFIO). (See Figure 7.)

The AFIO modules perform all update functions within the active area. The CRJE command processors issue AFIO macro instructions to initiate insert, delete, and replace functions on the proper user's active file. The fetching and saving of data sets in user libraries is accomplished by the overlapping of library I/O and AFIO operations. The addition and deletion of data sets in user libraries (CRJE data sets) is performed by Library I/O.

System library information, copied into the global files of the active area at start-up time, is accessed by AFIO, facilitating faster system information retrieval and updating.

AFIO and library I/O are macro driven programs. All requests for librarian service are issued via macros. Macro expansion provides the required codes necessary to perform the requested service.

All requests for manipulation of a user's active file cause AFIO to dynamically define what I/O is necessary to perform the requested operation and to dynamically build the required Channel Command Program, schedule, initiate, and monitor that program, and to check the results of the I/O operation. Figure 5 gives an overview of the modules of AFIO interacting with other modules of the CRJE system.

In most cases, macro request for Library I/O operation will cause standard OS/360 BPAM/BSAM manipulation of the system and user libraries.

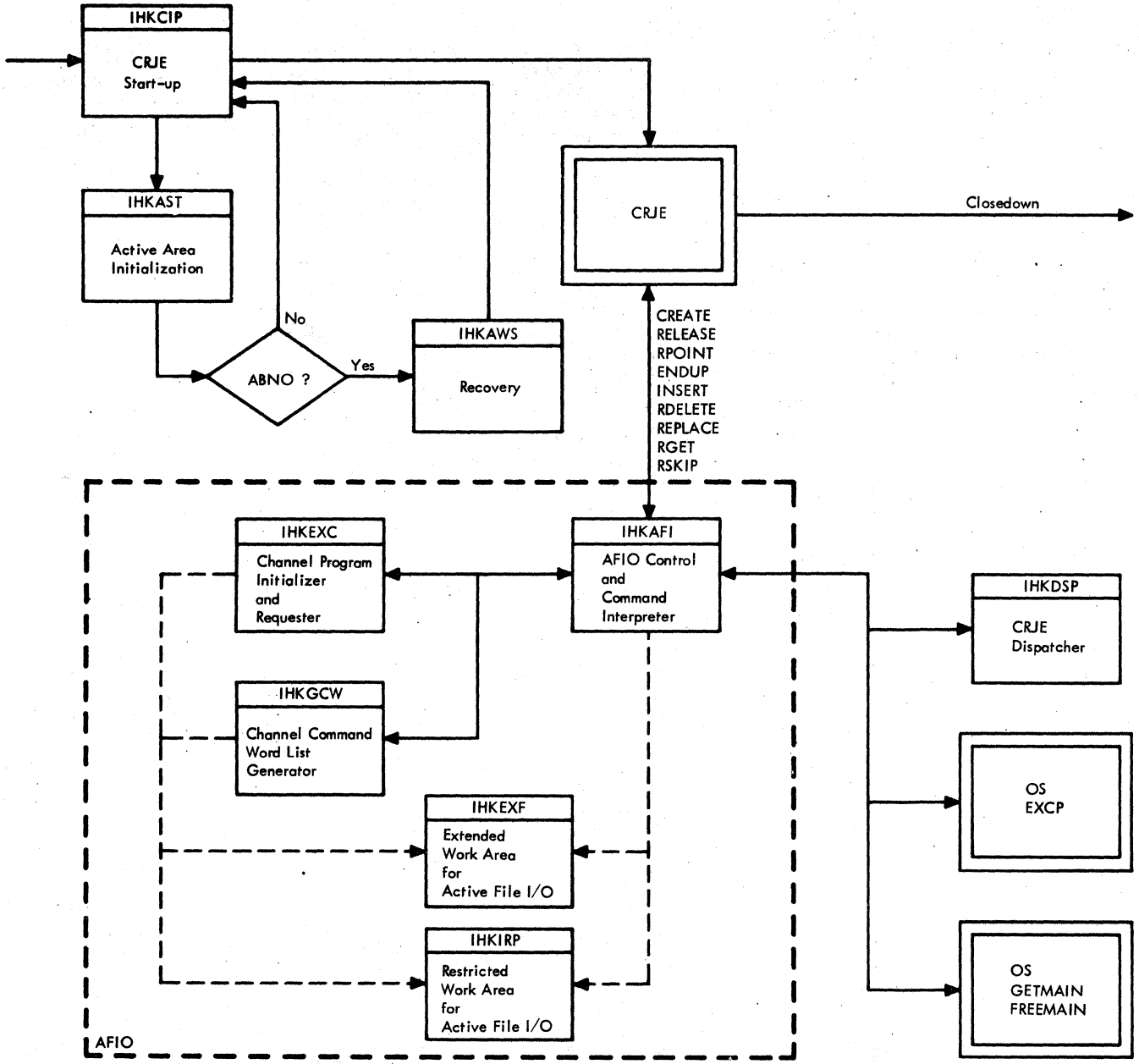


Figure 7. Overview of AFIO

AFIO/Library I/O Constants, Control Fields, and Work Areas (IHKNBX)

Entry Point

This module contains no executable code; therefore, entry points, as such, do not exist.

Function

The constants, control fields, and work areas that this module contain are required by the library I/O module (IHKBPM) and by the active file I/O modules (IHKAFI, IHKEXC, and IHKGCW). Most fields and areas within the KONBOX are sufficiently defined by accompanying comments. However, several areas require further explanation as to their purpose and use.

RECTABFW - This area contains the definitions of the records supported in the active area. These records are defined with the RECDEF macros that specify key length, data length, and an eight-byte work area that is used by the CREATE macro interpreter.

BUFTABFW - This area contains the definitions of the different types of buffer configurations that AFIO supports for CRJE. The buffer configurations are specified with BUFDEF macros, which allow for defining the block size, logical record size, key position, and data position. The resulting constants that are generated by the BUFDEF macros are used by AFIO at macro interpretation time to define the type of buffer being used, where the key begins, and where the data begins. This information is part of the data that is mapped in the IHKEXF and IHKIRP modules.

RECTYPFW - This area specifies which of the record types (defined in RECTABFW) is used with each of the eight global files and the active files. The record types are specified with the AFRECTYP macro.

BUFTYPFW - This area specifies, for each of the eight global files and the active files, the type of buffer for AFIO to use in each of the six AFIO operations that require a buffer. This information is specified with the AFBUFTYP macro. The macro operand is coded with the buffer definition code for each buffer type desired for the following AFIO operations:

- Point with retrieval
- Insert single
- Insert multiple
- Replace
- RGET single
- RGET multiple

This information is also part of the data that is mapped in AFIO work areas IHKEXF and IHKIRP.

GBFORGFW - A 56-byte work area for each of the eight global files is defined in this area. The first two words of each of the eight areas are the queue control element and the queue element for the particular global file. These are used by the queuing module IHKRNQ.

AFIO Extended Work Area (IHKEXF) and AFIO Restricted Work Area (IHKIRP)

Entry Point

These two modules contain no executable code; therefore, entry points do not exist.

Function

The IHKIRP module contains a set of constants that describe the record and buffer formats for the particular file being processed. The IHKEXF module contains all the fields in the IHKIRP module plus the active area DCB, IOB, and a large work area (4K) that is used for transient I/O requirements.

The restricted work area (IHKIRP) is used by the macro interpreter routines in the IHKAFI module and the requester routines in the IHKEXC module, which request channel program execution. The extended work area (IHKEXF) is used primarily by the IHKEXC module and the IHKGCW module for channel program initialization, CCW list generation, channel program execution, and cleanup.

Each AFIO macro request may be in one of three states at a given time:

- interpretation
- execution
- waiting for execution

Interpretation consists mainly of determining which channel program to execute to accomplish the requested function. Interpretation requires a small amount of working storage; therefore, the IHKIRP module is used by AFIO for interpretation. Execution consists of generating, executing, and waiting for completion of channel programs. Execution requires a large work area; therefore, the IHKEXF module is used by AFIO for execution.

The reason for the overlap of fields in the IHKIRP and IHKEXF modules is to facilitate some degree of overlap of the AFIO interpretation and execution processes. An interpreting operation may be overlapped with an execution operation within AFIO. At the time the interpretation process finishes and an execution is requested, the next requested execution is queued for access to the extended work area (IHKEXF). If the IHKEXF module is available for use (no other execution is in process), the requested execution process may begin. If the IHKEXF module is not immediately available, the request is queued and a wait in the CRJE dispatcher results.

AFIO Fields in the Terminal User Block (TUB)

The terminal user block is discussed in detail in the section Data Area Layouts. The fields that are pertinent to AFIO and library I/O are specified, and a small explanation is given for each field. Following is a more complete description of fields in the TUB that are pertinent to AFIO and library I/O:

TUBRAFBF - buffer pointer used in AFIO operations. This field must be initialized to point to the buffer being used for input operations. For output operations, AFIO initializes this field to point to the dynamically allocated buffer into which the records were read by AFIO.

TUBAFQEL - queue element used by the IHKAFI module in AFIO to request access to the AFIO extended work area (IHKEXF).

TUBGBLQL - queue element used by the IHAKFI module in AFIO to request access to the global files in the active area.

TUBAFPR1
TUBAFPR2 - switches used by the macro interpreter routines in AFIO. These switches are initialized by the AFIO macro expansions.

TUBREZUM - half-word linkage save area for internal saving of return displacements within AFIO.

TUBNCCWS - work area used by the IHAKFI module to save the computed maximum number of CCWs required for a particular channel program.

TUBNSRCH - work area used by the IHKAFI module to save the computed maximum number of search arguments required for a channel program.

TUBXTNDF - work area that is initialized by the IHAKFI module after determining the total amount of storage required for a channel program --including buffers, CCW lists, DCB, and IOB. The IHAKFI module of AFIO uses internally the high-order byte of this field to indicate that a buffer allocation is desired.

TUBTRKFW - track address save area used by the IHAKFI module.

TUBACTNM - indicator that is initialized by the AFIO macro CREATE to indicate to the macro interpreter in the IHAKFI module the type of private active file requested.

TUBGBLNM - same as TUBACTNM except that the relation is to the global files in the active area.

TUBGBLKY - key value of the record the processor is attempting to manipulate. This field must be initialized by the module that wishes to issue an AFIO macro for a global file in the active area. In certain AFIO macro forms, the value of the key of a record residing in a global file in the active area is returned to the processor in this field.

TUBBPQEL - queue element used for a request to the IHKRNQ module to gain access to the library I/O facilities.

TUBPAMSW - switch that is initialized by the library I/O macros to inform the IHKBPM module of the options available in the various forms of the macros. The IHKBPM module also uses this field to reflect the status of the library I/O activity that was generated as a result of the macro request.

TUBCNTFS - block definition for multiple operations. This field is shared by AFIO and library I/O to reflect the number of records read for a multiple read request. This field must be initialized by the command processor to reflect the number of records the processor wishes to write for multiple write requests.

TUBNXKEY - key value of the record the processor is attempting tomanipulate. This field must be initialized by the processor that is to issue an AFIO macro for a private active file. In certain AFIO macro forms, the value of

the key of a record residing in a private active file is returned to the processor in this field.

TUBAFCTL - fourteen-word area that contains AFIO control information concerning a user's active file. This information is kept as long as the user's active file exists. Control information, as mapped in the AFCTLDS macro, is stored in this work area.

TUBLNSEQ - a unique sequence number that is assigned to each TUB as it is allocated to a user. AFIO uses this field as part of the process of identifying a particular user with his private active file.

AFIO Macros

These macros are used by the CRJE command processors to gain access to and to modify the global files and private files residing in the active area. For global files, the relative file number is taken from TUBGBLNM and the key from TUBGBLKY; for private files, the relative file number is taken from TUBACTNM and the key from TUBNXKEY.

CREATE

The CREATE macro is used to assign space in the active area to a user for his active file.

Operation	Operand
CREATE	{ IS } [,ACTUM=value] [,FTYPE={GBL }]
	{ O } [ ] [{XGBL}]

IS

Creates new indexed sequential file by performing the following functions:

- Assigning and initializing the index track;
- Updating the master index to point to the index track;
- Modifying TUBACTNM to equal the relative file number if for private file (relative file number=ACTNUM+number of global files is used for master index reference and record description constants in the KONBOX);
- Computing the remainder of the record description constants in the KONBOX;
- Initializing the file control area (TUBAFCTL) to null file, positioned at EOD.

O

Reinitializes the file control area to refer to a dormant file by performing the following functions:

- Modifying TUBACTNM to equal the relative file number if a private file (relative file number=ACTNUM+number of global files is used for the master index reference and record description constants in the KONBOX);

- Computing the remainder of record description constants in the KONBOX;

- Reading index record track to locate the most recent index record, the length of the track and segment bits, and the index track balance.

  Note: The file is not positioned when O is specified on the CREATE macro.

ACTNUM=

For private files this is the relative private file number and TUBACTNM is set accordingly.

Default: ACTNUM=1.

For global files this is the relative global file number and TUBGBLNM is set accordingly.

Default: TUBGBLNM is unchanged.

FTYPE=
GBL and XGBL means that reference is to a global file specified by TUBGBLNM, which must be initialized. XGBL means that exclusive control of the file will be maintained for this line. (The description of this parameter is the same for all the AFIO macros.)

Return Codes:
00 - normal
04 - not one track available to assign for index record (IS only)
08 - GETMAIN failure
12 - I/O error or incorrect macro usage (i.e., CREATE O for nonexistent file)

RELEASE

This macro is issued when a file is no longer needed. The master index is updated to indicate that this active file no longer exists. And all tracks (including index track) assigned to the file (i.e., all extended storage) are released.

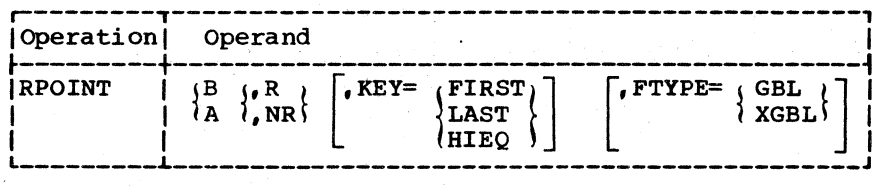
Operation	Operand
RELEASE	A [,FTYPE={GBL / XGBL}]

Return Codes:
00 - normal
04 - not used
08 - GETMAIN failure
12 - I/O error or incorrect macro usage. RELEASE should be assumed completed; the tracks will not be recognized.

RPOINT

This macro is used to position before or after a particular record in the file. For a private file the key of the record is in TUBNXKEY; for global files the key is in TUBGBLKY.

NOTE: If the eighth byte of the key is X'00' or X'FF', the reference is to the first or last record of the file respectively. This is the equivalent of coding KEY=FIRST or KEY=LAST respectively. If the reference is to the last record of the file, the actual key of this record will replace TUBNXKEY or TUBGBLKY.



B
Position is before record

A
Position is after record

R
Retrieval of record and pointer to buffer in TUBRAFBF

NR
No retrieval of record

KEY=FIRST
Position is to first record of file

KEY=LAST
Position is to last record of file

KEY=HIEQ
Indication that the key may not exist or that there are several adjacent records with the same key. If the key does not exist, the record with the highest key not in excess of the specified key is the record referred to (the return code will be set to 4). If there is more than one record with the specified key, the first of the group is the referred to.

Return Codes:
- 00 - normal
- 04 - specified record does not exist, with the following exceptions for a null file:
  - POINT B,NR,FTYPE=GBL/XGBL with 8th byte of key equal to X'00' (i.e., point before first without retrieval).
  - POINT A,NR,FTYPE=GBL/XGBL with 8th byte of key equal to X'FF' (i.e., point after last without retrieval); R is ignored.
- 08 - GETMAIN failure
- 12 - I/O error or incorrect usage of macro. File is not positioned.

Note: Inconsistent parameter specifications:

- R is ignored if KEY=HIEQ is specified.
- KEY=HIEQ is ignored if last or first is the record referred to. R is not ignored in this case.

ENDUP

This macro insures that any pending updates are done. After this macro is executed the file is no longer positioned, and the file control area can be used for other purposes. (If the file control area

(TUBAFCTL) is used for some other purpose, it must be restored by the macro CREATE 0,FTYPE=GBL before any other macro is executed.)

```
┌---------┬-------------------┐
|Operation|  Operand          |
├---------┼-------------------┤
|ENDUP    | [FTYPE= {GBL }]   |
|         | [       {XGBL}]   |
└---------┴-------------------┘
```

Return Codes:

00	-	normal
04	-	not used
08	-	GETMAIN failure
12	-	I/O error or incorrect macro usage. File is no longer positioned and the last set of updates may be incomplete.

INSERT

This macro inserts one record or multiple records into the file at its present location. (No check is made on the keys; it is assumed they are in order.) TUBRAFBF points to the buffer containing a single record or a block of multiple records. The buffer is always released.

```
┌---------┬-------------------------------------┐
|Operation|  Operand                            |
├---------┼-------------------------------------┤
|INSERT   | {S} {,B}  [,FTYPE= {GBL }]          |
|         | {M} {,A}  [        {XGBL}]          |
└---------┴-------------------------------------┘
```

S

Single record

M

Multiple records - TUBCNTFS must have been initialized to the relative first logical record of the block (normally zero). TUBCNTFS + 1 must have been initialized to the relative last logical record of the block + 1 (normally the number of logical records to be inserted). If the I/O completed operation without error, on exit TUBCBNFS equals TUBCNTFS +1.

B

Final position is before the last record that was inserted.

A

Final position is after the last record that was completely inserted

Return Codes:

0	-	normal
4	-	not enough external storage available for records to be inserted (not all records inserted)
8	-	GETMAIN failure
12	-	I/O error or incorrect macro usage. The file is no longer positioned and the insert process is incomplete.

NOTE: If file was not positioned, results are unpredictable.

RDELETE

This macro is used to delete a record and optionally to read the next key. If a key is read, it is read into TUBNXKEY for private files or TUBGBLKY for global files.

```
┌---------┬-------------------------------------------------┐
|Operation|  Operand                                        |
├---------┼-------------------------------------------------┤
|RDELETE  | {NXK} {,B} [,REC=PREVIOUS]  ,FTYPE= [{GBL }]    |
|         | {NK } {,A}                          [{XGBL}]    |
└---------┴-------------------------------------------------┘
```

NXK
No key retrieval. A is the assumed option for the second parameter.

XK
Retrieval of the key of the record following the deleted record. B is the assumed option for the second positional parameter.

B
Position is before the deleted record if NXK is specified. If XK is specified, position is before the record with the next key.

A
Position is after the deleted record if NXK is specified. If XK is specified, position is after the record with the next key.

REC=PREVIOUS

Deletion of the previous record in the file rather than the next record. If this option is not specified, the next record is assumed.

Return Codes:
- 00 - normal
- 04 - end-of-data occurred before end of operation (i.e., previous or next record does not exist, which is beginning or end of file, or XK record does not exist, which means last record in file was deleted).
- 08 - GETMAIN failure
- 12 - I/O error or incorrect macro usage; file is no longer positioned and delete is incomplete.

NOTE: On EOD, the file is no longer positioned.

REPLACE

This macro is used to replace a record and, optionally, to read the next key. If a key is read, it is read into TUBNXKEY for private files or TUBGBLKY for global files.

```
┌---------┬-------------------------------------------------┐
|         |                                                 |
|Operation|  Operand                                        |
├---------┼-------------------------------------------------┤
|REPLACE  | {NXK} {,B} [,REC=PREVIOUS] [,FTYPE= {GBL }]     |
|         | {XK } {,A}                         [{XGBL}]     |
└---------┴-------------------------------------------------┘
```

NXK
No key retrieval. A is the assumed option for the second positional parameter.

XK
Retrieval of the key of the record following the replaced record. B is the assumed option for the second positional parameter.

B
Position is before the replaced record if NXK is specified. If XK is specified, position is before the record with the next key.

A
Position is after the replaced record if NXK is specified. If XK is specified, position is after the record with the next key.

REC=PREVIOUS
Replacement of the previous record in the file rather than the next record. If this option is not specified, the next record is assumed.

Return Codes:
- 00 - normal
- 04 - end-of-data reached before end of operation (i.e., no previous record with REC=PREVIOUS, no next record, or no next key after next record or previous record)
- 08 - GETMAIN failure
- 12 - I/O error or incorrect usage of macro; file is no longer positioned and replacement is incomplete.

NOTE: On EOD, the file is no longer positioned.

RGET

This macro is used to retrieve a single record, a block of records, or a key from the file. Record retrieval causes dynamic buffer allocation. The pointer to the buffer is put in the TUBRAFBF field. If a key is retrieved, it is put in the TUBNXKEY field for private files or in the TUBGBLKY field for global files.

Operation	Operand
RGET	{R / XK} {,S / ,M} {,B / ,A} [,REC=PREVIOUS] [,FTYPE={GBL / XGBL}]

R
Retrieval of one record or multiple records

XK
Retrieval of a key

S
Retrieval of a single record or key

M
Retrieval of multiple records (multiple key retrieval not supported). TUBCNTFS must be set equal to the relative logical record number of the first logical buffer or the block (normally zero). TUBCNTFS + 1 must be set to the maximum logical buffer to be used +1 (normally the maximum number of logical records to retrieve). On completion, TUBCNTFS +1 because the last record retrieved +1 is equal to the number of records retrieved.)

B
Position is before the last, or only, record or key referred to.

A
Position is after the last, or only, record or key referred to.

REC=PREVIOUS

The first, or only, record of the operation is to be the previous record rather than the next record. If this option is not specified, the next record is assumed.

Return Codes:

0	-	normal
4	-	end-of-data reached before end of operation. No buffer allocated or partial block retrieval, depending upon the S or M parameter.
8	-	GETMAIN failure
12	-	I/O error or incorrect usage of macro; file is no longer positioned and no buffer is allocated.

NOTE: On EOD the file is no longer positioned.

RSKIP

This macro is used to skip a record and, optionally, to retrieve a key from the file. If a key is retrieved, it is put in the TUBNXKEY field for private files or the TUBGBLKY field for global files.

Operation	Operand
RSKIP	{NXK / XK} {,B / ,A} [,REC=PREVIOUS] [,FTYPE= {GBL / XGBL}]

NXK
No key retrieval. A is the assumed option for the second position parameter.

XK
Retrieval of the key of the record following the record skipped. B is the assumed option for the second positional parameter.

B
Position is before the record skipped if NXK is specified. If XK is specified, position is before the record with the next key.

A
Position is after the record skipped if NXK is specified. If XK is specified, position is after the record with the next key.

REC=PREVIOUS
Indication that the previous record rather than the next record is to be skipped. If not specified, the next record is skipped.

Return Codes:

0	-	normal
4	-	end-of-data reached before end of operation
8	-	GETMAIN failure
12	-	I/O error or incorrect usage of macro; file is no longer positioned.

NOTE: On EOD the file is no longer positioned.

Active Area Organization

The active area must be allocated on a 2311, 2314, or 2319 Direct-Access Storage Device (DASD) as a single extent. Multiple extents are not supported. A performance degradation occurs if defective tracks, with alternate track assignments, are in the space allocated to the active area. Therefore, an effort should be made to allocate to the active area space that is free from defective tracks to achieve maximum efficiency of AFIO.

Active files are allocated from within the active area a track at a time. Therefore, an active file seldom has adjacent tracks assigned to it. The availability of a track for allocation to an active file is determined by checking the track allocation table (TAT). The TAT is built and initialized at CRJE start-up time by the active area initialization module IHKAST. The active area DCB, which is located in the extended work area module IHKEXF, is opened by the IHKAST module at the same time. The process of initializing the active area is illustrated in the start-up and initialization functional diagram and is discussed in the Initialization and Start-up section of this book.

ACTIVE AREA TRACK ALLOCATION TABLE (TAT): The TAT resides in a GETMAIN area with a pointer to it in the KONBOX. It is initialized at CRJE start-up time to reflect all tracks in the active area available for allocation. The mean cylinder within the extent allocated to the active area is determined by examining the upper and lower extent limits in the Data Extent Block (DEB). The TAT is constructed by assigning relative cylinder number values to each cylinder in turn from both sides of the mean.

Example:

Mean cylinder	=	10
Relative cylinder# + 1	=	11
Relative cylinder# - 1	=	09
Relative cylinder# + 2	=	12
Relative cylinder# - 2	=	08

The TAT is just a sequence of bits, with each bit representing a track. A bit value of 1 means that the associated track is available for allocation to an active file. A bit value of 0 means that the track is not available. The bits are ordered in strings. Each string represents a cylinder of a 2311, 2314, or 2319 disk. Bit 0 of a given string corresponds to track 0 of the cylinder. Bits corresponding to tracks on the first and last cylinders of the active area extent that are not included in the extent are set to zero.

Track allocation to active files is made starting at the mean cylinder and progressing outward is both directions from the mean, a track at a time. Figure 8 gives two examples of Track Allocation Tables. One is for a 2311 DASD and the other for a 2314 DASD.

CRJE ACTIVE AREA FILE ORGANIZATION: The active area is allocated as a sequential data set. The active files are areas within the active area that are allocated to individual active users. Active files contain sequential sets of unblocked records. Each record is 80 bytes long and has an associated 8-byte key, which is the line number. The keys and their associated records are logically arranged in increasing sequential order by key value.

Since tracks within the active area are allocated to active files one at a time, adjacent records are not necessarily on adjacent tracks. Records that reside on different tracks but are part of the same active file are kept in logical sequential order by grouping them into segments and ordering the segments sequentially by using a track chain (see Active Area Track Formats).

A segment is defined as consisting of records that are logically adjacent to each other and reside on the same track. A segment must exist on one track. But more than one segment may exist on a track. If more than one segment exists on the same track, they cannot be logically sequential segments; otherwise, they would exist as one segment.

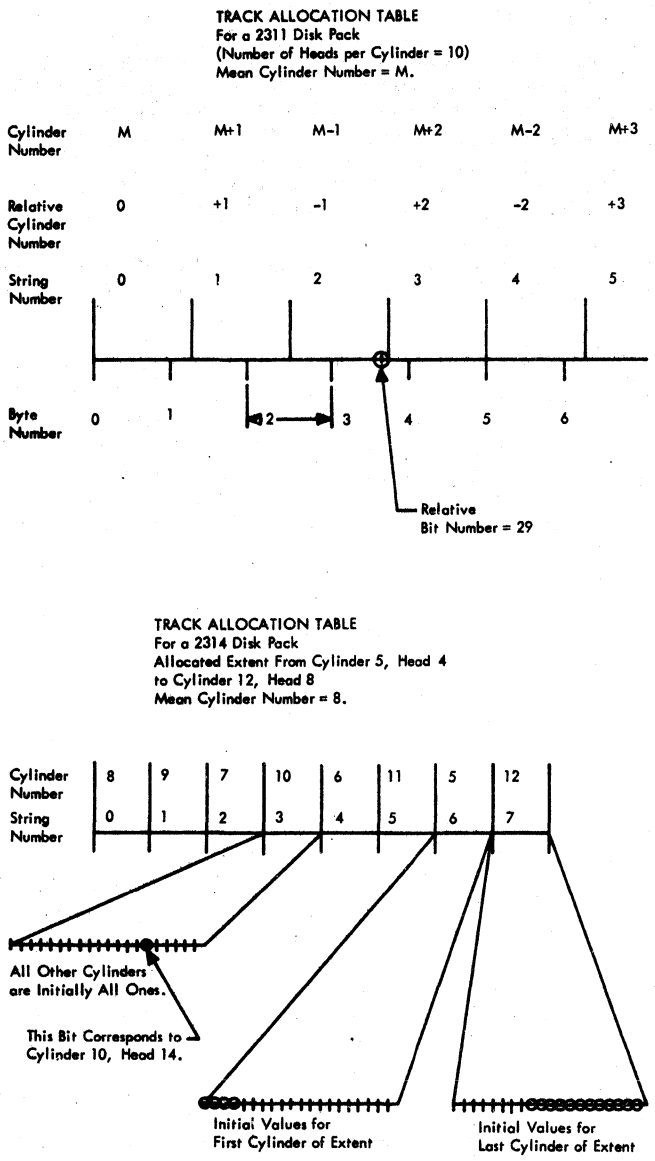
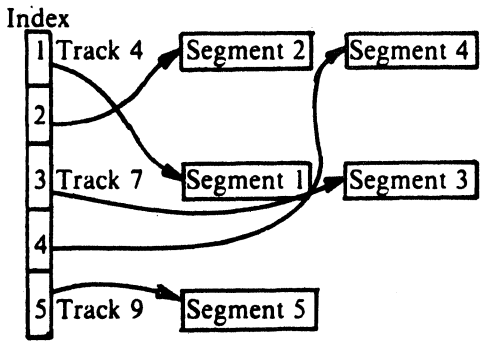


Figure 8. Track Allocation Table

Example:



Each segment is assigned a relative number within a user's active file. A file index is added to facilitate rapid positioning within the file (see Active Area Track Formats). The file index contains one entry for each segment in the file. These entries are in order by segment number. Each entry in the file index contains the following information about the segment:

- track address of segment
- number of records in segment
- physical record number of first and last records in segment
- key value of last record in segment

When file update operations are performed, the index for that file is read into main storage. And if it is necessary to locate a particular segment, the index is searched.

Another part of the active area file organization that facilitates rapid positioning is the master index, which resides on the master index track (see Active Area Track Formats). The master index has an entry for each possible active file in the active area. Each entry in the master index points to the index track of a particular active file and also points to the first data record of that file.

ACTIVE AREA TRACK FORMATS: There are four types of tracks that exist with the active area:

- master index track
- file index track
- data track
- unassigned track (available for file index track or data track)

Master Index Track

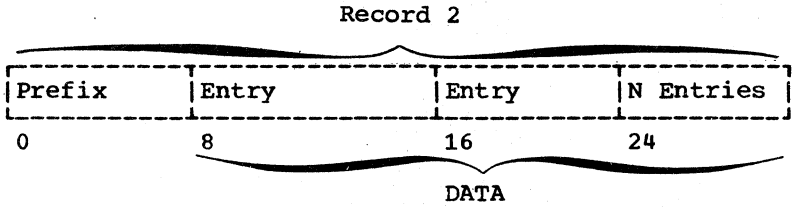
There is only one master index track in the CRJE system. It is the first track allocated in the active area. It is allocated and initialized as a result of the first CREATE macro issued by a command

processor. Subsequent CREATE and RELEASE macros cause the master index to be referred to and rewritten. The RPOINT macro in certain instances may refer to the index but not cause it to be rewritten.

The master index is written on track 0 of the mean cylinder. The master index is composed of two records, which are formatted as follows:

Record 1

Record 1 - (1 byte) not used

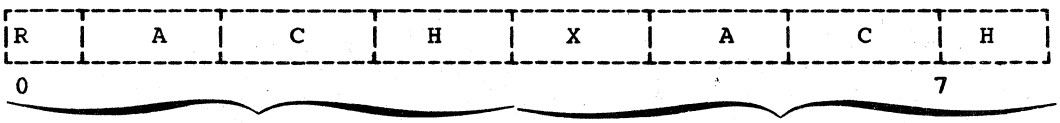


Record 2 - Master Index Record

Prefix - (8 bytes) contains:
- number of global files in system
- number of private files allowed per user
- maximum number of active users
- length of record

Data - contains one entry for each global file and one set (currently 2) of entries for each potential active file allowed a user at one time.

One Entry



Pointer to first record of file

Pointer to index track for this file

R - record number (hardware defined record)
A - access index (always 1; multiple extents not supported)
C - cylinder number
H - head number
X - not used

File Index Track

The file index track is allocated and formatted when an active file is created. Therefore, there is a file index track for each existing active file. This track contains information concerning the data tracks allocated to this active file and information concerning the segments within this active file. Figure 9 shows the format of the file index track.

A file index track for a user's active file or for a global file is one record divided into three sections: count, key, data.

Key

The key section of the record has a 16-byte prefix that is not used. The remainder of the key is the variable-length track list. The track list is composed of a 5-byte entry for each data track that is allocated to this active file but is not full. If a data track is allocated but is full, it has no entry in the track list. The first three bytes of the entry contain a pointer to the data track. The fourth byte contains the number of record entries available on the data track. The fifth byte is the record positon (hardware) of the first unused record.

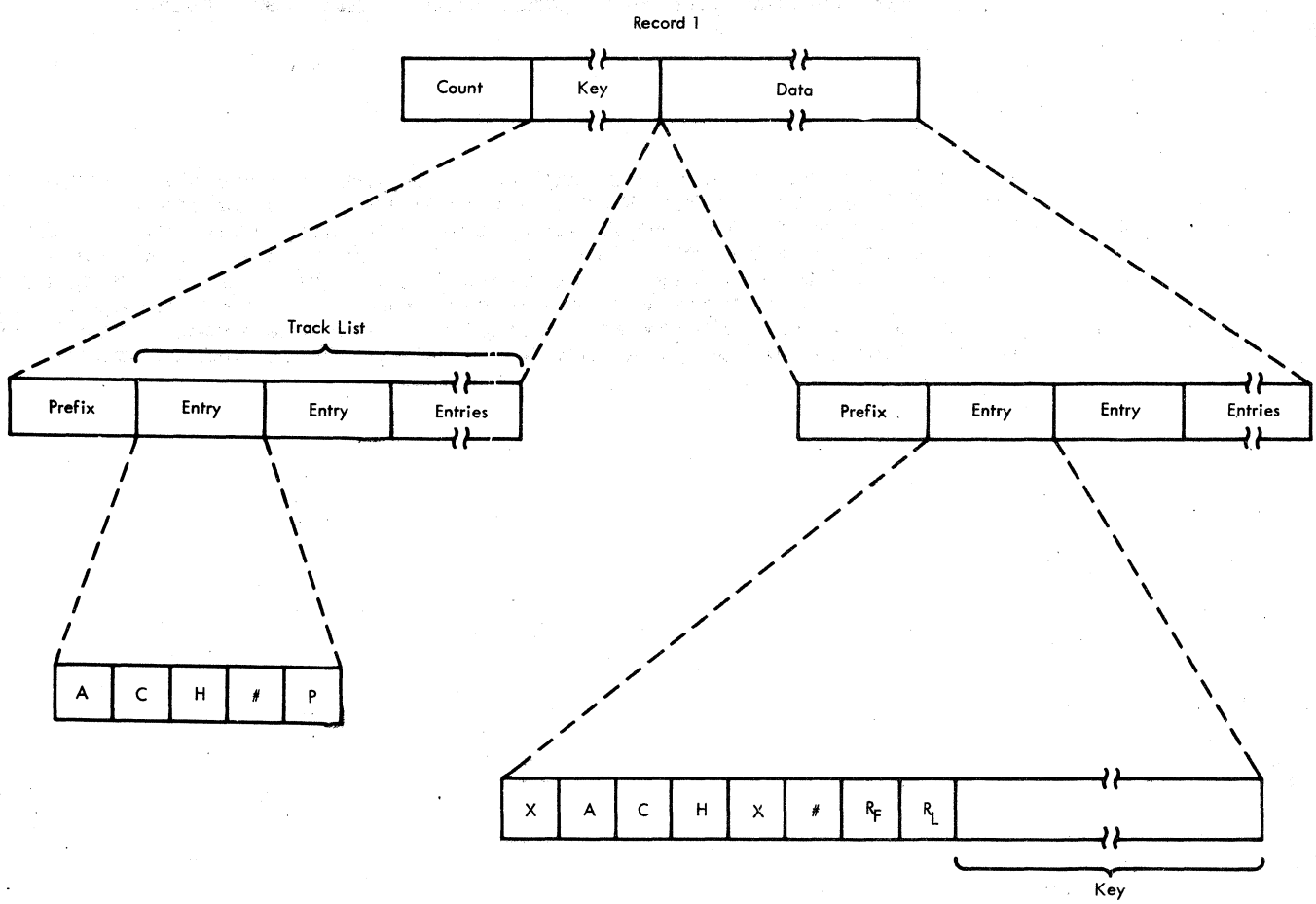


Figure 9. File Index Track

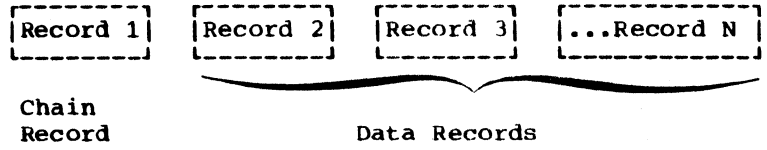
Data

The data section of the record also has an unused sixteen-byte prefix. The remainder of the data section is of variable length and contains a sixteen-byte entry for each segment of the file. The entries are in ascending order by the key value contained in the last eight bytes. Figure 9 shows the format of the entry. The following information is specified in each entry:

Bytes	Symbol	Explanation
0	X	Not used
1-3	AC'H	Pointer to data track
4	X	Not used
5	#	Number of records in segment
6	RF	First (hardware) record in segment
7	RL	Last (hardware) record in segment
8-15	KEY	Key value (left-adjusted) of last data record in segment associated with this entry.

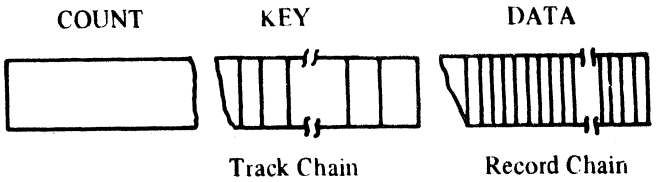
Data Track

There are from one to two hundred data tracks allocated to each active file. (Two hundred is the capacity of the file index track.) Data tracks are allocated to active files one at a time. Each data track consists of two types of records. The first record on the track (hardware record 1) is the chain record. All records following the chain record are data records.

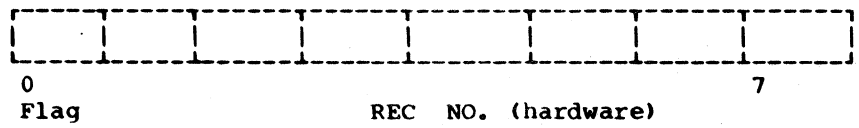


Data records in the active area have 8-byte keys and 80-byte data areas. The format of the data records in a buffer after they have been read from the active area depends upon the values specified in buffer definitions (BUFDEF) in the KONBOX (IHKNBX). Regardless of the format, all data records residing on data tracks in the active area have key lengths of 8 bytes and data lengths of 80 bytes.

The key area of the chain record is a track chain, and the data area is a record chain.



The track chain is of variable length and consists of a 3-byte entry for each data record on the track. The value of the entry is zero unless this entry is for the last record of a segment. If it is for the last record, the value of the entry is a pointer (ACH) to the data track of the next sequential segment. The record chain is also of variable length and contains a one-byte entry for each data area on this data track. (A data record need not occupy this data area. The entry is for the space not for the particular record.) The entry has the following format:



The first entry refers to the first hardware record area, second entry refers to second hardware record area, etc. The value of each entry is the next sequential record number (logical, not hardware), or the value is zero if the record area is not being used. The flag bit is on (1) if the next sequential record (logical, not hardware) is on the next data track. Otherwise, the flag bit is off (0). The record chain facilitates the reading of data records from a data track in logical sequential order, even though they may not be in physical sequential order.

AFIO General Theory

The ten major functions of AFIO are discussed in this section. AFIO refers to the three executable modules, IHKAFI, IHKEXC, and IHKGCW, and to the two non-executable modules, IHKIRP and IHKEXF. The functional

desciptions are not easily defined as being in one particular module or another because of the interaction between modules to perform a particular function. The interpreter (IHKAFI), the requester (IHKEXC), and the generator (IHKGCW) communicate via nonstandard linkages. The ten major functions of AFIO are listed below along with the module that is responsible for initiating the function:

- I/O Scheduler - IHKAFI
- Macro Initializer - IHKAFI
- Channel Program Selection and Macro Interpreter - IHKAFI
- Exit Handler - IHKAFI
- Buffer and Storage Acquisition - IHKAFI
- Channel Program Initializer - IHKEXC
- Channel Program Generator - IHKGCW
- Search Program/Track Data Analysis - IHKEXC
- Channel Program Execution/Monitor - IHKAFI
- Storage Release/Error Handler - IHKEXC.

The result of all AFIO processing is the execution of a channel program to perform the desired or required manipulations on the CRJE user's active file. Figure 10 shows the order in which the functions of AFIO are performed. A short functional analysis is given for each of the ten major functions.

I/O Scheduler

The I/O scheduler is the master control routine for every AFIO macro request received from a command processor. It handles the interface with the issuing processor and provides linkage to the specialized AFIO subroutines that maintain the AFIO data organization.

Macro Initializer

The AFIO macro initializer handles the macro argument that is passed as part of the calling sequence. If a global file reference is requested, this routine handles the queuing for access to the global file. Requests for global file access are serviced on a FIFO basis. The macro argument passed to AFIO is used to locate the proper macro interpreter entry point. Also contained in the macro argument is information concerning multiple or single operation, record or key retrieval, etc. (The macro argument, TUBAFPAR, is discussed later.) The macro initializer also determines whether a buffer is required; and if so, the size and format definition of the buffer to be used for this macro request.

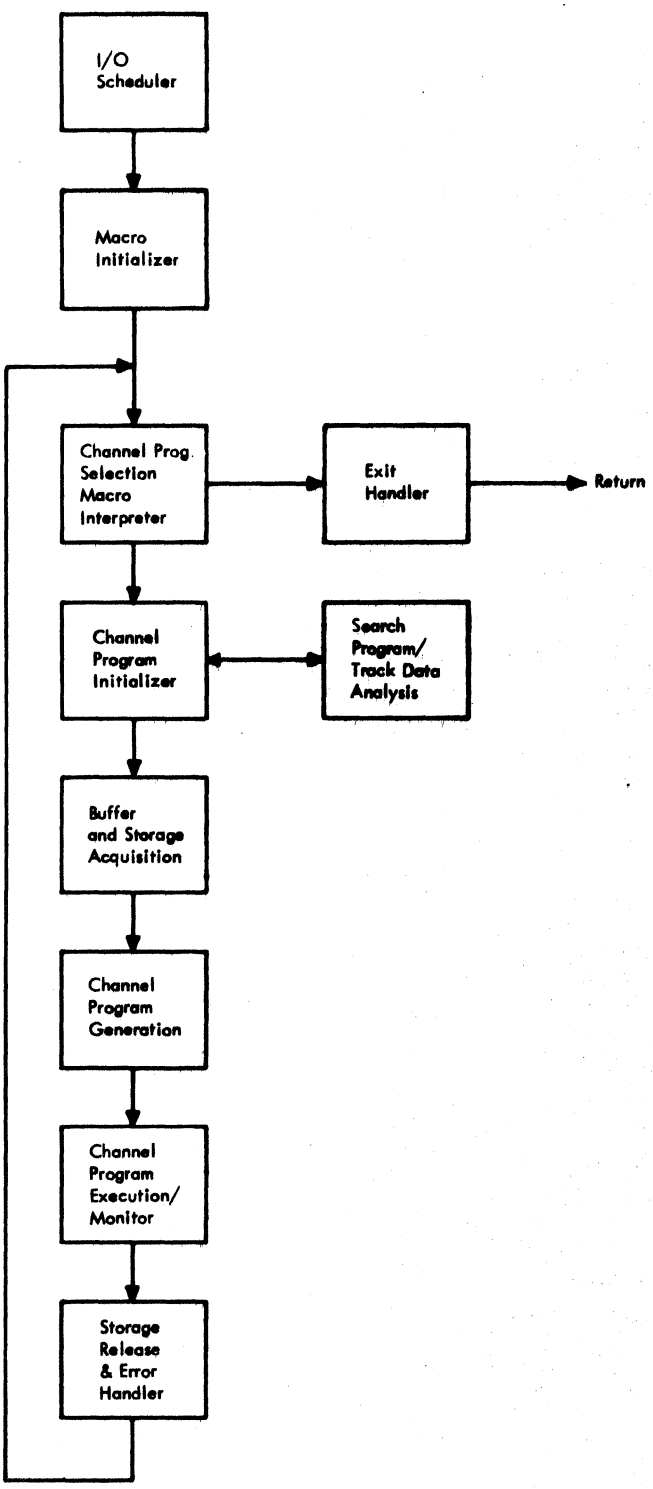


Figure 10 AFIO Macro Request

Channel Program Selection and Macro Interpreters

This routine initially receives control from the macro initializer and then from the storage release handler after I/O is complete. It is

assumed that before this routine is given control the restricted work area has been assigned and initialized and that the primary linkage register (LINK1RG) contains the half-word displacement entry address to the macro interpreter. This routine provides flexible linkage between interpreter and requester routine, the result being an exit to the processor that issued the macro or a request for I/O.

The macro interpreter routines select the next channel program or exit to the issuing processor. The requester routines in the IHKEXC module request an appropriate amount of storage and provide a track address for the channel program. Actually, due to an optional no-op return from the requester, the channel program selection process is not so clearly defined. Common subroutines in IHKEXC can be used by the interpreter routines in AFIO. Many of the interpreter channel program selections are conditional upon requester logic.

Exit Handler

This routine receives control when the execution of the channel programs generated by an AFIO macro is complete. If the access was to a global file, the line is dequeued and the next request is posted. If XGBL was specified, the line is not dequeued. Standard return is made to the processor that issued the macro.

Buffer and Storage Acquisition

Storage acquisition provides and initializes all storage required to generate and to execute the requested channel program and to allocate a buffer if requested. All transient storage is assigned from the extended work area (IHKEXF). Only one request at a time is allowed to use this area; therefore, subsequent requests are queued. When the work area is free, it is assigned to the first line on the queue.

Storage in excess of the minimum extra storage required is assigned to the space provided immediately following the DCB in the IHKEXF module. This area precedes the CCW list and corresponds to the dummy CCW lists in the IHKGCW module. The number of search arguments, the CCW list base, and the search argument base are left in registers for the channel program initializer.

If a buffer is required, a GETMAIN is issued to obtain a buffer of the proper size. AFIO allows each command processor to provide its own buffer for input and/or output operations. If it is providing the buffer, it must turn on the first bit of the TUBAFISW field. AFIO provides a dynamic buffer of the proper size for the RPOINT R macro and for the RGET macro. AFIO also frees the buffers that are used for the INSERT macro and for the REPLACE macro. These services are not provided if the first bit of the TUBAFISW field is set. The buffers provided by the user must be of the same configuration as those buffers gotten and freed by AFIO itself. For single operations to private files the buffer must be 96 bytes long. For single operations to global files the buffer must be 88 bytes long. The first eight bytes are not used by AFIO, so information starts at eight bytes beyond the start of the buffer. These eight bytes are used for control words when AFIO performs dynamic buffer control. For multiple operations to both private and global files, the buffer length is 888 bytes with information beginning 8 bytes beyond the beginning of the buffer. This length (888) supports multiples of 10 and 11 for user libraries and the system library respectively. Smaller multiples may be used but the TUBCNTFS field should be initialized accordingly. The TUBRAFBF field must point to the beginning of the buffer. When a processor is managing its own buffer, it must initialize and maintain the TUBRAFBF field.

Channel Program Initializer and Generator

The I/O operations that are required to provide the requested function are determined by the IHKEXC module. Channel program

initialization routines in the IHKEXC module construct a table of indices that reference channel program elements within the IHKGCW module. Routines within the IHKGCW module then use this table to construct the actual CCW lists.

Search Program/Track Data Analysis

This function is accomplished by a complex set of routines in the IHKEXC module. These routines determine and analyze track content, segmentation, record changing, and other pertinent data concerning data tracks. This is done to ensure that the most efficient channel program will be generated.

Channel Program Execution/Monitor

The functions of these routines consist of executing one or more EXCP macros, monitoring errors, and communicating with the execution control routines in the IHKEXC module. After an EXCP macro is issued, a call is made to the CRJE dispatcher to WAIT for completion of the I/O. Upon return from the dispatcher, error monitoring is performed. Then control is returned (via storage release) to the processor that issued the macro or more I/O is performed, depending upon return displacements generated by the IHKEXC module.

Storage Release/Error Handler

If any storage was acquired for buffers, it is freed. The line is dequeued from the extended work area queue, the next request is posted, and control is returned (via exit handler) to the processor that issued the macro.

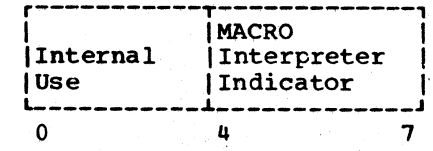
AFIO Macro Argument (TUBAFPAR)

Within every expansion of the AFIO macros is the instruction STH 0,TUBAFPAR. TUBAFPAR is a half word field in the terminal user block (TUB). At the time the STH instruction is issued register 0 contains a unique numeric value. This value is computed by the inner macros issued by the AFIO macros. The following inner macros reside on CRJE.MACLIB:

```
VALARG
RAFIOARG
RAFIOXK
```

The macros examine the operands specified on the AFIO macro and put in register 0 the numeric value that causes AFIO to perform the function requested.

The half-word field TUBAFPAR is actually structured as two one-byte fields named TUBAFPR1 (AFPAR1FS) and TUBAFPR2 (AFPAR2FS). The TUBAFPR1 field is formatted as follows:



Bits 0-3 are not used for input arguments to AFIO but are used instead to reflect certain error conditions and control logic information internal to AFIO

Bits	0 1 2 3	
	1 1 0 0	I/O error
	0 1 0 0	File exception
	0 0 0 1	Internal indication to proceed to next segment.
	0 0 1 0	Internal indication to return to macro interpreter.

Bits 4-7 are initialized by the AFIO macros to reflect the numeric value of the macro being issued. This value is used within the IHKAFI module to determine the proper entry point to the macro interpreters.

All eight bits of the TUBAFPR2 field are initialized by the AFIO macros to the numeric value that reflects the operand options coded on the AFIO macro. Several bit patterns have more than one meaning. The proper meaning of these similar bit patterns is determined by the macro interpreter, which is entered in the IHKAFI module.

Example:

TUBAFPR2	=	X'01'	means old file to CREATE macro interpreter
		X'01'	means final position is before record processed to macro interpreters other than CREATE.

Following are some more examples of how the field is used:

TUBAFPR2	=	X'02'	Read key
	=	X'04'	Read record
	=	X'08'	Exclusive global
	=	X'10'	Position not first or last
	=	X'20'	Position after last
	=	X'20'	Multiple request
	=	X'40'	Next record
	=	X'40'	Positioned to last record
	=	X'80'	Search by record number.

The TUBAFPR2 field is used in the IHKAFI and IHKEXC modules, but it is used more extensively in the search program/track data analysis routines within the IHKEXC module.

AFIO Search/Track Data Analysis Routines

The search program is used to position within a previously determined segment. The chain record is always read and checked. The chain pointer is followed from the first record of the segment to an end-of-segment entry. If the chain does not terminate within the maximum number of records on a data track, or if any entry point is outside the chain, the chain is not considered valid and an I/O error is indicated. The number of available recording areas on the track is determined from the chain record. If there is a track list entry (at least one available record) but its positon is unknown, a switch is set indicating that a track list scan is necessary.

Entry point SRCHRQ is used for sequential positioning. The search program reads the record chain into temporary storage, checks its validity, and moves it to the TUB. If the file was not last positioned on this same track, it is positioned to the beginning or end of the segment for forward or backward processing respectively. The record address has been determined by the chain record update program or the index update program (backward processing necessitates index reference). If the file was last positioned on this track but processing was interrupted for an index record update, the position within the segment is already specified and is not changed. If the record number that

specifies the present position is not found within this segment on the record chain, the search program sets the I/O error indication.

Entry point SRCHXKRQ supports the XK option on the RDELETE, REPLACE, SKIP, and RGET macros. It is used if the key retrieval requires positioning to a new track. If it is not necessary to combine tracks, the file is positioned after the first record of the next segment. The record chain is read into temporary storage, checked for validity, and moved to the TUB. The key of the first record of the next segment is read into temporary storage and moved to the area specified by NXTKEYFW.

After the index record program has deleted the intervening segment, the RDELETE macro interpreter may discover that the previous and next segments are located on the same track. In this case the macro interpreter indicates that the search program is to combine these two segments. If combined segments is specified and XK is not also specified, the file is positioned before the first record of the next segment. When XK is also specified, the file is positioned after the first record of the next segment. To combine segments, the track chain and record chain are read into temporary storage. The end-of-segment entry in the record chain for the first of the two combined segments is replaced by the record number of the first record of the next segment. The end-of-segment entry for the first of these segments is eliminated from the track chain (but saved in the track chain save area). I/O is then initiated to write the updated chain record and write-check on the track. If XK is specified, the key of the first record of the second segment is retrieved. It is later moved from temporary storage to the area specified by NXTKEYFW. The record chain is moved to the TUB.

Entry point SRCHPTRQ is used by the RPOINT macro interpreter. This entry can be used to RPOINT FIRST or LAST, position to an exact key, or search high or equal on a given key. If FIRST is specified, the index record (or the master index if no index record update is needed) is used to locate the first segment of the file, and the search program positions the file after the first record. If LAST is specified, the index record is used to locate the last segment, and the search program positions the file after the last record. The record chain is read into temporary storage, checked for validity, and moved to the TUB. The first or last record of the segment is referred to by physical record number for key and/or text retrieval. If the record number of the last record (RPOINT LAST) is not found within the segment definition in the record chain, the search program indicates an I/O error. If record retrieval is specified, a buffer is requested, and the key and text are read into it. If key retrieval is also specified, the key is moved from the buffer to the area specified by NXTKEYFW. If key retrieval but not record retrieval is specified, the key is read into temporary storage and then moved to the area specified by NXTKEYFW.

When the RPOINT macro is used to position before or after a given key, the macro interpreter uses the index record update program to position the file to the proper segment (if the key is within the file range). Then the search routine is entered with search on exact key specified. The record chain is read into temporary storage, checked for validity, and moved to the TUB. The track is searched for the specified key, and if it is found, its record number (part of count field) is obtained. If the user specified text retrieval, a buffer is obtained and the text is read into it. The key is moved into the buffer later and the file is positioned after the specified record. If the specified key is not found by the channel program, file exception is indicated. If the key is found, but that record is not within the segment according to the record chain (record has been deleted), the search program sets file exception. In both cases the search program positions the file before the first record of the segment. If a buffer was obtained, its release is requested.

When the user specifies the high-or-equal option on the RPOINT macro, the index record update program locates the segment (if the key is within the file range). The search program reads the record chain into

temporary storage, checks its validity, and moves it to the TUB. All the keys on the track are read into temporary storage. The keys of the segment are searched in logical order (according to the record chain) for one that is at least as high as the given key. If the key which satisfies the search is equal to the given key, the file is positioned after that record. If the key that satisfies the search is higher than the given key, file exception is indicated and the file is positioned before that record. (If key retrieval is specified, its key is returned in the area specified by NXTKEYFW.) If no key within the segment satisfies the search, an I/O error is indicated. The search high-or-equal option does not support text retrieval.

When a file contains duplicate keys, RPOINT on exact key gives unpredictable results. Thus the RPOINT high-or-equal option must be specified. If the key that satisfies a search is one of a string of duplicates, it will necessarily be the first of the string. If the key that satisfies a search is equal to the given key, the file is positioned after the first record of the string. If the key that satisfies the search is higher than the given key, the file is positioned before the first record of the string.

If the RPOINT macro (exact key or high-or-equal option) refers to a key that is higher than any key in the file, the index record update program detects this condition, positions to the last segment of the file, and passes an end-of-data indication to the search program. The search program positions the file after the last record of the segment and sets file exception. The record chain is read into temporary storage, checked, and moved to the TUB.

AFIO Register Usage

On all calls to and exits from AFIO, register usage is standard. However, register usage within the three AFIO modules (IHKAFI, IHKEXC, and IHKGCW) and register usage for linkage between the modules is nonstandard.

Several registers within AFIO are fixed. Their contents and/or their usage are not changed from entry to exit. Other registers are variable. Their use changes from subroutine to subroutine within a module. These registers are listed with a short explanation of their purpose.

Fixed Registers

Register 8 (KBXRG) - used as the base register. It is initialized upon entry to AFIO and is not used for any other purpose.

Register 9 (AFWRG) - used as the base register for the restricted work area (IHKIRP) and the extended work area (IHKEXF). These work areas are defined with the EXCPFWDS TYPE=DSECT macro.

Register 10 (BASERG) - used as the program base register for all three of the modules that comprise AFIO. When linkage is made to either the IHKEXC module or the IHKGCW module from the IHKAFI module, this register is initialized with the proper base for V-type constants in the KONBOX.

Register 11 (TUBRG) - used as the base register for the terminal user block (TUB). It is initialized upon entry to AFIO and is not altered during AFIO processing.

Register 12 (FBXRG) used as the work area base register. Work area for private files is the TUB and for global files is the GBFORGFW in the KONVOX. Either area is defined with the dsect macro AFCTLDS. This is the work area into which information from the master index track and the user's file index track is read.

Register 13 (REGD) - used as the AFIO save area register.

Variable Registers

Register 0 (GRORG-GRERG) - used as work register.

Register 1 (GR1RG-GRORG-INCRG) - used as work register.

Register 2 (GR2RG-GRDRG-CYLERG) - used as work register.

Register 3 (GR3RG-GRD2RE-CYLORG) - used as work register.

Register 4 (INTLNKRG-HEADRG) - used most extensively as the branch register for internal AFIO closed subroutines. It is also used as a work register to contain head number information in the AFIO active area track allocation routines.

Register 5 (CCWRG) - used mainly to contain the number of CCWs required for the Channel Program. It is also used in the IHKGCW module as the base register for the CCW list.

Register 6 (SRCHRG) - used mainly as the search argument list base register. The search argument list is generated by the IHKEXC module and used by the IHKGCW and IHKEXC modules.

Register 7 (ICRG) - used as the parameter register for TUBAFPAR (AFIO macro arguments) and as the offset register for internal AFIO linkages. It is also used as the index register to access the proper global file work area in the KONBOX. These work areas are defined by the macro AFCTLDS. It is also used extensively by all three AFIO modules to contain offsets and index values for Channel Program requesting and generating.

Register 14 (LINK1RG)
Register 15 (LINK2RG) - used as linkage registers within and between the controller/interpreter (IHKAFI), the executor/requester (IHKEXC), and the generator (IHKGCW). The linkage conventions used are discussed later.

Subroutines Within the AFIO I/O Scheduler (IHKAFI)

The following routines are in the scheduler and interpreter sections of the AFIO control routine (IHKAFI). These routines are called from various locations within the IHKAFI module. They are listed in alphabetical order with a short description of their function.

BACK2INT

This subroutine restores register 15 (LINK2RG) with the REZOOMLK field; (the IOREQ subroutine stored register 15 in the REZOOMLK field). This subroutine puts in register 15 (LINK2RG) the next required entry offset to the IHKEXC module, and then returns to the NEXTIO subroutine.

EXNTRY0

This subroutine loads register 4 (INTLNKRG) with the entry point of the IHKEXC module and then goes to the EXNTRY1 subroutine.

EXNTRY1

This subroutine initializes register 10 (BASERG) for the IHKEXC module and loads register 15 (LINK2RG) with the secondary return address. This subroutine is called from the IOREQ subroutine by using register 14 (LINK1RG). Therefore LINK1RG contains the primary return address and is used by the requestor routine to return to the IOREQ subroutine if I/O is not requested. Register 15 (LINK2RG) is loaded with LINK1RG +8 and is used by the requestor routine to return the the IOREQ subroutine if I/O is requested. Both the primary and secondary returns are saved and a branch on register 4 (INTLNKRG) is made to the proper requestor routine in the IHKEXC module.

EXNTRY2

Upon entry to this subroutine register 4 (INTLNKRG) is loaded with the proper entry point to the IHKGCW module (generator). Also at this time the number of search arguments and the number of CCWs required (as computed by the IHKEXC module) are known and entry is being made to the IHKGCW module to build the Channel Program that will be executed.

GETFNUM

If the I/O request is for a private file, this subroutine inserts the file number (TUBACTNM) into register 7 (ICRG), loads the address of the TUBNXKEY field into register 2 (GR2RG), and returns. If the I/O request is for a global file, this subroutine inserts the global file number (TUBGBLNM) into register 7 (ICRG), loads the address of the TUBGBLKY field into register 2 (GR2RG), and returns.

GETGBASE

This subroutine loads register 1 with the address of the beginning of the proper (determined by TUBGBLNM) global file work area in the KONBO and then returns to the caller.

GTFBASEO

If a private file is being accessed, this subroutine returns to the caller with register 12 loaded with the address of the TUBAFCTL field, which is the AFIO work area in the TUB. If a global file is being

accessed, this subroutine goes to the GETGBASE subroutine; upon return loads register 12 with register 1; and returns to the caller with the address of the SETCON subroutine in register 15 (LINK2RG).

IOREQ

This subroutine is called from the NEXTIO subroutine if the macro interpreter (called from NEXTIO) requests I/O operation. The linkage offset entry to the IHKEXC module that is in register 14 (LINK1RG) is saved in register 7 (ICRG). The half-word return address is computed and stored in the REZOOMLK field so that macro interpretation will be resumed at the proper place. Register 2 (GR2RG) is loaded with the offset to the beginning of the extended work area in the IHKEXF module. Then the subroutine performs a branch and link on register 14 (LINK1RG) to the EXNTRY0 subroutine.

LOCINTRO

The macro code number is gotten from the TUBAFPR1 field; the macro interpreter entry point displacement in register 3 (GR3RG) is computed; and a return is made to the caller with the macro interpreter entry point in register 15 (LINK2RG).

NEXTIO

This subroutine branches to the proper macro interpreter. The offset to the proper interpreter from the label MACTAB is in register 15 (LINK1RG). Return from the macro interpreter is on register 14 (LINK1RG) or register 4 (INTLNKRG) to return to the processor that issued the macro.

SETCON

A branch is made to the GETFNUM subroutine to get the file number and next key pointer. Using register 7 (ICRG), the record type, record definition, and buffer description constants are moved from the KONBOX into the restricted work area (IHKIRP). This subroutine returns to the processor that issued the macro.

AFIO Internal Parameter Passing and Linkage

Parameter passing and module linkages within the AFIO modules (IHKAFI, IHKEXC, and IHKGCW) are nonstandard to increase performance and to lower the main storage requirements. There are several basic concepts involved in the processing of building and executing a channel program.

A basic AFIO concept is that the I/O required for two similar functions may require two dissimilar channel programs to be generated. The search/track data analysis routines of the IHKEXC module does a lot of checking to determine position, segment arrangement, record location, empty records, etc. This is done in order that the most efficient channel program will be generated.

In support of this concept there is a table of half-word displacements located at the beginning of the generator module (IHKGCW). These half-word constants are displacements to the twenty-seven CCW list generating routines within the IHKGCW module. The IHKEXC module also has a table of displacements and constants at its beginning. Although this table is constructed differently, its function is the same -- to allow flexibility in channel program generation. The displacements are

to the forty-seven different requestor and executor routines in the IHKEXC module. Linkage to these routines is invoked by the macro interpreter routines in the IHKAFI module.

The initial entry to the proper macro interpreter routine is made by the I/O scheduler at the NEXTIO label in the IHKAFI module. Register 15 (LINK2RG) has been previously loaded with the offset to the proper interpreter (from MACTAB table). The macro interpreter is entered with the following sequence:

```
BAL     LINK1RG,INTERPRETER
B       I/O REQUEST
B       RETURN
```

The interpreter returns on LINK1RG +4 if the macro request is invalid. Normally return is made on LINK1RG.

The exit from the interpreter is with the following sequence:

```
BALR    LINK1RG,LINK1RG
DK      REQUESTOR INDEX
DK      EXECUTOR INDEX
```

The DK macros expand to a one-byte value that indexes to the proper half-word value at the beginning of the IHKEXC module. The first index is to the requestor routines and the second is to the executor routines in the IHKEXC module. The BALR instruction serves a dual purpose. It returns control to the location that initially called the interpreter (NEXTIO). And it leaves in LINK1RG the address of the first index value (DK REQUESTOR).

Upon return to the NEXTIO subroutine from the interpreter, a branch to the IOREQ label is made. At this time the interpreter return address (and a DK index value) is saved as a half-word displacement in the REZOOMLK field. A branch is then made (via the EXNTRY0 routine in the IHKAFI module) to the proper requestor routine in the IHKEXC module with the following sequence:

```
BAL     LINK1RG,REQUESTOR
B       BACK2INT
        request I/O
```

If the requestor routine returns on LINK1RG, a branch is made to the BACK2INT label. The REZOOMLK field is used to load register 15 (LINK2RG) with the interpreter return address, bumped passed the DK indices, and a return is made to the interpreter routine via the NEXTIO subroutine again, and the process is repeated.

If the requestor routine returns on LINK1RG +4 (I/O request) the following occurs: The line is put on the extended work area queue and control is lost to the CRJE dispatcher until this request is returned, a buffer is requested if required and the control fields, search argument list, and CCW lists (DUMMY) are initialized. LINK1RG is loaded with the REZOOMLK field, bumped by one to reference the second DK index (executor), and a branch is made to the proper executor routine in the IHKEXC module via the EXNTRY0 subroutine in the IHKAFI module. The following sequence is used:

```
BAL     LINK1RG,EXECUTOR (initial entry)
B       SORT
B       NOSORT
```

SORT and NOSORT refer to the two entry points in the generator.

Each channel program is constructed in two phases. The executor routine constructs a table of indexes. These indexes reference various channel program elements in the following manner. The table of indexes is made up of one-byte values that are used as an index into the

half-word displacement table at the beginning of the IHKGCW module. The generator uses this table to branch to various CCW generating routines within itself to construct the desired channel program. The SORT/NOSORT entries are used to either sort the search argument list or to leave it as it exists. The decision is made in the executor module and the proper return is made to the I/O scheduler on either LINK1RG or LINK1RG +4.

Upon return from the executor module, the return address to the executor module is in register 14 (LINK1RG). And this value is saved in the EXRETLK field, and the proper branch (SORT/NOSORT) is taken to the generator module via the EXNTRY2 subroutine in the IHKAFI module.

The search argument list and the dummy CCW list are made available to the IHKGCW module by saving the location in the restricted work area. The table of index entries built by the executor module is actually a one-byte field in the search argument list entries. The IHKGCW uses these values to index to its displacement table and branch to the proper routine to build a section of the channel program. This process is repeated until the search argument list is completed. At this time the channel program is complete and return is made to the I/O scheduler routine.

The scheduler routine issues an EXCP macro and goes to the CRJE dispatcher to await I/O completion. Error checking is then performed by the scheduler. The EXRETLK field (return to executor after I/O) is used to load LINK1RG. Return is then made to the executor with the following sequence:

```
BAL     LINK1RG,EXECUTOR
B       ENDIO
B       EXCP
```

The executor returns on LINK1RG if processing is complete or on LINK1RG +4 if the channel program is to be issued again (for write checking). If return from the executor is on LINK1RG, the one final call is made to the executor. The return from that call is used to determine if a buffer should be released or not. If required, the buffer is released, the line is removed from the extended work area queue, and return is made to the processor that issued the macro.

LIBRARY INPUT/OUTPUT

Library I/O Macros

RFIND

This macro initiates library I/O operations. It must be issued before any other library I/O macros except the RSCRATCH macro. If no operands are specified, the options used previously are assumed.

Operation	Operand
RFIND	[O / I / F] [,S / ,NS]

O(utput)
Output operations are initiated. RWRITE and RCLOSE macros can be issued.

I(nput)
    Input operations are initiated. RREAD and RCLOSE macros can be issued.

F(DI)
    The directory of a BPAM data set is read. RREAD and RCLOSE macros can be issued.

S
    Standard library I/O access is requested. Information is obtained as follows: library name of the form CRJE.LIB.userid, where userid has a maximum of eight characters and is taken from the TUBUSRNM field; the member name is taken from TUBPMFNM; ddname for OS job control is taken from the BPPARMFS field in the KONBOX.

NS
    Nonstandard library I/O access is initiated. The data set name must be of the form CRJE.LIB.userid where the userid has a maximum of eight characters. The BPPTRFS field in the KONBOX must have been initialized to point to an address list of the form:

```
DC  A(dsname)
DC  A(ddname)
```

Return code on completion:
- 00 - file name found (if applicable); no error
- 04 - file name not found (if applicable)
- 12 - I/O error in reading directory
- 16 - directory full (O,S operands only)

RREAD

The RREAD macro reads a block from a user library or from the system library.

Operation	Operand
RREAD	none

Return code on completion:
- 00 - normal
- 04 - end of data (no block retrieved)
- 08 - GETMAIN failure
- 12 - I/O error in reading block.

If no errors are encountered, the number of logical records read is returned as a half-word value in the TUBCNTFS field. If EOD is returned, the TUBCNTFS field is unpredictable. The buffer address is returned in the SFBUFF1 field of the KONBOX. If EOD or I/O error is encountered, the buffer is released before control is returned to the caller.

Prerequisites: a return code of 0 or 16 from the RFIND macro.

RWRITE

This macro writes a block of records in a user library or in the system library.

Operation	Operand
RWRITE	none

Return code on completion: none.

The buffer address must be in the SFBUFF1 field of the KONBOX. The number of logical records to be written must be placed in the first byte of the TUBCNTFS field by the calling routine. Zero is put in this byte before returning to the caller.

Control is returned to the caller before the I/O operation has completed. To insure that the write has finished and to determine the results of that operation, the IHKWTR module must be called. Upon return from the IHKWTR routine the return codes are:

00 - last write normal
04 - end-of-volume
08 - GETMAIN failure
12 - I/O error in writing block.

The routine issuing the macro must ensure that the last write has finished before issuing another library I/O macro for that file.

Prerequisites: a return code of 0 or 4 from the RFIND macro and a return code of 0 on all previous RWRITE macros.

RSCRATCH

The purpose of this macro is to delete a data set (member) from a user library.

Operation	Operand
RSCRATCH	none

Return code on completion:
00 - normal
04 - name not found
08 - GETMAIN failure
12 - I/O error in reading/writing directory

Input fields are the same as described under RFIND, S(tandard) except that file attributes are ignored.

Prerequisities: Same as RFIND S(tandard). Currently only S(tandard) access is supported.

RCLOSE

The purpose of this macro is to terminate library I/O operations.

Operation	Operand
RCLOSE	[STOW] [NOSTOW]

STOW
A library I/O STOW is performed to record or update the file and FDI in the directory. The RCLOSE macro is ignored if

- the RFIND macro was type F or I,
- an I/O error occurred, or
- no RWRITE macros were completed successfully.

Return code on completion:

00	-	normal
04	-	EOV
08	-	GETMAIN failure
12	-	I/O error
16	-	directory full

Prerequisities: library I/O operations must have been initiated for this file, and the last RWRITE macro must have been completed.

NOTE: All library I/O operations except the RSCRATCH macro must be terminated by issuing the RCLOSE macro. No other library I/O access is possible until a line releases control of the module by use of this macro.

Librarian Queue Module (IHKRNQ)

Entry Point

IHKRNQ

Function

The IHKRNQ module controls the queuing necessary for sequential access to the following CRJE system resources: user libraries, system global files, and the serially reusable functions of the syntax checker interface and the CRJE SUBMIT command.

Each queue handled by this module has associated with it a queue control element. Each routine requesting access (command processors or AFIO) has a queue element to associate with the queue control element. The control element and the queue elements that are associated with the resources that the queue module controls are as follows:

Q Control Element*	Q Element	Resource
AFQCTLFW	TUBAFQEL	AFIO extended work area
BPQCTLFW	TUBBPQEL	User libraries
GBQCTLFW	TUBGBLQL	Global files (One Q control element exists for each global file.)
SUBQCTEL	TUBSUBQL	Job submission
SYNQCTEL	**	Syntax checker interface

*All queue control elements are located in the KONBOX.

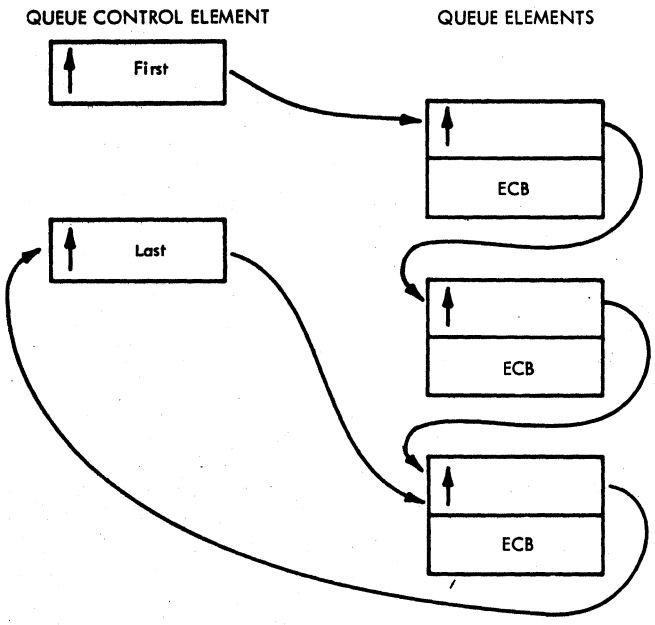
**The syntax checker interface module gets main storage dynamically for its required queue elements.

Queue Elements

When this module receives control, register 1 must point to a parameter list that contains the address of the queue element and the address of the queue control element. Register 13 must point to a 72-byte save area provided by the calling routine.

A check is made to determine if the specified queue is empty. If it is empty, the queue element is placed as the first in the queue and control is returned. If the queue is not empty, the queue element provided by the calling routine is placed on the end of the queue, a GETMAIN is issued for a save area, and a call is made to the CRJE dispatcher to wait until the queue element for this request is posted.

When the queue element is posted, the dispatcher returns control to the IHKRNQ module. Then this module frees the storage for the associated work area and returns control to the calling routine.



The IHKAFI module dequeues all requests for the global files and the AFIO extended work area. The IHKBPM module dequeues all requests for user library accesses. The SUBMIT command processor and the syntax checker interface modules dequeue themselves. The dequeuing process consists of pointing the first word of the queue control element to the next queue element in line and posting the ECB of that queue element. The dequeuing sequence includes the posting of the ECB associated with the next queue element in the queue. This ECB is the ECB provided for the dispatcher by the queuing routine (IHKRNQ) when it calls the dispatcher.

External Routines

IHKDSP - CRJE dispatcher, to wait for queue element to be posted.

Tables/Work Areas

TUB (see chart of queue elements)
KONBOX (see chart of queue elements)

Exits

Normal - branch is to dispatcher if requested queue is not empty.
- to the calling routine if queue is empty or when queue element is first on the queue with a return code of 0 in register 15.

Error - return is to calling routine with a return code of 4 in register 15 indicating a GETMAIN failure.

Attributes

Resident and reentrant

Library I/O Module (IHKBPM)

Entry Point

IHKBPM

Function

This library I/O module is the interface module responsible for gaining access to user libraries and to the CRJE system library. Access to these libraries is obtained by issuing CRJE library I/O macros. The IHKBPM module uses OS BSAM and BPAM to provide the library I/O requested.

This module allows only one line at a time to gain access to a user library. No other requests for library I/O are processed until the library currently being processed is closed (RCLOSE) or scratched (RSCRATCH). The user verification module (IHKUTM) employs the services of the queuing module (IHKRNQ) to queue all requests for library I/O. The requests are serviced in FIFO order.

One DCB in this module is used for reading, writing, and directory processing of all user and system libraries.

Since the functions this module provides is dependent upon the macro issued, it can be described best by these macros. Each library I/O macro has a processor within this module that provides the function requested.

RFIND

- Synchronous routines are established (end-of-data, synad exit).
- Data set is opened, if necessary, by the read job file control block (RDJFCB) and OPENJ macro. The data set name field is generated in the JFCB and merged fields are zeroed out before the OPENJ is issued. If the data set is to be opened for input, the DCB EXIT is also established before the OPENJ is issued.
- The BLKSIZE and LRECL fields are set to 256 if the directory is to be read. Otherwise, they are checked for nonzero entries. If LRECL is zero, 88 is the assumed value. If BLKSIZE is zero, the assumed value is the maximum buffer size (880).
- The build list is created for the user libraries.
- BLDL is issued to locate the data set(s).
- Directory space is checked if 0(output) is specified.
- Directory is updated for user libraries.
- BPAM FIND C is issued to prepare for the first read if opened for input.
- Control is returned to the routine that issued the macro with one of the following return codes:
  00 - normal
  04 - name not found (if applicable)

08 - directory full
12 - I/O error in reading directory.

RREAD

- Input buffer is obtained.
- BSAM read is initiated.
- Wait for I/O completion.
- Record count is computed and stored in COUNTFS.
- If 80-character logical records are read, they are expanded to 88-character records.
- Buffer is released if end-of-data or I/O error.
- Control is returned to issuing routine with one of the following return codes:
  00 - normal
  04 - end-of-data
  08 - directory full
  12 - I/O error.

RWRITE

- Block size is computed from logical record size and record count.
- If the user library is an 80-character library, the blocks are truncated from 88 to 80-character records.
- BSAM WRITE is initiated.
- Control is returned immediately if I/O is not complete.
- If I/O is complete, a check is made for EOV or other error and buffer is released.
- Control is returned to the routine that issued the macro.

Note: The return code is set by the wait module (IHKWTR) when I/O is complete. I/O can be overlapped by initiating other I/O operations before the previous BPAM write has been completed. A check must be made to ensure that the BPAM write is completed, before requesting further BPAM services. The IHKWTR module may return one of the following codes:

00 - normal
04 - EOV
08 - not used
12 - I/O error.

RSCRATCH

- Data set is opened by the first two functions described under the RFIND macro.
- Data set is deleted by the macro STOW D.
- Data set is closed.
- User is dequeued.
- Return to issuing routine with one of the following condition codes:

00 - normal
04 - name not found
08 - not used
12 - I/O error.

RCLOSE

- Build list set up for STOW macro.
- Exit to close data set if requested or required by previous events.
- If output and writes were done without error, a STOW A or R is issued to store file name in directory and write end of data.
- If a user library was opened for input and if the name was found, a STOW R is issued to update the directory.
- Data set is closed.
- Line is dequeued.
- Return to routine that issued macro with one of the following codes:

00 - normal
04 - EOV
08 - directory full
12 - I/O error.

External Routines

None

Tables/Work Areas

TUB
KONBOX

Exits

Normal - return to routine that issued macro with proper return code
Error - none

Attributes

Serially reusable and resident

Library I/O Wait Module (IHKWTR)

Entry Point

IHKWTR - register 1 must contain a pointer to the TUB.

Function

The purpose of this wait module is to allow overlapping of I/O when writing to a user library or CRJE system library. Overlapping is achieved by the RWRITE (library I/O) macro initiating the I/O and this WAIT module testing for errors and exceptional conditions at the completion of the I/O. A call to this routine must be made for each RWRITE macro before another RWRITE macro may be invoked.

Upon entry this module checks whether or not the I/O is complete. If the I/O is not complete this module waits in the CRJE dispatcher until the I/O is posted as complete. A CHECK macro is then issued to inspect the results. If an error occurred, the CHECK macro passes control to the SYNAD routine in the library I/O module, which sets the I/O error flag in the TUBPAMSW field and returns. When control is returned from the CHECK macro, this module frees the output buffer, moves the TUB buffer pointer to the BPAM output buffer location, gets the completion code from the TUBPAMSW field, and returns to the caller.

If the routine using this wait module specifies that it will control its own buffers (bit 0 of TUBAFISW field set to 1), no freeing of the output buffer or switching of the TUB AFIO buffer pointer (TUBRAFBF) to the library I/O output buffer (SFBUFF1 in KONBOX) is done.

If the I/O is posted as complete when this module gets control, a check is made for the following two exceptional conditions:

End-of-volume - When end-of-volume is detected by the RWRITE processor (IHKBPM), the end-of-volume switch in the TUBPAMSW field is set and the output buffer is freed before control is returned to the caller (unless buffer control is specified).

I/O Error - When the library I/O module (IHKBPM) determines that START I/O failed on the write operation, it sets the I/O error switch in the TUBPAMSW field and frees the output buffer (unless t error switch in the TUBPAMSW field and frees the output buffer (unless buffer control is specified).

If either of these conditions is found, the issuing of the CHECK macro and the freeing of the output buffer are bypassed by the IHKWTR module. If either of these two conditions is not found, processing is resumed at the point where the CHECK macro is issued.

External Routines

IHKDSP - CRJE dispatcher to wait on I/O
BPAM (CHECK) - to inspect results of I/O

Tables/Work Areas

TUB
- TUBSAVE - 18-word save area in TUB
- TUBPAMSW - (BPAM switch byte) bits 6 and 7 used to set return code; bits 4 and 6 inspected for I/O error
- TUBRAFBF - (AFIO buffer pointer) moved to library I/O buffer pointer

KONBOX
- SFBUFF1 - (library I/O buffer pointer) replaced by TUBAFBF

Exits

Normal - return to calling routine with one of the following return codes in register 15:
- 00 - normal return, I/O complete with no errors
- 04 - end-of-volume
- 12 - library I/O error

Error - none.

Attributes

Resident and reentrant

Library Condense Module (IHKCDP)

Entry Point

IHKCDP - register 1 must point to a two-word parameter list containing the address of the TUB and the address of the AVT.

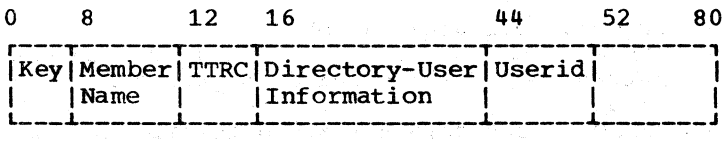
Function

The function of this module is to recover the lost space within a partitioned data set. This space becomes unavailable whenever a member of a PDS is deleted or updated and then written back. This module copies the members to the beginning of the PDS, thereby recovering the lost space and making it available for use at the end of the data set.

Condense is a serially reusable resource. When it gains control, it requests exclusive control of global file #6 (utility sort file reserved for condense). Once the condense module has control of this utility file, it becomes serially reusable, and all subsequent requests for this utility file are delayed until the condense module relinquishes exclusive control of it.

The queuing module (IHKRNQ) is used to get the condense module (IHKCDP) on the queue for library I/O. Since the condense process is time consuming, the IHKCDP module only takes exclusive control of library I/O when it is the only function requesting the resource. This is accomplished in the following manner: The IHKCDP module requests control of library I/O via a call to the IHKRNQ module. (Control is not returned from the IHKRNQ module until that request for condense is at the top of the queue for library I/O.) The IHKCDP dequeues itself from the library I/O queue and checks to see if it was the only requester on the queue. If it was not, this module goes back to the queuing module again and continues this loop until it is the only one on the queue. Once the condense module is the only requester on the library I/O queue, the IHKRNQ module is called to place the condense module on the library I/O queue.

The partitioned data sets directory is opened for input. All directory entries are read and sorted into global file #6 with the TTR and five blanks as the eight-byte key. The format of the records in the utility sort file is as follows:



When all members of the PDS have been sorted on the TTR, the output DCB (contained in the IHKCDP module) is opened and the utility sort file is positioned to the member with the lowest TTR. The address of the next block to be read (on processing the first member, the address is the block following the directory) is moved from the input DCB to the output DCB, and the TTR in the output DCB for the beginning of this member is updated to reflect the address of the next block to be read.

A member from the utility sort file (lowest TTR not processed) is retrieved, and the input DCB is positioned to the beginning of that member. The member is then read using the input DCB, then the type of operation is determined.

Three modes of operation can exist: skip, copy, or dump. In skip mode the member is already located in its most condensed form in the data set, no unused space has been encountered. In copy mode at least two tracks of unused space have been encountered. The member is to be

copied from its old position in the data set to a new position so that no unused space exists. Two tracks of unused space must exist because the records are written using a write of count, key, and data. This form of the write erases the portion of the track following the data. All existing data on the track before a write count, key, and data is lost. If a block that is to be written will not fit in the space remaining on a track, the access method (BSAM) will write it at the beginning of the next track and erase the remainder of the track. Thus, before a member can be copied from one portion of the data set to another, at least two tracks of unused space must be available so that no data is lost.

If a member needs to be copied but two unused tracks do not exist, dump mode is entered to provide at least two tracks. In dump mode the member is spilled to a second private file in the active area, keeping a count of the number of records and number of members dumped. When the last member of the PDS has been copied, all files that were spilled will be copied back to the PDS.

Each member, using the input DCB, is read and is either skipped, copied, or dumped. For each member skipped or copied, the record in the utility sort file for that member is deleted. In addition, on a copy the new TTR of the member is stored in the directory. For each member dumped, the utility sort file record is replaced using the count of records dumped as the key. The actual data records are spilled sequentially to the private file that was created. If the first member spilled contained four records and the second member spilled contained three records, the record counts used as the keys would be four and seven respectively.

When all members of the partitioned data set have been read, the spilled members (if any) are copied back to the PDS, the directory entry is stored, and the utility sort file record is deleted.

When all spilled members have been restored to the non-condensed PDS, a check is made to see if any writes were done to the PDS. (If all members were deleted, it is possible that the library was out of space but contained no members.) In this case an SVC 25 (stand alone seek) is executed to ensure that the data set control block will be updated at close time (first available space was updated in the output DCB immediately after it was opened).

If any I/O errors were encountered, the DCBOFLGS (bit 0) field is set off to prevent the updating of the DSCB at close time. The output DCB is closed, the input DCB is closed, the spill private file is released, the utility sort file is released, and the old private active file is recreated.

External Routines

IHKAFI	-	to handle I/O in active area
IHKBPM	-	to handle library I/O (on bottom of queue)
IHKRNQ	-	to get ddname for library and get on library I/O queue
BPAM	-	STOW, to update directory entries
SVC 25	-	stand alone seek (establish track balance if no writes attempted)

Tables/Work Areas

18-word save area
88-byte DCB
TUB
AFIO and library I/O fields are modified for the various macros.

Exits

Normal - return to calling routine with zero in register 15
Error - return to calling routine with one of the following return codes in register 15:

- 04 - AFIO I/O error, library not modified
- 08 - AFIO I/O error, library modification started, members may be lost or invalid
- 12 - library I/O error, library not modified
- 16 - library I/O error, library modification started, members may be lost or invalid
- 20 - GETMAIN failure

Attributes

Reentrant and nonresident

SERVICE ROUTINES

SCAN ROUTINE (IHKCCS)

Entry Point

IHKCCA - register 1 must point to a three-word area containing the following:

- the starting address of the scan;
- the stopping address of the scan (if a maximum scan length is specified, the contents of this word is arbitrary);
- the address of the following parameter list, which starts on a half-word boundary:
  1. In the first half-word is the maximum scan length (if the stopping address is specified, this half-word must contain binary zero).
  2. In the second half-word is the number of characters to which the scan is sensitive this number is binary zero, the scan is for (if the first nonblank character).
  3. In the following bytes is a character string containing the characters to which the scan is sensitive. The maximum length of this string is 12.

Function

Given a character string and a starting location, this routine scans in the direction of high main storage. The scan terminates with the first character observed that belongs to the character string. Alternatively, a scan may be made for the first nonblank character. The user must specify either a maximum number of characters over which to scan or a stopping address.

External Routines

None

Tables/Work Areas

256-byte work area (TRTTAB) is used to build a translation table sensitive to the proper number of characters.

Exits

Normal - return to calling routine with one of the following codes in register 15:
- 00 - no sensitive character or nonblank character was found.
- 04 - a find was made on a request for the first nonblank character
- 4j - a find was made on the jth character of the character string.

Register 1 always contains the address of the character found. It contains 0 if no find was made.

Attributes

Serially reusable and resident

NUMERIC VERIFICATION MODULE (IHKNUM)

Entry Point

IHKNUM - register 1 must point to a variable parameter list containing the following:
1. the address of a one-byte length field;
2. the address of a one to eight-byte data field (dependent on length parameter).
3. the address of an eight-byte area; (the high-order bit of the last parameter in the list is set to one).

Function

The purpose of this routine is to verify a field of eight or fewer numerics and, optionally, to move the field. The length byte of the parameter list is checked first. If the length is less than one or greater than eight, then a code of 4 is placed in register 15 and control is returned. The characters in the data field are checked for all numerics. If any character in the field is not numeric, then control is returned with a 4 in register 15. If each character is a numeric and if the data field address is the last address in the parameter list, then control is returned with a 0 in register 15. If the third address is present in the parameter list, then the data field is moved into the eight-byte area, right justified with leading numeric zeros. Then control is returned to the calling routine.

External Routines

None

Tables/Work Areas

None

Exits

Normal - return to the calling routine with a 0 in register 15
Error - return to the calling routine with a 4 in register 15 when either length or data is invalid

Attributes

Reentrant and resident

FORTRAN AND PL/1 CONVERSATIONAL SYNTAX CHECKER INTERFACE (IHKSYN)

Entry Point

IHKSYN - register 1 must point to a one-word area containing the address of the TUB

Function

This interface module builds the parameter list for the FORTRAN and PL/1 syntax checkers, reads lines from the active file, and passes them to the checker. The TUB contains a pointer (TUBIRLSA) to the range of lines to be scanned. Both checkers return with a pointer to an error message and an indication if scanning can continue. The interface goes to IHKMSG to queue the message.

The line number range passed to the interface in input mode represents one statement. For PL/1 input mode, the lines in the range are read one at a time, and chained until the end of data or maximum continuation is reached. The group then is passed to the checker to be scanned. For FORTRAN input mode, the next line is read and kept, while the current line is passed to the checker with the return-12-expected bit set. For standard FORTRAN, the checker will always return a code of 12 when this bit is set, indicating that another line is to be passed. The current line is freed, the next line already read is set up, and the succeeding line is read. When the end of data is reached, the return-12-expected bit is not set, and the current line is passed to the checker. The group collected by the checker is then scanned. When the statement has been completely scanned in input mode and any error messages have been queued, the bits for correction mode are set, the bits indicating changes made are turned off, and control passes to the caller.

The line number range passed to the interface in delayed scan mode (indicated by a bit in the TUB--TUBDLAYN) may be more than one statement. For PL/1, the lines are read one at a time, chained to the previous line if there is one, and passed to the checker with the return-12-expected bit set. If the last line is not the end of a statement, the checker returns a code of 12, indicating which, if any, lines in the group may be freed. The interface then reads the next line, chains it, sets the return-12-response bit, and passes the lines to the checker. If the end of data or maximum continuation is reached, the interface sets the return-12-response bit and again calls the checker. For maximum continuation, the interface queues an error message.

For FORTRAN, in delayed scan mode, the interface reads a line ahead, checking for a "C" in column 1 to indicate a comment, or a nonblank and nonzero character in column 6 to indicate continuation. For a group of comments, or several within a continuation group, all but the first are freed immediately. When the next line is a comment or a continuation, the return-12-expected bit is set, and the current line already read is set up, and the succeeding line is read. When the next line is not a comment or continuation, or when the end of data is reached, the return-12-expected bit is turned off, and the current line is passed. The collected group is then scanned.

For both FORTRAN and PL/1 in delayed scan mode, if no errors have been found in a statement, the interface begins collecting the next statement. When error messages have been queued, the interface stores the line number of the next line in the active file in the location containing the first line to be scanned (or a maximum number if the end of data has been reached), and returns to the caller.

External Routines

FORTRAN Syntax Checker
PL/1 Syntax Checker
IHKMSG - (entry point:IHKMSG01) to queue maximum continuation or wrong FORTRAN level message
- (entry point;IHKMSG02) to queue messages from checkers
IHKAFI - to read lines from active file

Tables/Work Areas

CCT

CCTFRTFG CCTPLIFG	- bits set to indicate old or new input, and response to return a code of 12
CCTPL1MX	- used to count continuation for PL/1

TUB

TUBFLG1	- (TUBPL1 and TUBFOR) contain attribute of active file
TUBFLG3	- (TUBDLAYN) delayed scan mode
	- (TUBCORRN and TUBCORCN) set for corrections to be made
TUBIRAD	- address of directory entry for data set in active file
TUBIRLSA	- pointer to area containing range of line numbers to be scanned
TUBNXKEY	- line number used by AFIO
TUBPARM1-TUBPARM5	- used as parameter list for syntax checkers and for IHKMSG
TUBUFFAD	- address of buffer used for line numbers to be inserted in maximum continuation message
TUBRAFBF	- pointer to buffer for AFIO
TUBUSRID	- userid used by IHKMSG

DIR

DIRATTRL	- checked for FORTRAN level

Exits

Normal - return to calling routine with 0 in register 15
Error - return to calling routine with one of the following return codes in register 15:
- 04 - error message queued (input mode - corrections are being made; delayed scan mode - scan may be finished)
- 08 - GETMAIN failure
- 12 - active area I/O error
- 16 - message lost
- 20 - data set is wrong level of FORTRAN (message is queued)

Attributes

Serially reusable and resident

USER FILE MANAGER (IHKUTM)

Entry Point

IHKUTM - register 1 points to a two-word parameter list containing the address of the TUB and the address of a one-byte area. Th one-byte area contains one of the following hexadecimal request codes indicating which functions of the user verification file manager are desired:

X'80' - verify userid and queue for library I/O;
X'40' - verify userid and test active bit in control byte of user verification record (UVR);
X'60' - verify userid, test active bit, and set active bit on if not already on;
X'10' - turn off active bit in control byte;
X'18' - turn off active bit and set abnormal termination bit in control byte;
X'04' - set library inoperative bit in control byte.

Function

The user verification file manager is called to gain access to the global file to obtain the user verification record. An exclusive RPOINT macro with retrieval is issued using the userid, which is passed to this routine in the TUBGBLKY field, as the key. This routine makes use of the optional buffer control for this call to the IHKAFI module. If the user's verification record (UVR) cannot be found, control is returned to the calling routine with a return code of 4. If the record is found, then the userid is valid and the request code in the parameter list is tested to determine the functions to be performed. The request code is checked by testing the bits in the request code byte.

If the request code X'04' is set, the library inoperative bit is set in the control byte of the UVR. A REPLACE macro is issued to update the UVR, and control is returned to the calling routine.

If the request code X'40' is set, then the active bit in the UVR is tested. If the bit is set the user is active, and control is returned with a code of 20 in register 15. Otherwise, a test is made for the X'02' bit in the request code. If it is not set, then a check is made for the X'20' bit.

If the request code X'02' is set, then the password is moved from the UVR into the TUBPARM4 field. If the X'20' bit is not set, then control is returned with a code of zero. If the X'20' request code is set, then the active bit in the UVR is set. If the delayed message bit (UVRDMSG) and/or the message lost bit (UVRLMSGN) are set, the following bits in the TUB are set: TUBMSG and TUBLMSGN. The TUB sequence number is moved from the TUB into the UVRLNSEQ field. Console IDs are moved from the UVR into the TUBPARM3 field. A REPLACE macro is issued to replace the UVR in the global file with the updated copy of the UVR. Control is then returned with a 0 return code.

If the X'10' bit in the request code is set, the active bit in the UVR is turned off. The TUB sequence number in the UVRLNSEQ field is cleared to zeros. The delayed message bit (UVRDMSG) and the message lost bit (UVRLMSGN) in the control byte are set or turned off to correspond to the TUBMSG and TUBLMSGN fields respectively. If the X'08' bit in the request code is set, the abnormal termination bit in the control byte of the UVR is set. Console IDs are moved from the UVR into the TUBPARM3 field. The REPLACE macro is then issued to update the global file with a new copy of the UVR. Control is returned to the calling routine.

If neither X'04', X'40', X'60', X'42', X'10', nor X'18' is detected, then X'80 is assumed to be the request code. If the library inoperative bit is set, control is returned to the calling routine with a return code of 24 in register 15. Otherwise, the ddname in the UVR is checked for blanks. If tbe ddname is all blanks, control is returned with a code of 16 in register 15. Otherwise, the key in the TUBGBLKY field is moved into the TUBUSRNM field and the IHKRNQ module is called to queue the line for library I/O. Upon return from the IHKRNQ routine, the ddname is moved from the UVR into the BPPARMFS field of the KONBOX. Control is then returned to the calling routine.

In all cases, however, an ENDUP macro is issued to ensure completion of updates and to allow another routine to gain access to the global file.

External Routines

IHKRNQ - to queue for library I/O
IHKAFI - to manipulate global files

Tables/Work Areas

AVT
18-word save area
1-byte area with request code
KONBOX
- bpparmfs - ddname moved from UVR

TUB
- TUBGBLKY - contains key for AFIO macros
- TUBUSRNM - set equal to TUBGBLKY if requested
- TUBPRMLS - contains password on return from request code of X'60'
- TUBLNSEQ - contains TUB sequence number
- TUBLMSGN - tested or set or turned off
- TUBMSG - tested or set
- TUBRAFBF - buffer address for IHKAFI
- TUBUFFAD - address of user buffer
- TUBPARM3 - used to return console IDs to IHKLGF and IHKLGN
- TUBPARM1 - parameter list
- TUBPARM2 - parameter list
- TUBAFISW - buffer control switch
- TUBBPQEL - library I/O queue element

User Verification Record (UVR)
- UVRACTVN - active bit, set or turned off
- UVRABTM1 - abnormal termination, set
- UVRDDNAM - contains ddname
- UVRDMSG - tested or set
- UVRLMSGN - tested or set
- UVRLMSGN - tested or set
- UVRLNSEQ - TUB sequence number
- UVRCID1 - contains console IDs
- UVRPASSW - contains password

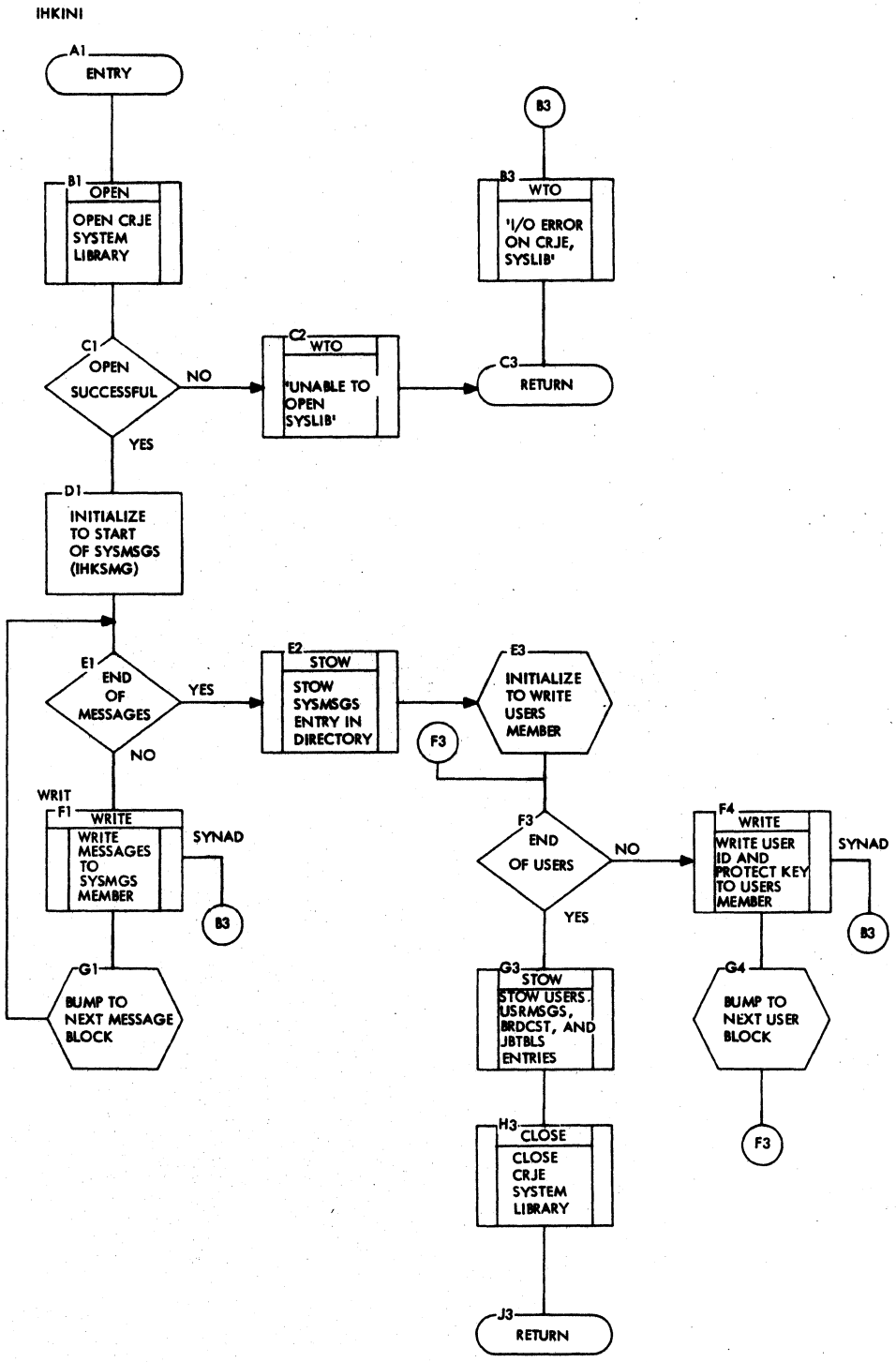
Exits

Normal - return to calling routine with 0 in register 15
Error - return to calling routine with one of the following return codes in register 15:
- 04 - userid not found
- 08 - GETMAIN failure
- 12 - active area I/O error
- 16 - ddname blank
- 20 - userid active
- 24 - library inoperative

Attributes

Reentrant and resident

• Chart AA. System Library Initialization Utility (IHKINI)



• Chart AF. START Command Processor (IHKBGN)

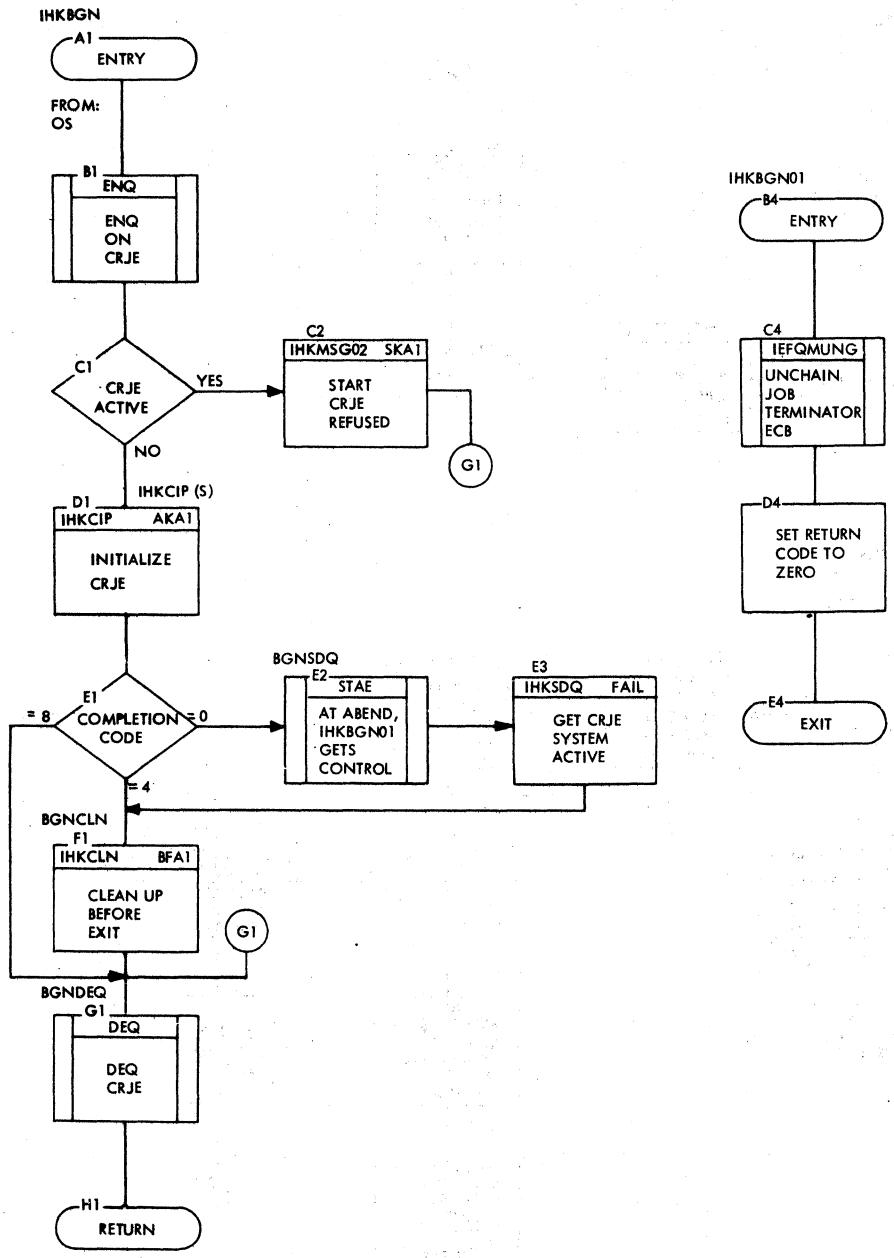
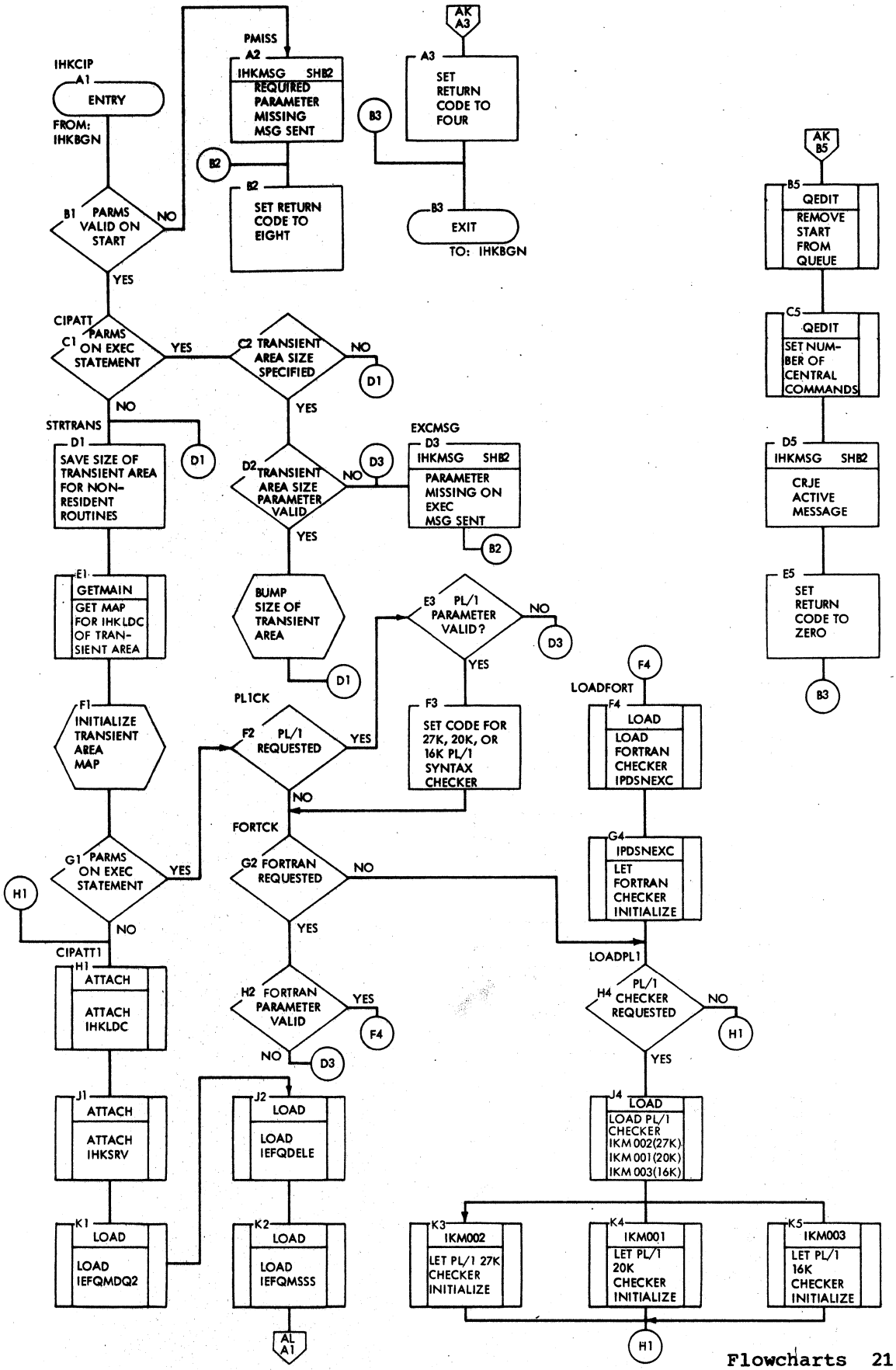
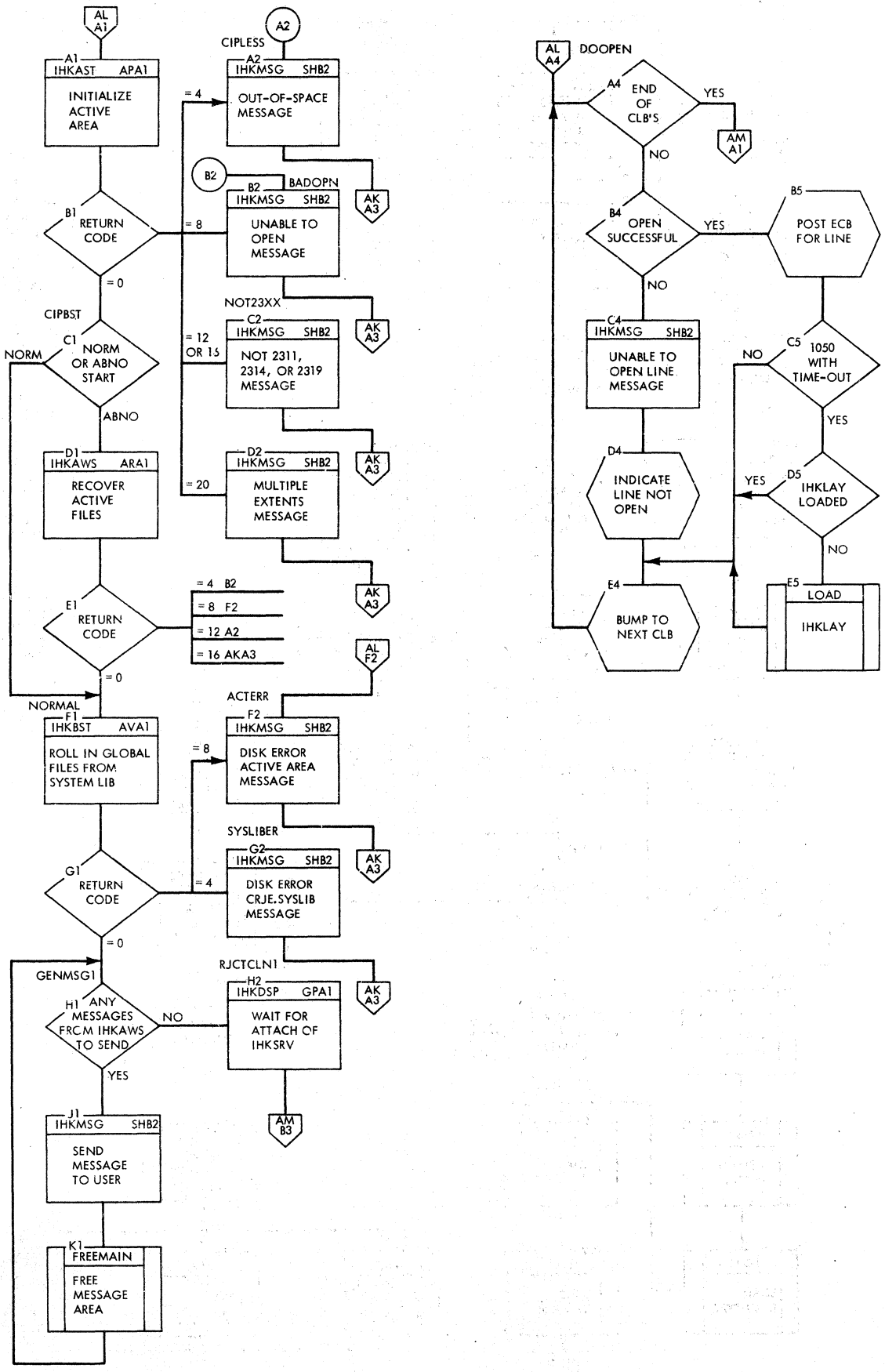


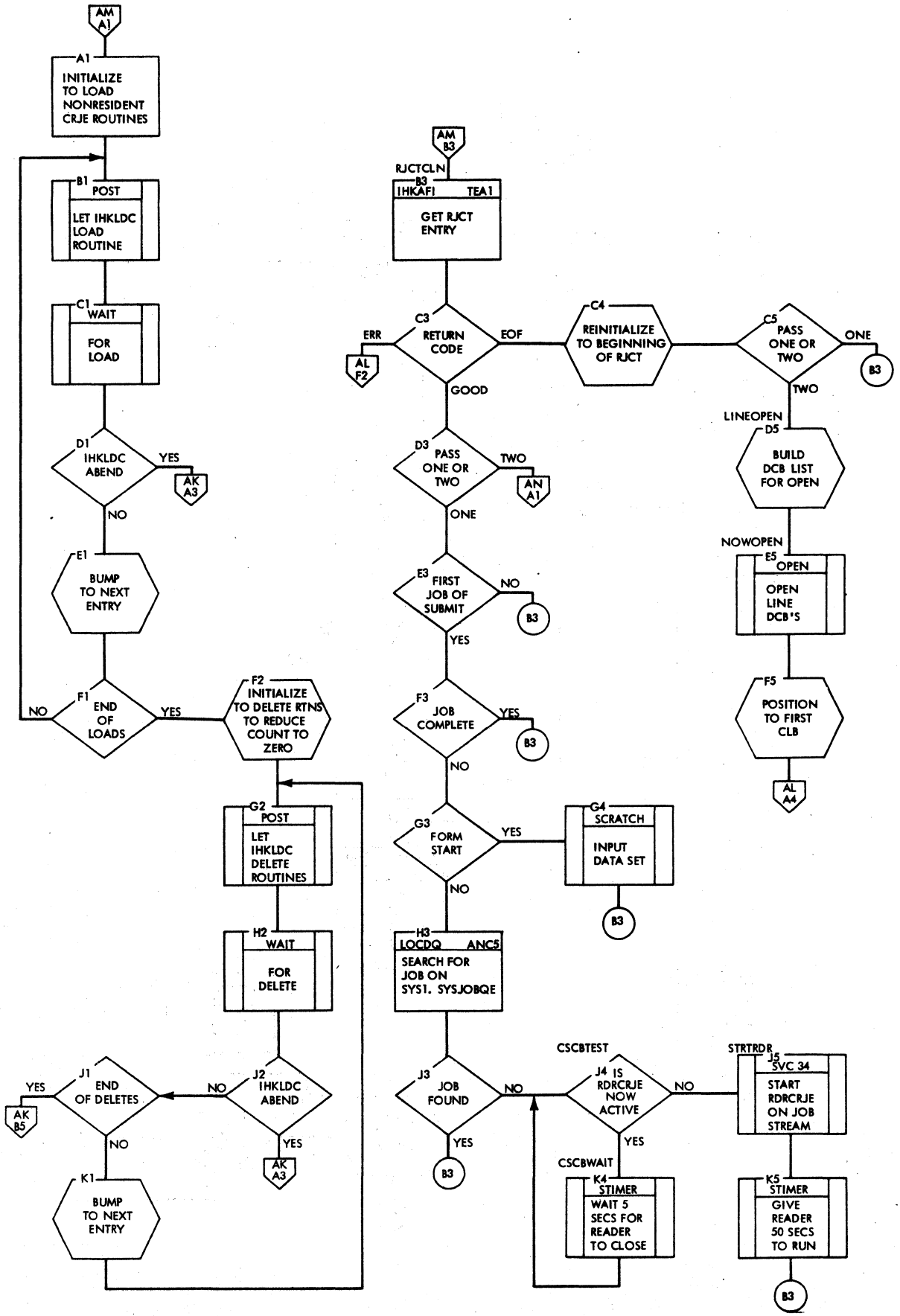
Chart AK. CRJE Initialization Routine (IHKCIP)



• Chart AL. CRJE Initialization Routine (IHKCIP)



• Chart AM. CRJE Initialization Routine (IHKCIP)



• Chart AN. CRJE Initialization Routine (IHKCIP)

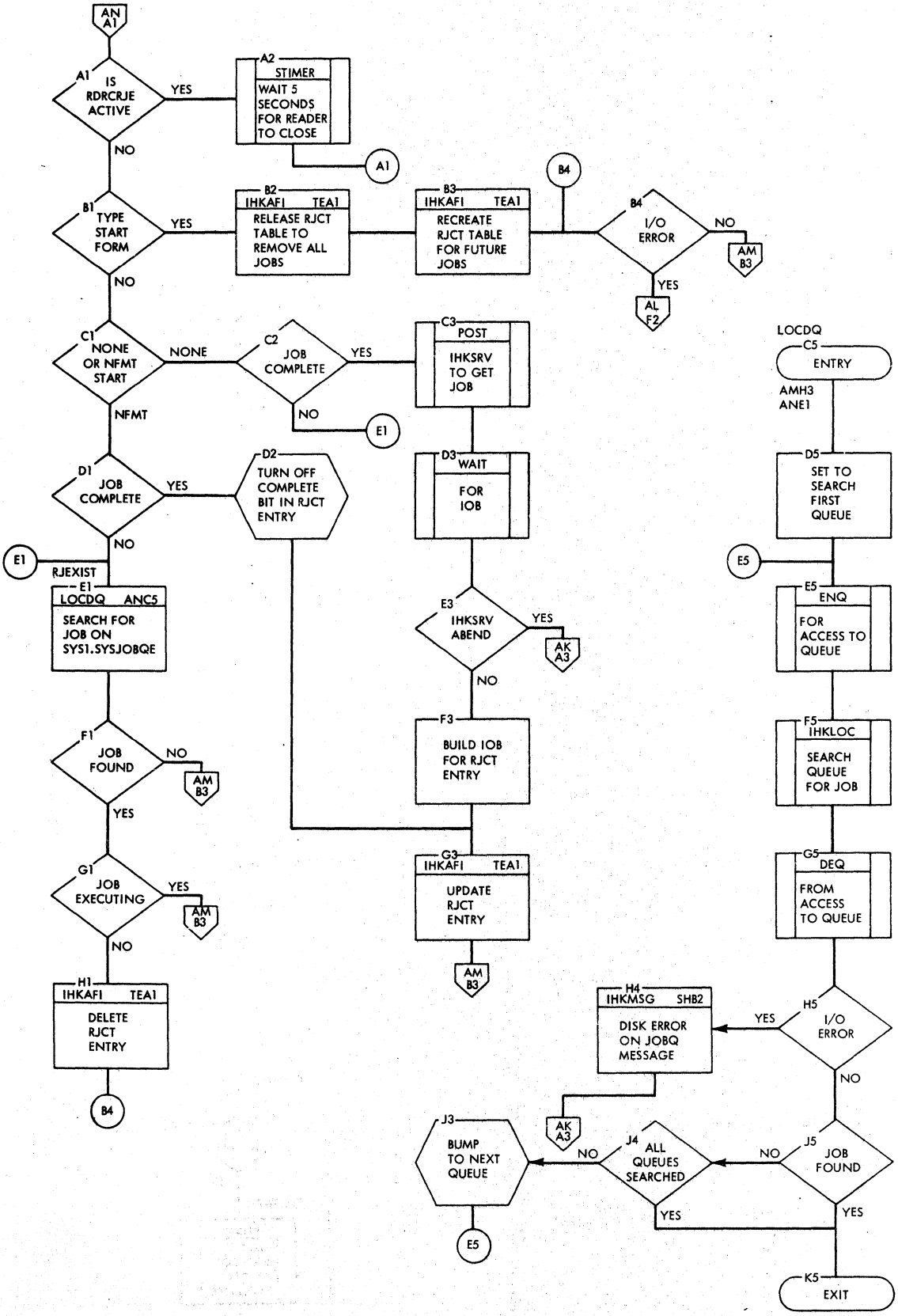


Chart AP. Active Area Start-up/Initialization Module (IHKAST)

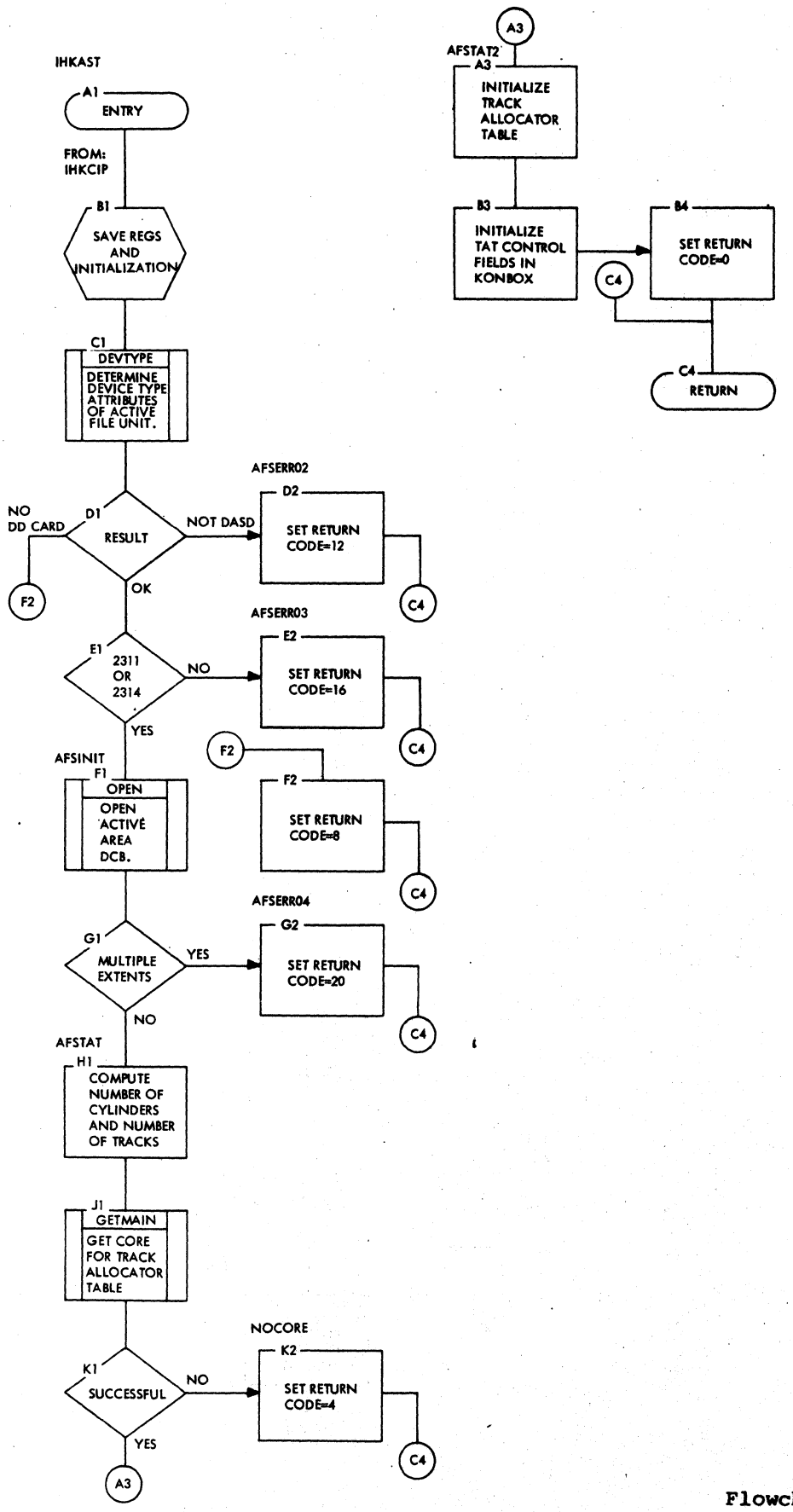
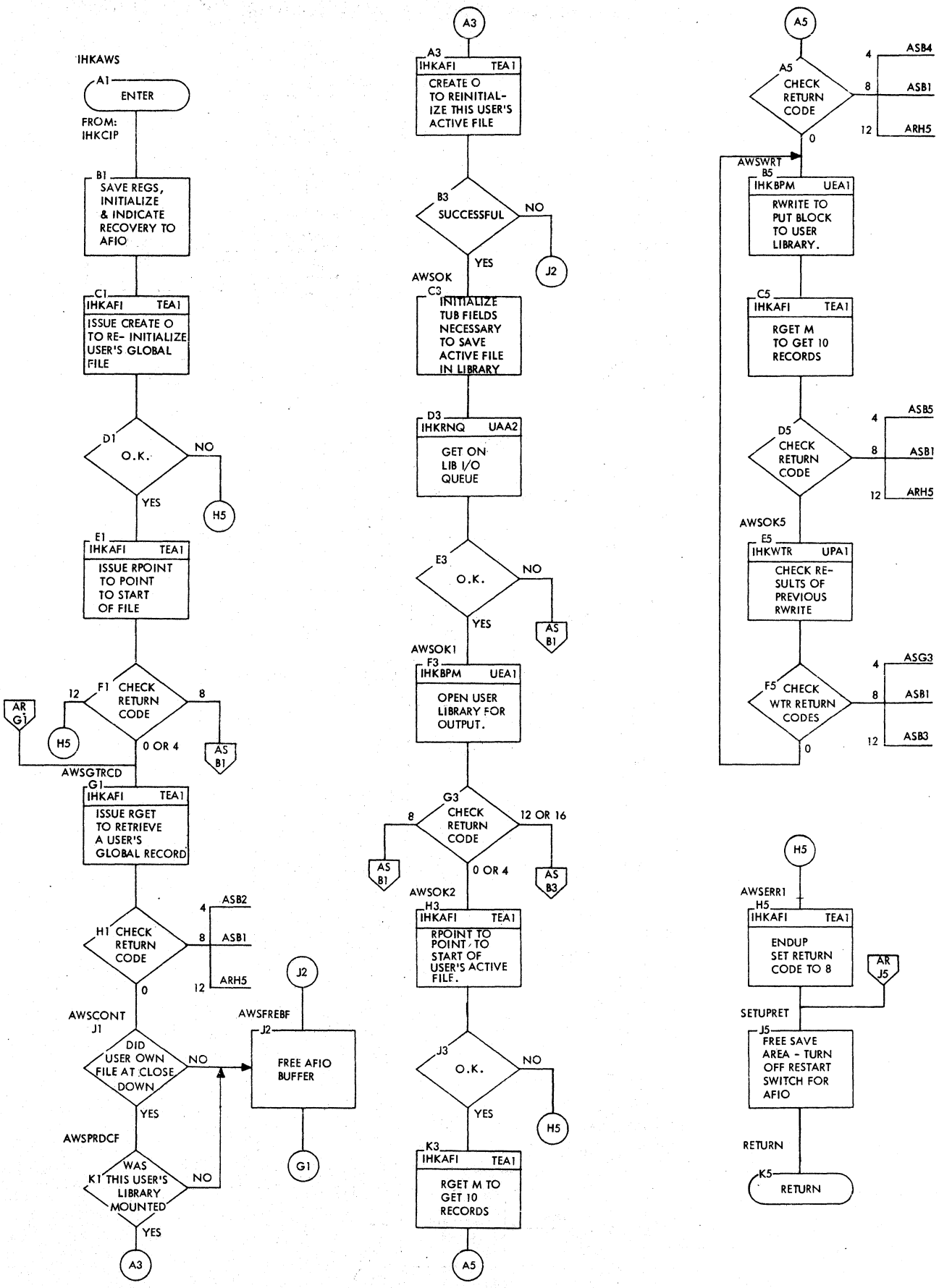


Chart AR. Active Area Recovery Module (IHKAWS)



• Chart AS. Active Area Recovery Module (IHKAWS)

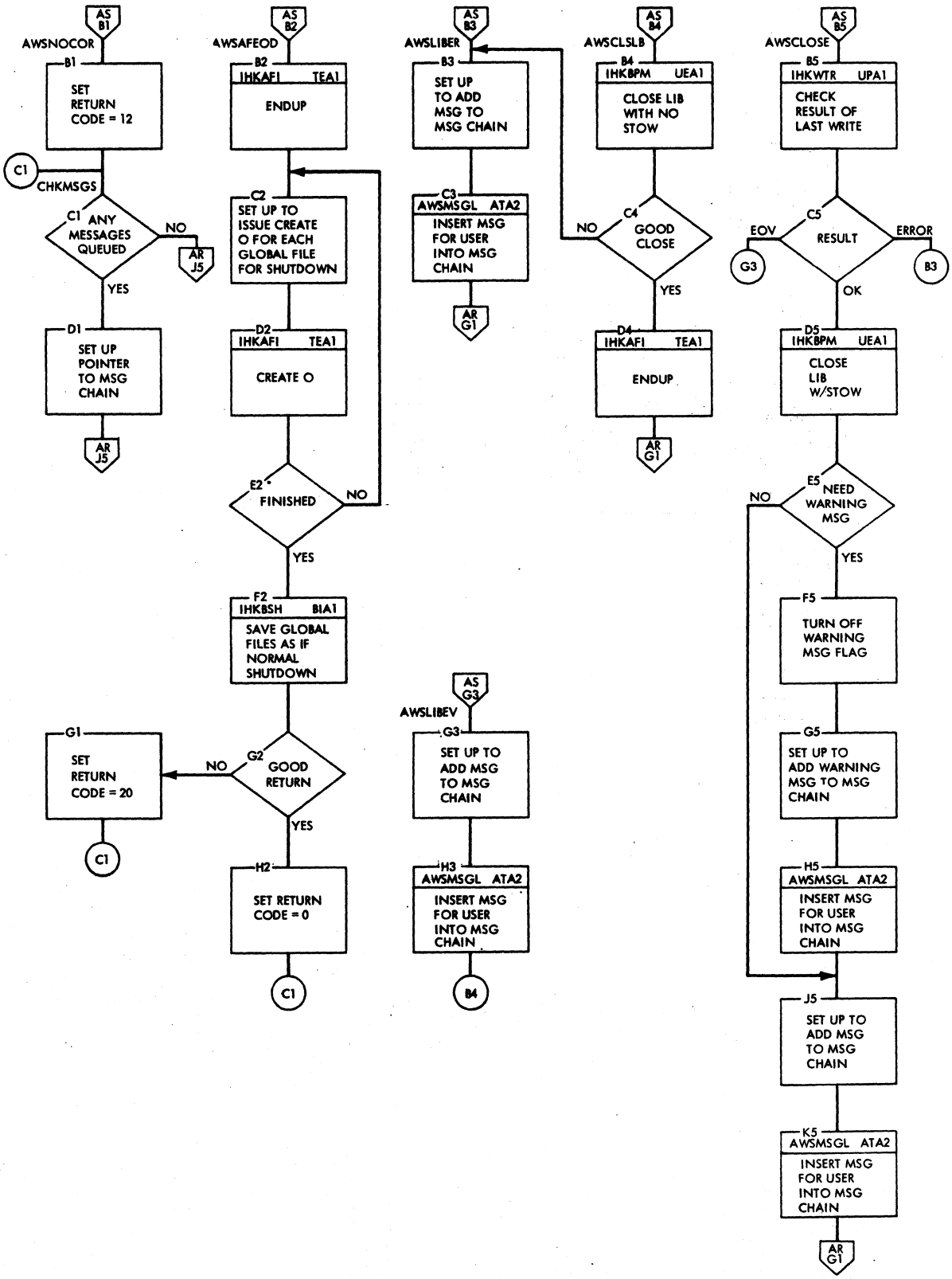


Chart AT. Active Area Recovery Module (IHKAWS)

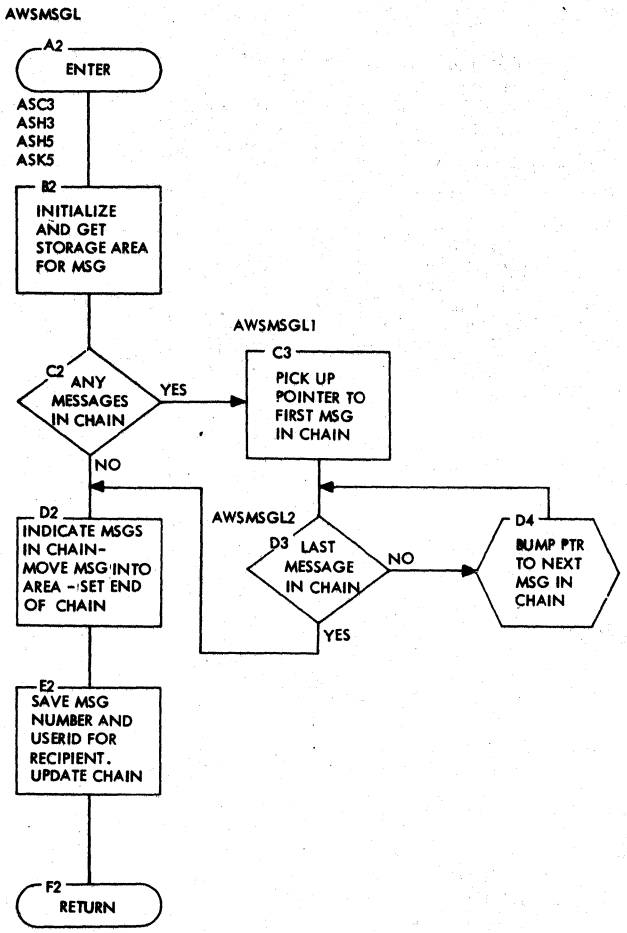


Chart AV. Library I/O Start-up Module (IHKBST)

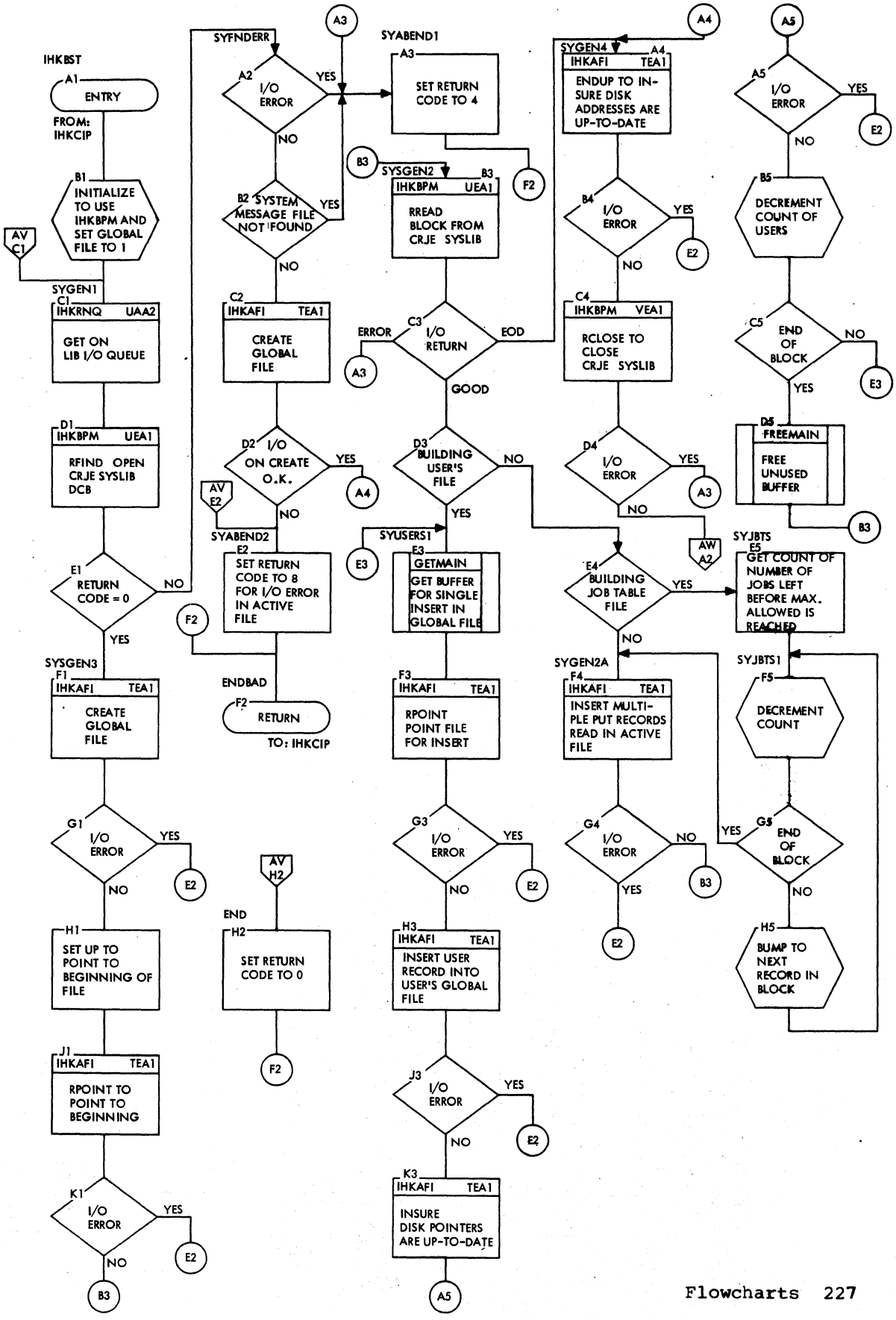
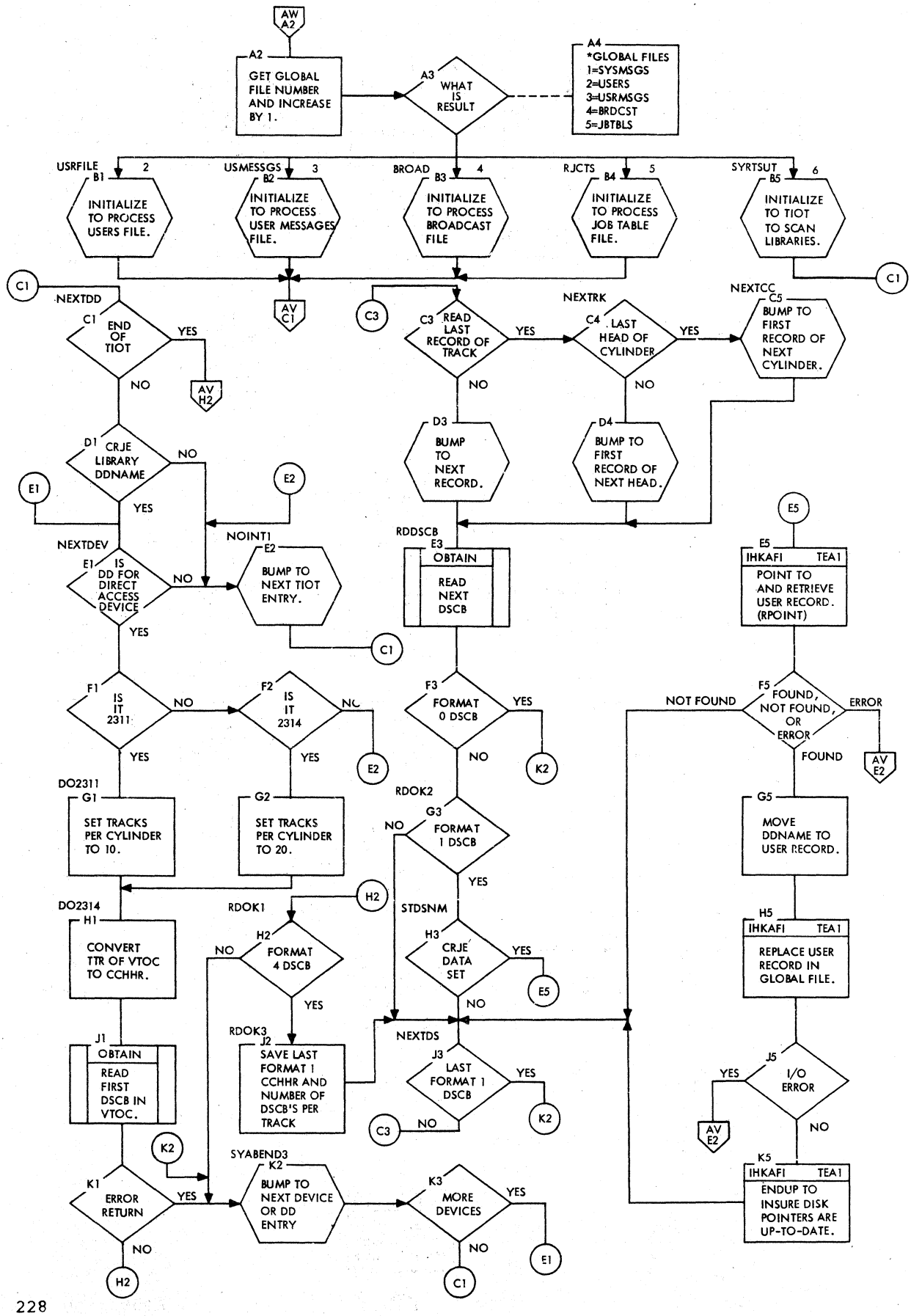
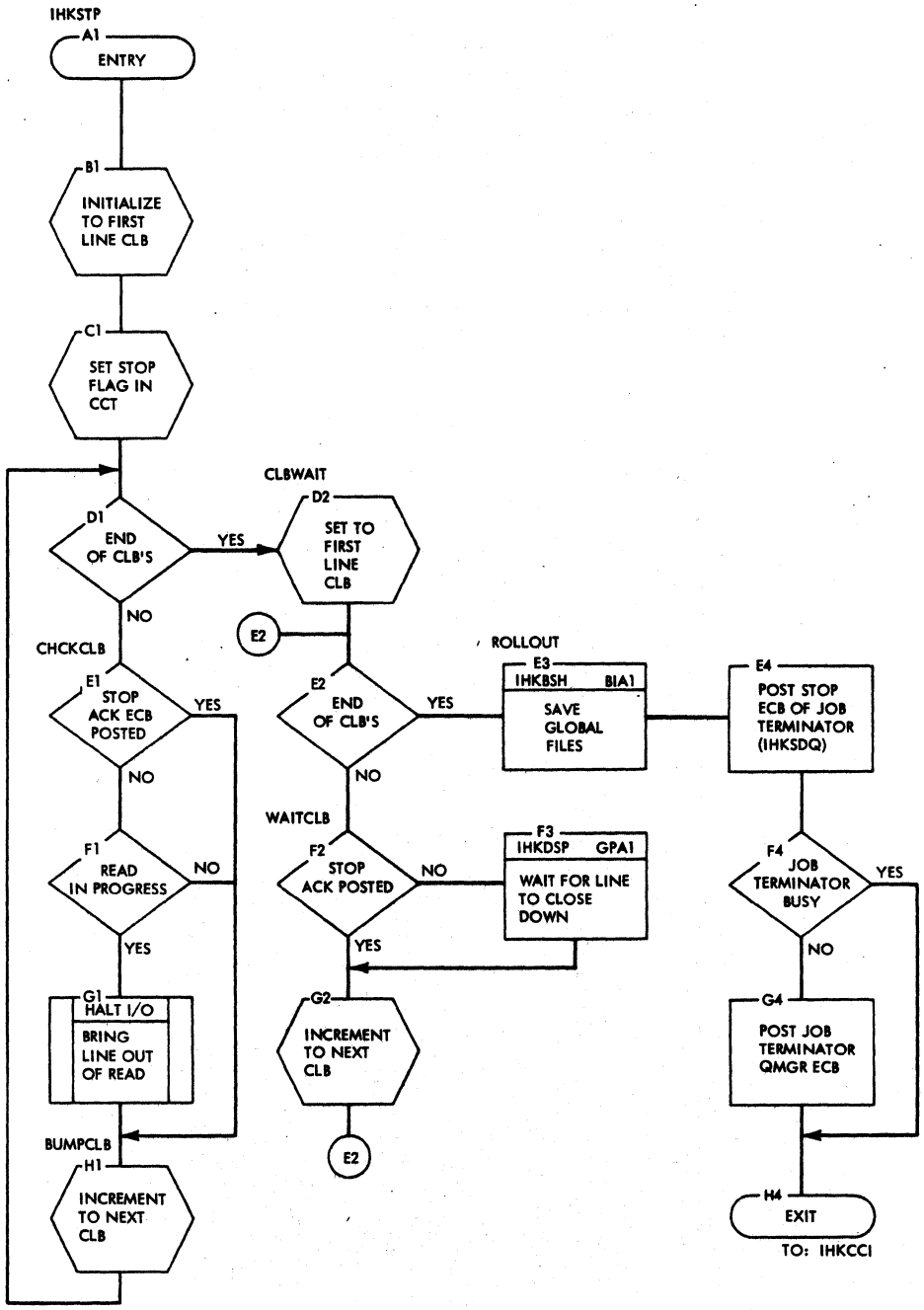


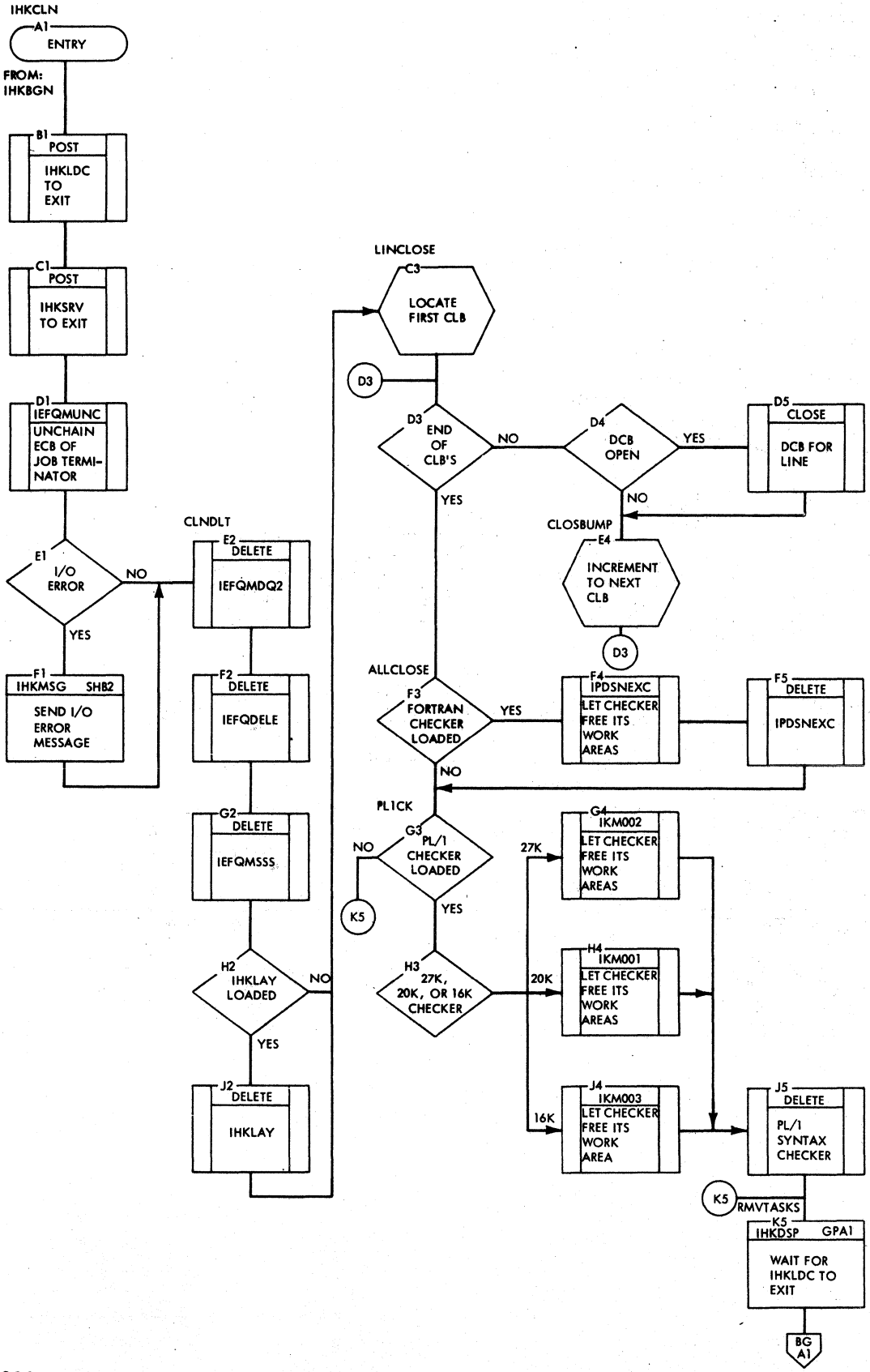
Chart AW. Library I/O Start-up Module (IHKBST)



• Chart BA. CRJE STOP Module (IHKSTP)



• Chart BF. CRJE Closedown Module (IHKCLN)



• Chart BG. CRJE Closedown Module (IHKCLN)

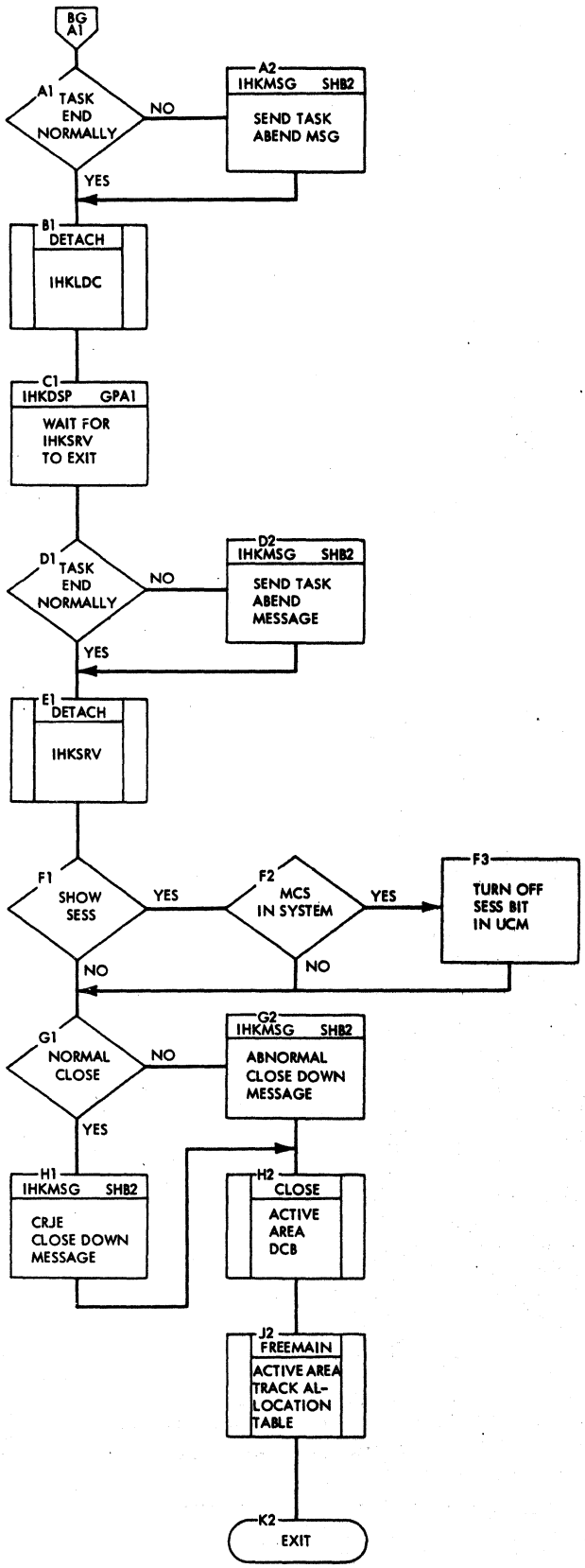


Chart BI. Library I/O Shutdown Module (IHKBSH)

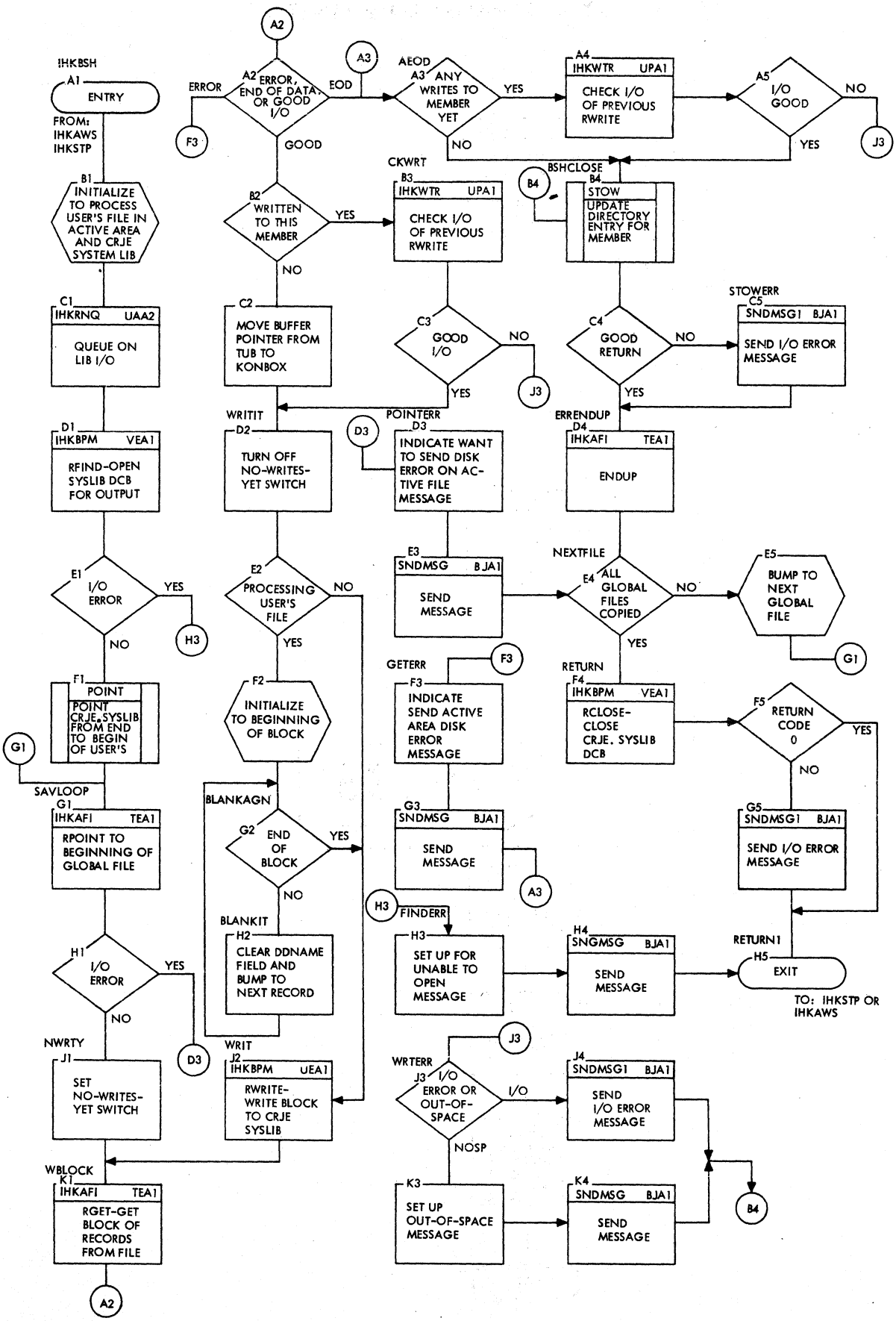
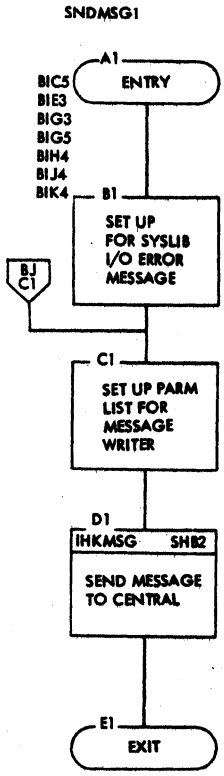


Chart BJ. Library I/O Shutdown Module (IHKBSH)



• Chart CA. START RDR, Allocate, Q Manager Service Task (IHKSRV)

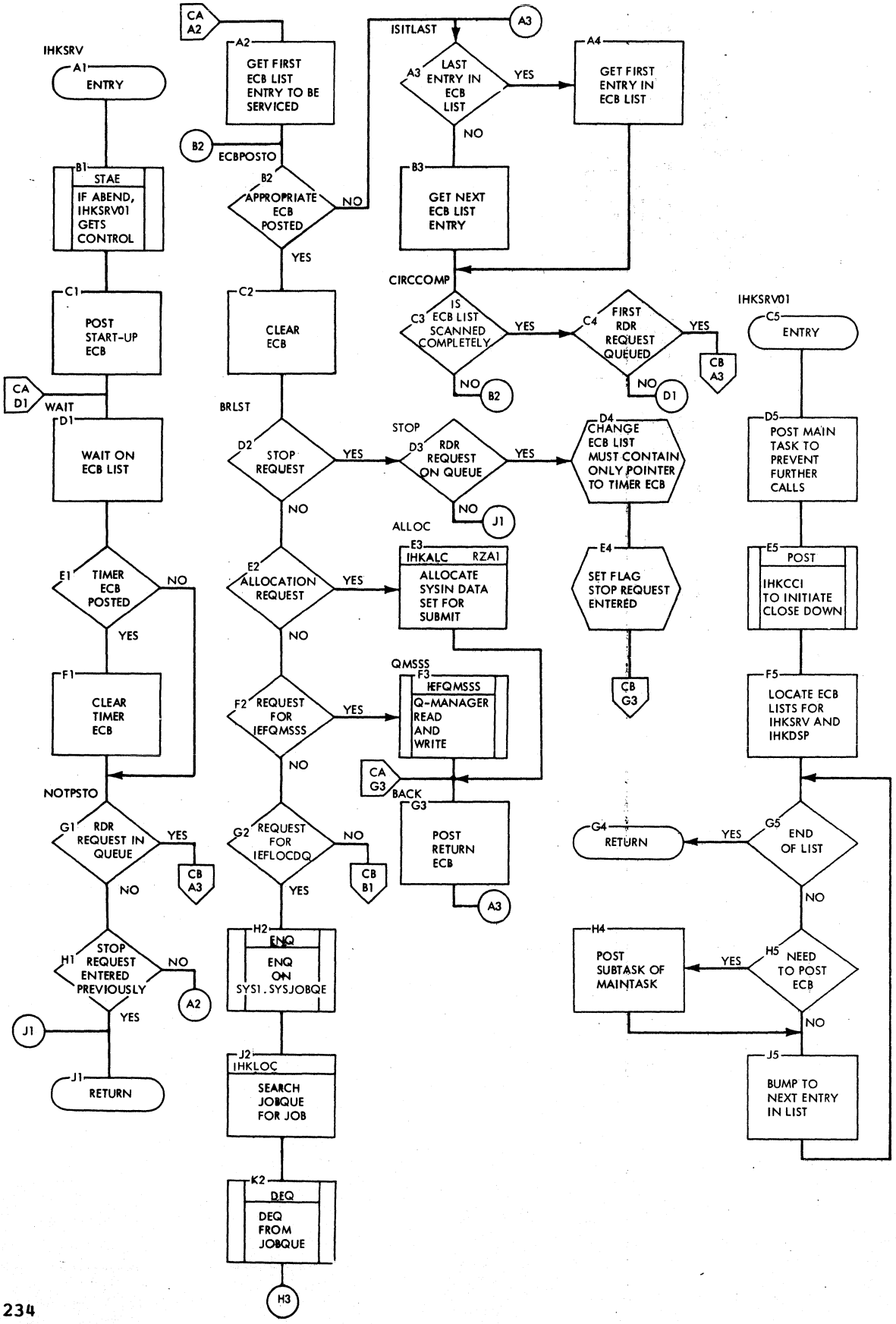
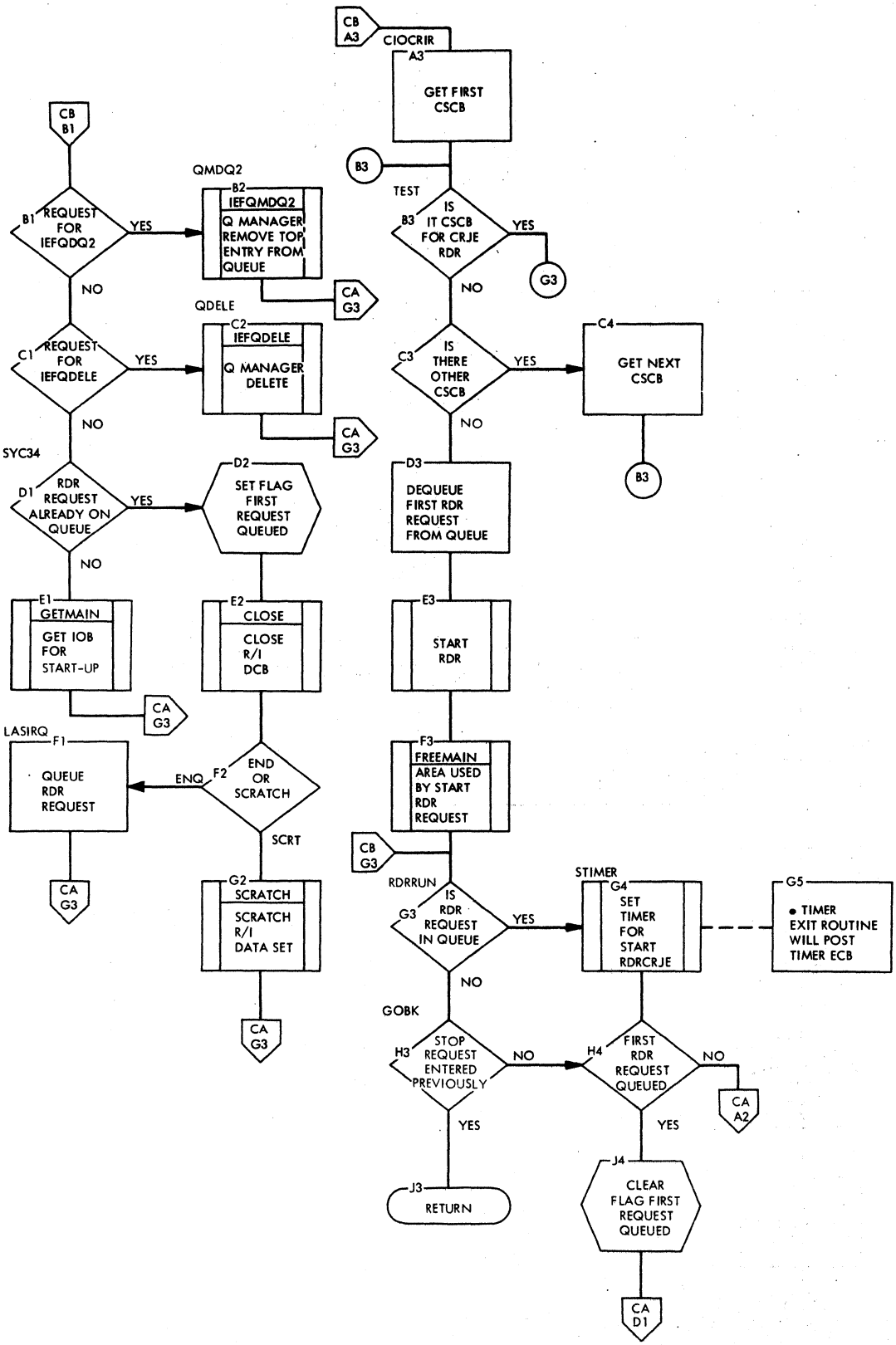
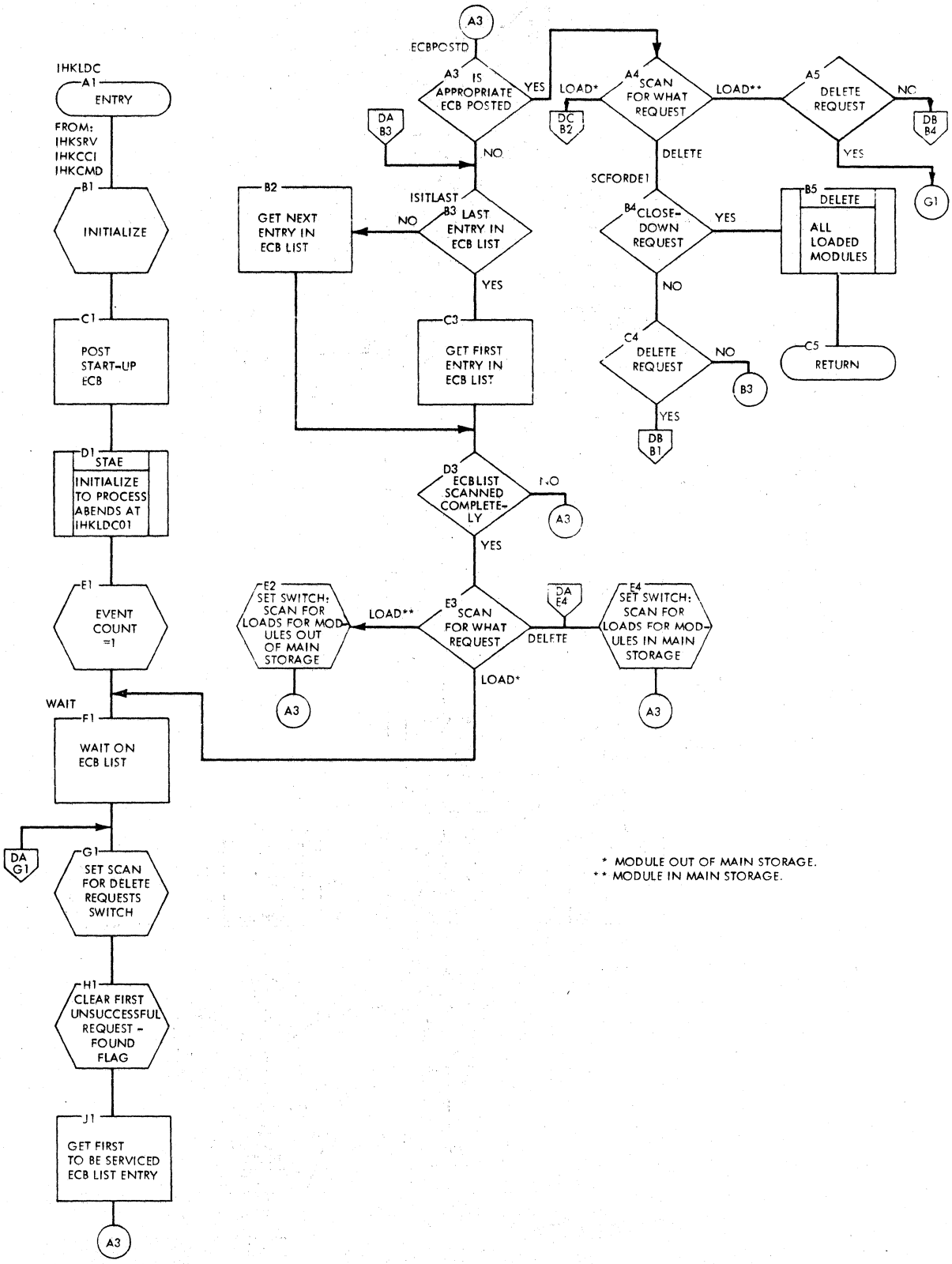


Chart CB. START RDR, Allocate, Q Manager Service Task (IHKSRV)



- Chart DA. Loader/Controller Module (IHKLDC)



* MODULE OUT OF MAIN STORAGE.
** MODULE IN MAIN STORAGE.

Chart DB. Loader/Controller Module (IHKLDC)

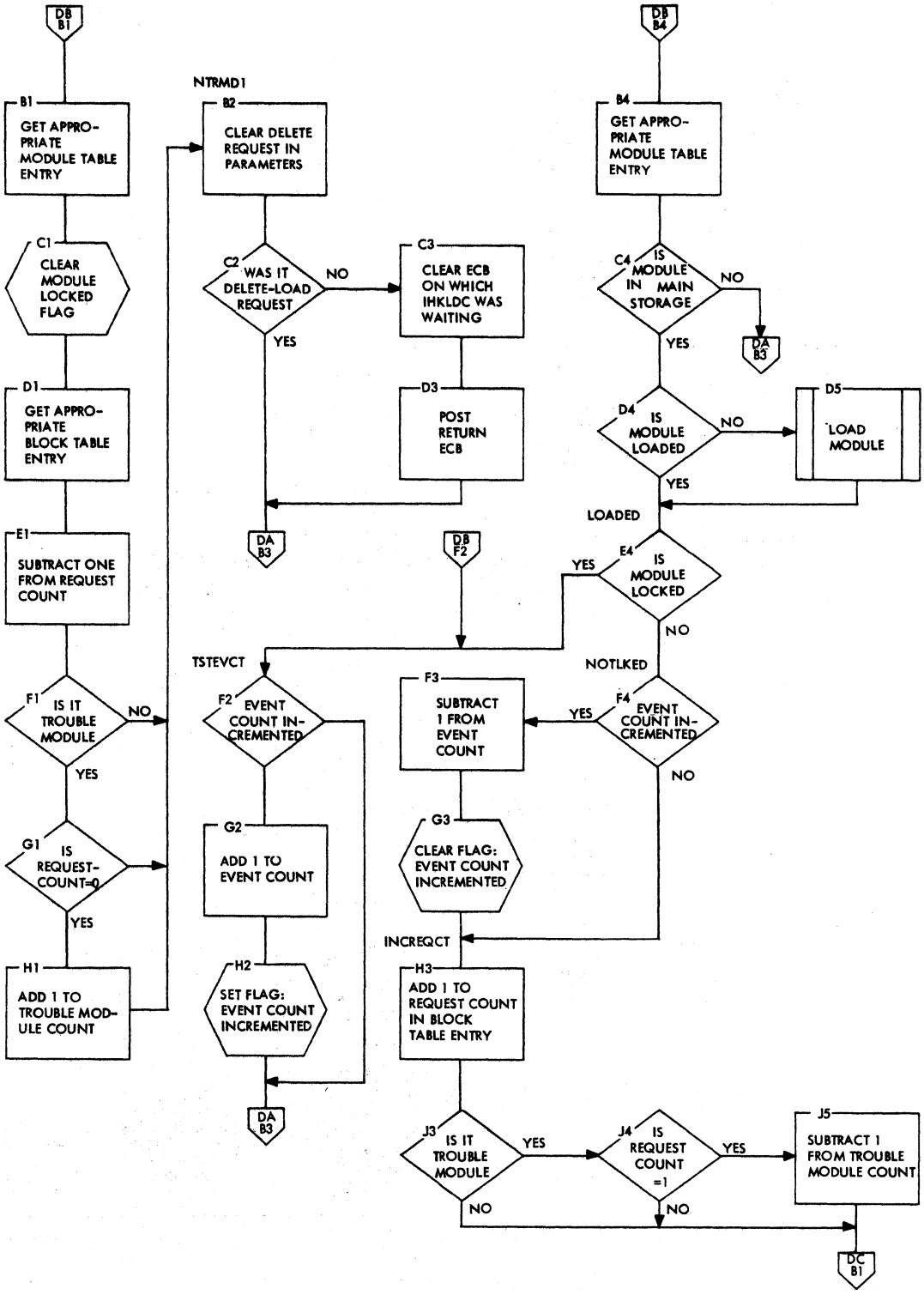
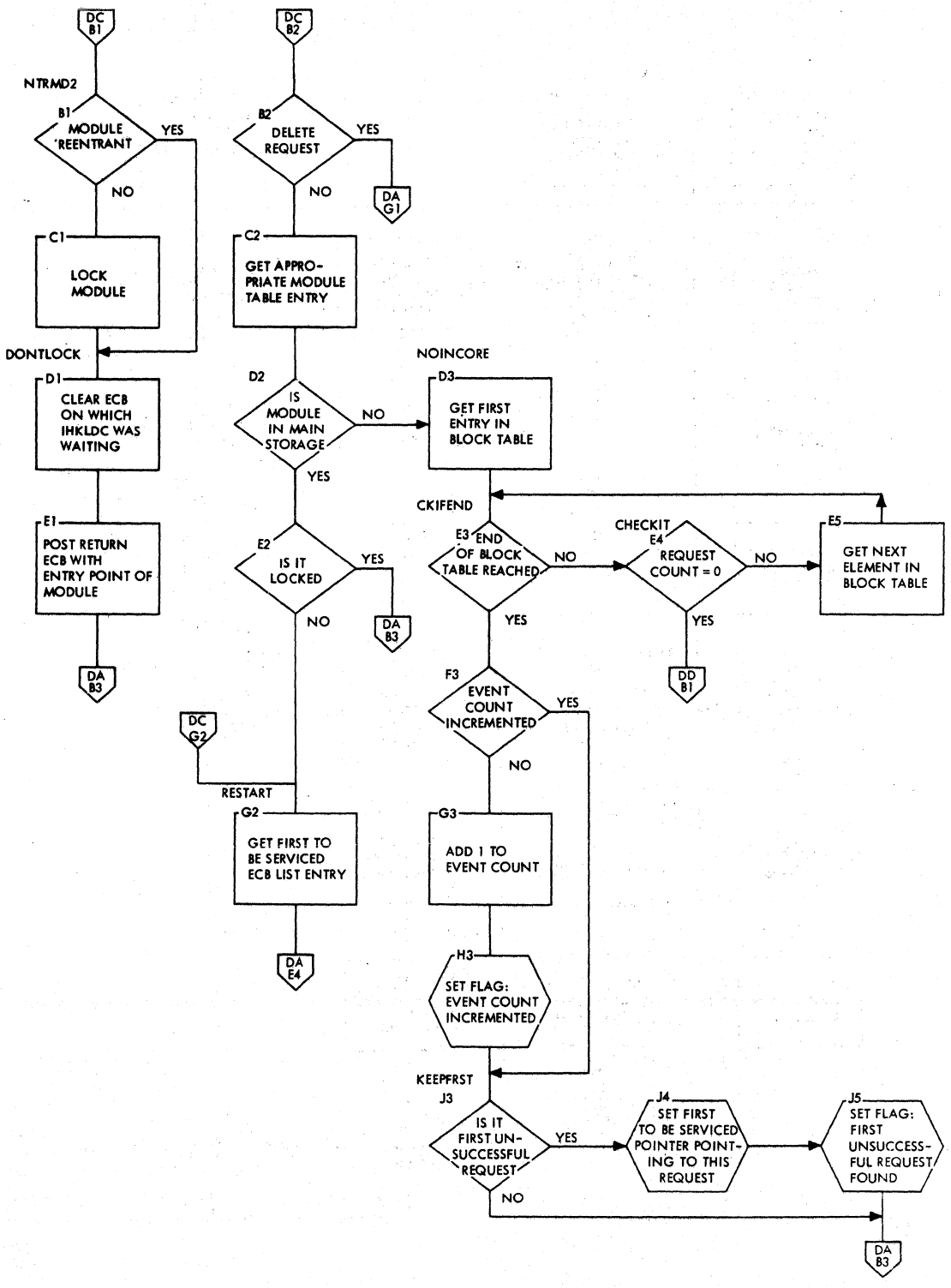


Chart DC. Loader/Controller Module (IHKLDC)



• Chart DD. Loader/Controller Module (IHKLDC)

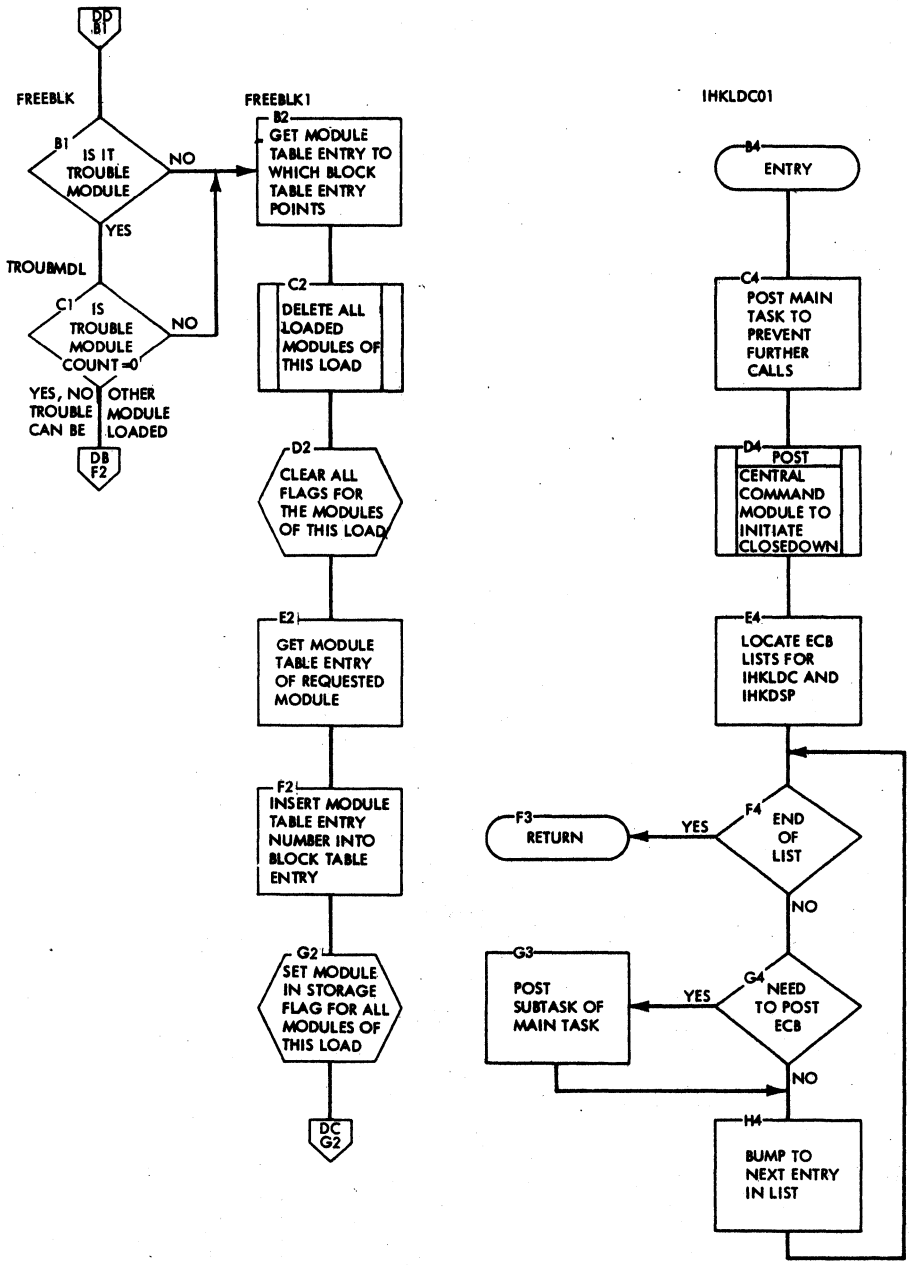


Chart DH. OS Date Set Open Module (IHKOPN)

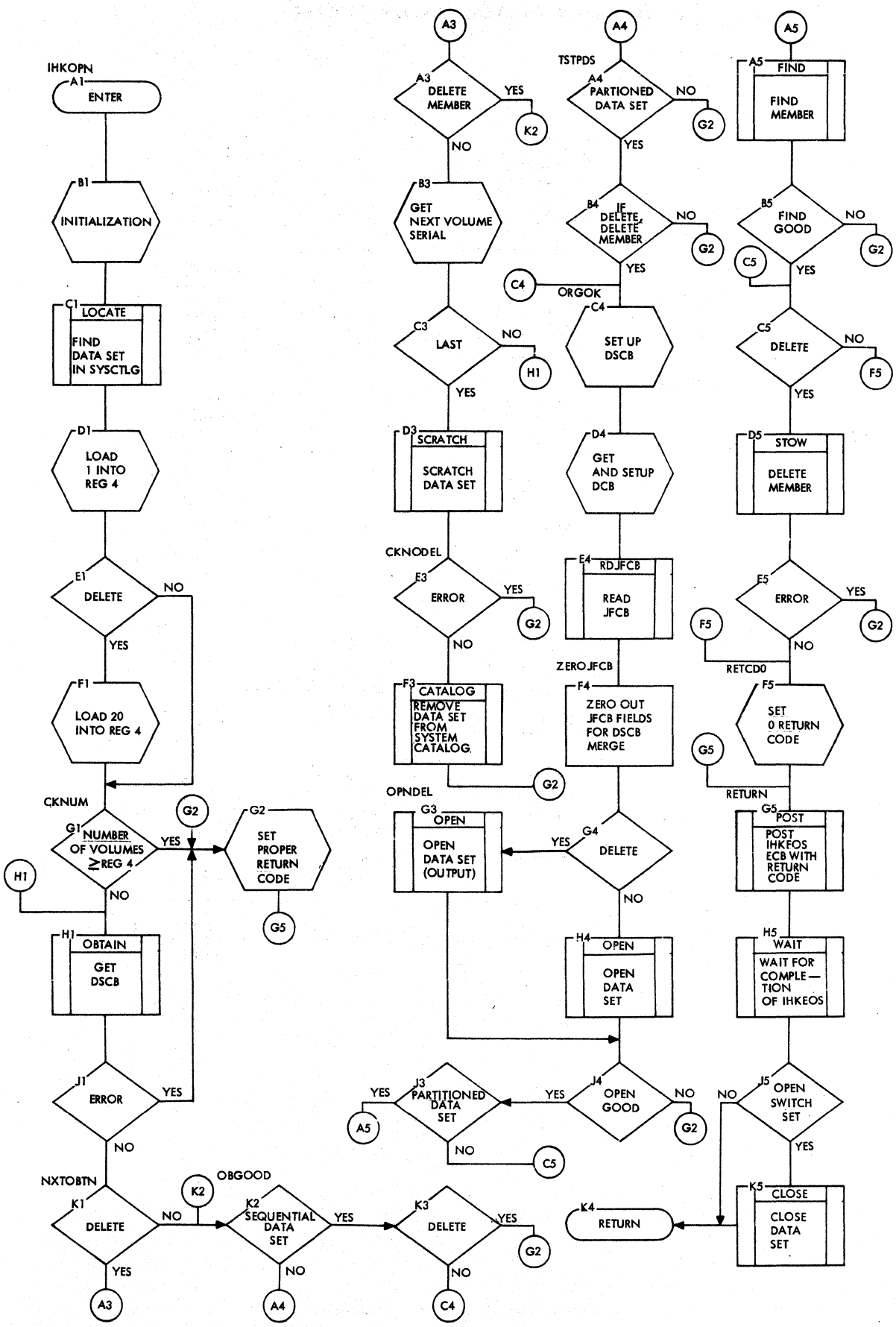
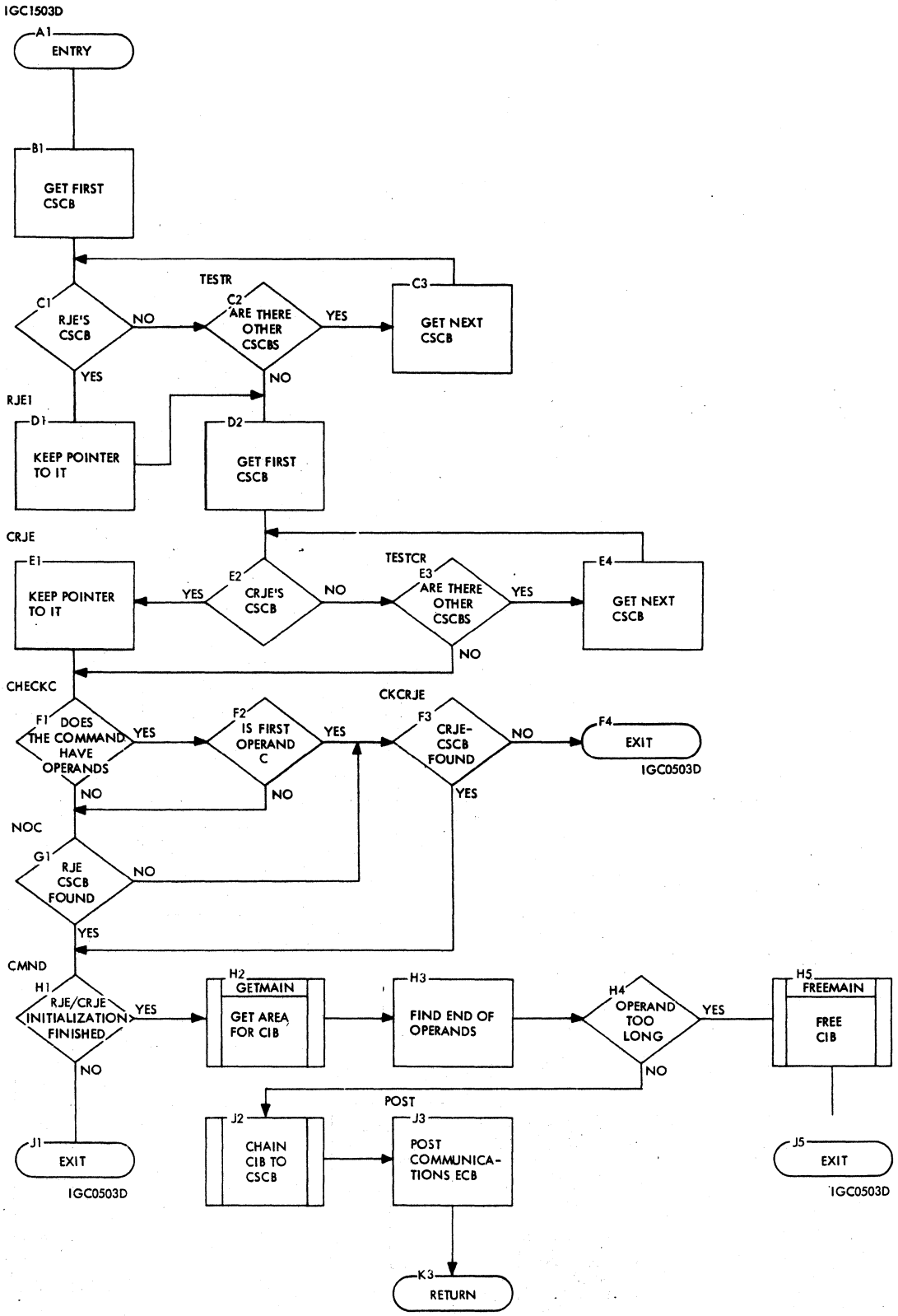


Chart EA. RJE/CRJE Central Command Scheduling Routine (IGC1503D)



• Chart EE. Central Command Interface Module (IHKCCI)

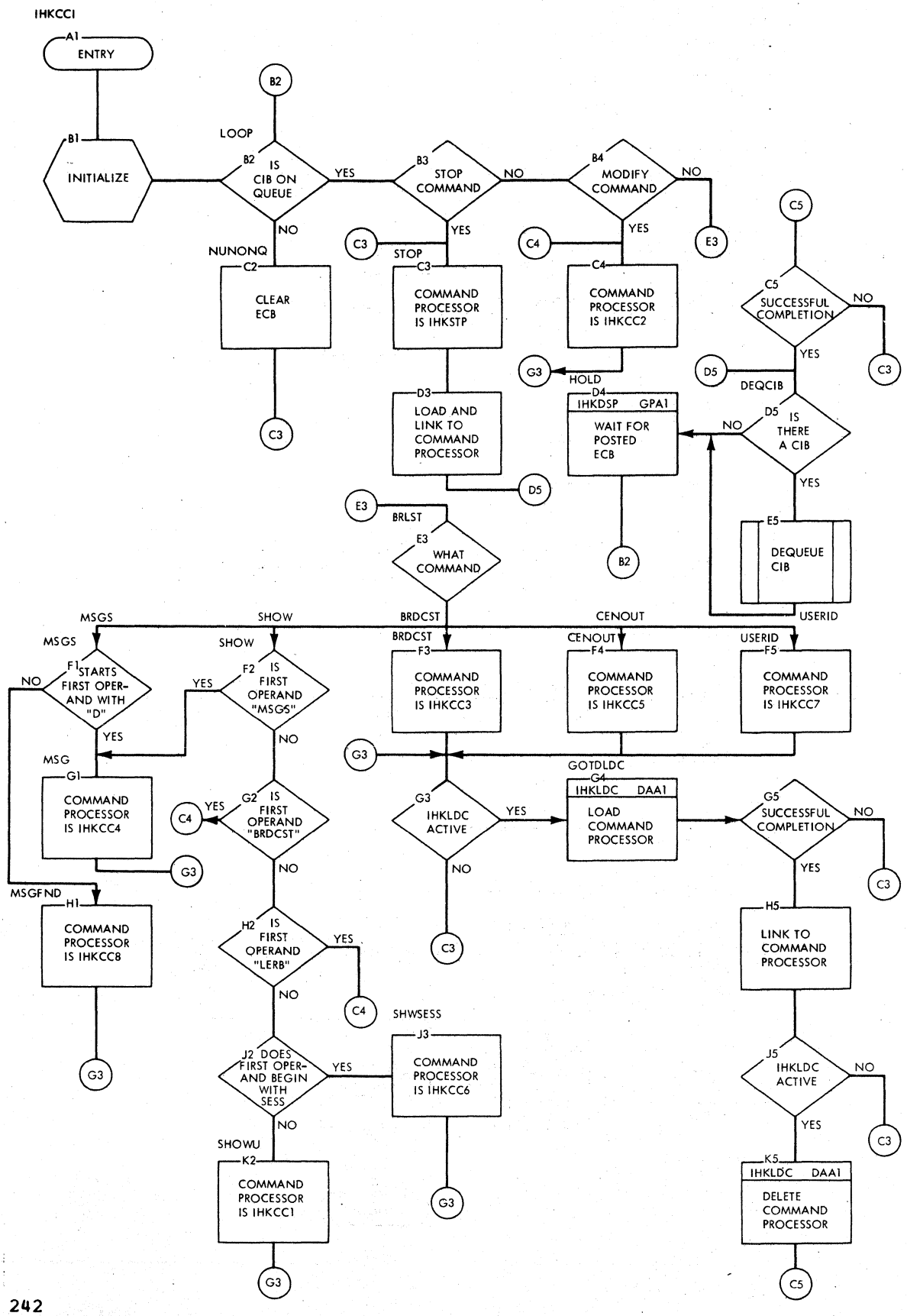


Chart EG. SHOW USERS and SHOW JOBS Central Command Processor (IHKCC1)

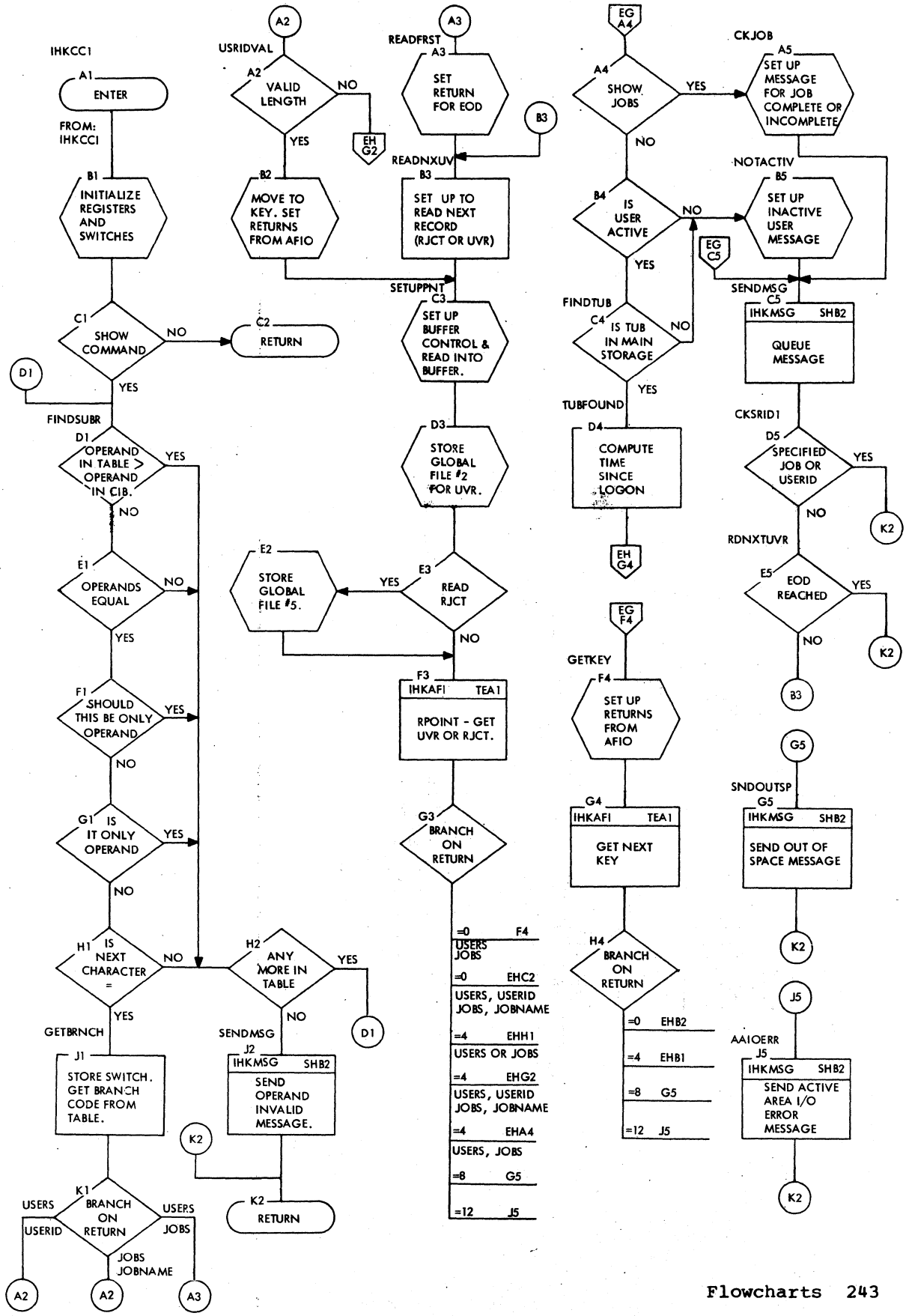


Chart EH. SHOW USERS and SHOW JOBS Central Command Processor (IHKCC1)

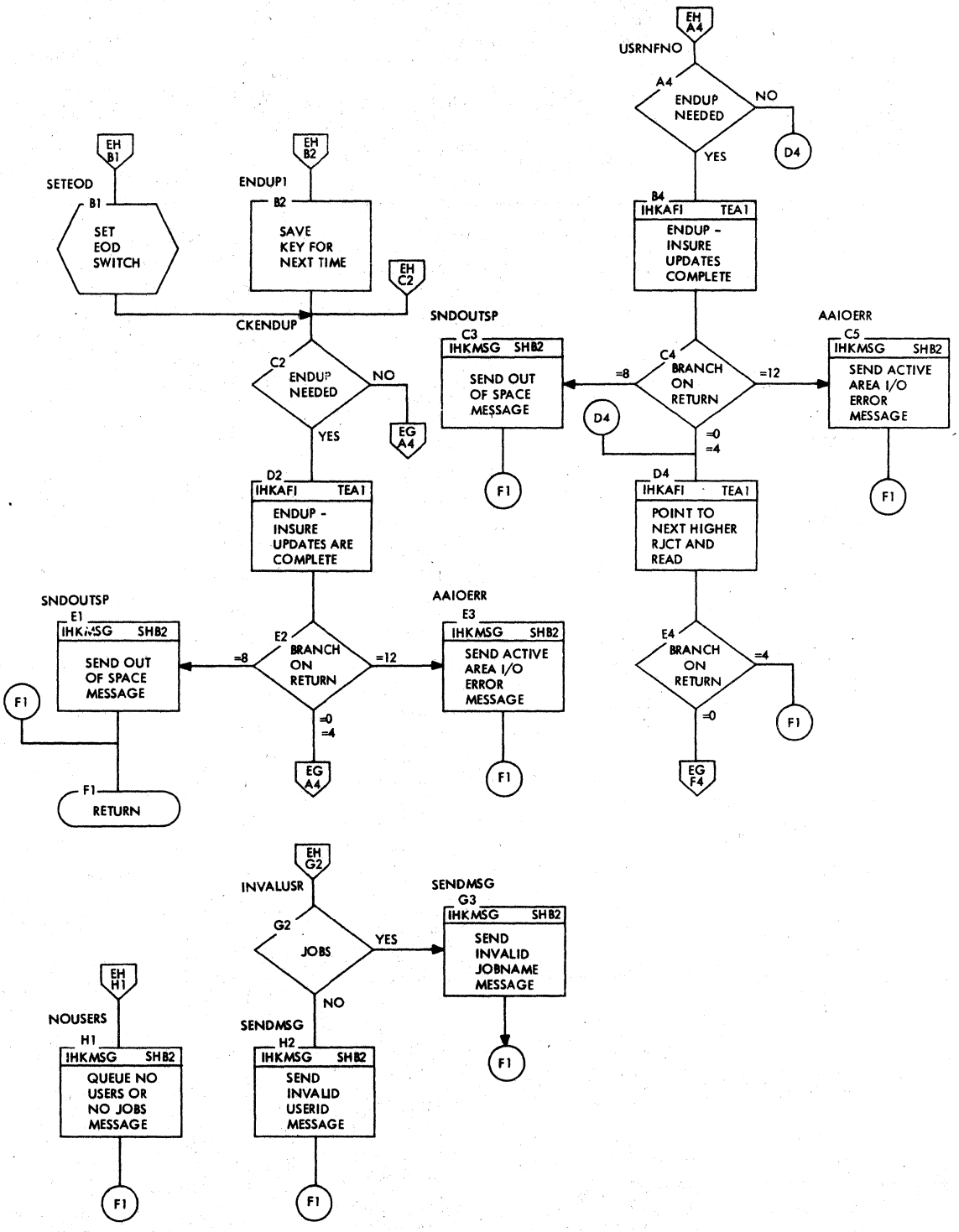


Chart EJ. SHOW LERB, SHOW BRDCST, and MODIFY Central Command Processor (IHKCC2)

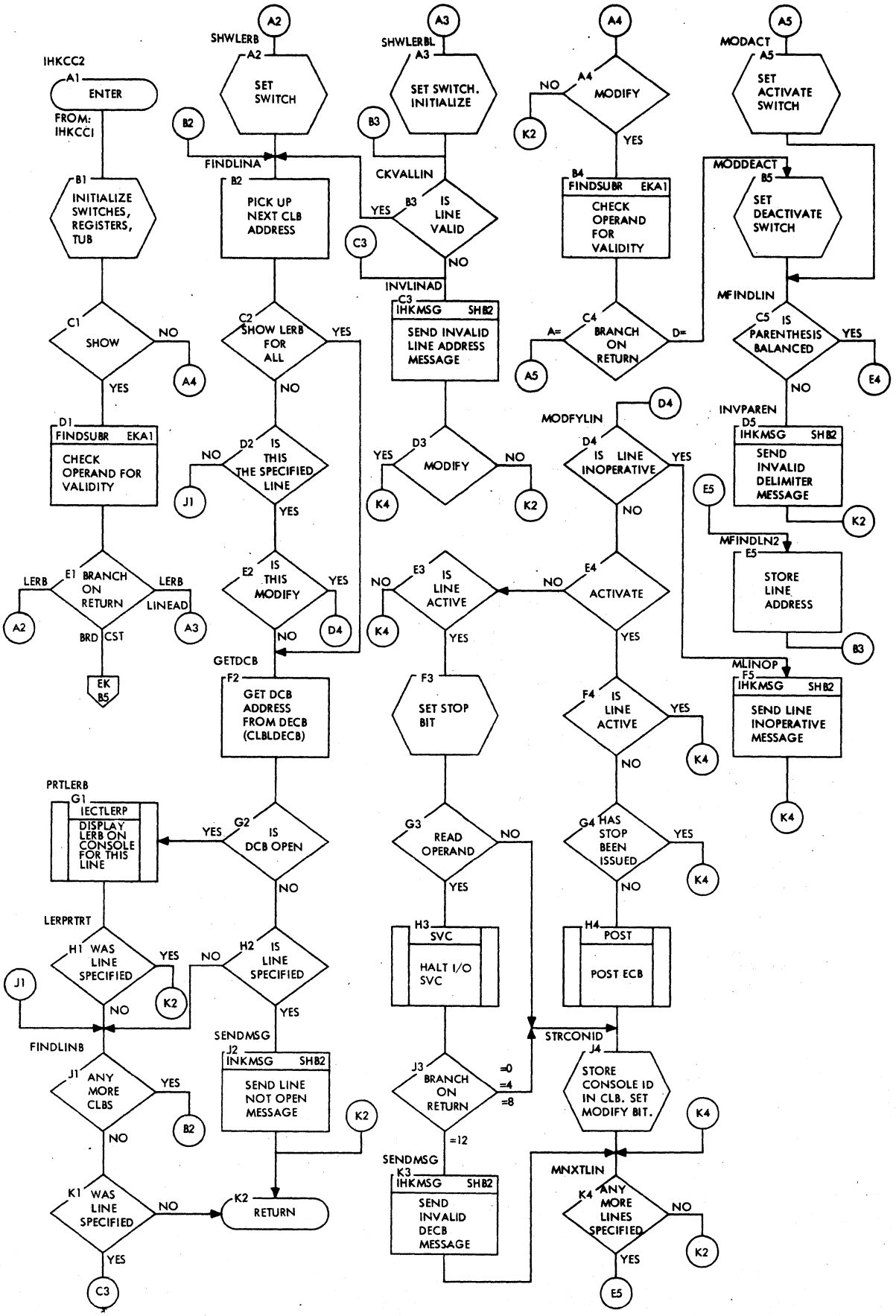


Chart EK. SHOW LERB, SHOW BRDCST, and MODIFY Central Command Processor (IHKCC2)

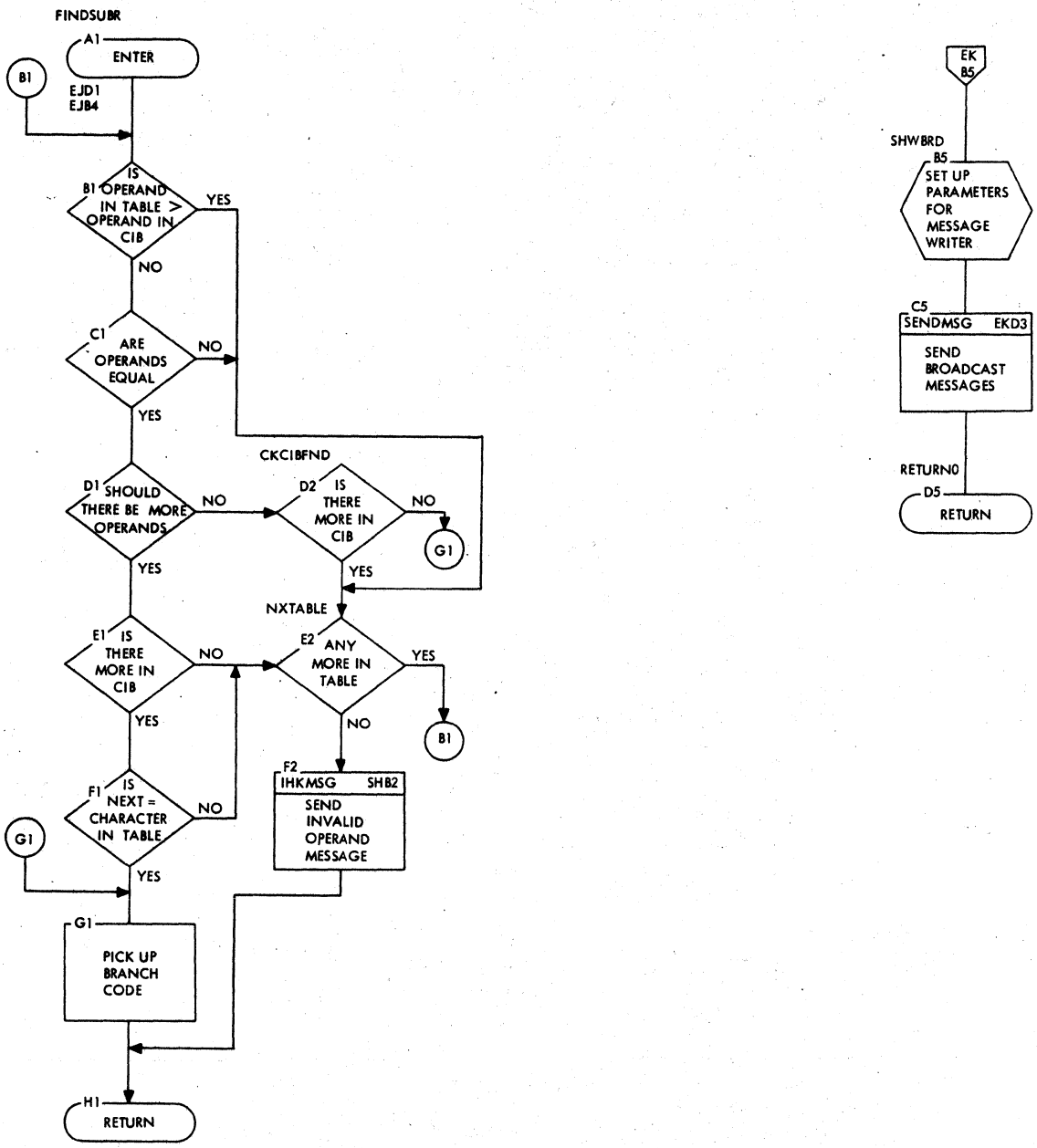


Chart EN. BRDCST Central Command Processor (IHKCC3)

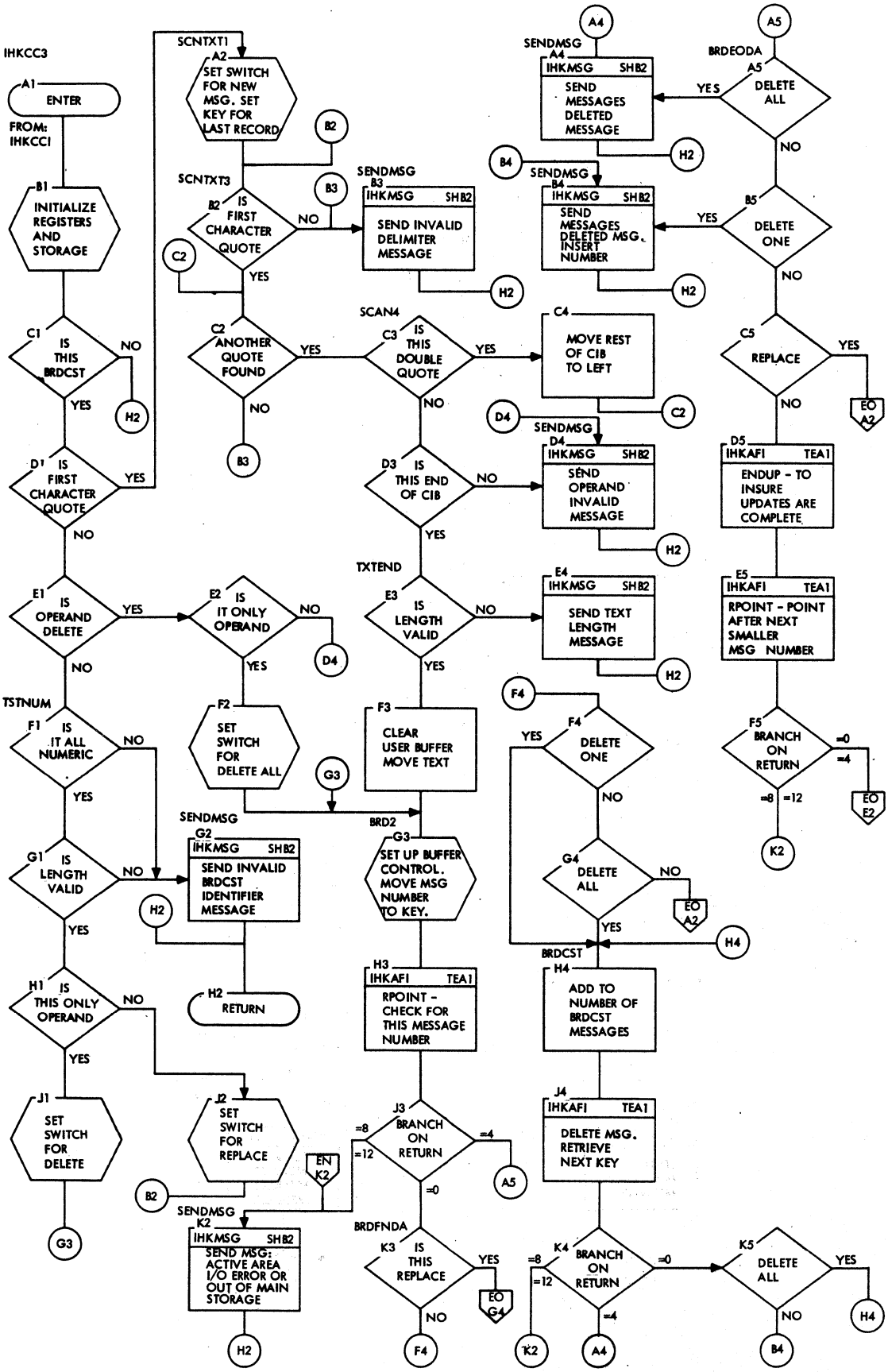
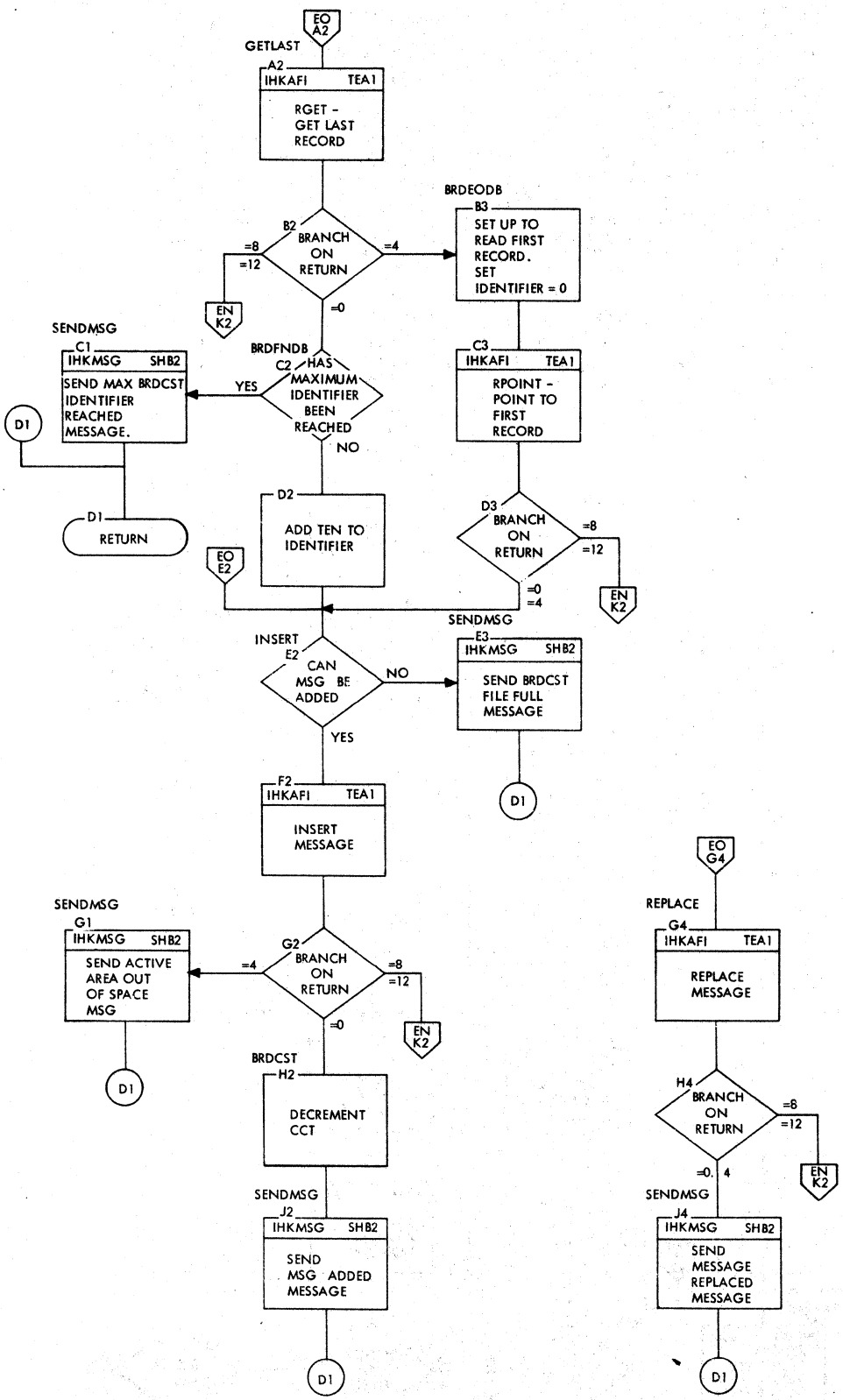


Chart EO. BRDCST Central Command Processor (IHKCC3)



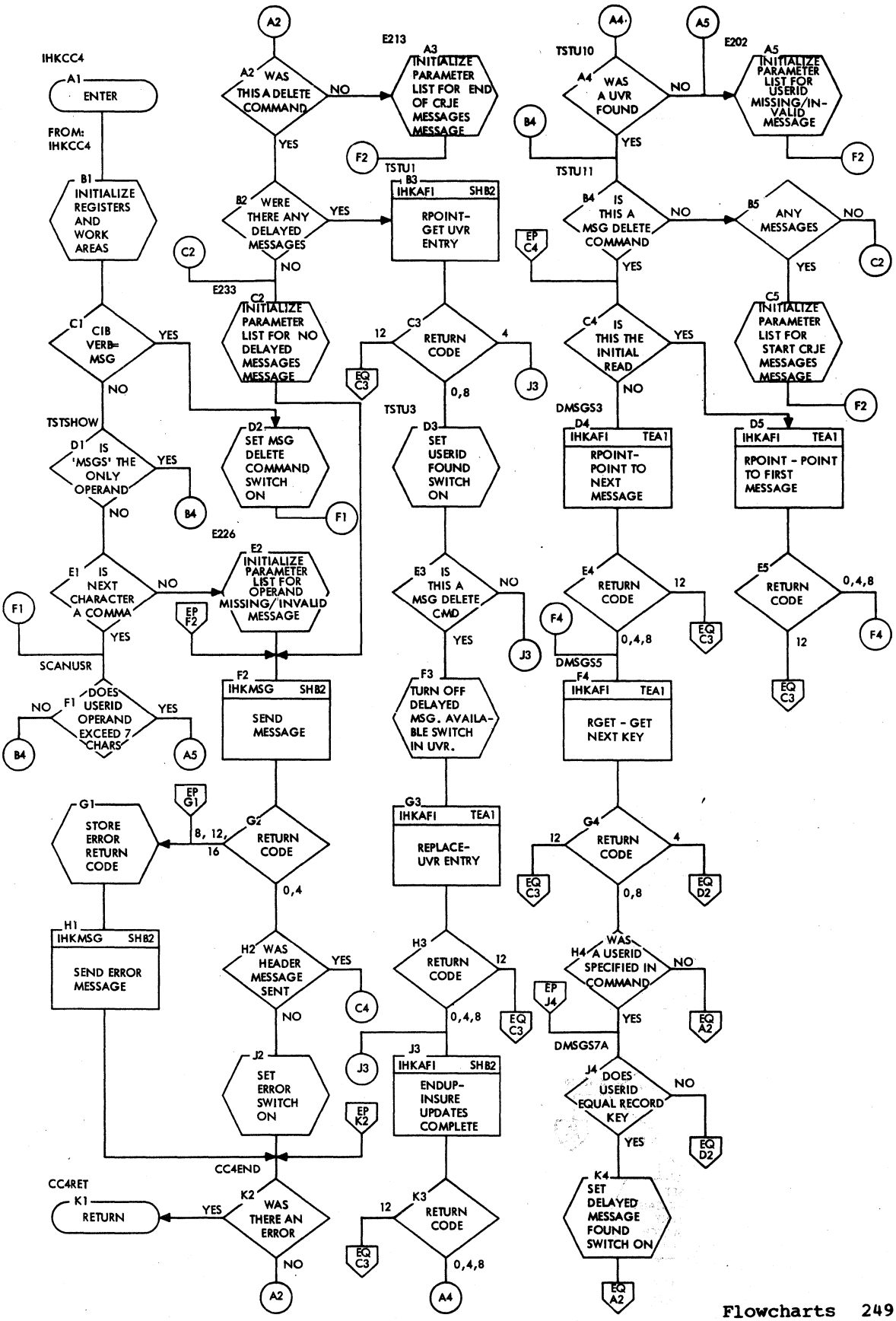

IHKCC4
A1 ENTER
FROM: IHKCC4
B1 INITIALIZE REGISTERS AND WORK AREAS
C1 CIB VERB= MSG
YES
NO
TSTSHOW
D1 IS 'MSGS' THE ONLY OPERAND
YES
NO
B4
E1 IS NEXT CHARACTER A COMMA
NO
YES
F1
SCANUSR
DOES USERID OPERAND EXCEED 7 CHARS
NO
YES
B4
A5
A2
A2 WAS THIS A DELETE COMMAND
NO
YES
E213
A3 INITIALIZE PARAMETER LIST FOR END OF CRJE MESSAGES MESSAGE
F2
TSTU1
B2 WERE THERE ANY DELAYED MESSAGES
YES
NO
B3 IHKAFI SHB2 RPOINT-GET UVR ENTRY
C2
E233
C2 INITIALIZE PARAMETER LIST FOR NO DELAYED MESSAGES MESSAGE
C3 RETURN CODE
12
4
EQ C3
0,8
J3
TSTU3
D2 SET MSG DELETE COMMAND SWITCH ON
F1
D3 SET USERID FOUND SWITCH ON
E226
E2 INITIALIZE PARAMETER LIST FOR OPERAND MISSING/INVALID MESSAGE
EP F2
E3 IS THIS A MSG DELETE CMD
NO
YES
J3
F2 IHKMSG SHB2 SEND MESSAGE
F3 TURN OFF DELAYED MSG. AVAILABLE SWITCH IN UVR.
EP G1
G1 STORE ERROR RETURN CODE
G2 RETURN CODE
8, 12, 16
0,4
G3 IHKAFI TEA1 REPLACE-UVR ENTRY
H1 IHKMSG SHB2 SEND ERROR MESSAGE
H2 WAS HEADER MESSAGE SENT
YES
NO
C4
H3 RETURN CODE
12
0,4,8
EQ C3
J3
J2 SET ERROR SWITCH ON
EP K2
J3 IHKAFI SHB2 ENDUP-INSURE UPDATES COMPLETE
CC4END
CC4RET
K1 RETURN
K2 WAS THERE AN ERROR
YES
NO
A2
K3 RETURN CODE
12
0,4,8
EQ C3
A4
TSTU10
A4 WAS A UVR FOUND
NO
YES
A5
E202
A5 INITIALIZE PARAMETER LIST FOR USERID MISSING/INVALID MESSAGE
F2
B4
TSTU11
B4 IS THIS A MSG DELETE COMMAND
NO
YES
EP C4
B5 ANY MESSAGES
NO
YES
C2
C4 IS THIS THE INITIAL READ
YES
NO
C5 INITIALIZE PARAMETER LIST FOR START CRJE MESSAGES MESSAGE
F2
DMSGS3
D4 IHKAFI TEA1 RPOINT-POINT TO NEXT MESSAGE
D5 IHKAFI TEA1 RPOINT - POINT TO FIRST MESSAGE
E4 RETURN CODE
12
0,4,8
EQ C3
F4
DMSGS5
F4 IHKAFI TEA1 RGET - GET NEXT KEY
E5 RETURN CODE
0,4,8
12
F4
EQ C3
G4 RETURN CODE
12
4
EQ C3
0,8
EQ D2
H4 WAS A USERID SPECIFIED IN COMMAND
NO
YES
EP J4
EQ A2
DMSGS7A
J4 DOES USERID EQUAL RECORD KEY
NO
YES
EQ D2
K4 SET DELAYED MESSAGE FOUND SWITCH ON
EQ A2

Chart EQ. SHOW MSGS and MSG D=userid Central Command Processor (IHKCC4)

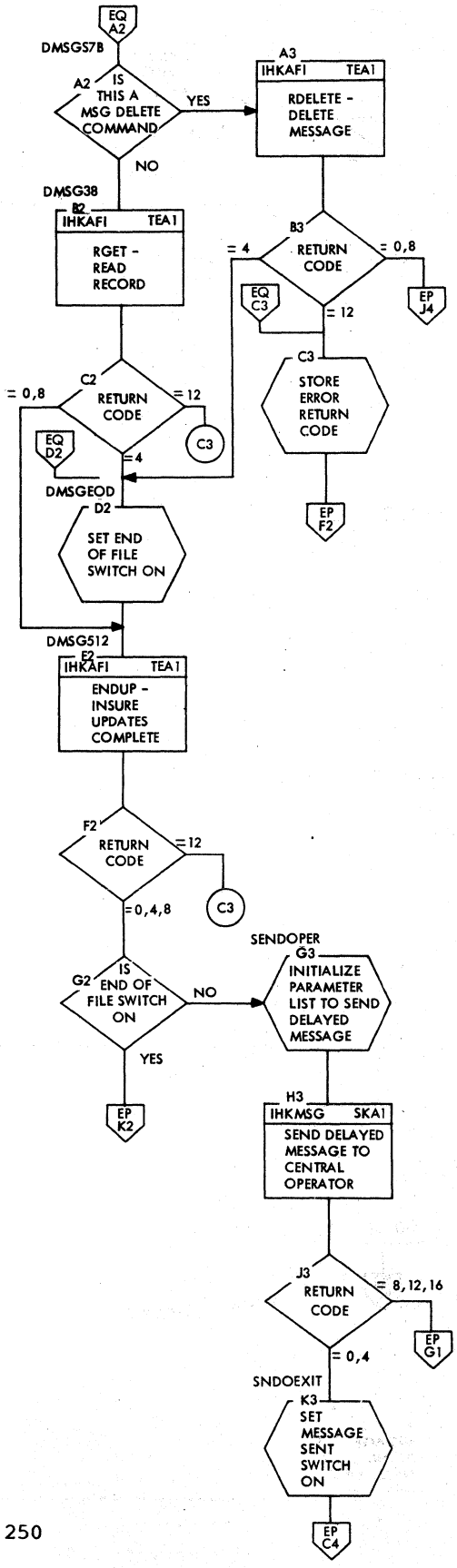


Chart ER. CENOUT Central Command Processor (IHKCC5)

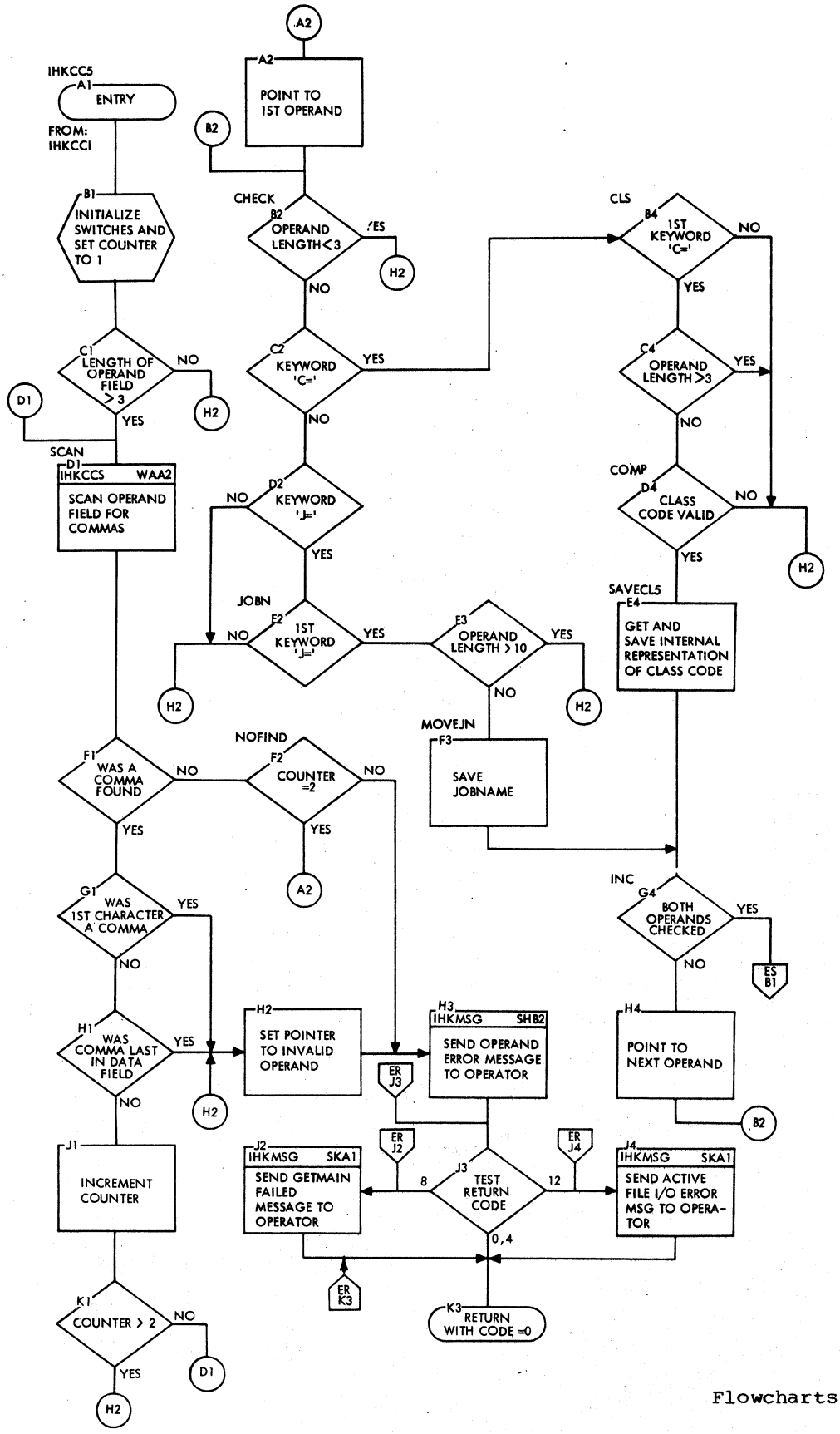
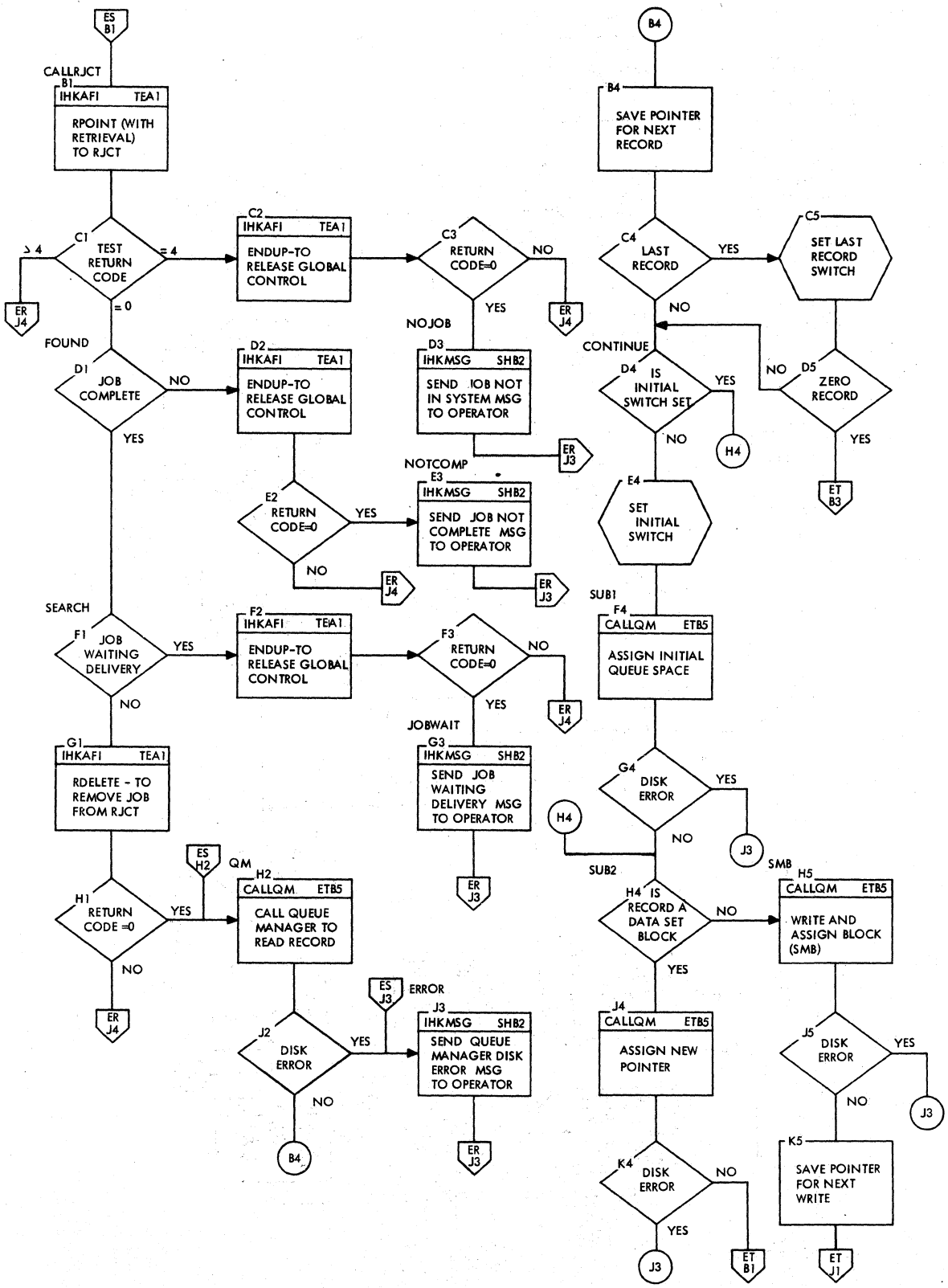


Chart ES. CENOUT Central Command Processor (IHKCC5)



• Chart ET. CENOUT Central Command Processor (IHKCC5)

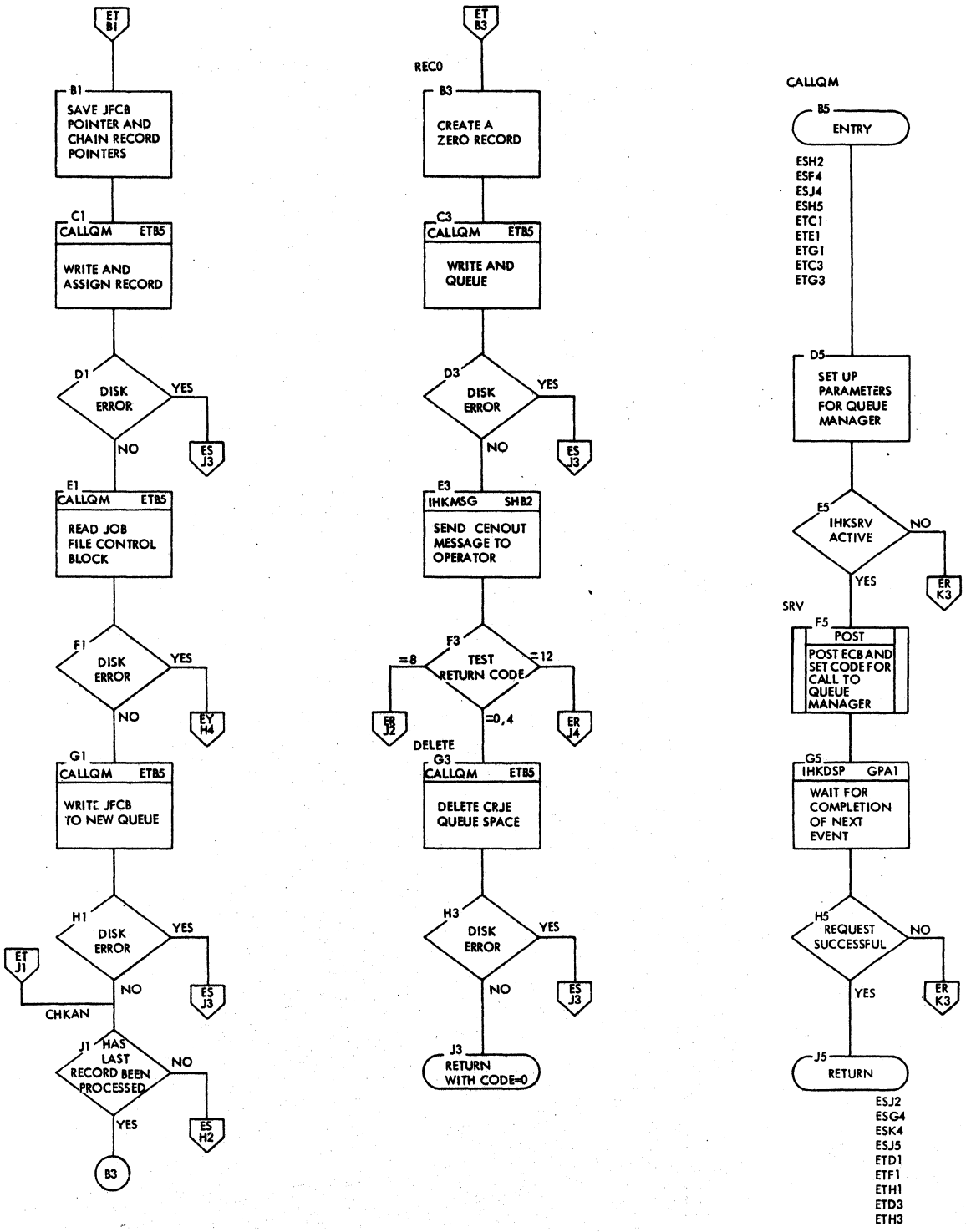
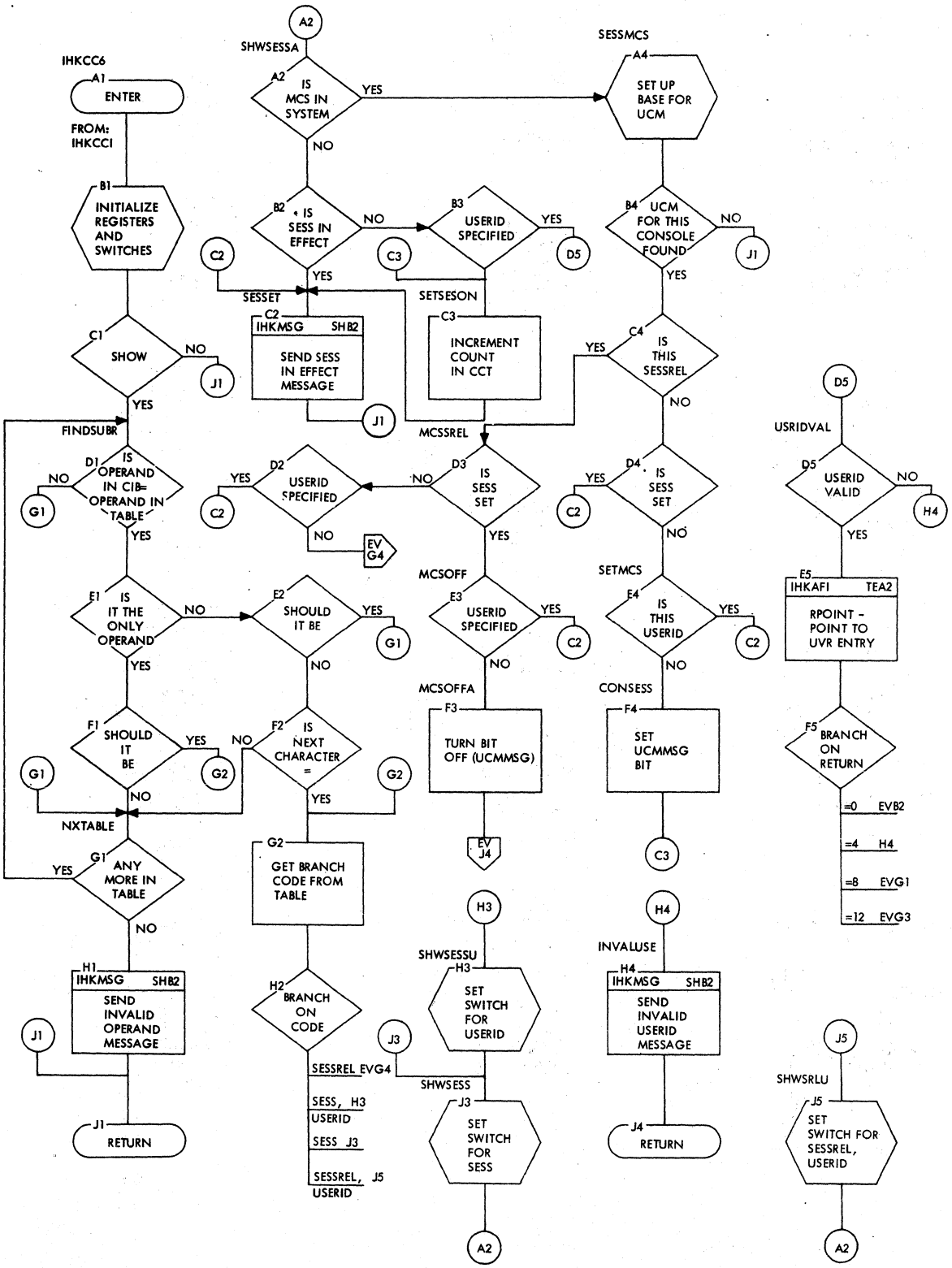


Chart EU. SHOW SESS and SHOW SESSREL Central Command Processor (IHKCC6)



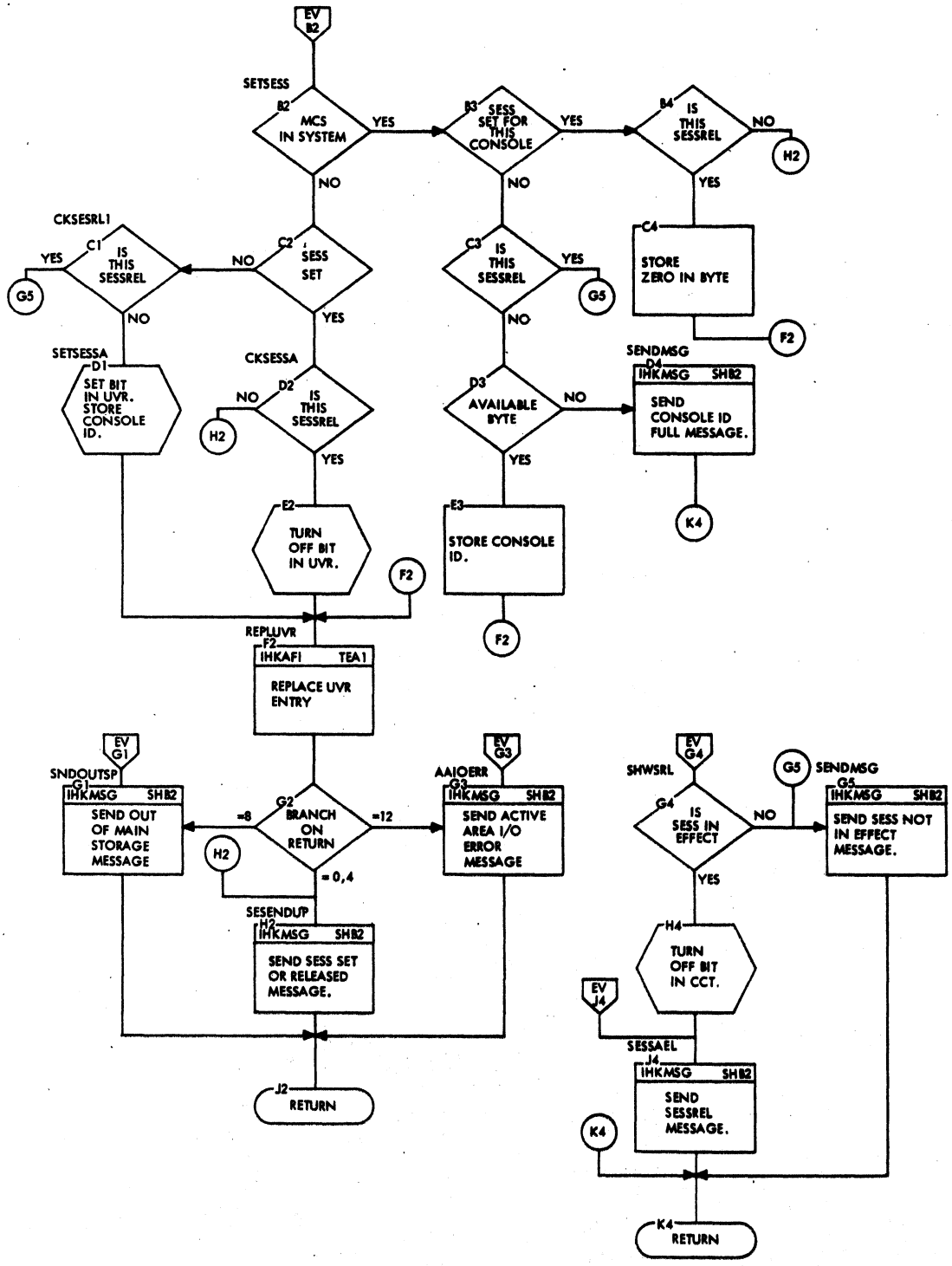
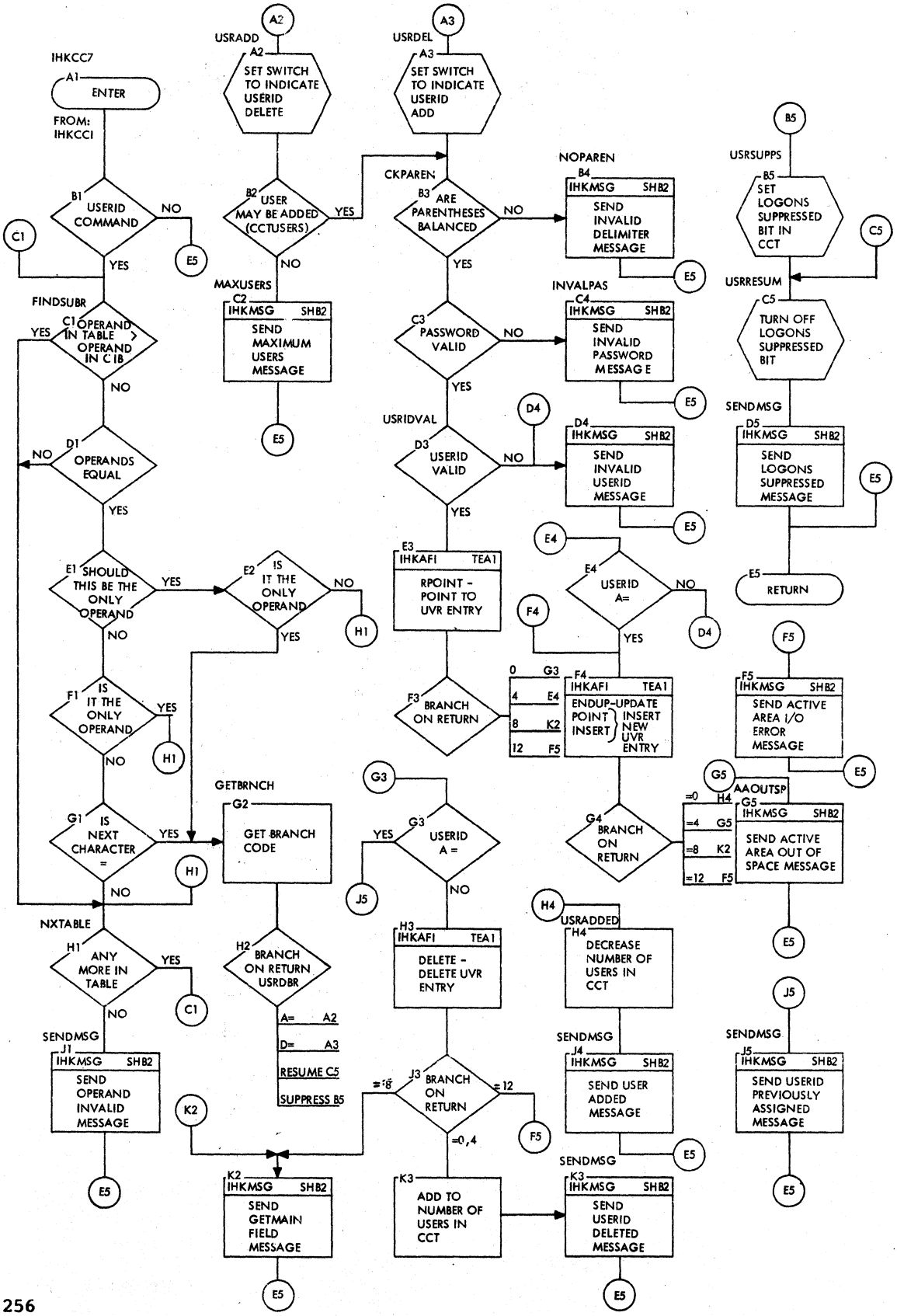

EV B2
SETSESS
B2 MCS IN SYSTEM
YES
NO
B3 SESS SET FOR THIS CONSOLE
YES
NO
B4 IS THIS SESSREL
NO
H2
YES
C4 STORE ZERO IN BYTE
F2
CKSESRL1
C1 IS THIS SESSREL
YES
G5
NO
C2 SESS SET
NO
YES
C3 IS THIS SESSREL
YES
G5
NO
SETSESSA
D1 SET BIT IN UVR. STORE CONSOLE ID.
CKSESSA
D2 IS THIS SESSREL
NO
H2
YES
D3 AVAILABLE BYTE
NO
YES
SENDMSG
D4 IHKMSG SHB2
SEND CONSOLE ID FULL MESSAGE.
K4
E2 TURN OFF BIT IN UVR.
F2
E3 STORE CONSOLE ID.
F2
REPLUVR
F2 IHKAFI TEA1
REPLACE UVR ENTRY
EV G1
SNDOUTSP
G1 IHKMSG SHB2
SEND OUT OF MAIN STORAGE MESSAGE
G2 BRANCH ON RETURN
=8
H2
=12
= 0,4
EV G3
AAIOERR
G3 IHKMSG SHB2
SEND ACTIVE AREA I/O ERROR MESSAGE
SESENDUP
H2 IHKMSG SHB2
SEND SESS SET OR RELEASED MESSAGE.
J2 RETURN
EV G4
SHWSRL
G4 IS SESS IN EFFECT
NO
G5 SENDMSG
G5 IHKMSG SHB2
SEND SESS NOT IN EFFECT MESSAGE.
YES
H4 TURN OFF BIT IN CCT.
EV J4
SESSAEL
J4 IHKMSG SHB2
SEND SESSREL MESSAGE.
K4
K4 RETURN

Chart EW. USERID Central Command Processor (IHKCC7)



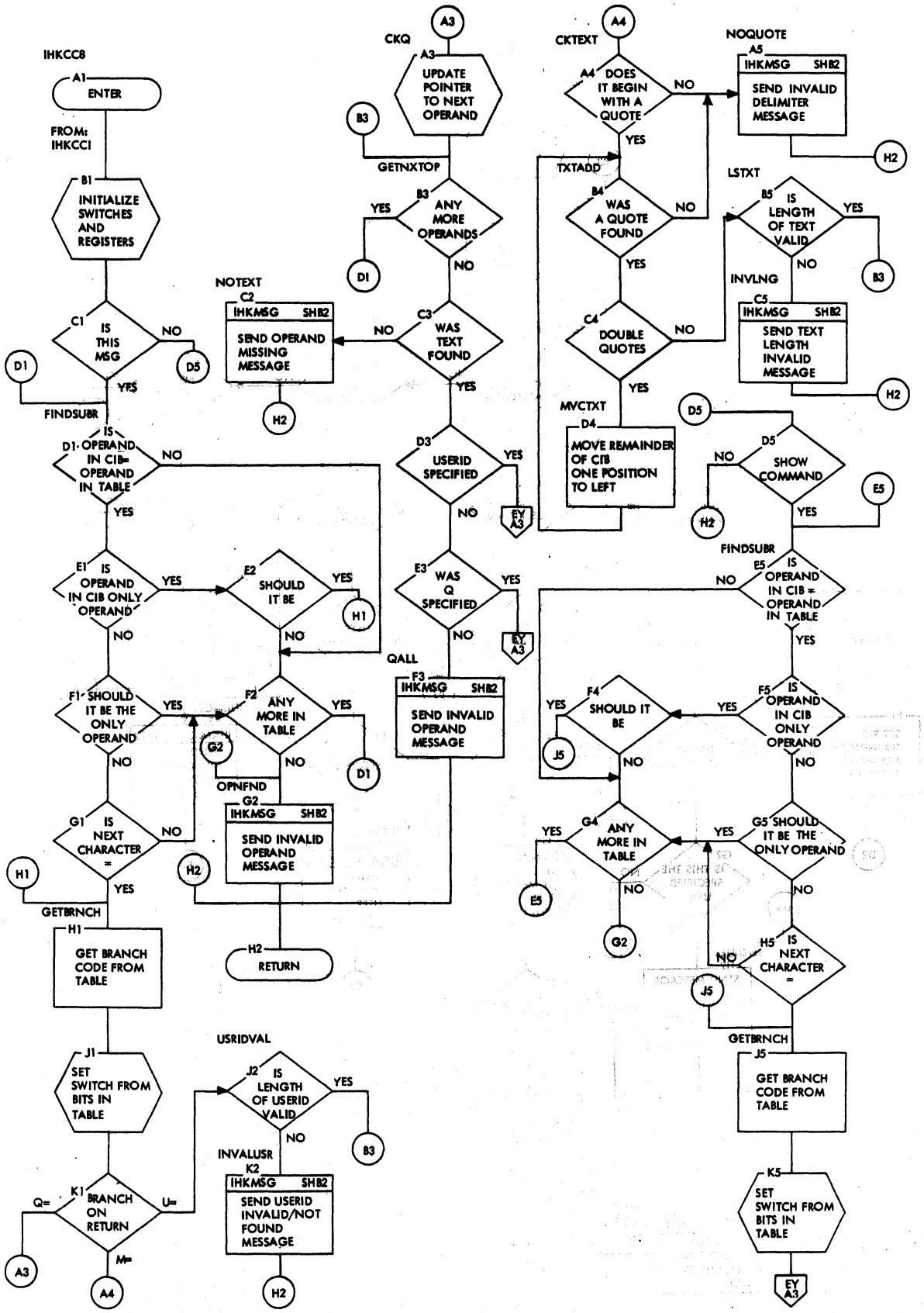

IHKCC8
A1
ENTER
FROM: IHKCCI
B1
INITIALIZE SWITCHES AND REGISTERS
C1
IS THIS MSG
NO
D5
YES
D1
FINDSUBR
D1
IS OPERAND IN CIB = OPERAND IN TABLE
NO
YES
E1
IS OPERAND IN CIB ONLY OPERAND
YES
E2
SHOULD IT BE
YES
H1
NO
NO
F1
SHOULD IT BE THE ONLY OPERAND
YES
F2
ANY MORE IN TABLE
YES
D1
G2
NO
NO
OPNFND
G2
IHKMSG SHB2
SEND INVALID OPERAND MESSAGE
G1
IS NEXT CHARACTER =
NO
YES
H1
H2
GETBRNCH
H1
GET BRANCH CODE FROM TABLE
H2
RETURN
J1
SET SWITCH FROM BITS IN TABLE
K1
BRANCH ON RETURN
Q=
U=
M=
A3
A4
USRIDVAL
J2
IS LENGTH OF USERID VALID
YES
B3
NO
INVALUSR
K2
IHKMSG SHB2
SEND USERID INVALID/NOT FOUND MESSAGE
H2
A3
CKQ
A3
UPDATE POINTER TO NEXT OPERAND
B3
GETNXTOP
B3
ANY MORE OPERANDS
YES
D1
NO
C3
WAS TEXT FOUND
NO
NOTEXT
C2
IHKMSG SHB2
SEND OPERAND MISSING MESSAGE
H2
YES
D3
USERID SPECIFIED
YES
EY A3
NO
E3
WAS Q SPECIFIED
YES
EY A3
NO
QALL
F3
IHKMSG SHB2
SEND INVALID OPERAND MESSAGE
A4
CKTEXT
A4
DOES IT BEGIN WITH A QUOTE
NO
YES
NOQUOTE
A5
IHKMSG SHB2
SEND INVALID DELIMITER MESSAGE
H2
TXTADD
B4
WAS A QUOTE FOUND
NO
YES
LSTXT
B5
IS LENGTH OF TEXT VALID
YES
B3
NO
INVLNG
C5
IHKMSG SHB2
SEND TEXT LENGTH INVALID MESSAGE
H2
C4
DOUBLE QUOTES
NO
YES
MVCTXT
D4
MOVE REMAINDER OF CIB ONE POSITION TO LEFT
D5
D5
SHOW COMMAND
NO
H2
YES
E5
FINDSUBR
E5
IS OPERAND IN CIB = OPERAND IN TABLE
NO
YES
F4
SHOULD IT BE
YES
J5
NO
F5
IS OPERAND IN CIB ONLY OPERAND
YES
NO
G4
ANY MORE IN TABLE
YES
E5
NO
G2
G5
SHOULD IT BE THE ONLY OPERAND
YES
NO
H5
IS NEXT CHARACTER =
NO
J5
GETBRNCH
J5
GET BRANCH CODE FROM TABLE
K5
SET SWITCH FROM BITS IN TABLE
EY A3

Chart EY. MSG and SHOW ACTIVE Central Command Processor (IHKCC8)

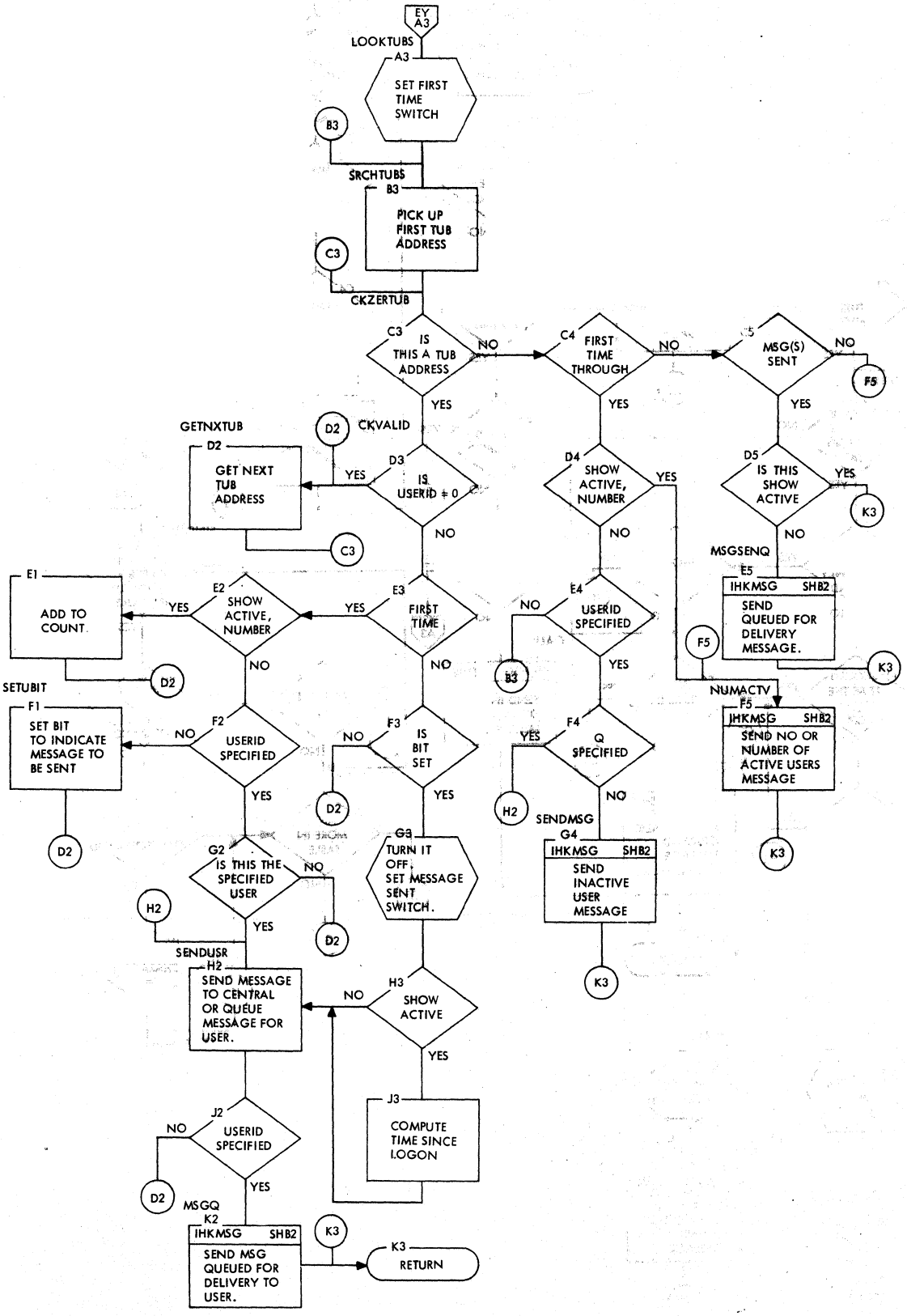
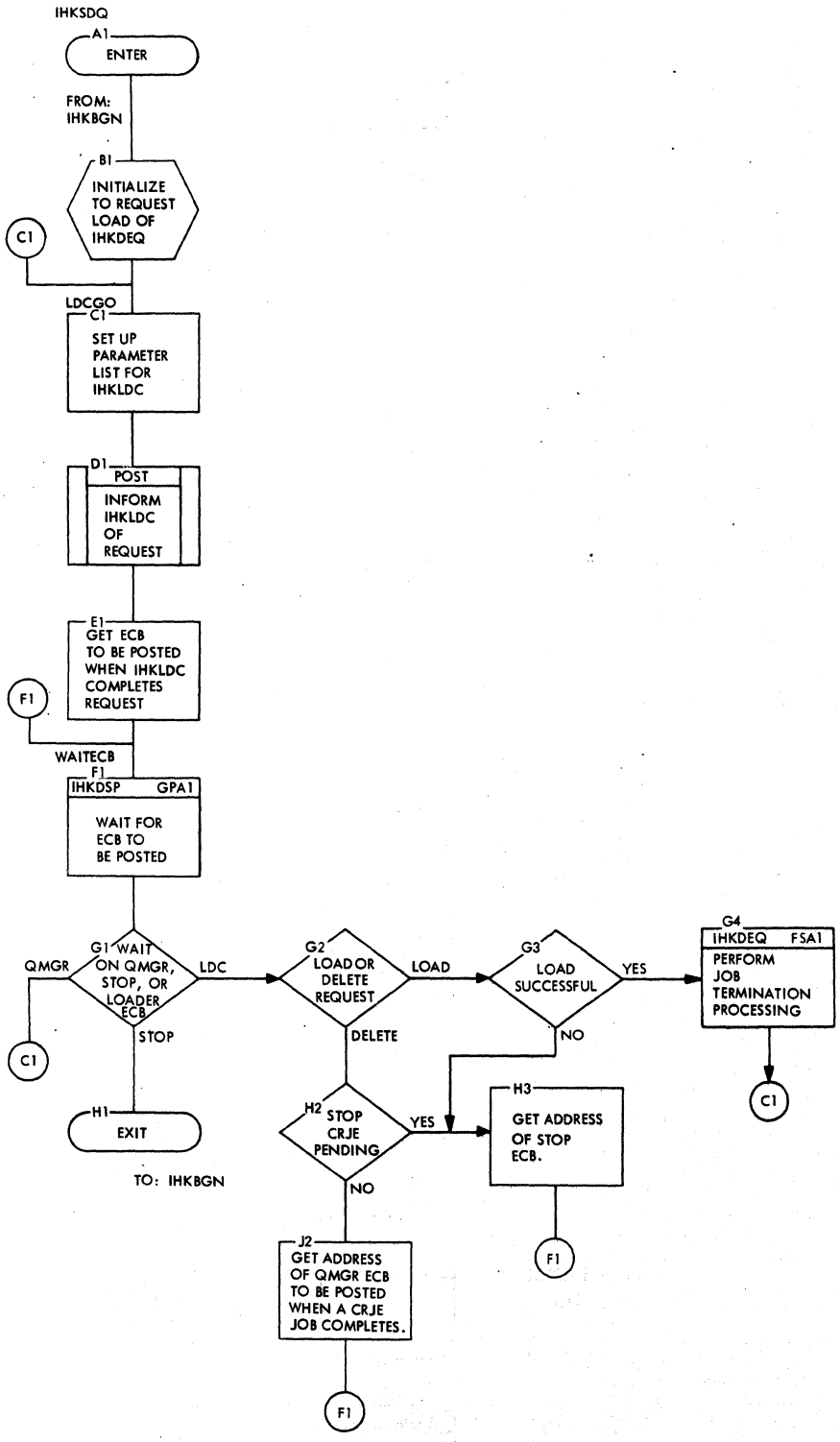
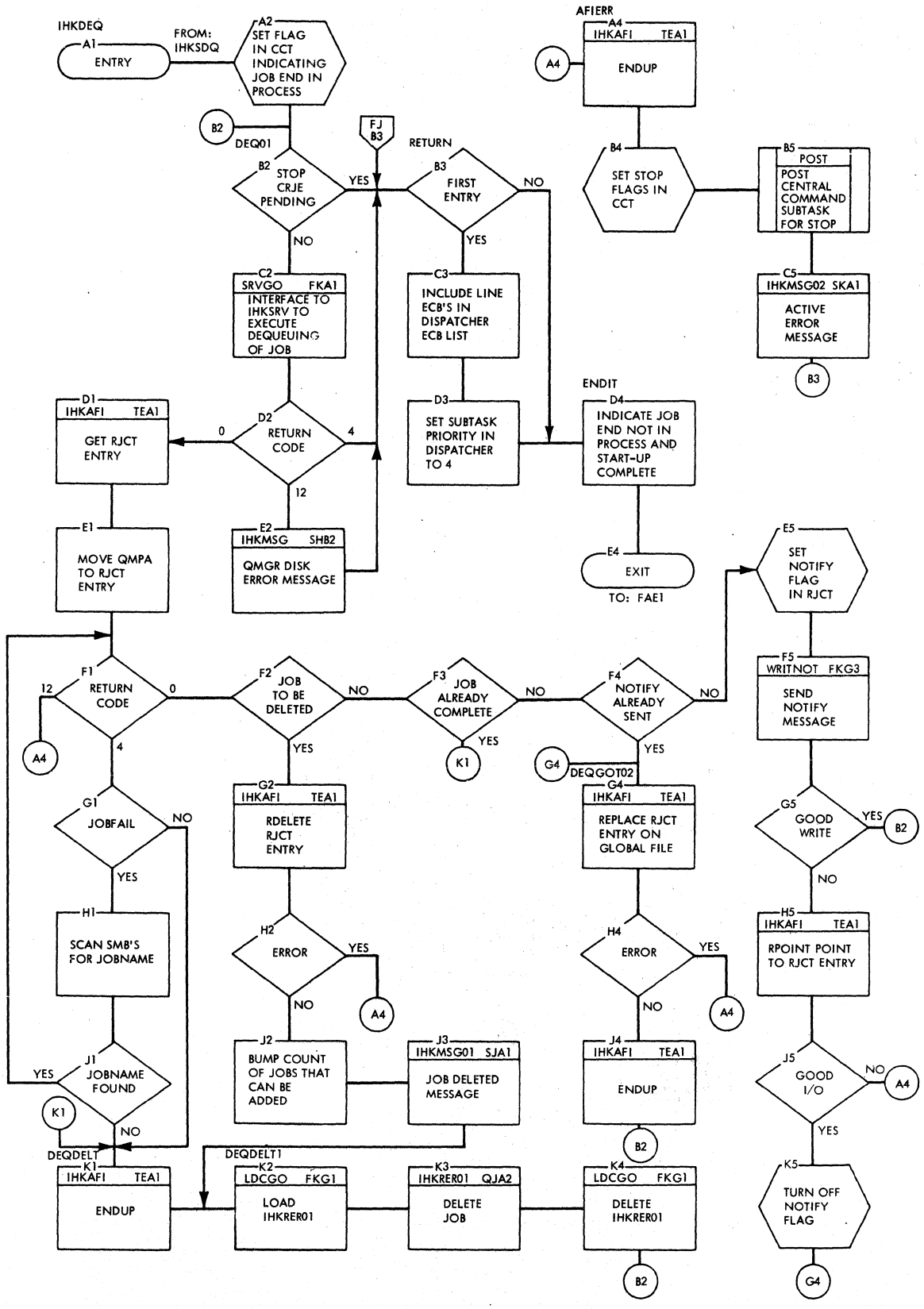


Chart FA. Job Termination Handling Module (IHKSDQ)



• Chart FJ. Dequeue/Job End Processor (IHKDEQ)



• Chart FK. Dequeue/Job End Processor (IHKDEQ)

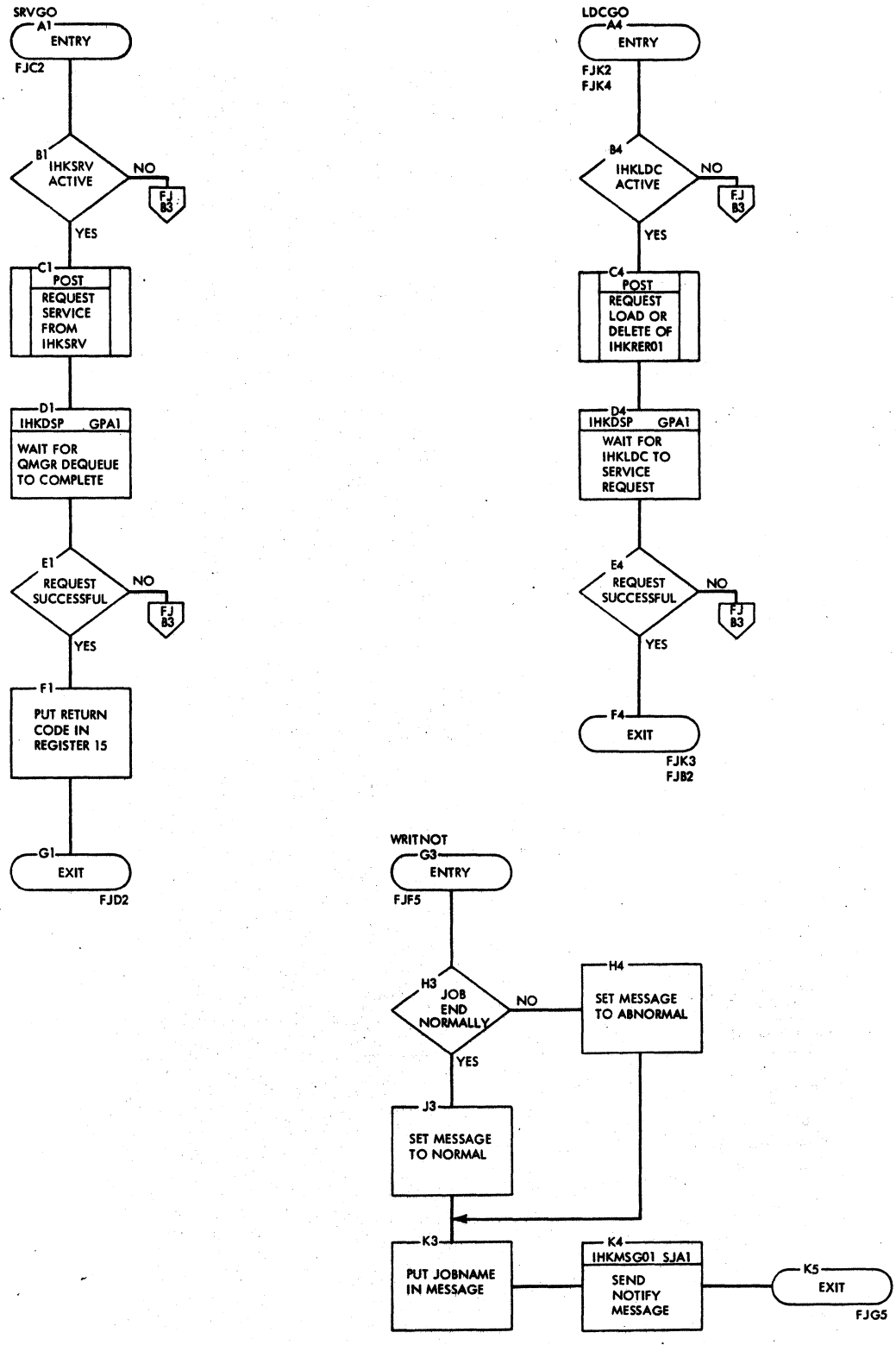


Chart GA. Command analyzer Module (IHKCMD)

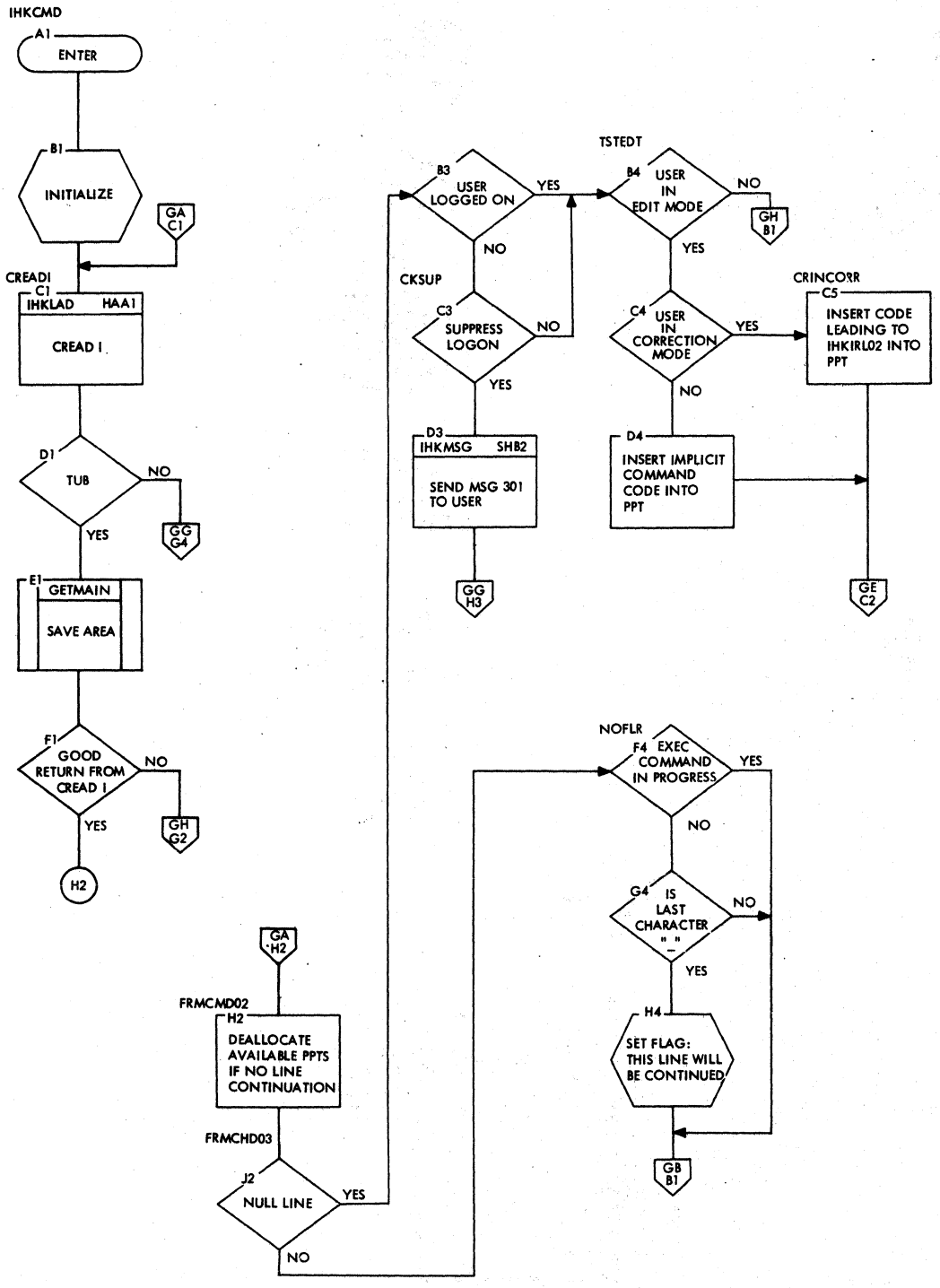


Chart GB. Command Analyzer Module (IHKCMD)

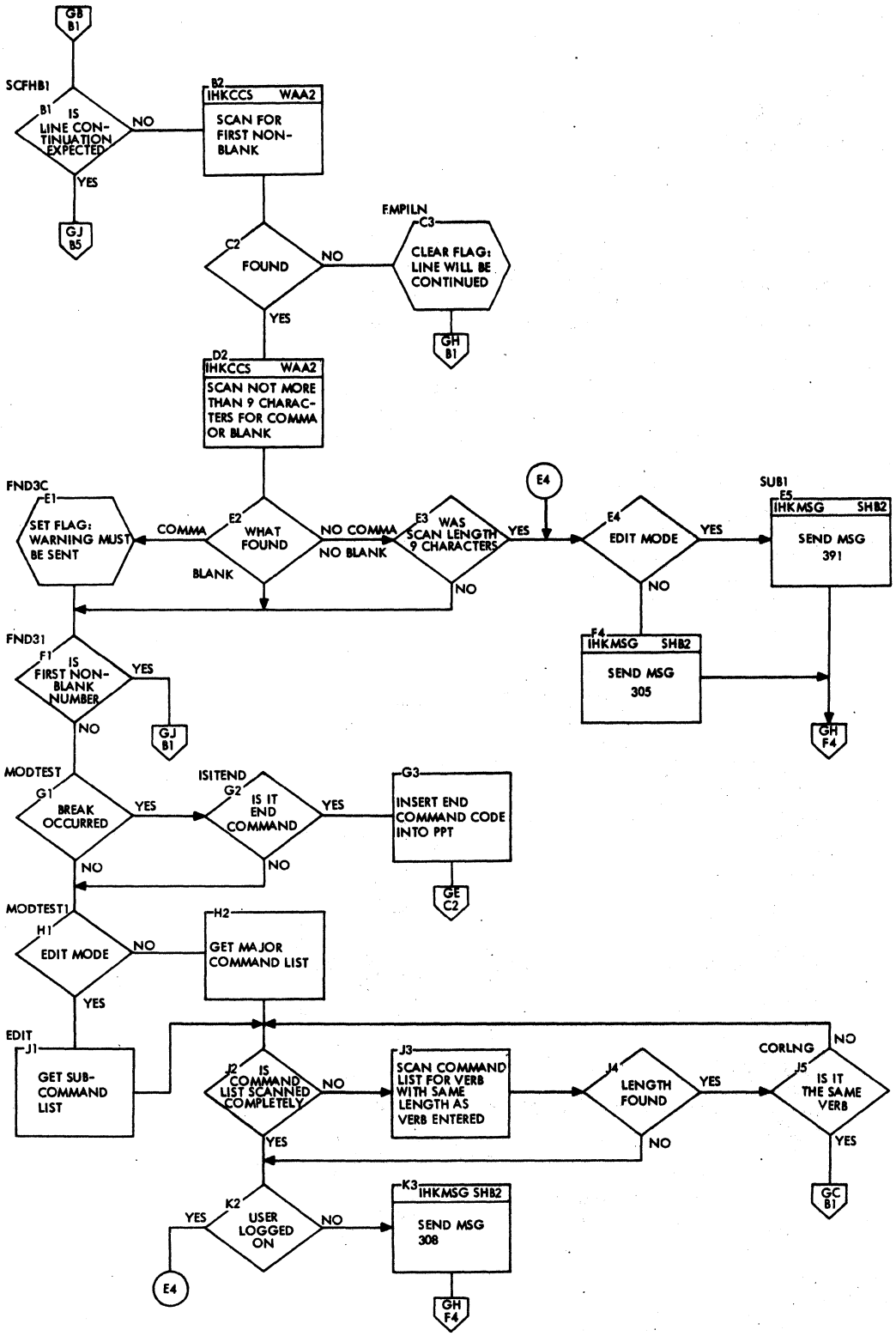


Chart GC. Command Analyzer Module (IHKCMD)

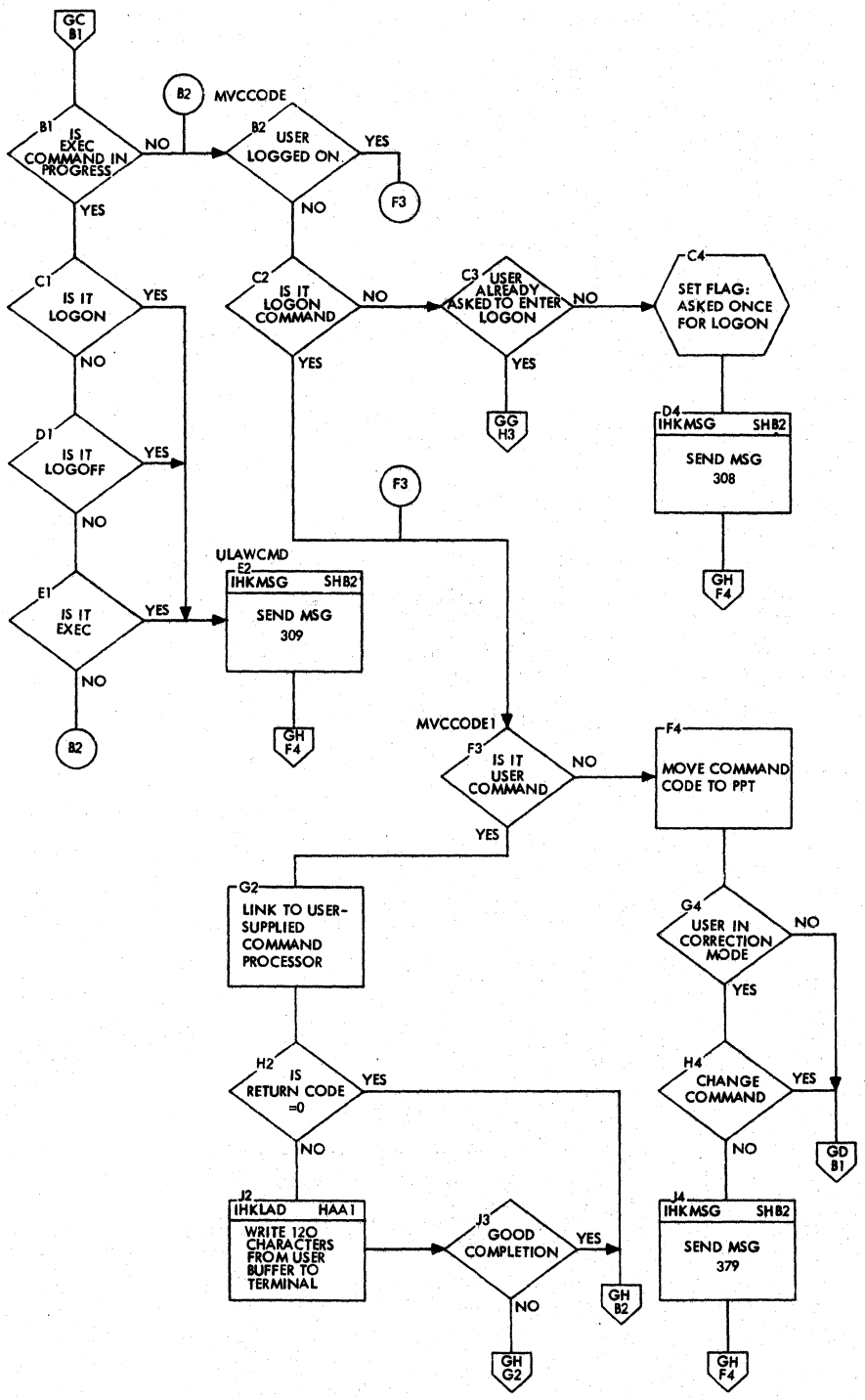


Chart GD. Command Analyzer Module (IHKCMD)

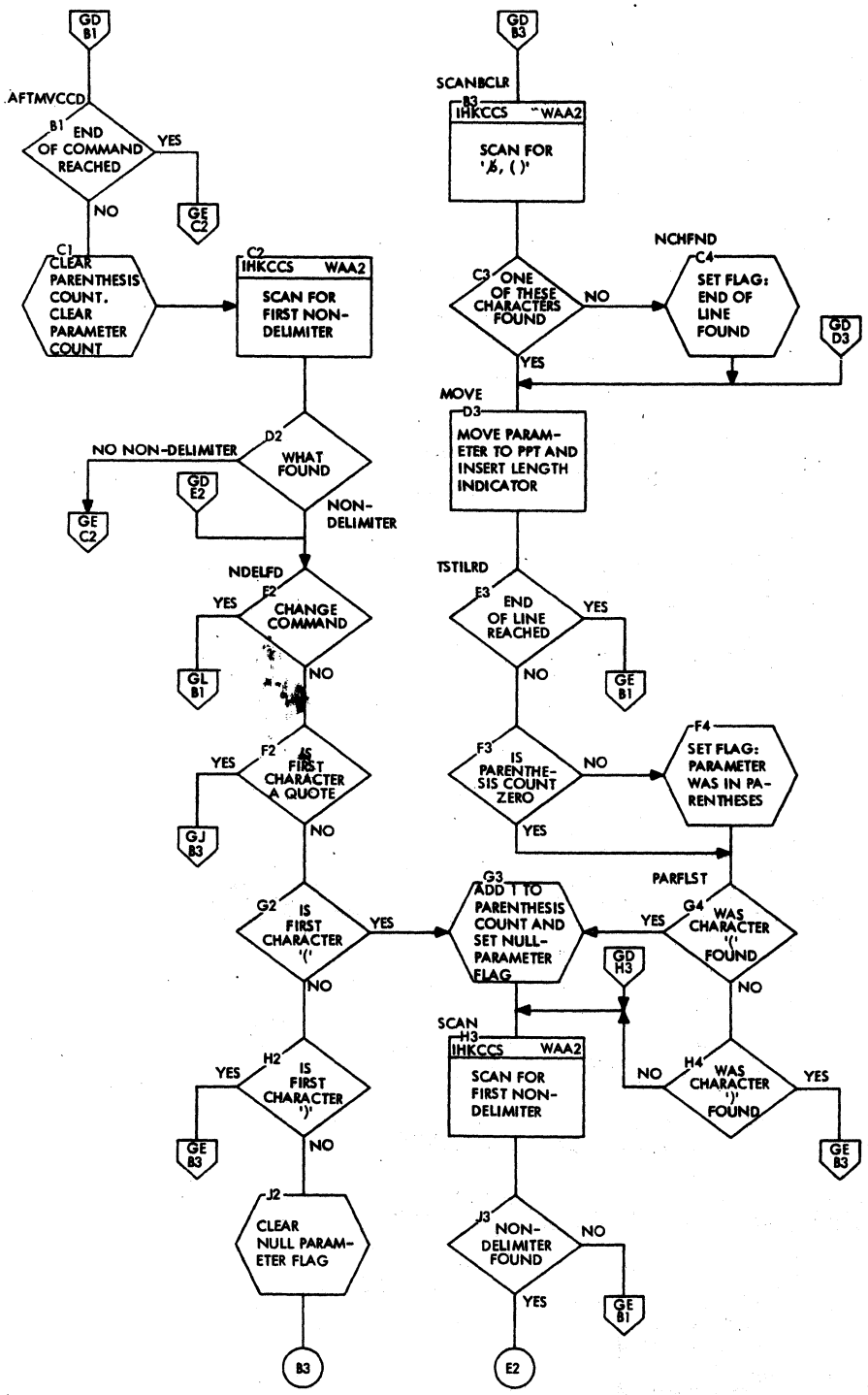
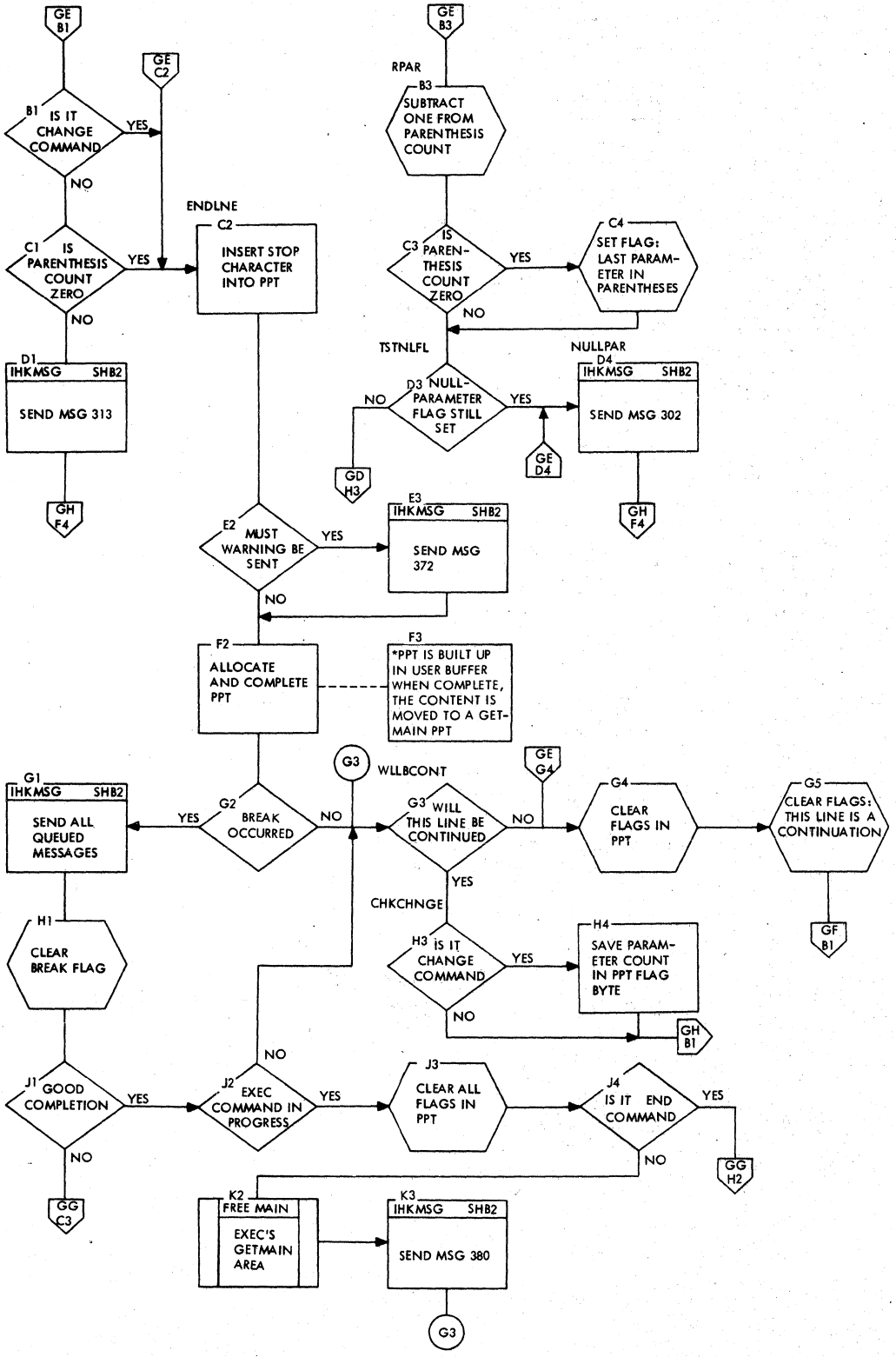
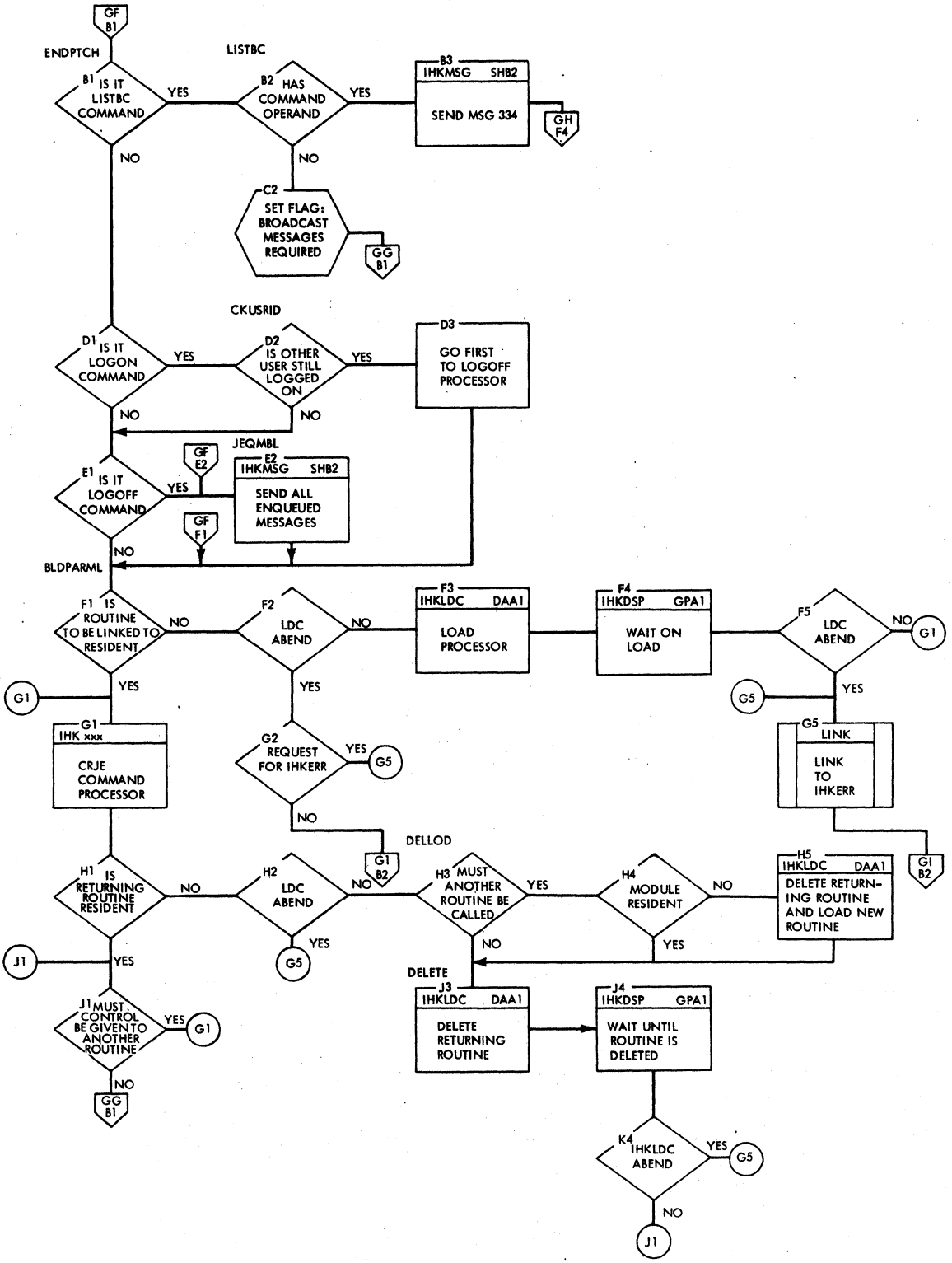


Chart GE. Command Analyzer Module (IHKCMD)



• Chart GF. Command Analyzer Module (IHKCMD)



• Chart GG. Command Analyzer Module (IHKCMD)

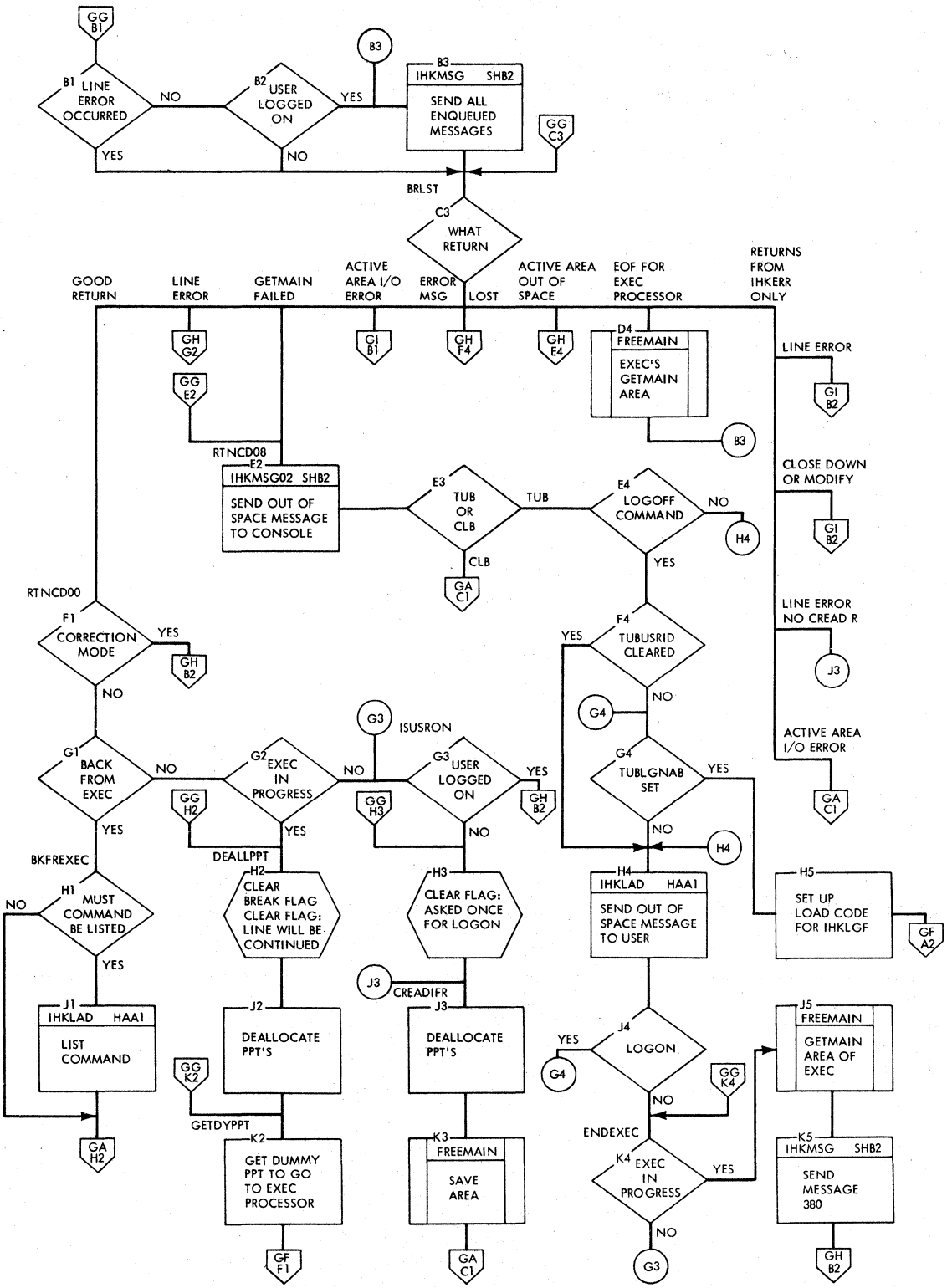


Chart GH. Command Analyzer Module (IHKCMD)

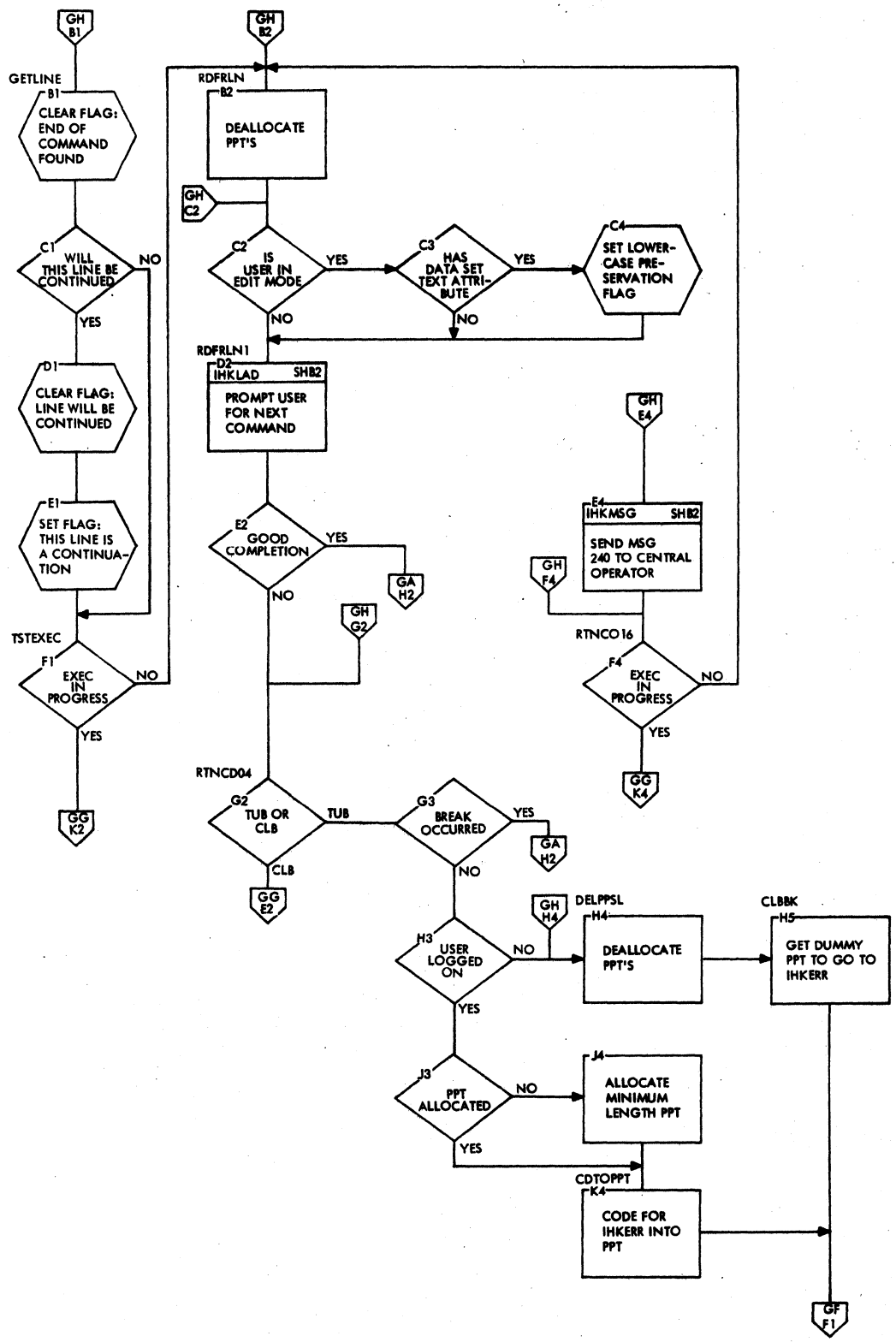


Chart GI. Command Analyzer Module (IHKCMD)

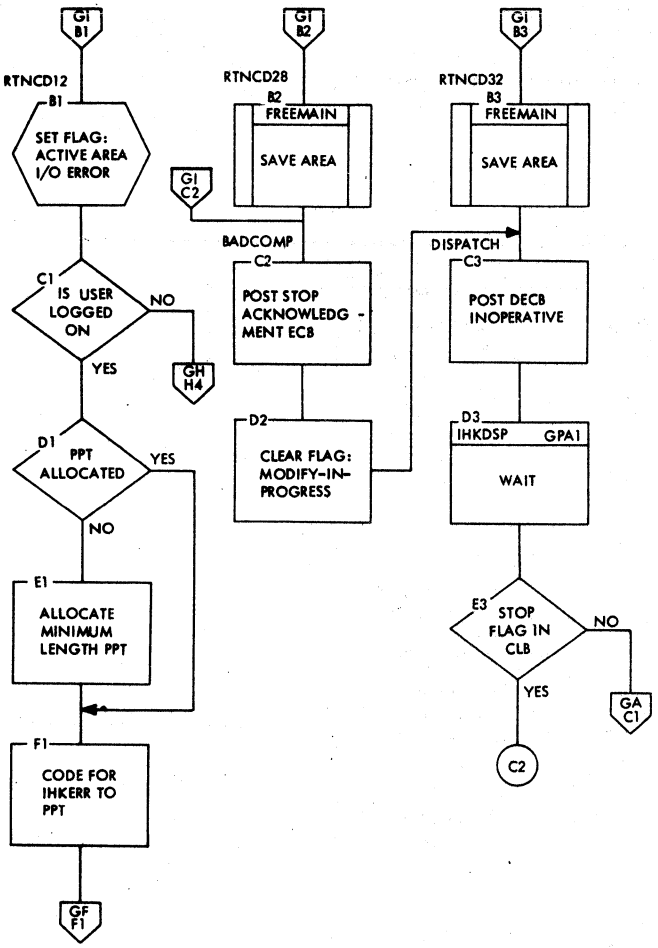


Chart GJ. Command Analyzer Module (IHKCMD)

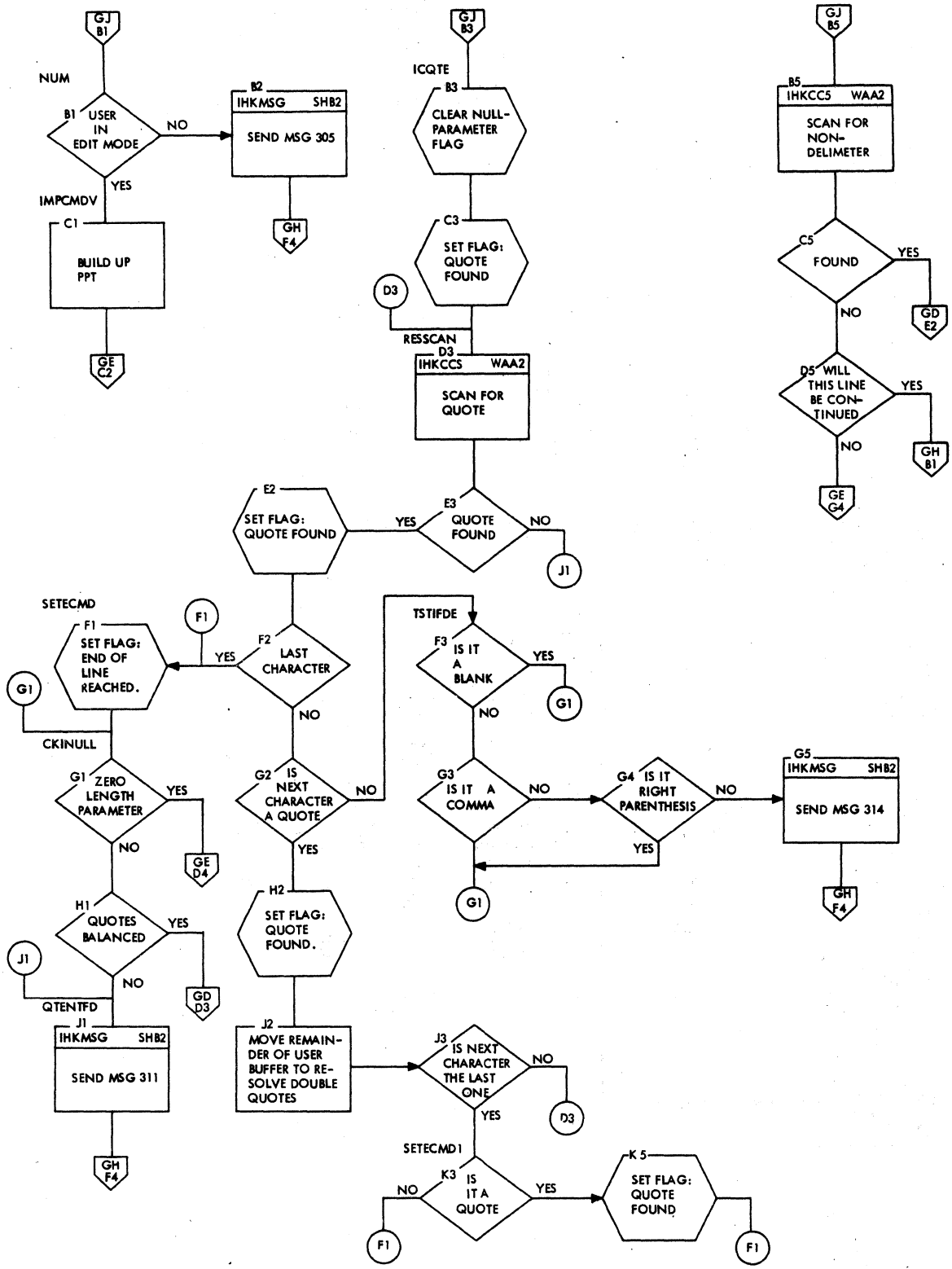


Chart GK. Command Analyzer Module (IHKCMD)

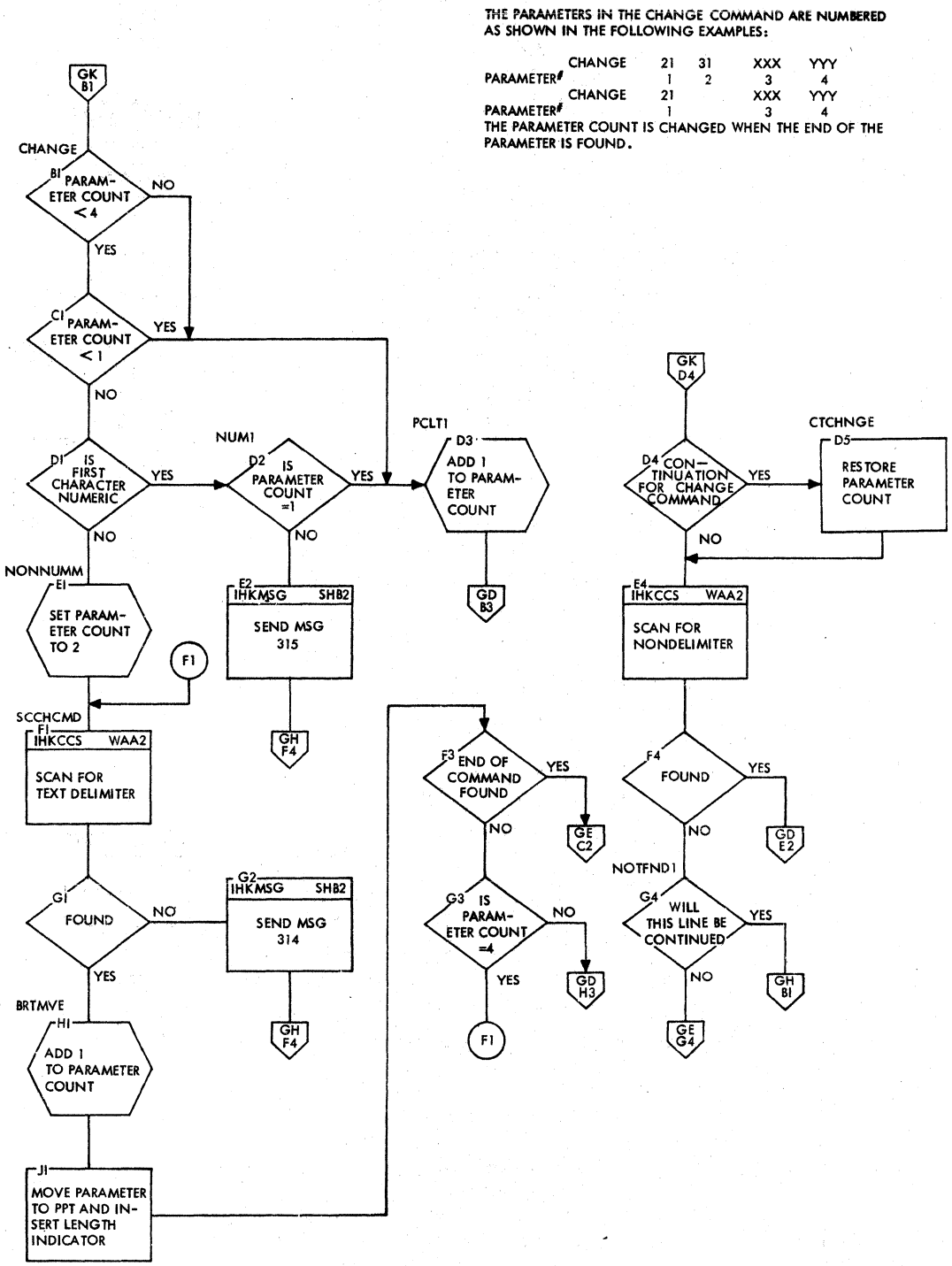


Chart GP. CRJE Dispatcher (IHKDSP)

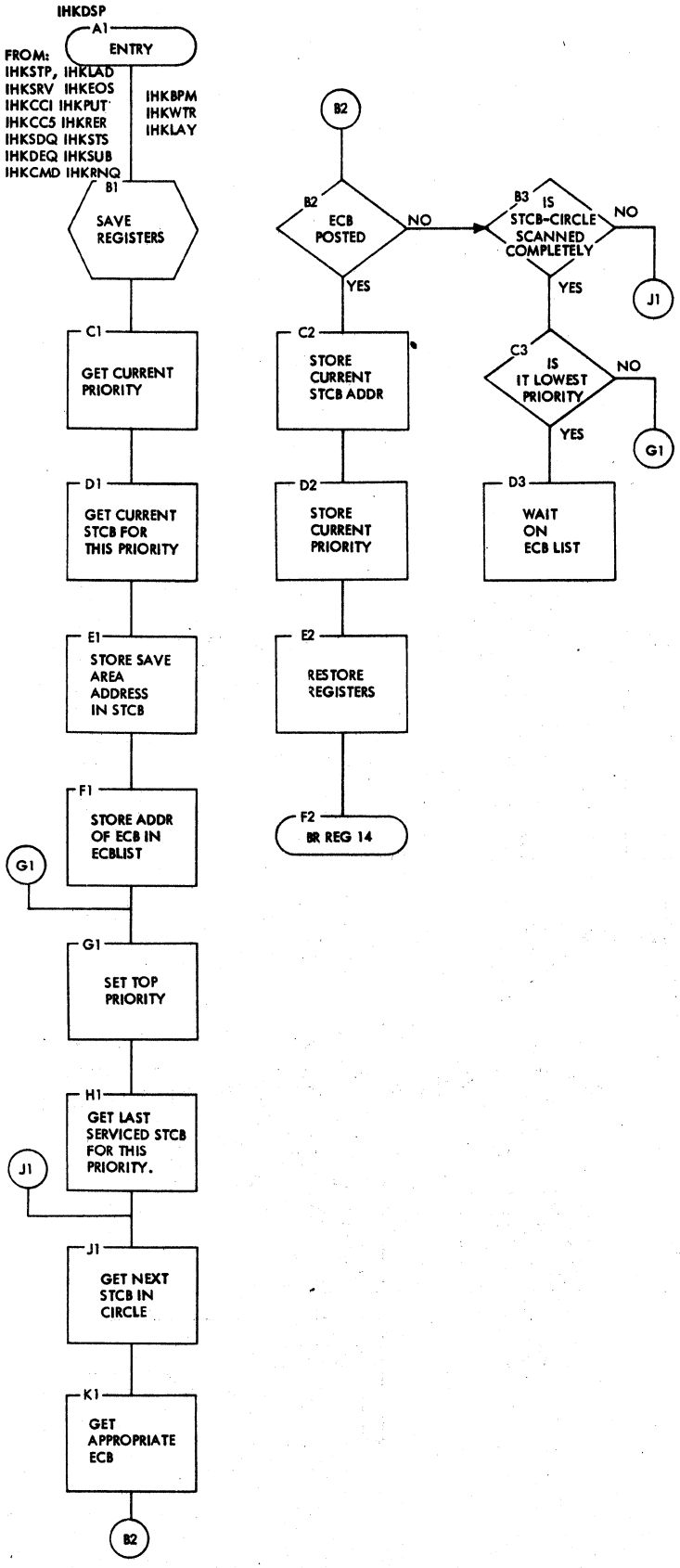


Chart GS. Line Error and Active Area I/O Error Recovery Module (IHKERR)

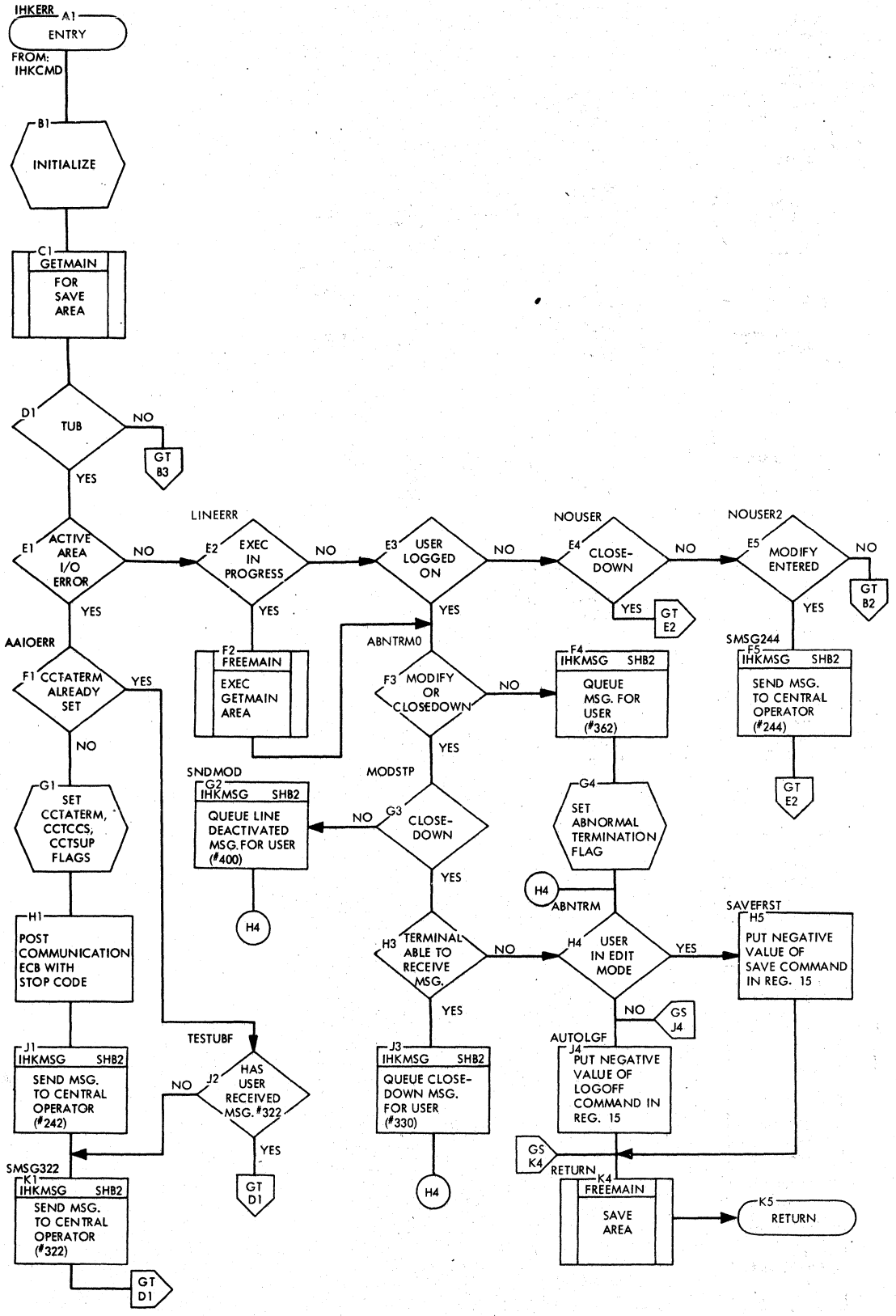


Chart GT. Line Error and Active Area I/O Error Recovery Module (IHKERR)

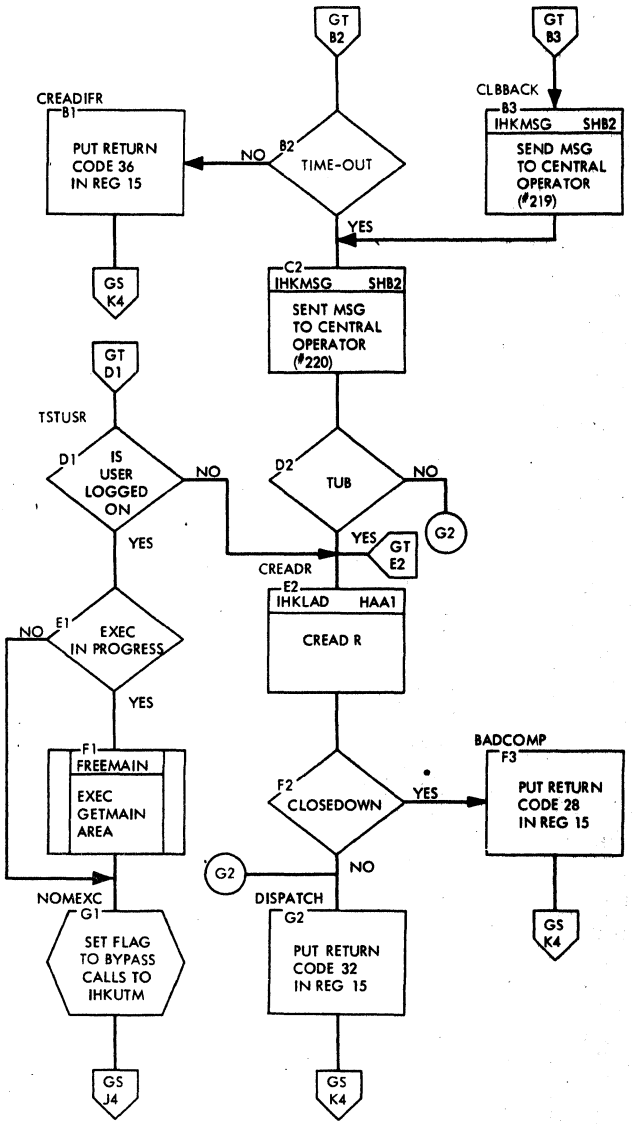


Chart HA. Communication Line Administrator Module (IHKLAD)

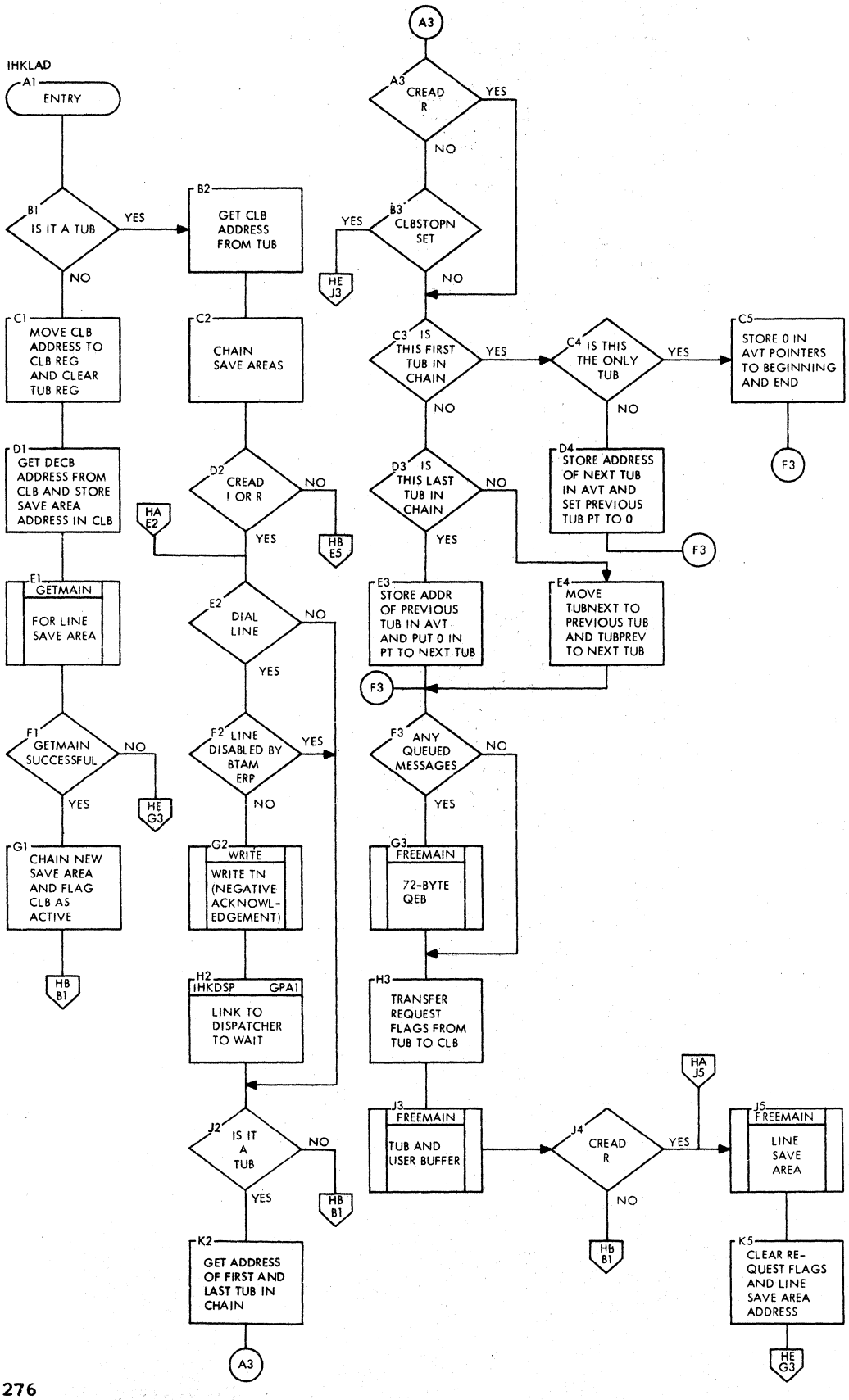


Chart HB. Communication Line Administrator Module (IHKLAD)

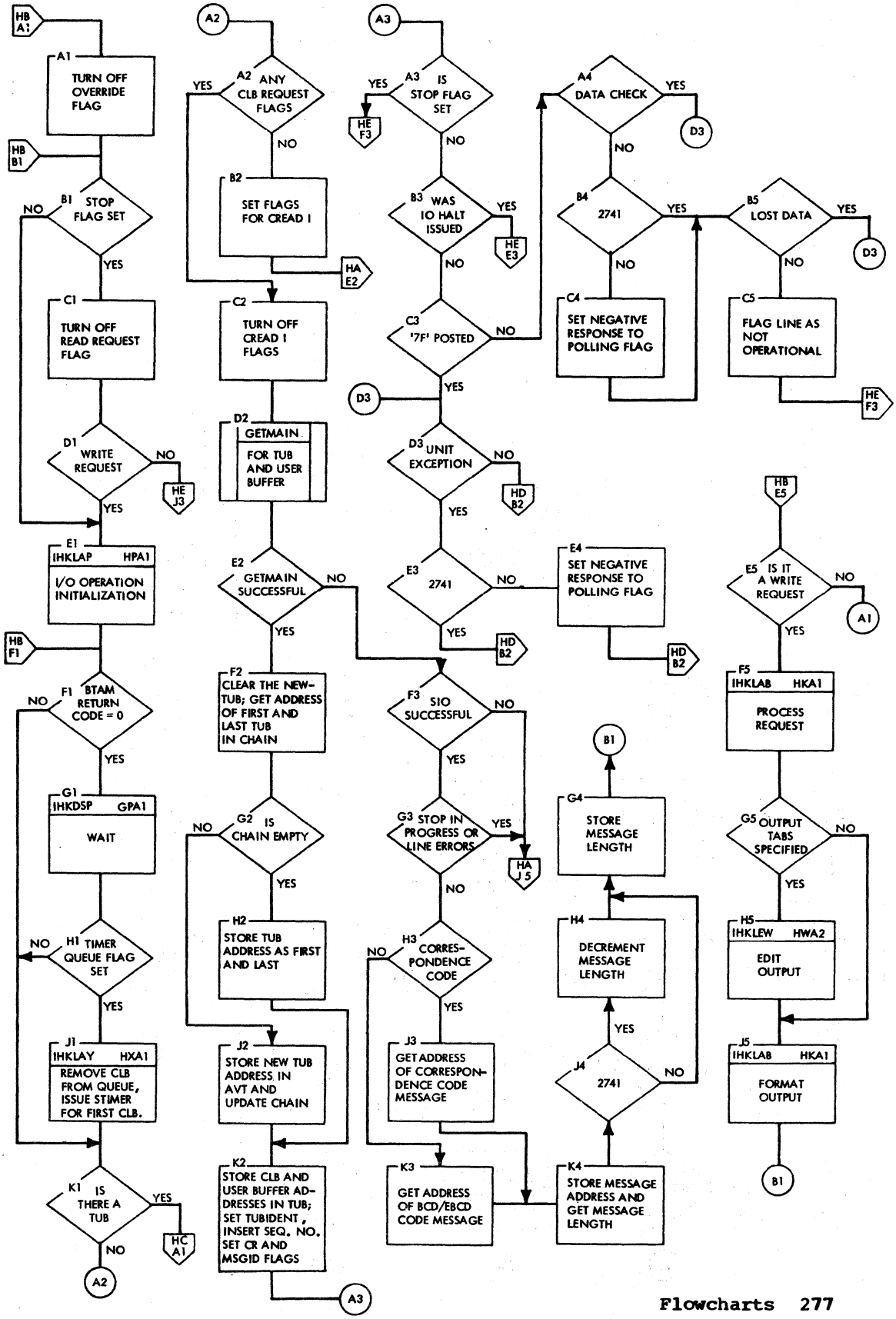


Chart HC. Communication Line Administrator Module (IHKLAD)

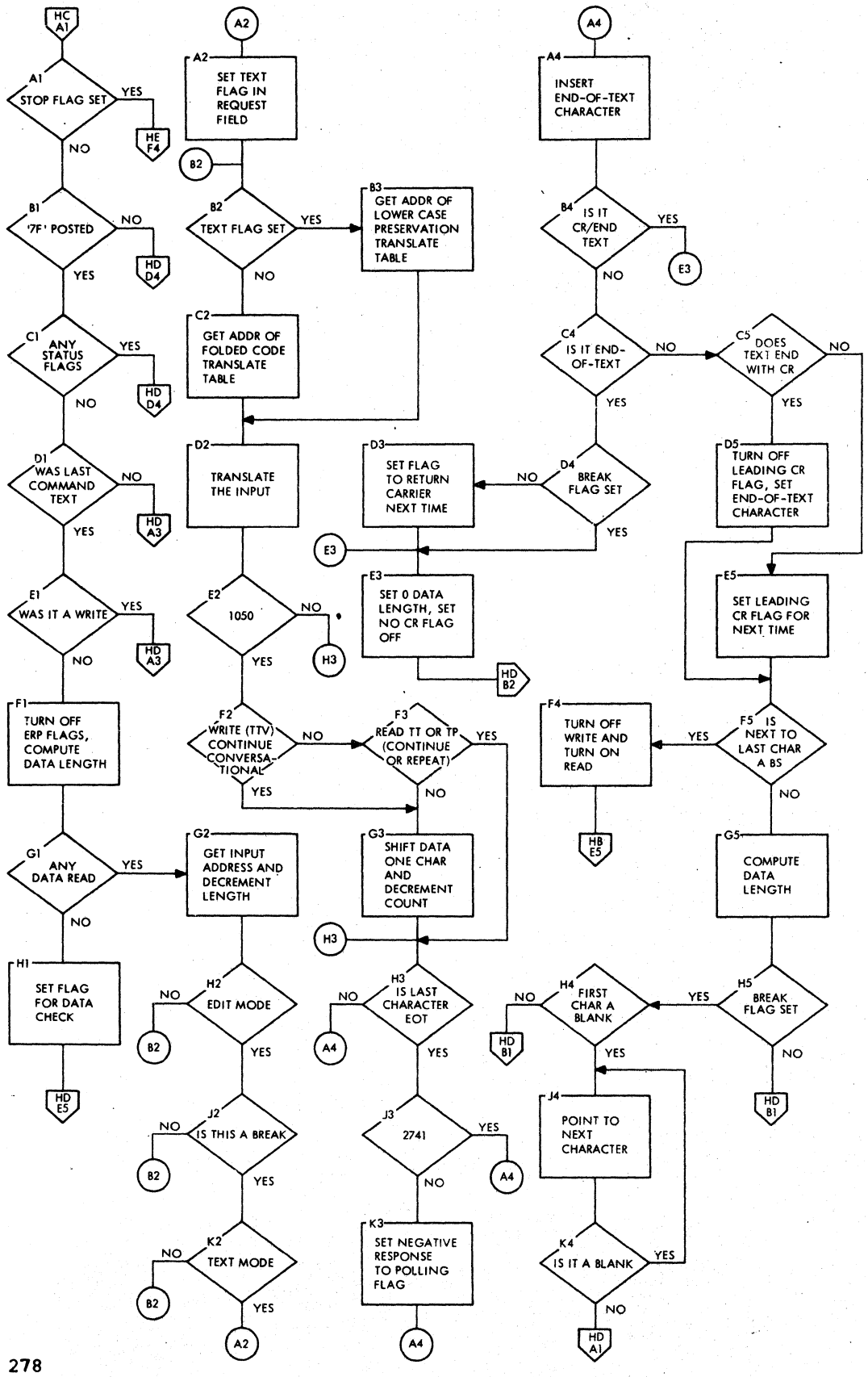


Chart HD. Communication Line Administrator Module (IHKLAD)

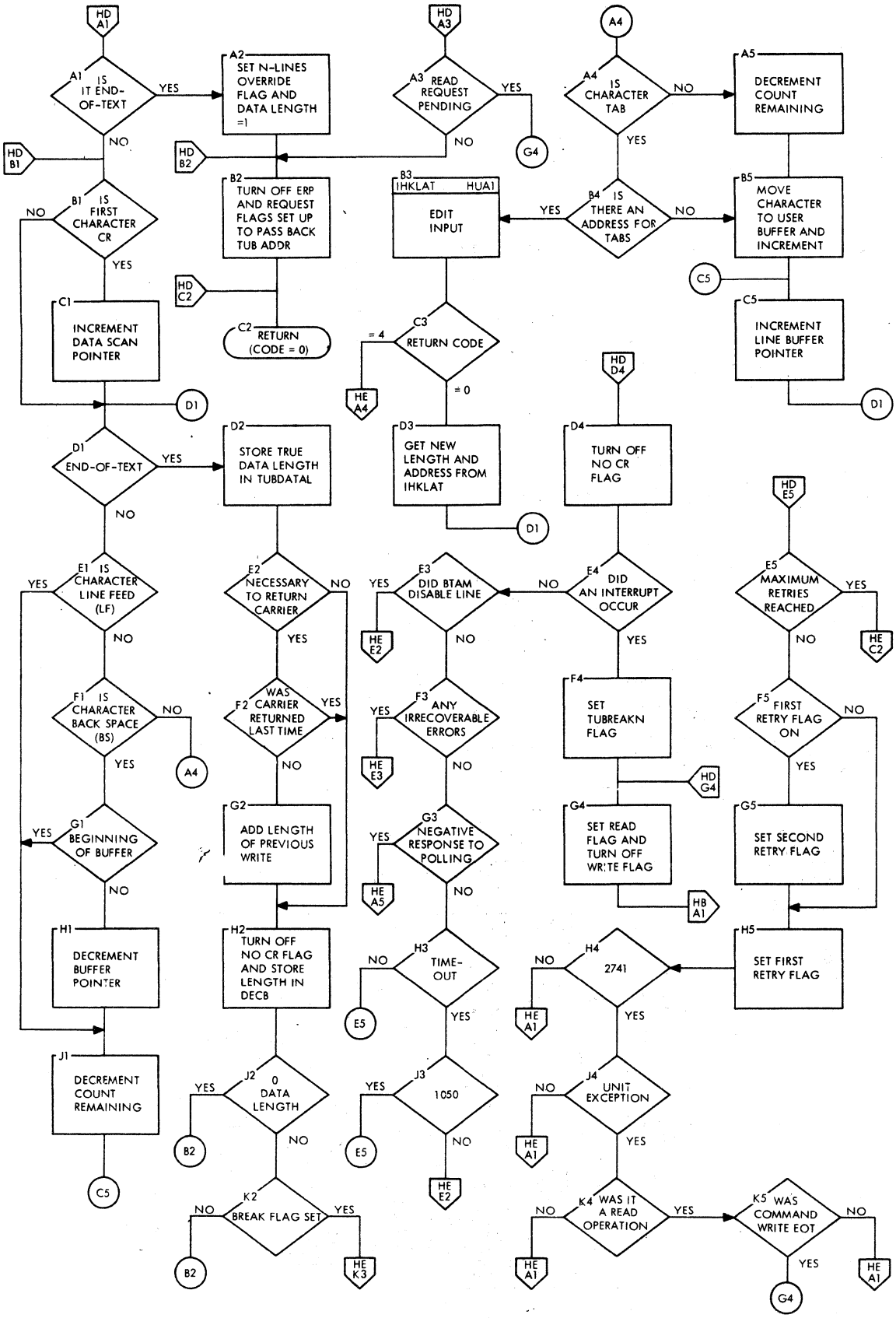


Chart HE. Communication Line Administrator Module (IHKLAD)

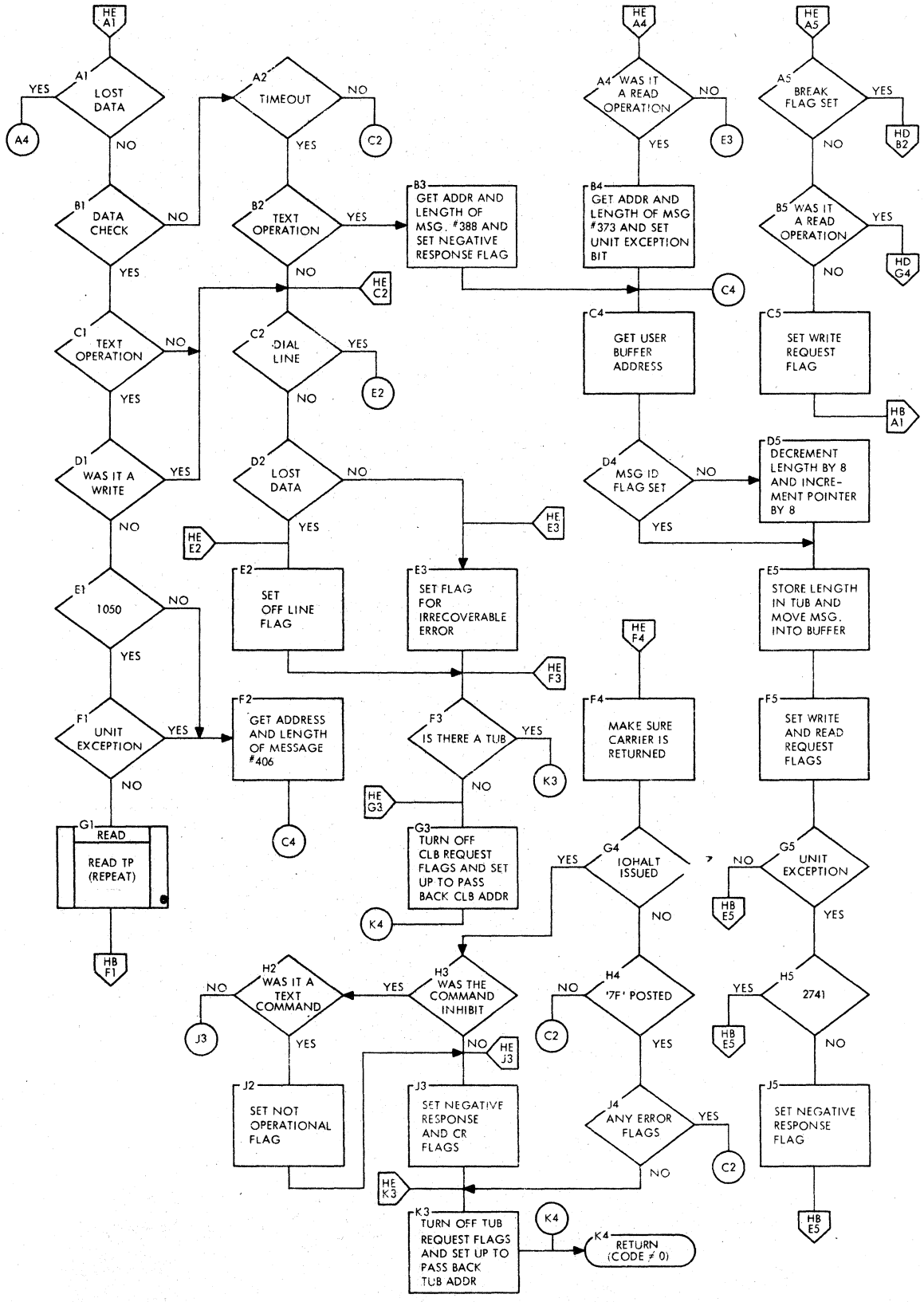


Chart HK. Input/Output Operation Initiation Module (IHKLAP)

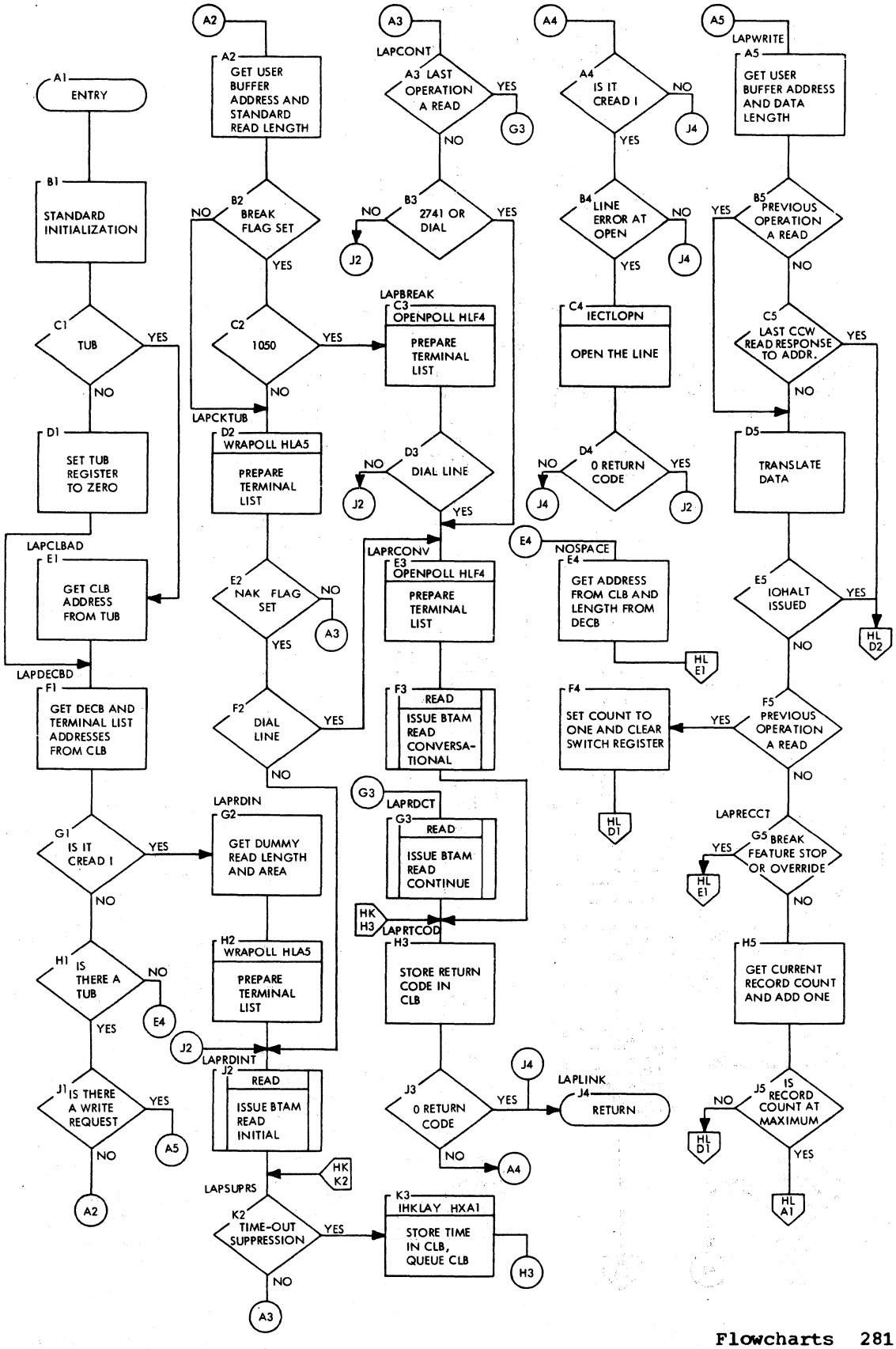


Chart HL. Input/Output Operation Initiation Module (IHKLAP)

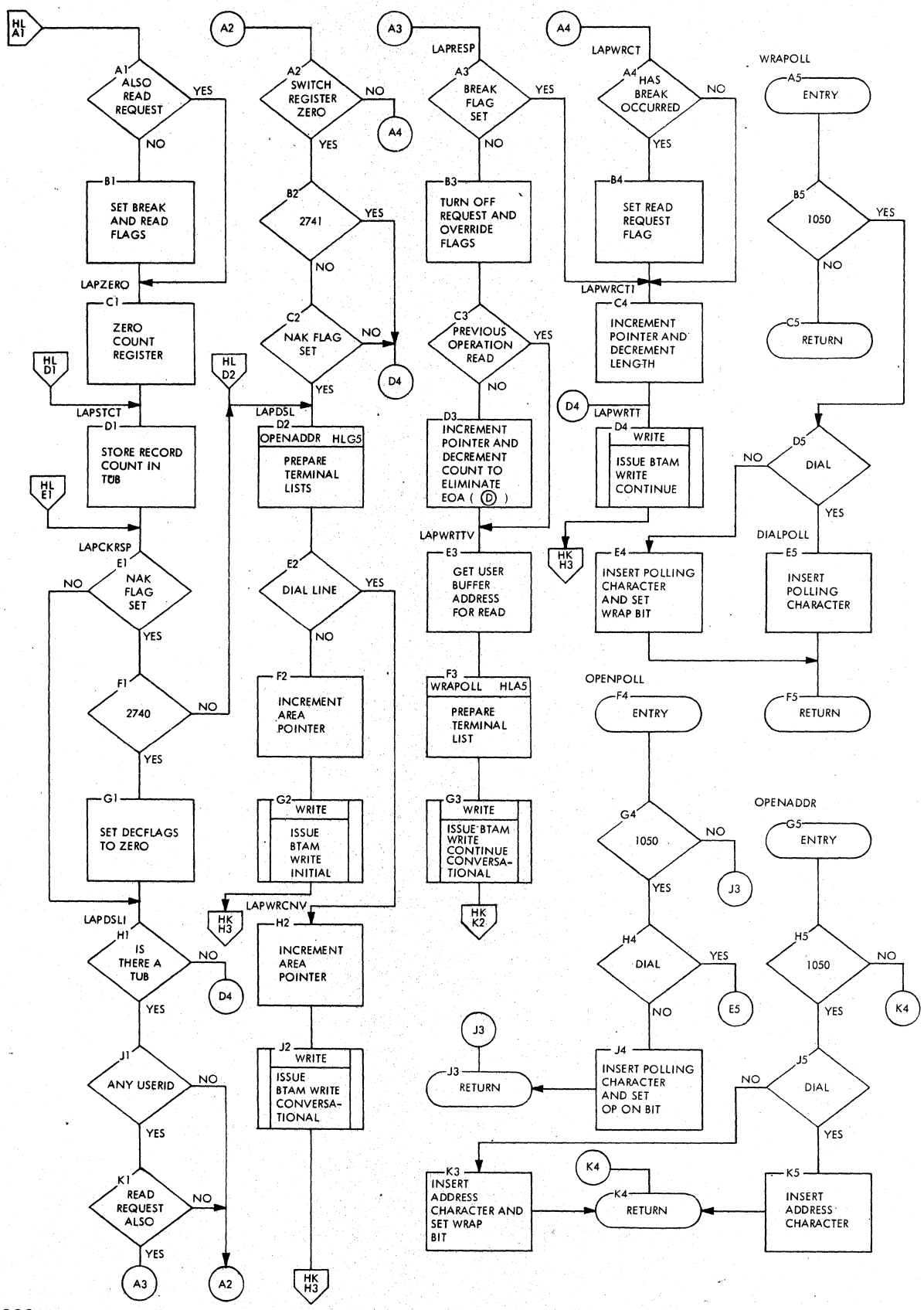


Chart HP. Output Text Formatting Module (IHKLAB)

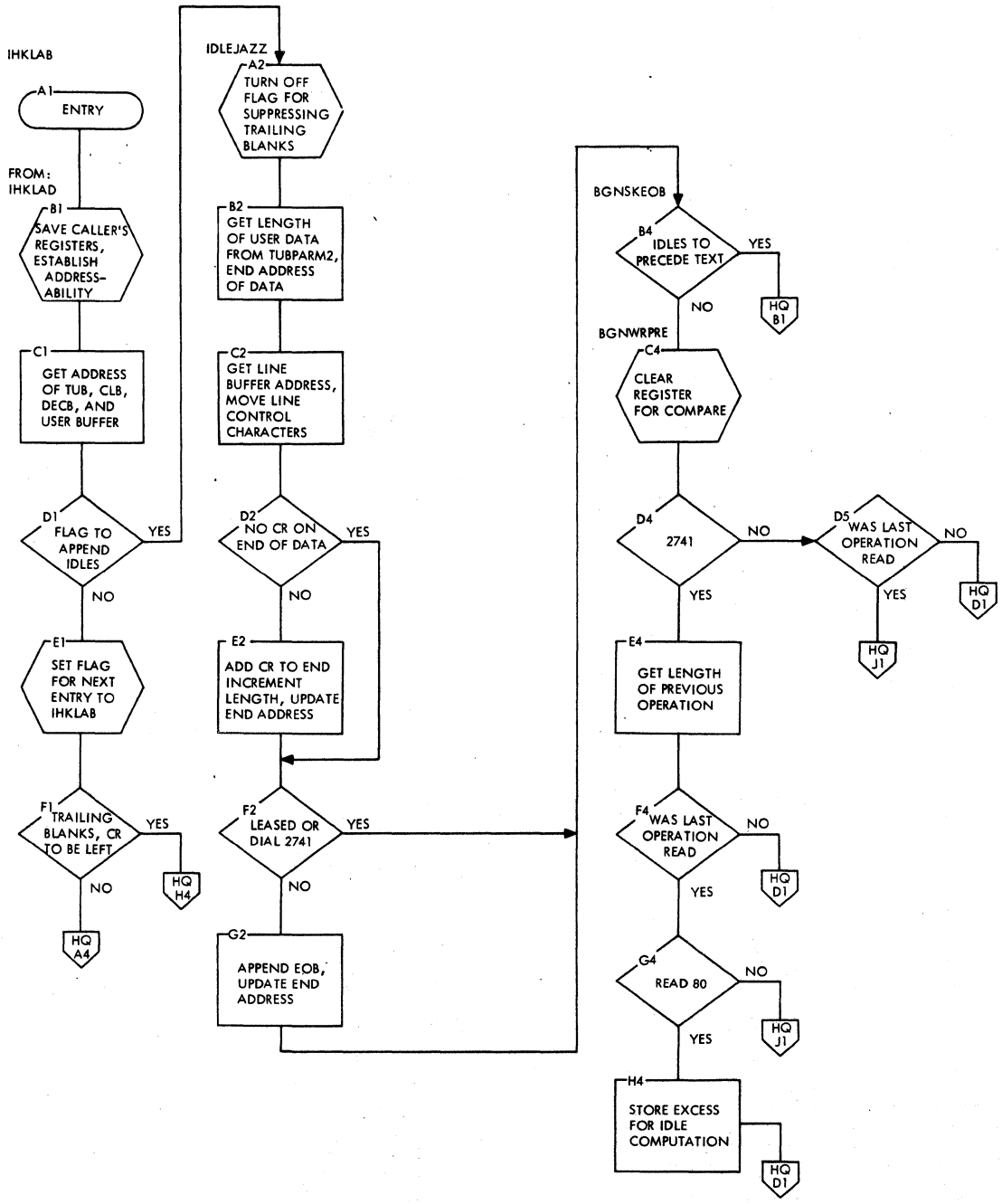


Chart HQ. Output Text Formatting Module (IHKLAB)

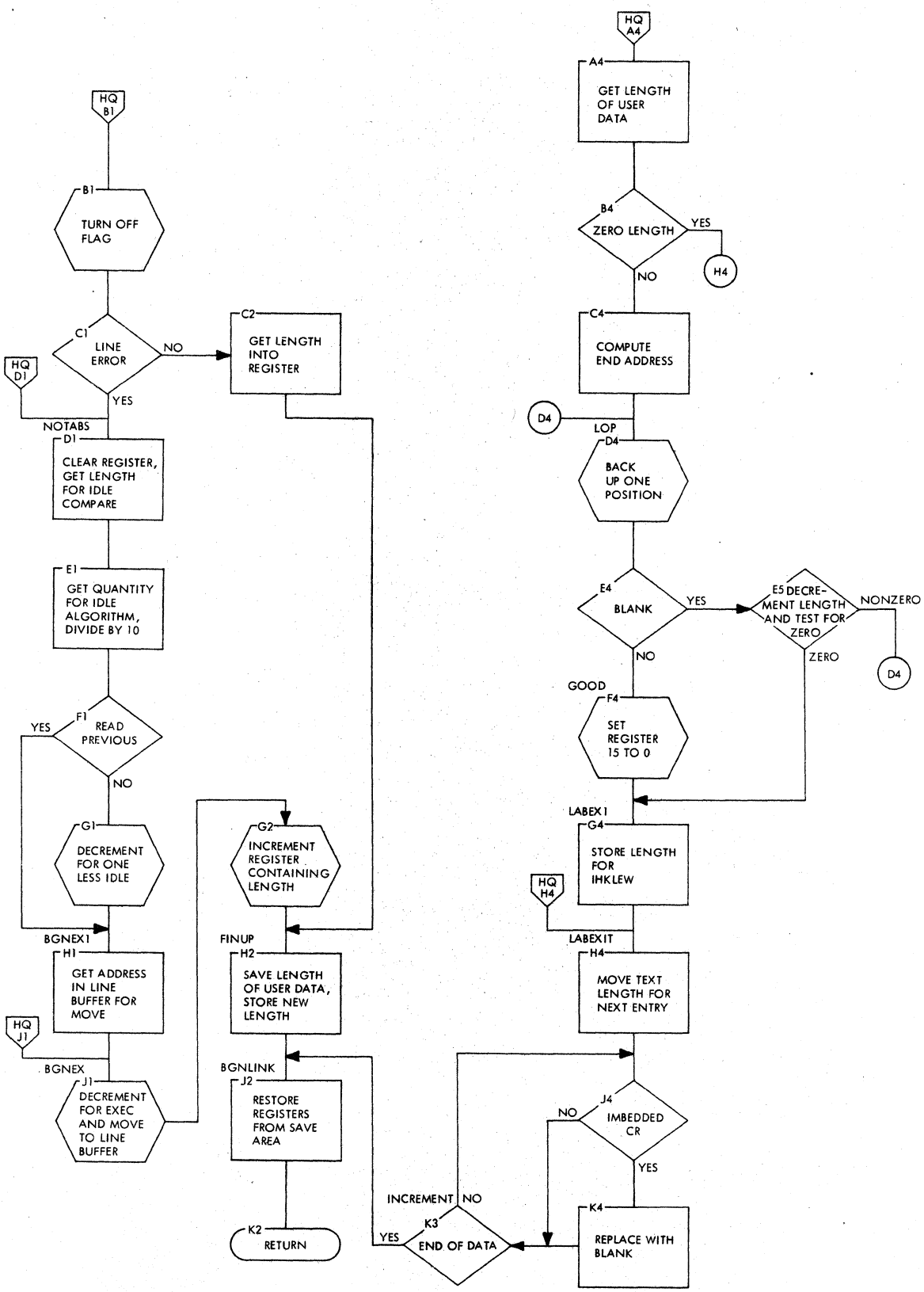


Chart HU. TABSET Edit Module (IHKLAT)

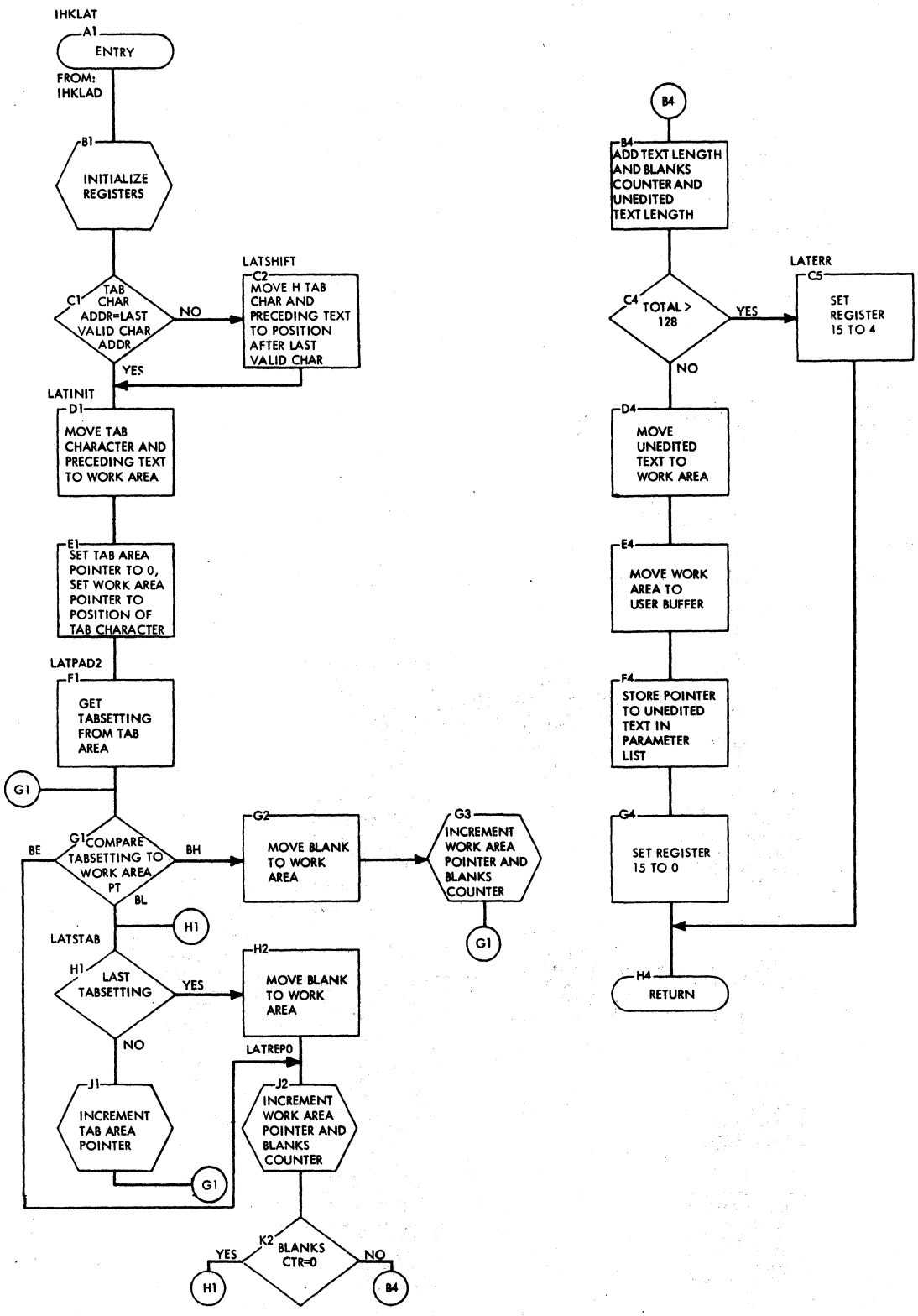


Chart HW. Line Edit Write Module (IHKLEW)

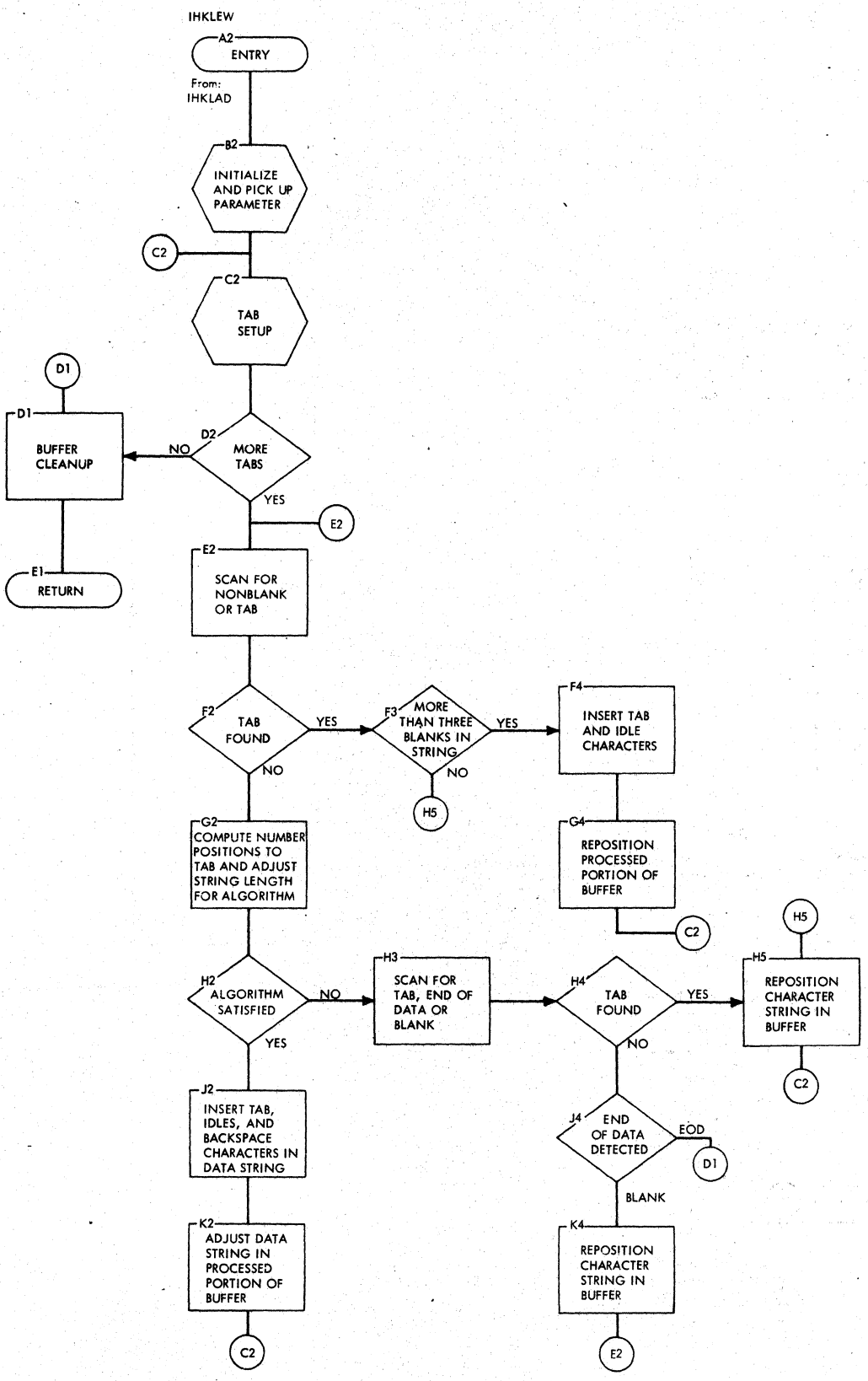


Chart HX. 1050X Programmed Time-Out Module (IHKLAY)

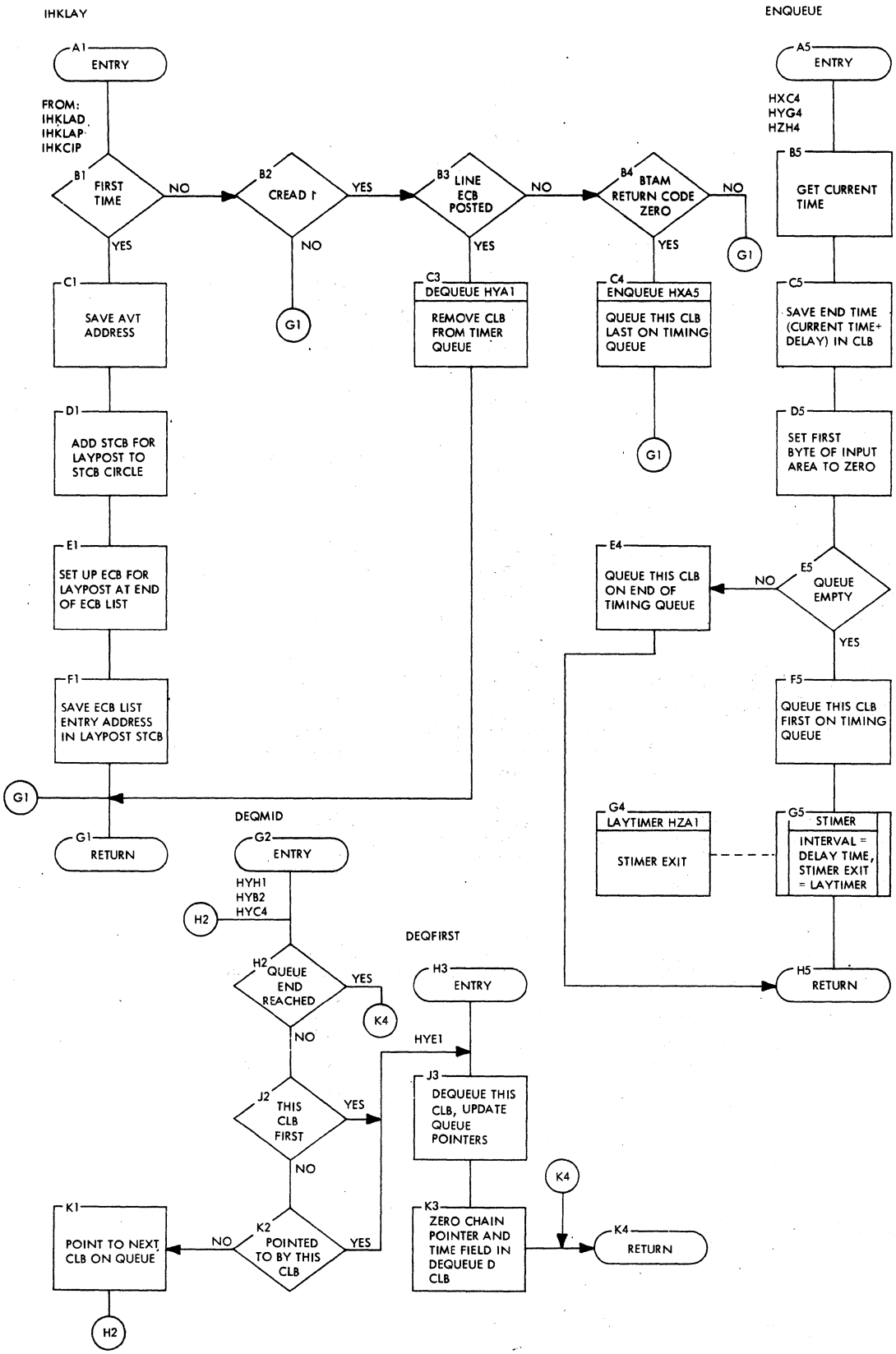


Chart HY. 1050X Programmed Time-Out Module (IHKLAY)

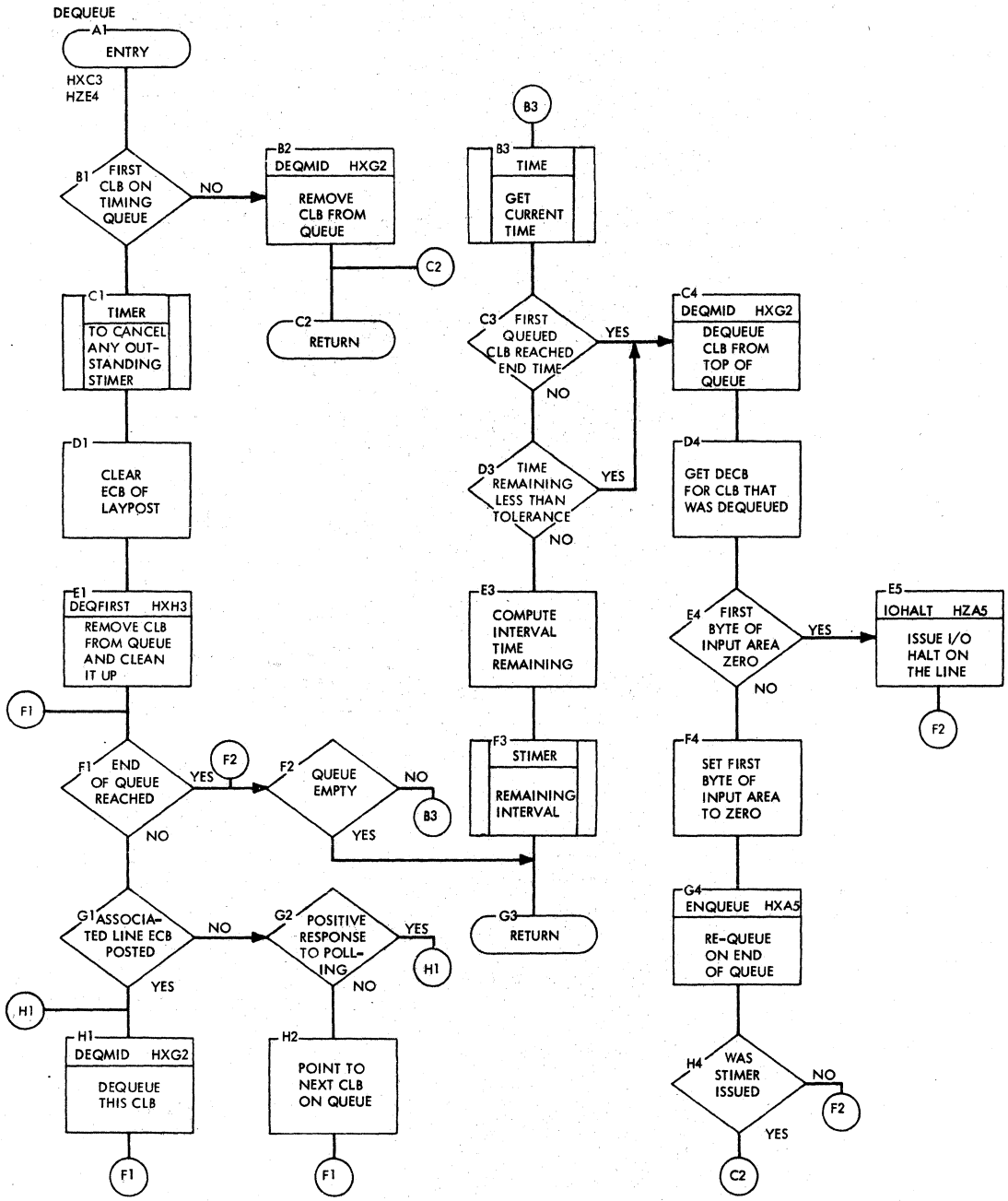


Chart HZ. 1050X Programmed Time-Out Module (IHKLAY)

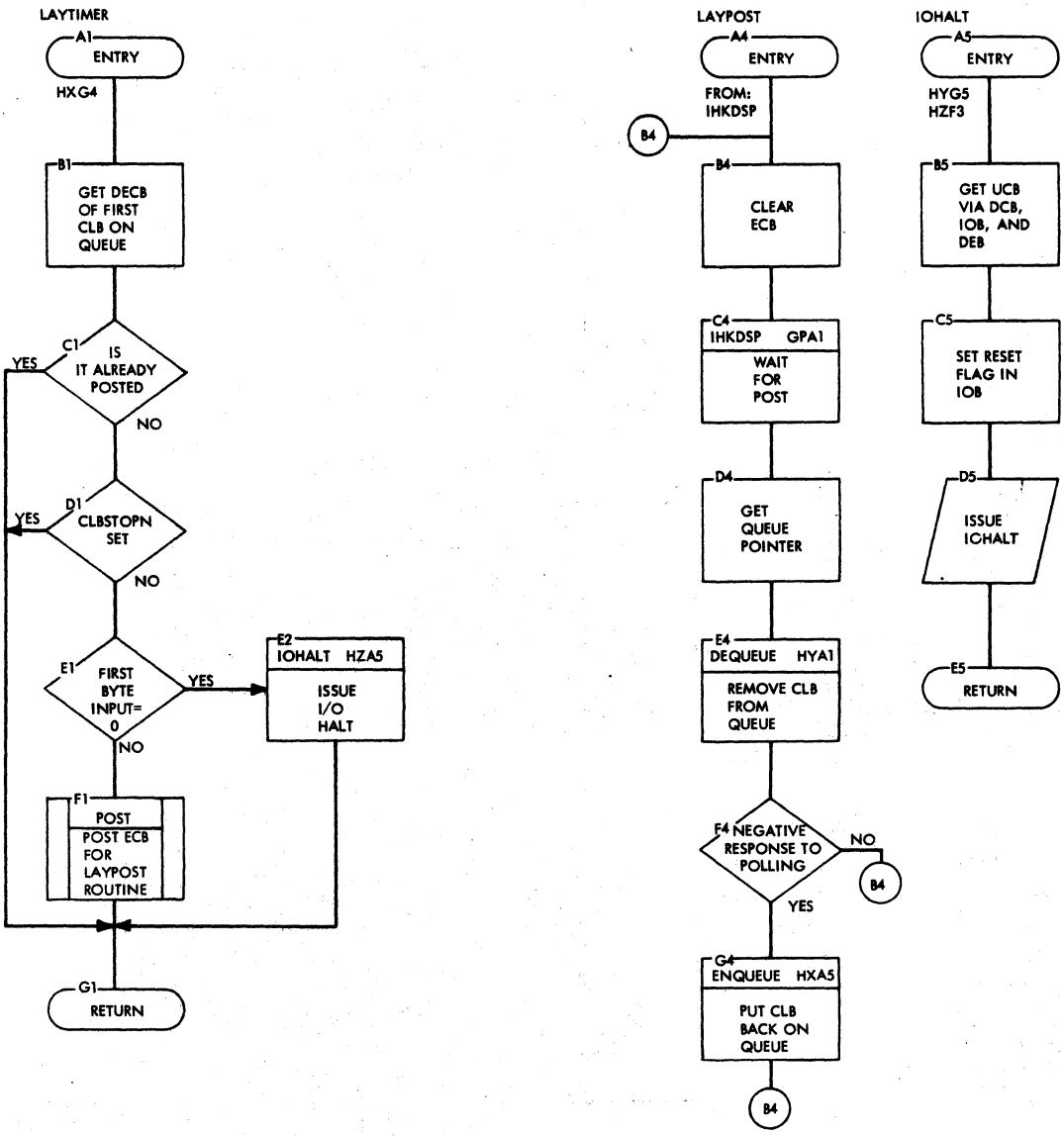


Chart MA. CHANGE Subcommand Processor (IHKCGN)

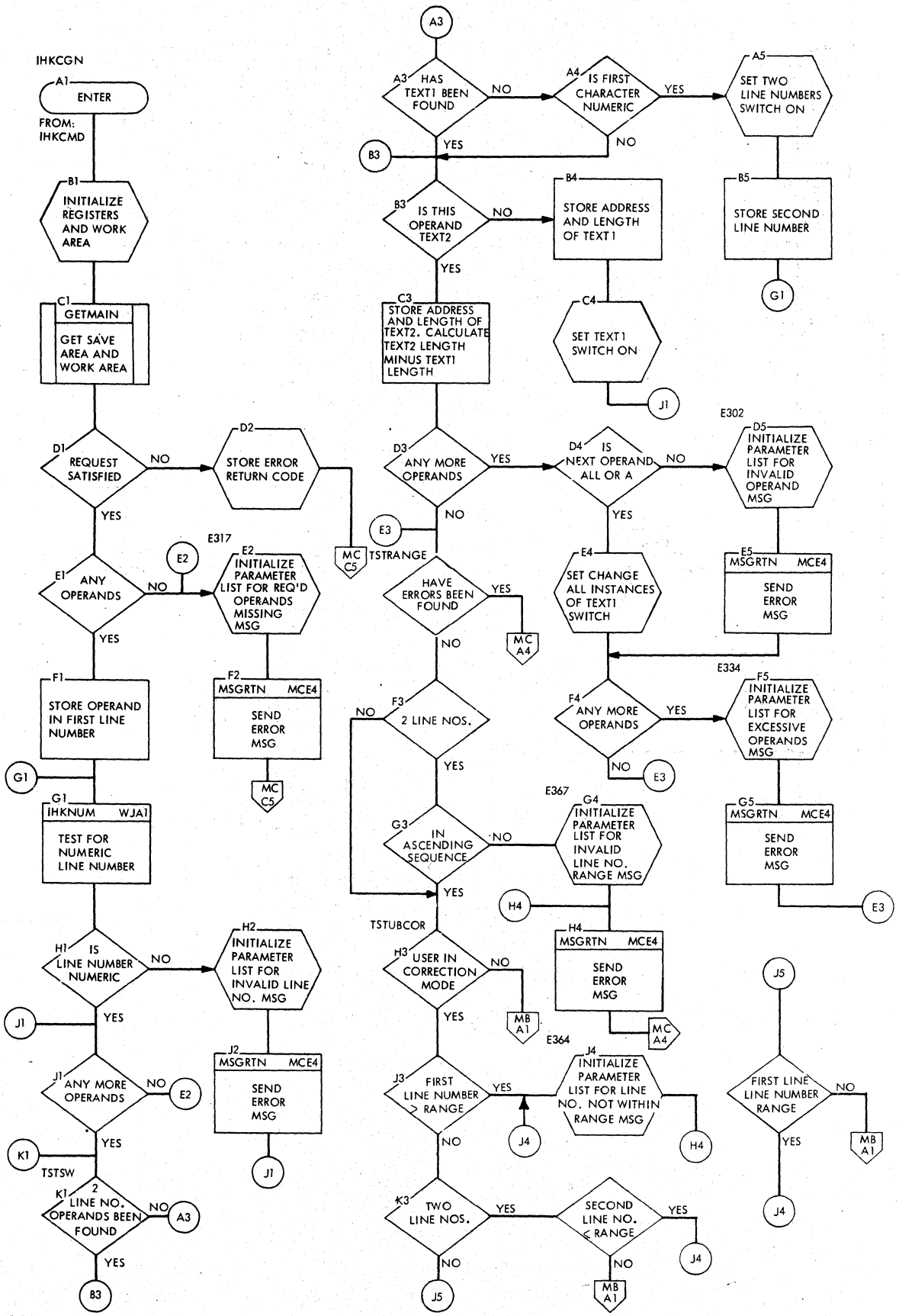


Chart MB. CHANGE Subcommand Processor (IHKCGN)

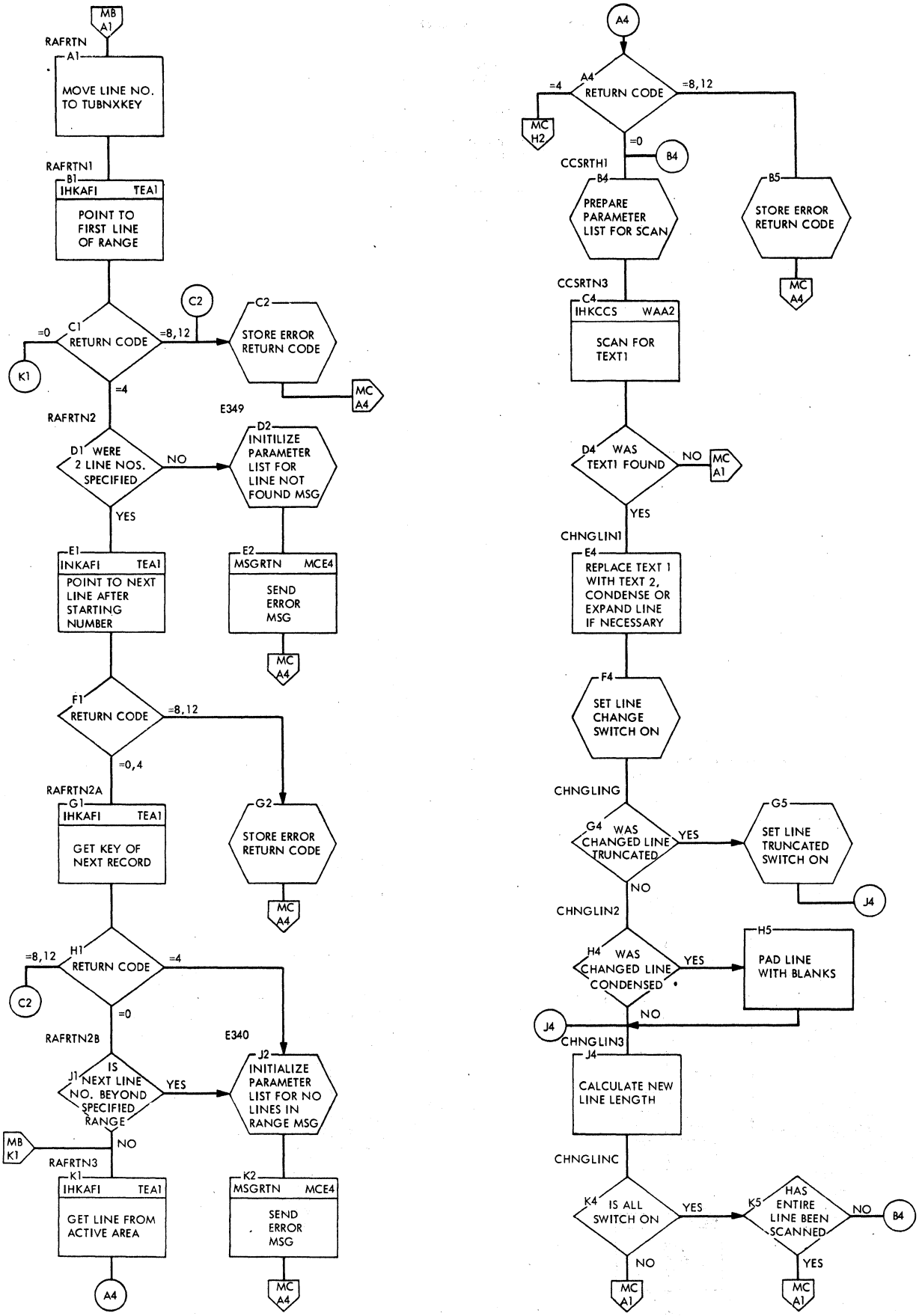


Chart MC. CHANGE Subcommand Processor (IHKCGN)

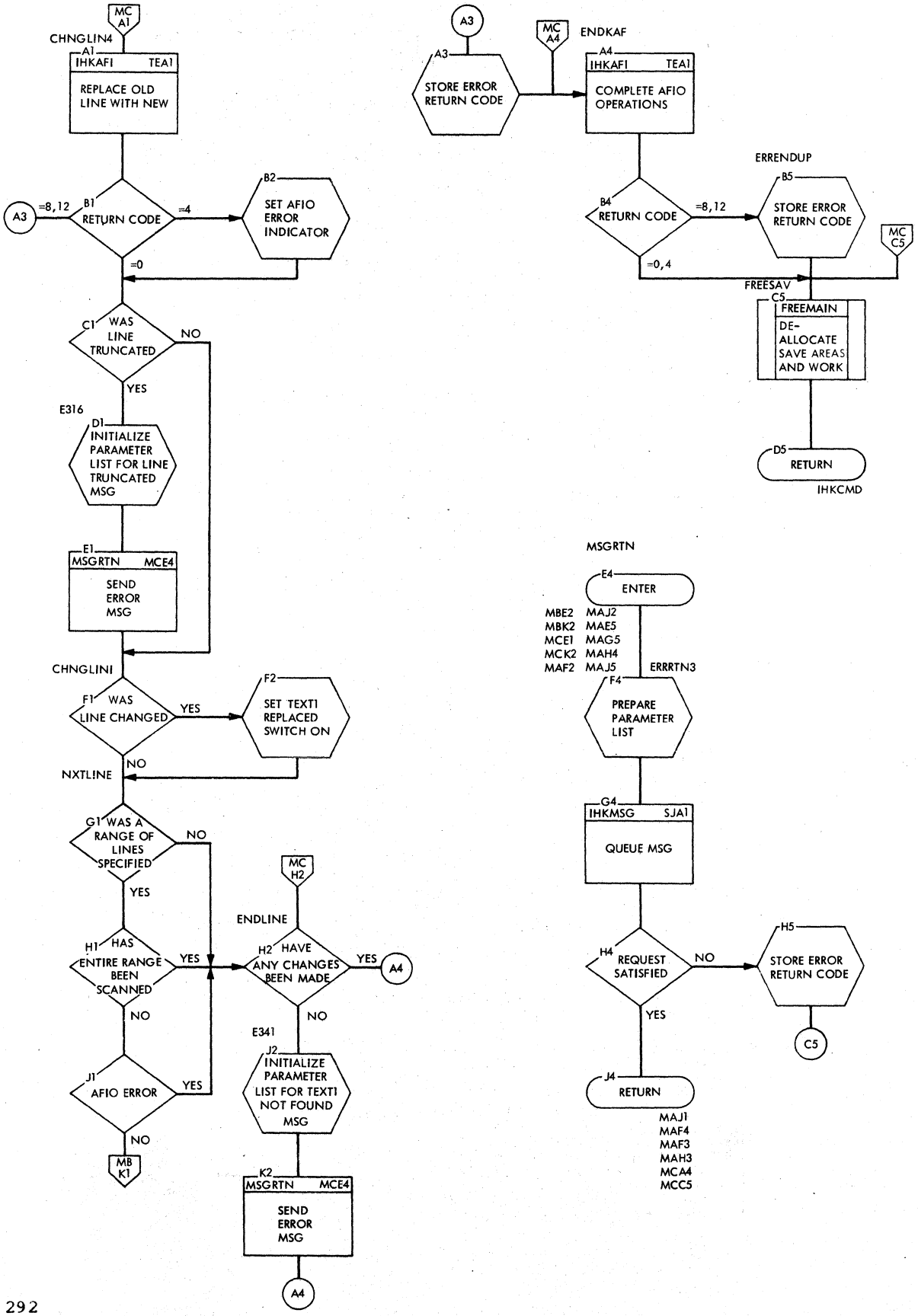


Chart MG. EDIT, DELETE, and EXEC Command Processor (IHKEDT)

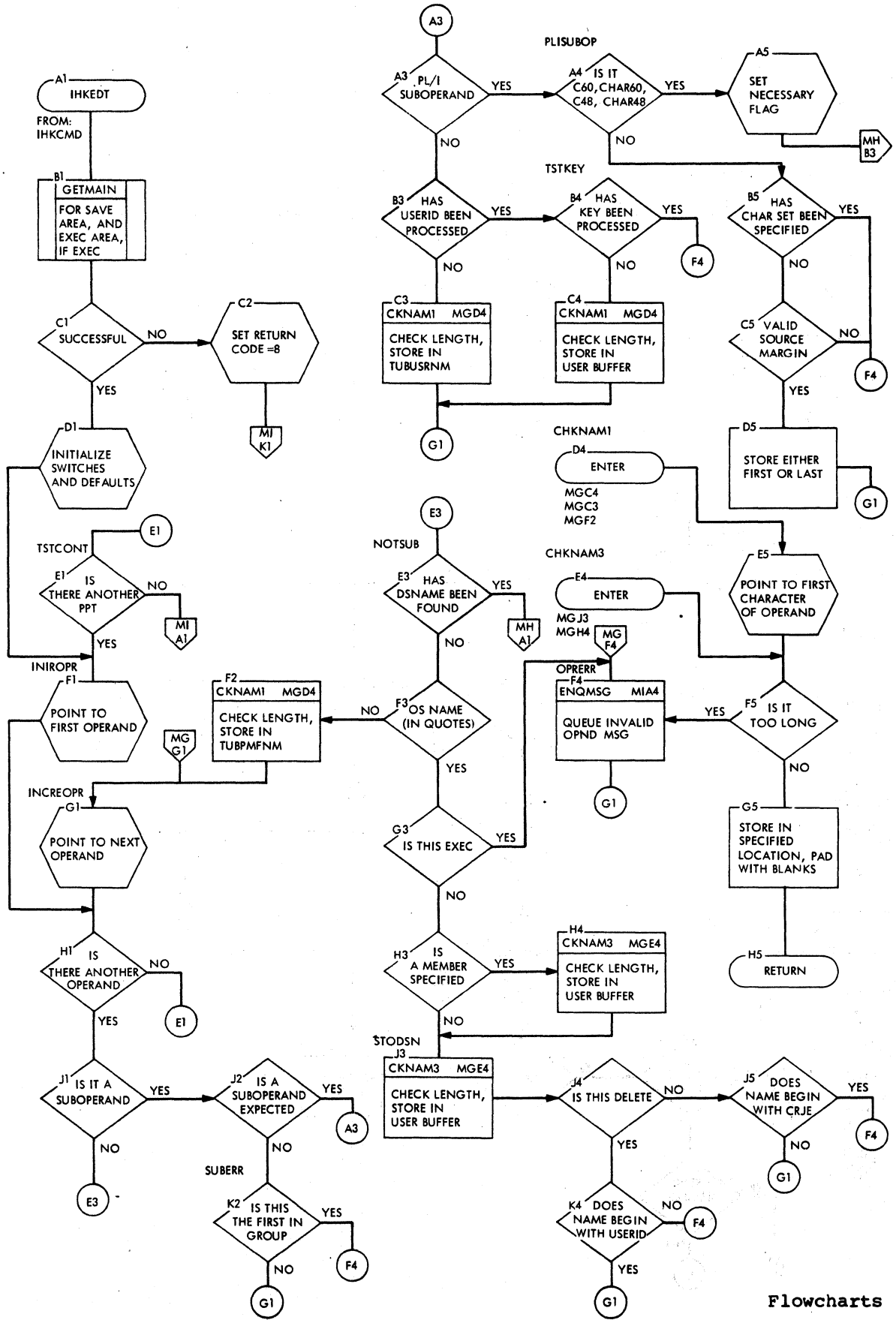


Chart MH. EDIT, DELETE, and EXEC Command Processor (IHKEDT)

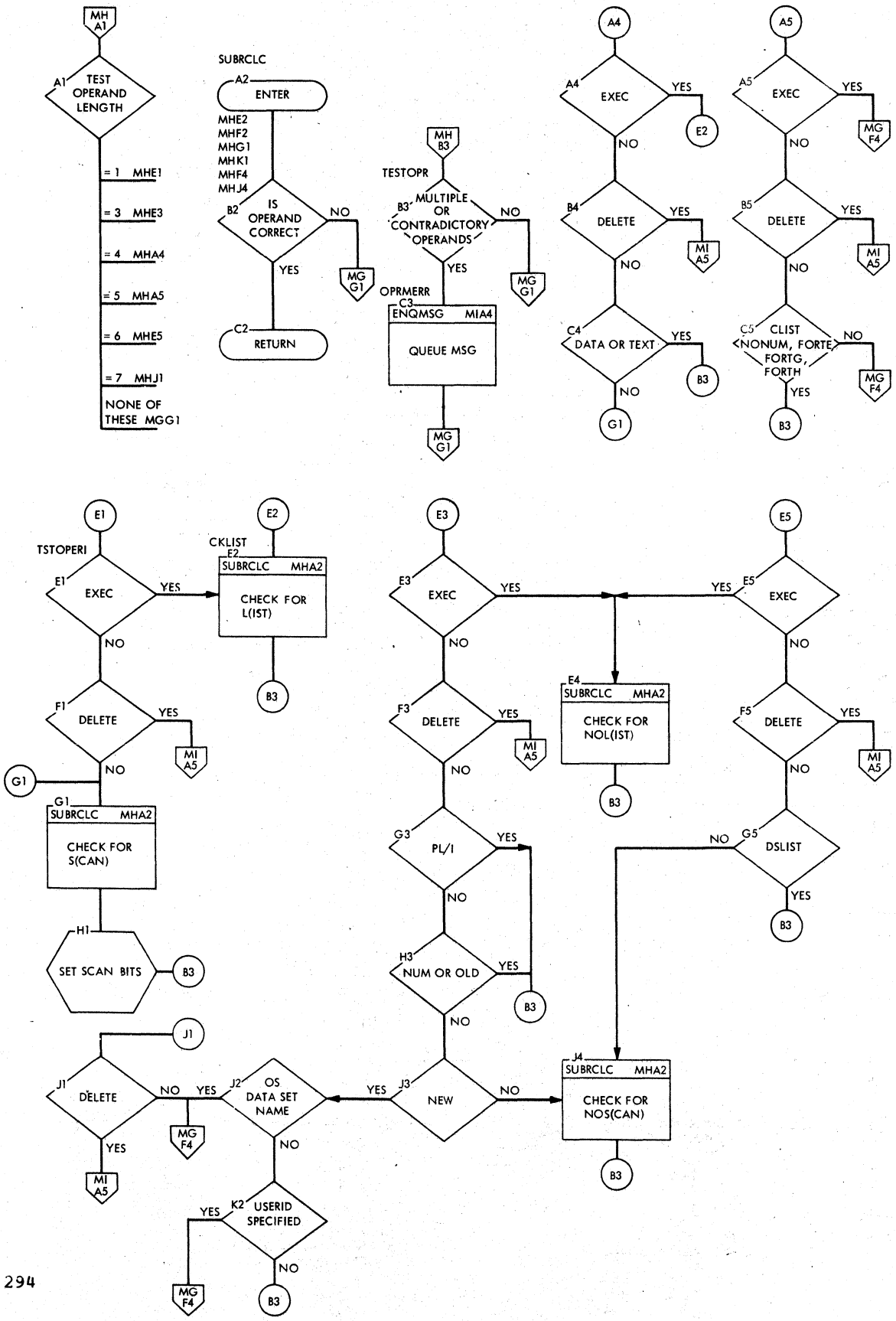
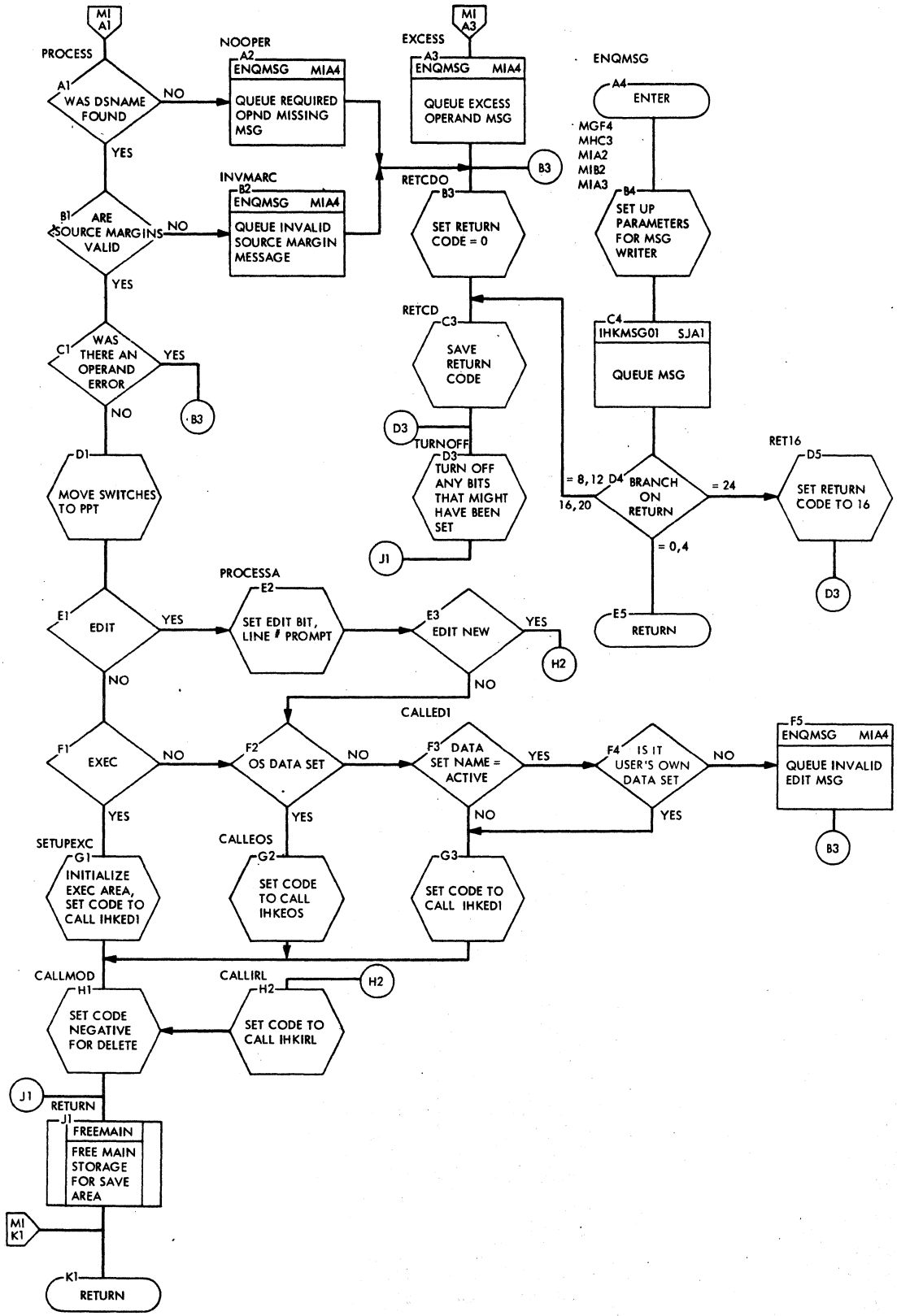


Chart MI. EDIT, DELETE, and EXEC Command Processor (IHKEDT)



• Chart MJ. EDIT, DELETE, and EXEC Command Processor (IHKED1)

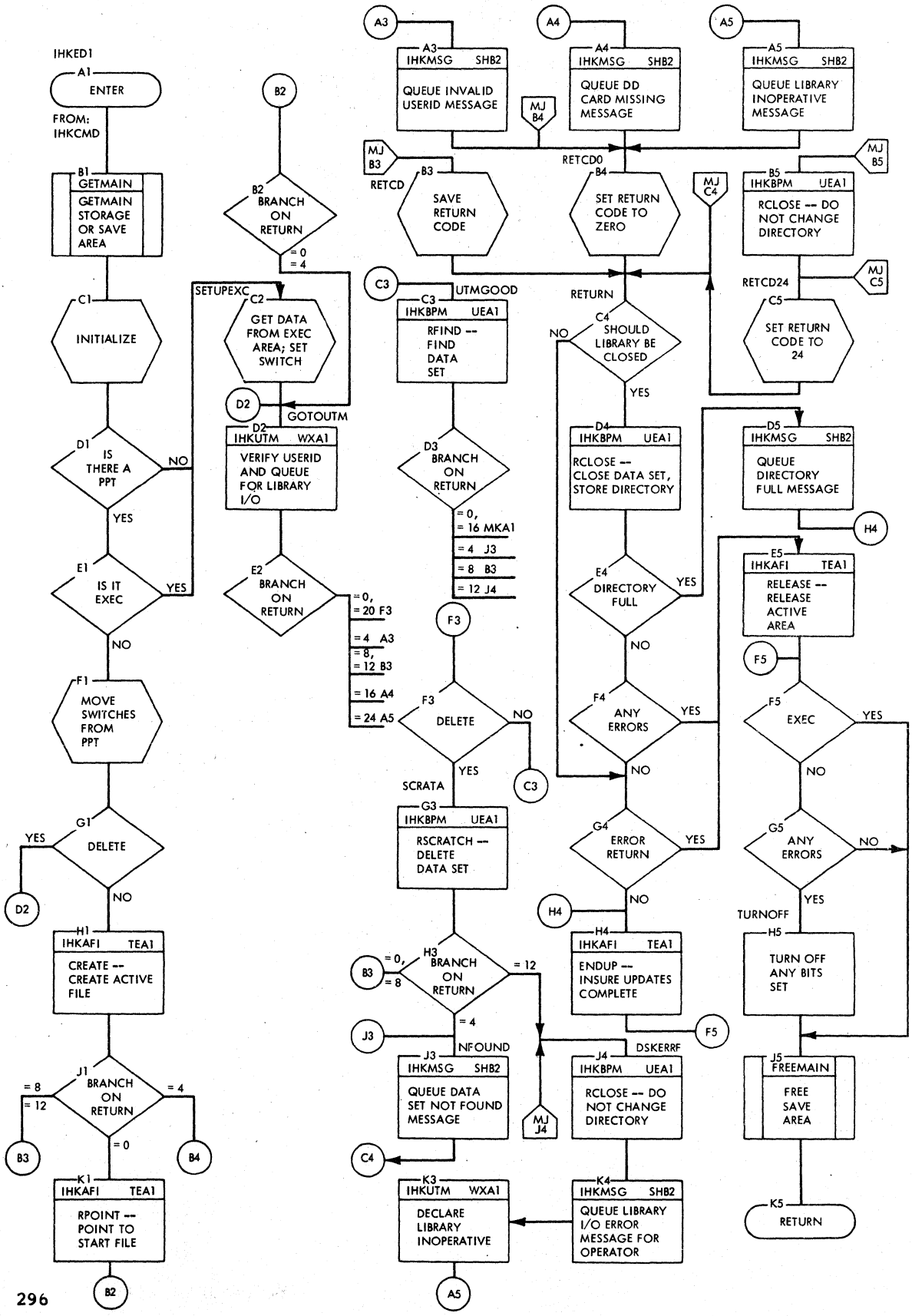
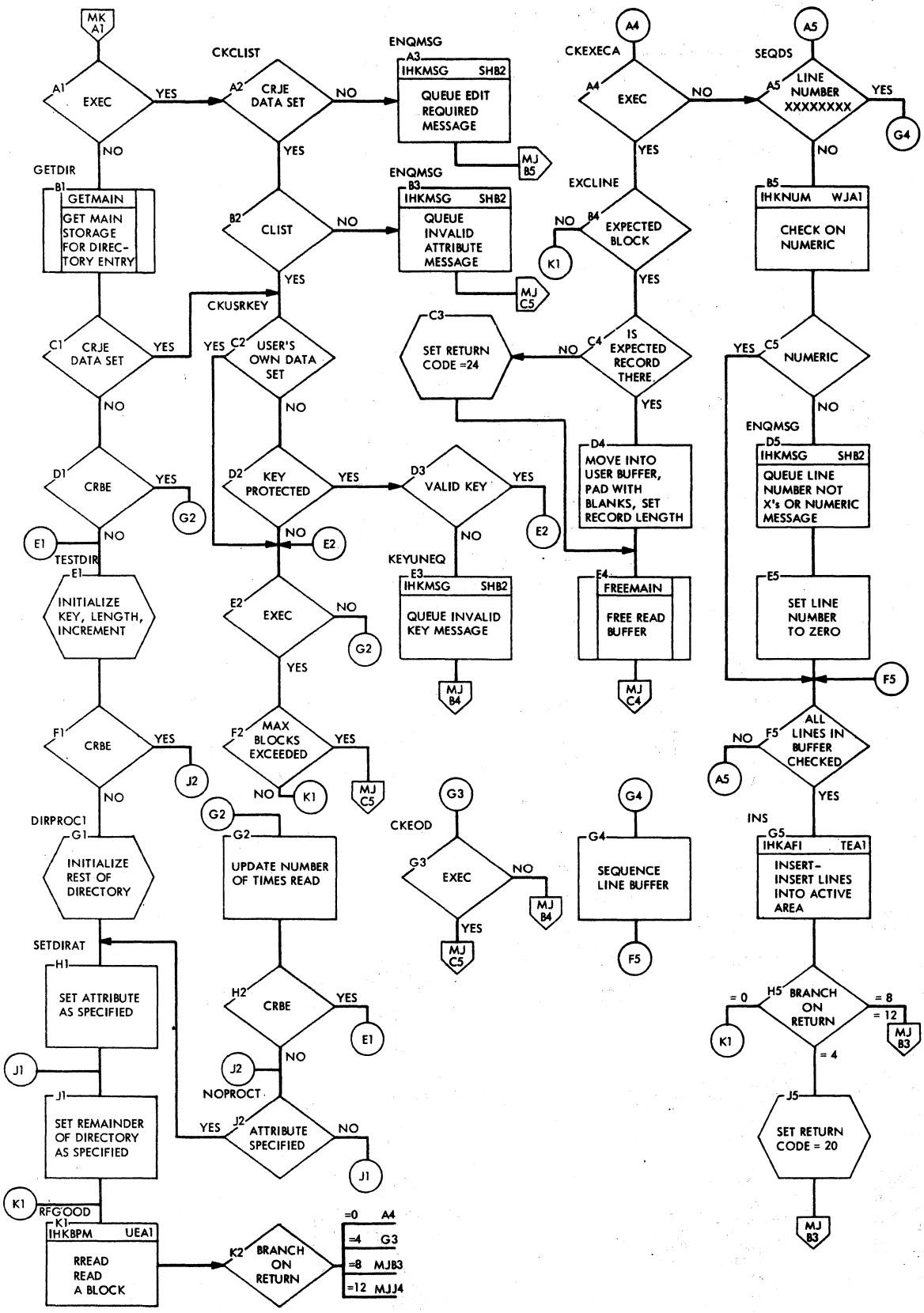
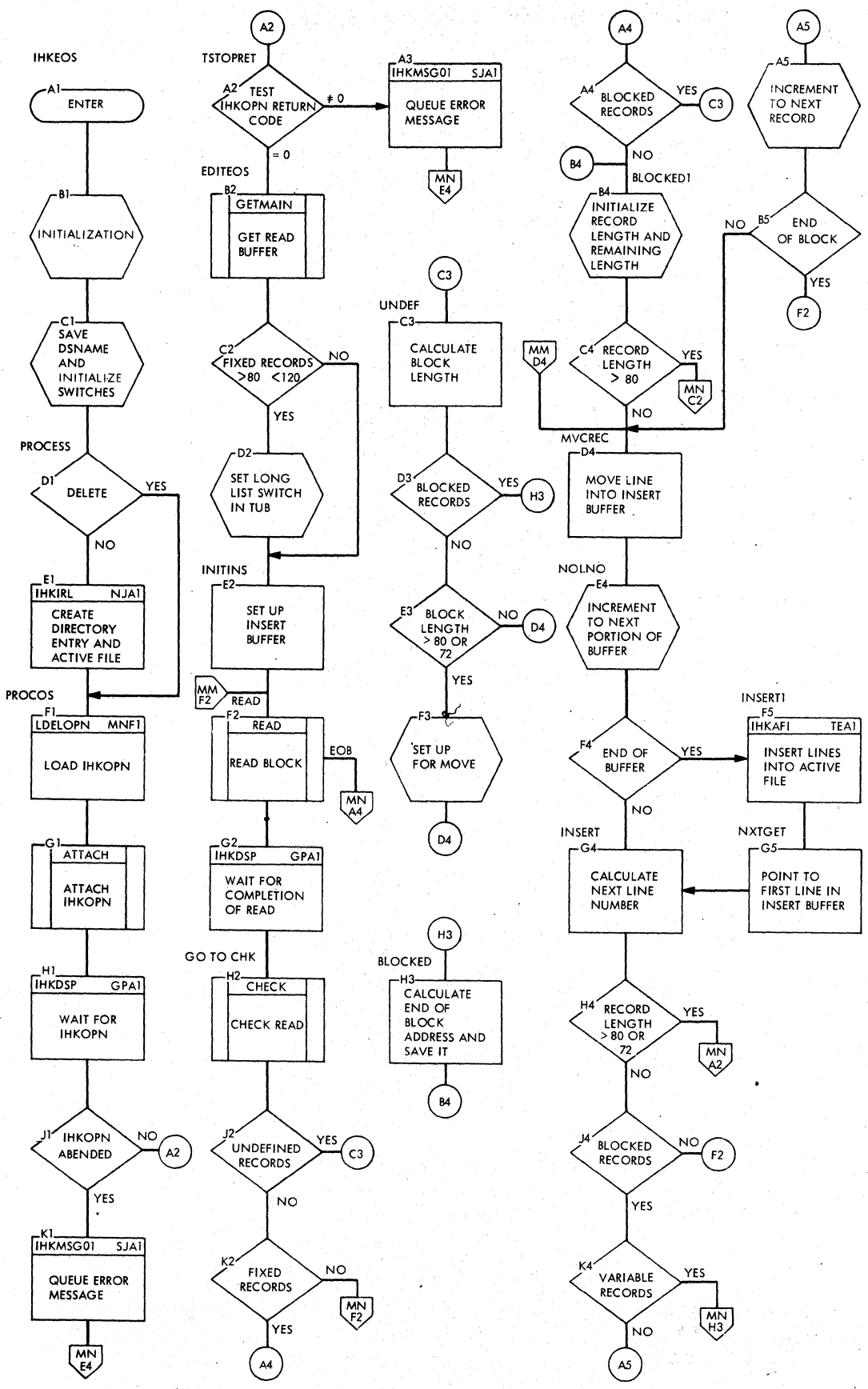


Chart MK. EDIT, DELETE, and EXEC Command Processor (IHKED1)



• Chart MM. EDIT Command Processor (IHKEOS)



• Chart MN. EDIT Command Processor (IHKEOS)

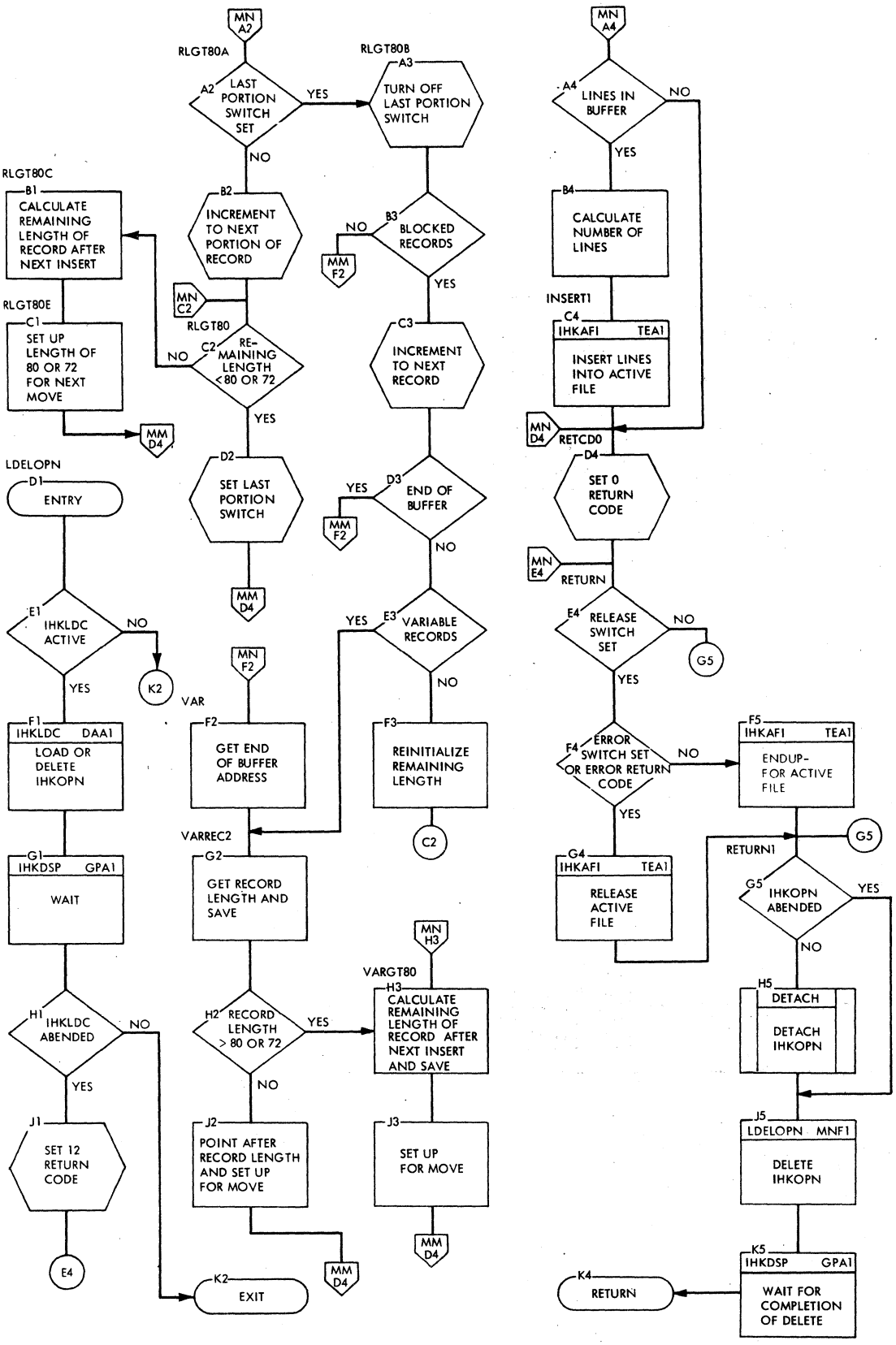


Chart MW. END Subcommand Processor (IHKEND)

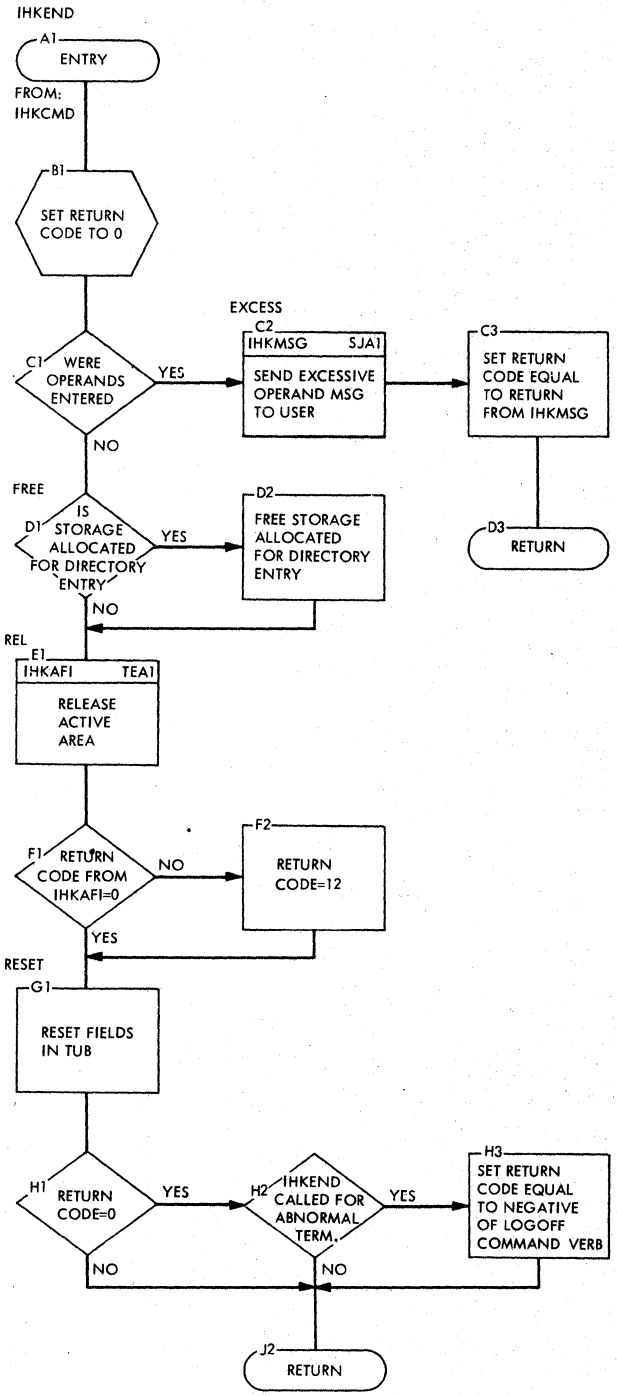


Chart NA. INPUT Subcommand Processor (IHKIPT)

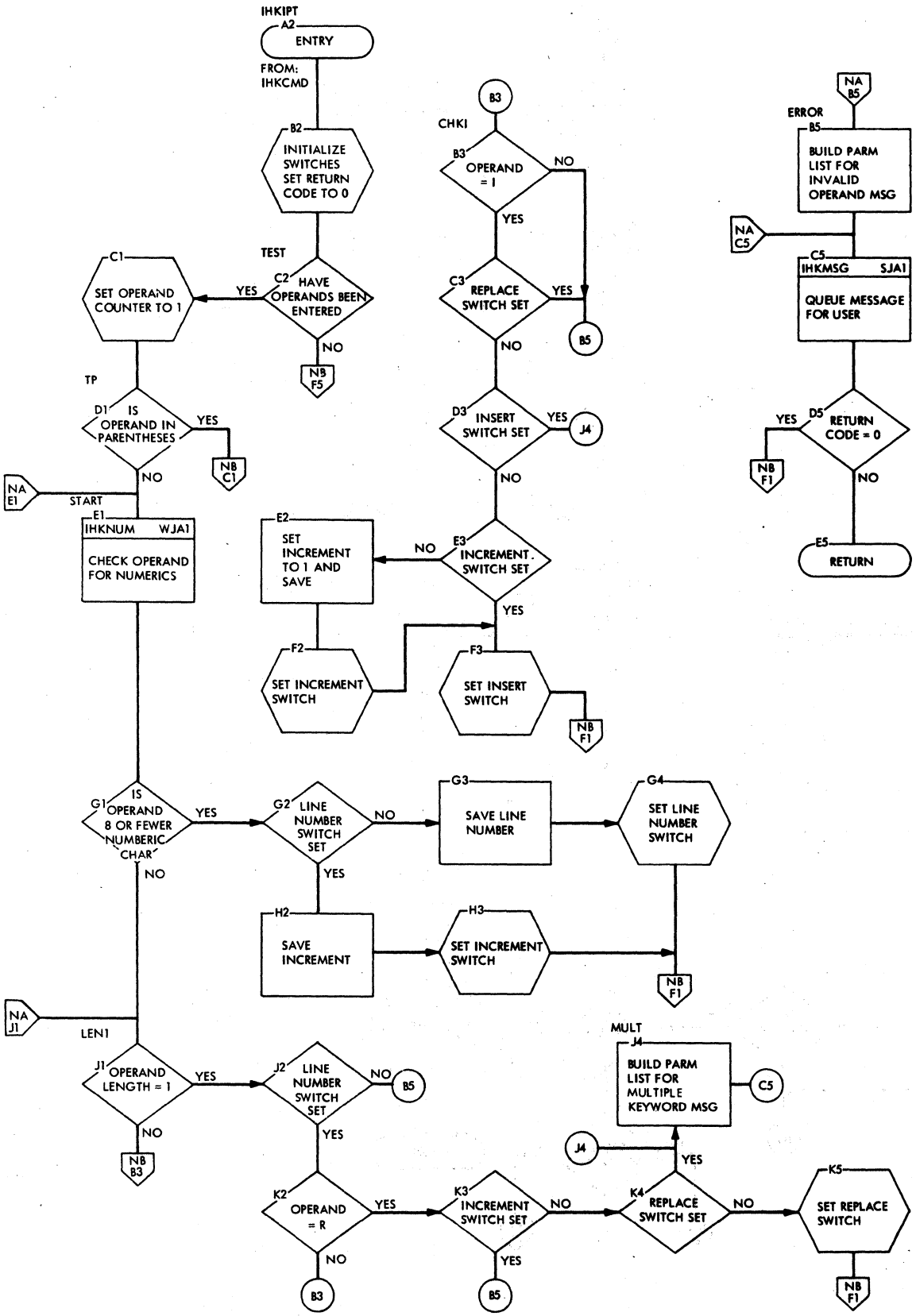


Chart NB. INPUT Subcommand Processor (IHKIPT)

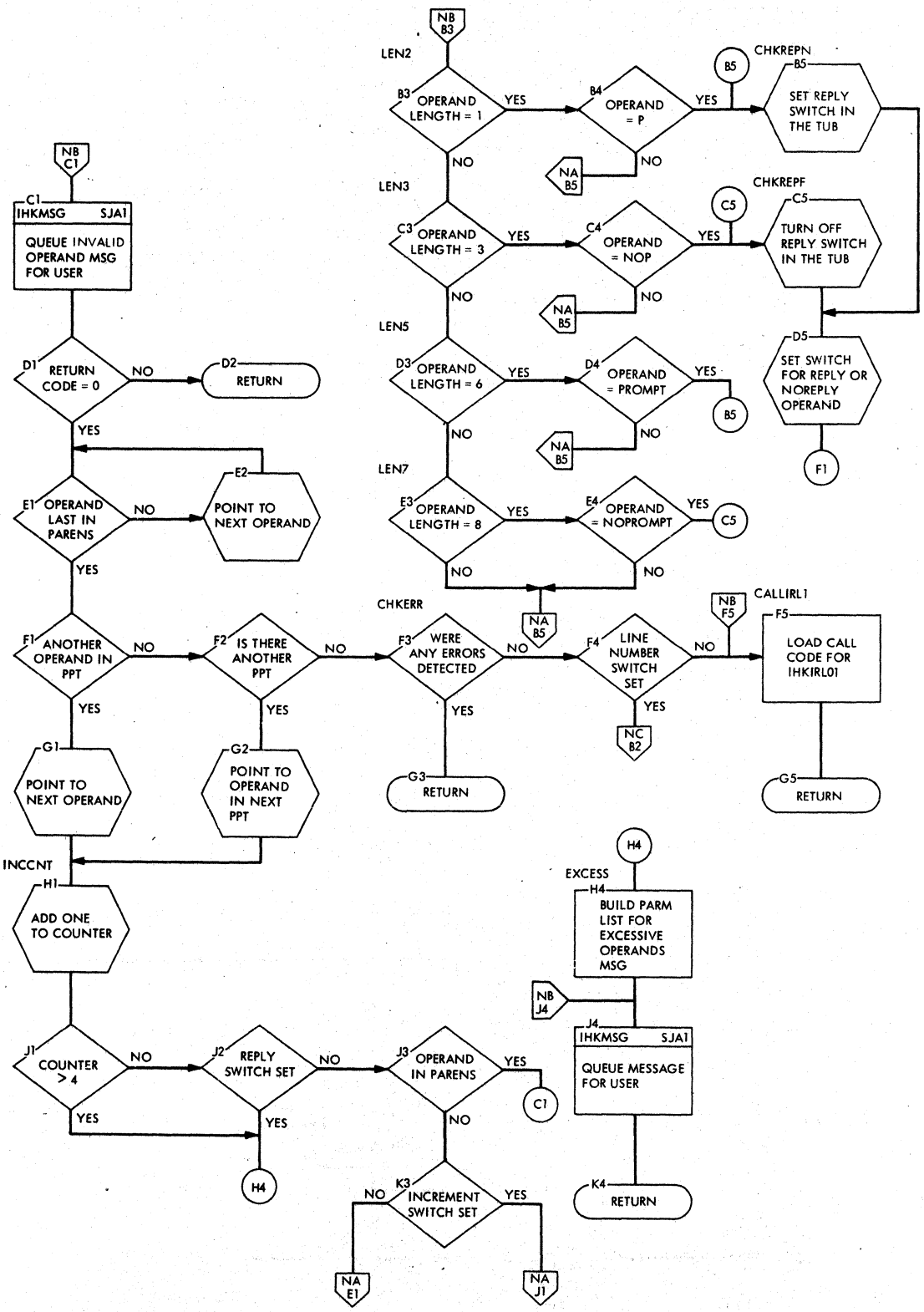


Chart NC. INPUT Subcommand Processor (IHKIPT)

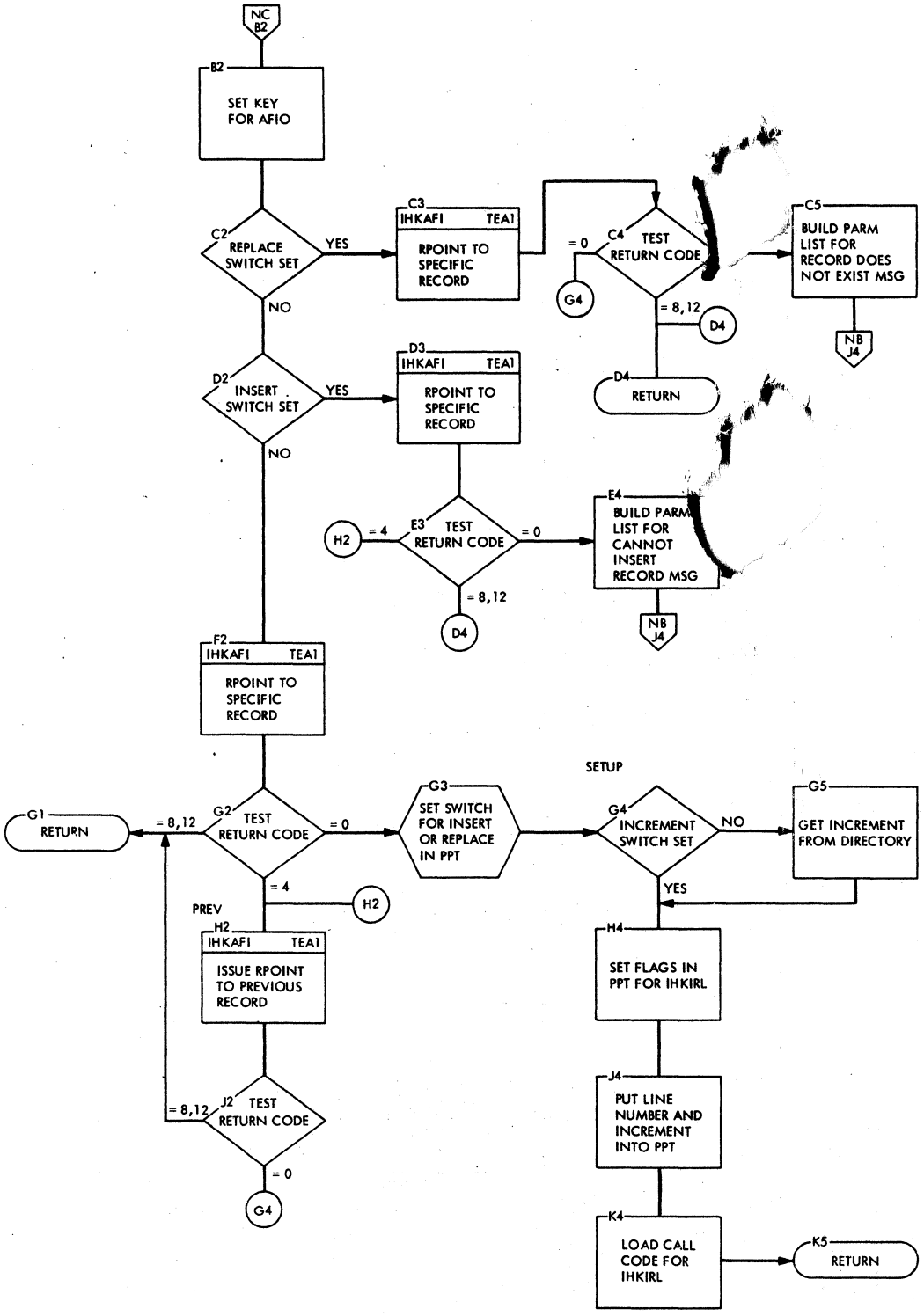


Chart NJ. Insert/Replace/Delete Processor (IHKIRL)

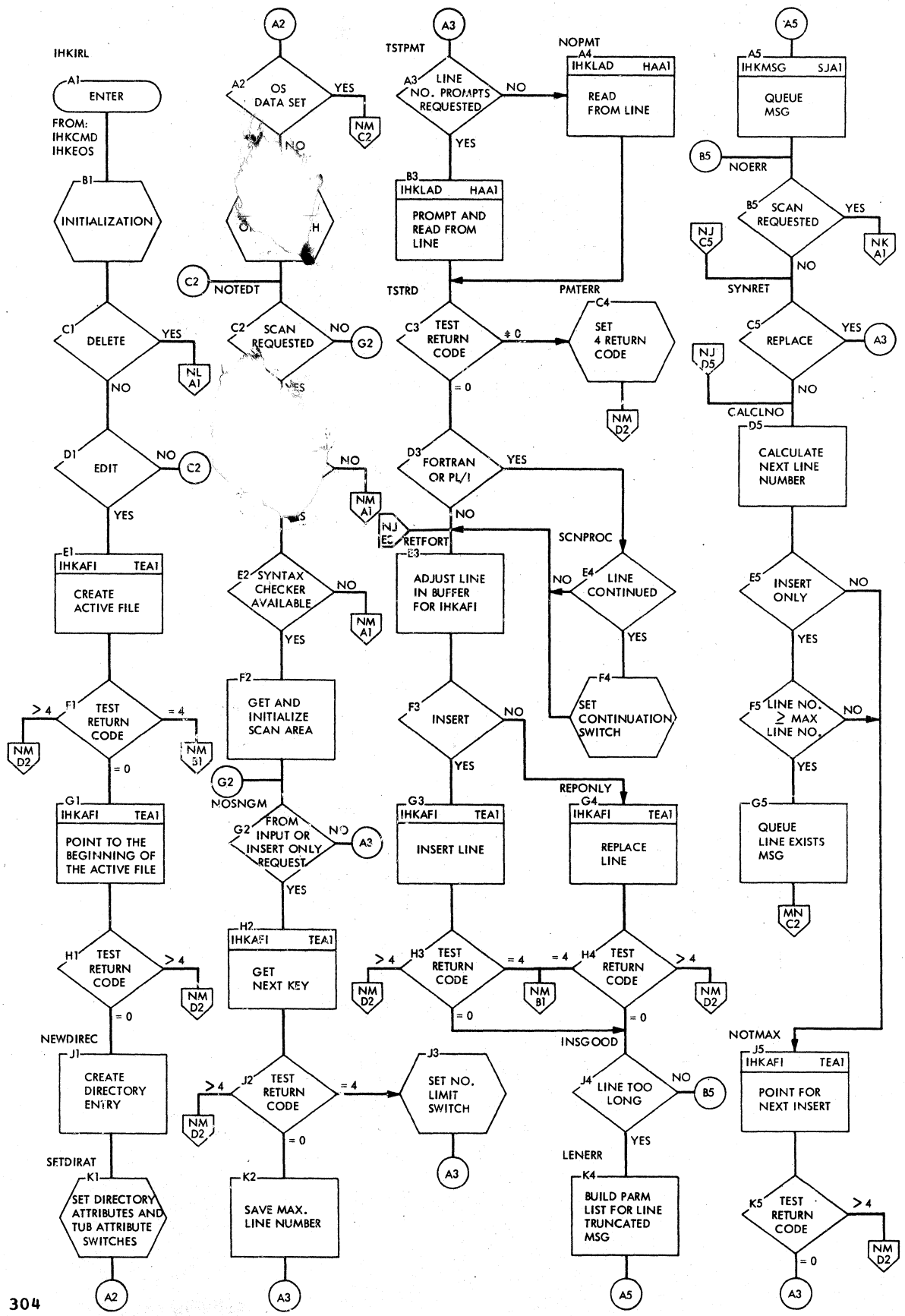


Chart NK. Insert/Replace/Delete Processor (IHKIRL)

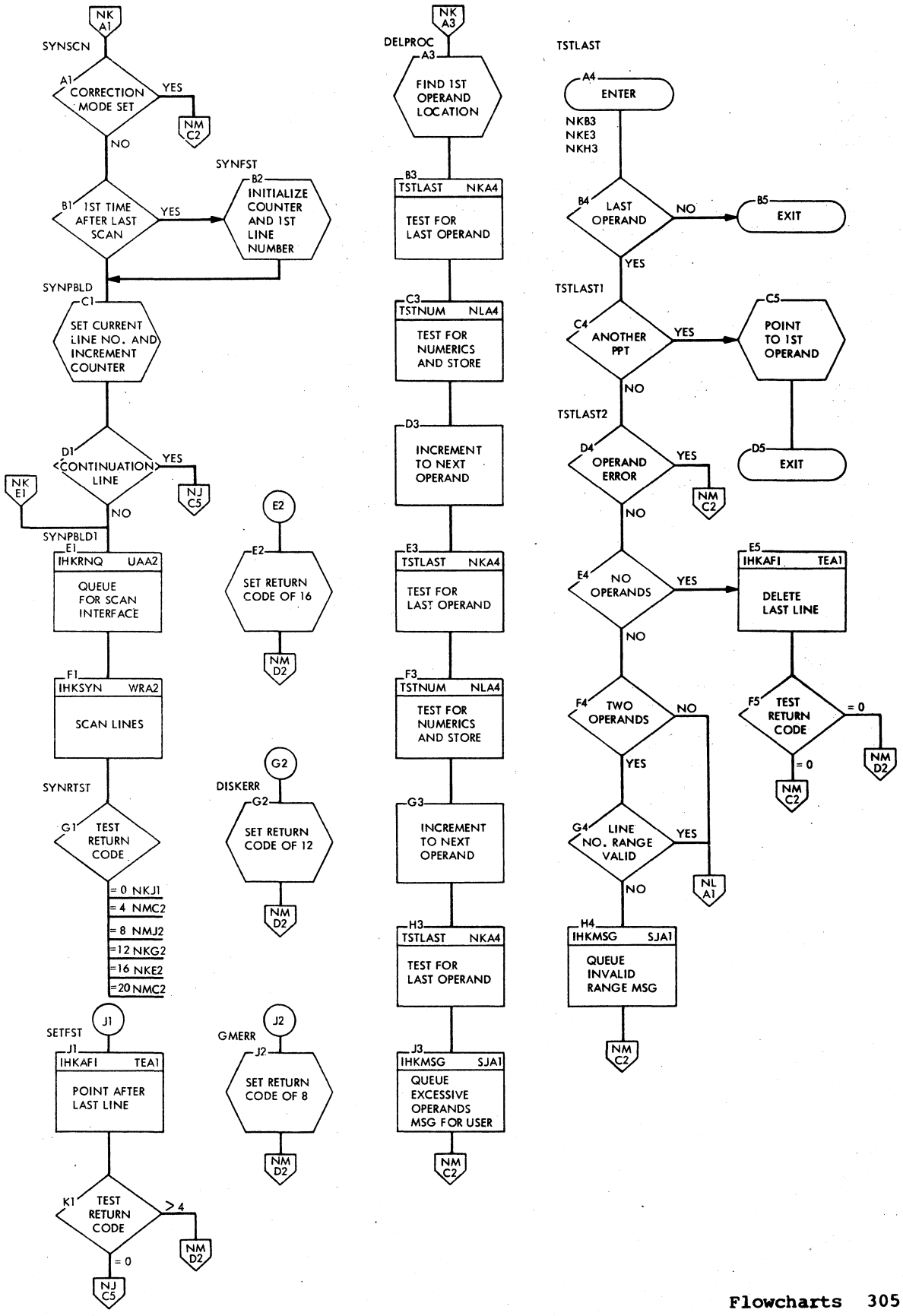


Chart NL. Insert/Replace/Delete Processor (IHKIRL)

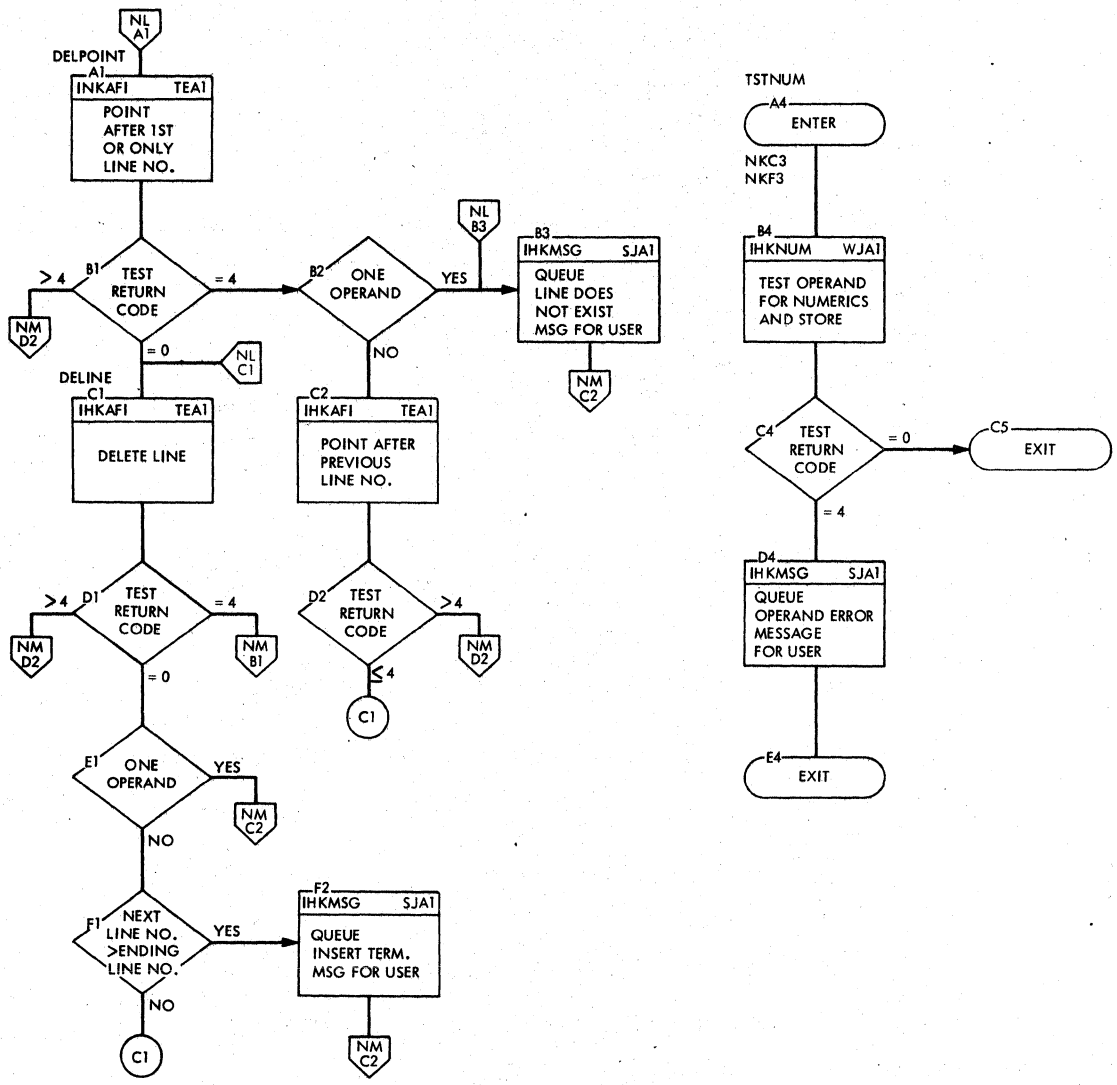


Chart NM. Insert/Replace/Delete Processor (IHKIRL)

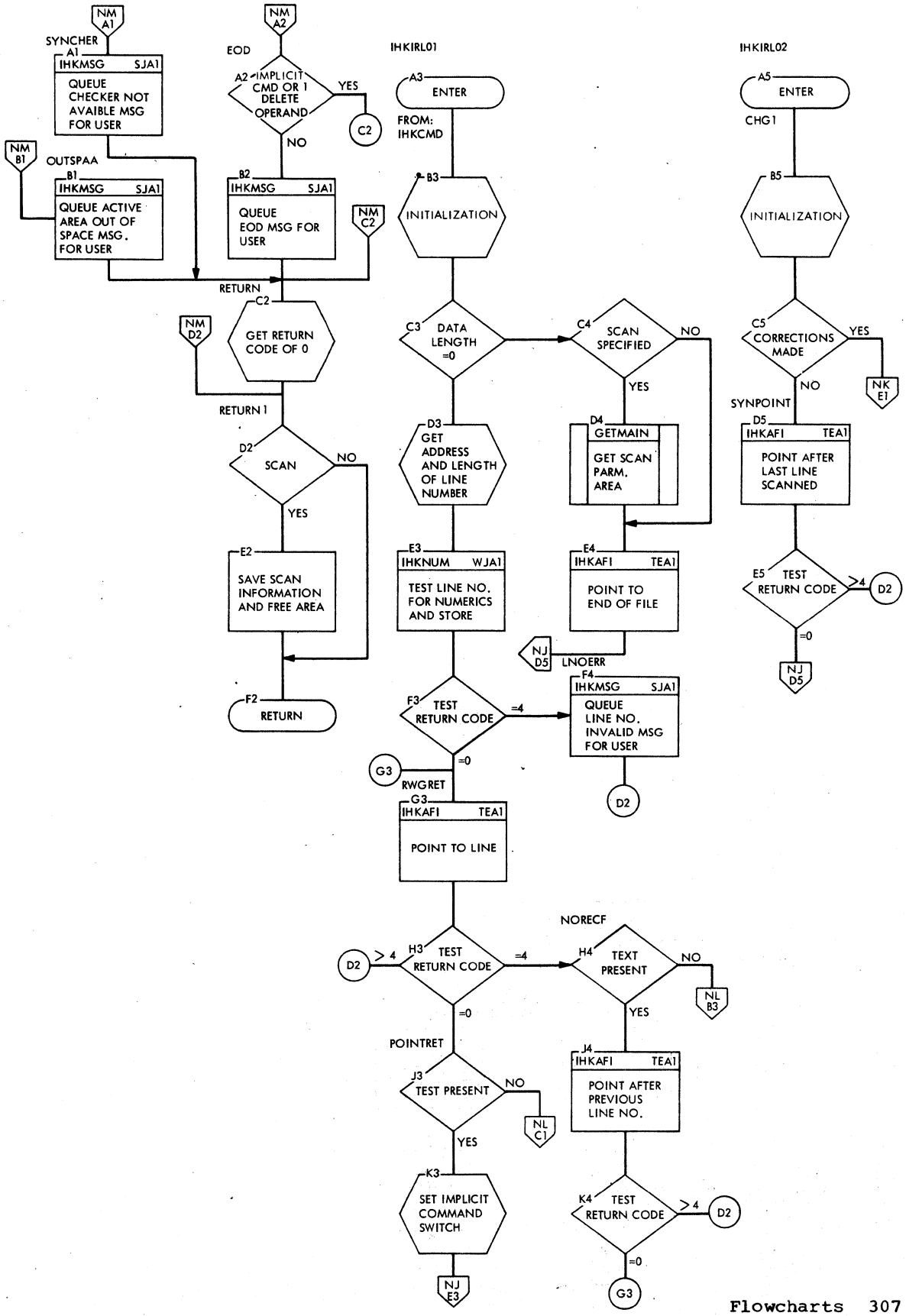


Chart NR. LIST Subcommand Processor (IHKLST)

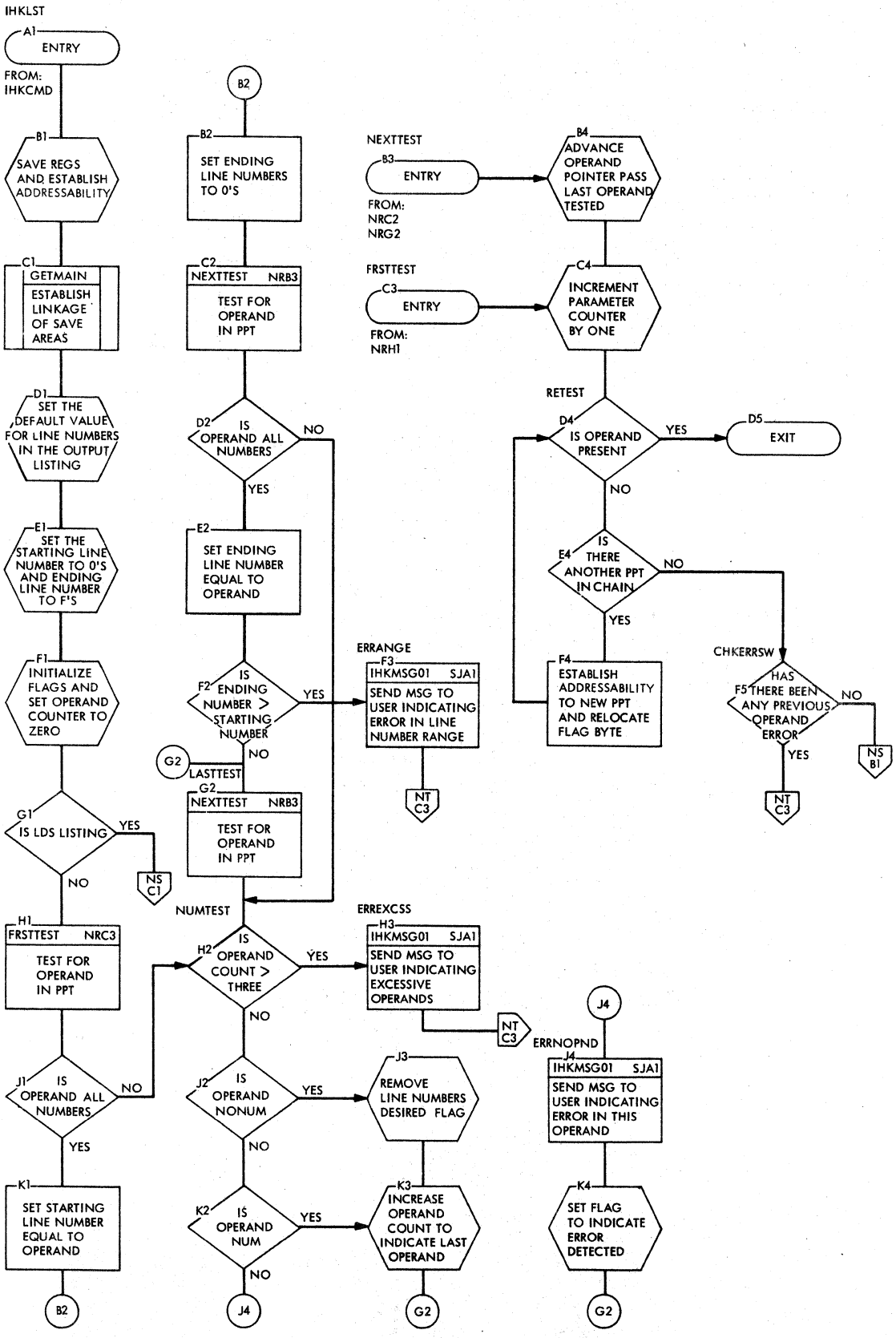


Chart NS. LIST Subcommand Processor (IHKLST)

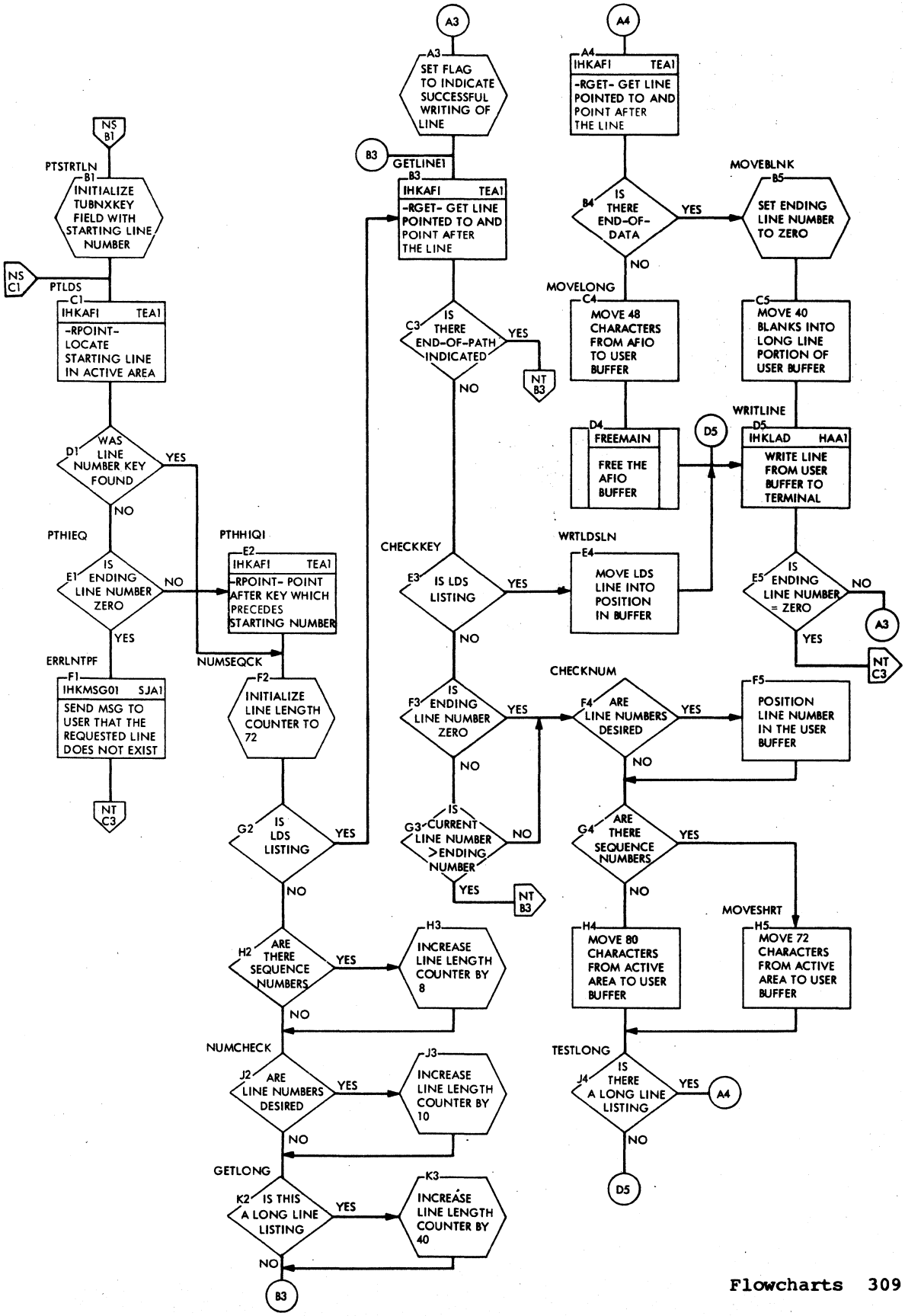


Chart NT. LIST Subcommand Processor (IHKLST)

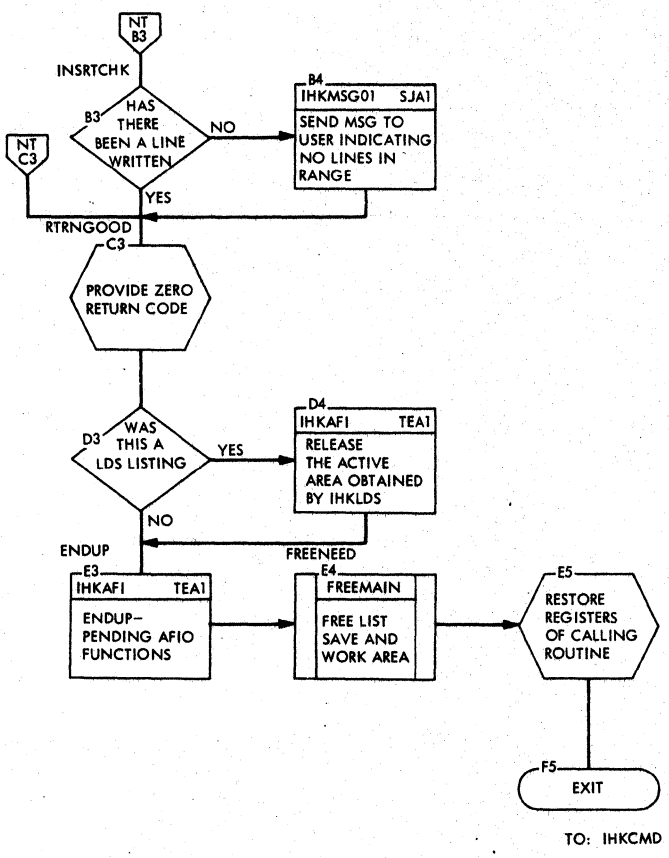


Chart NV. LISTDS and LISTLIB Command Processor (IHKLDS)

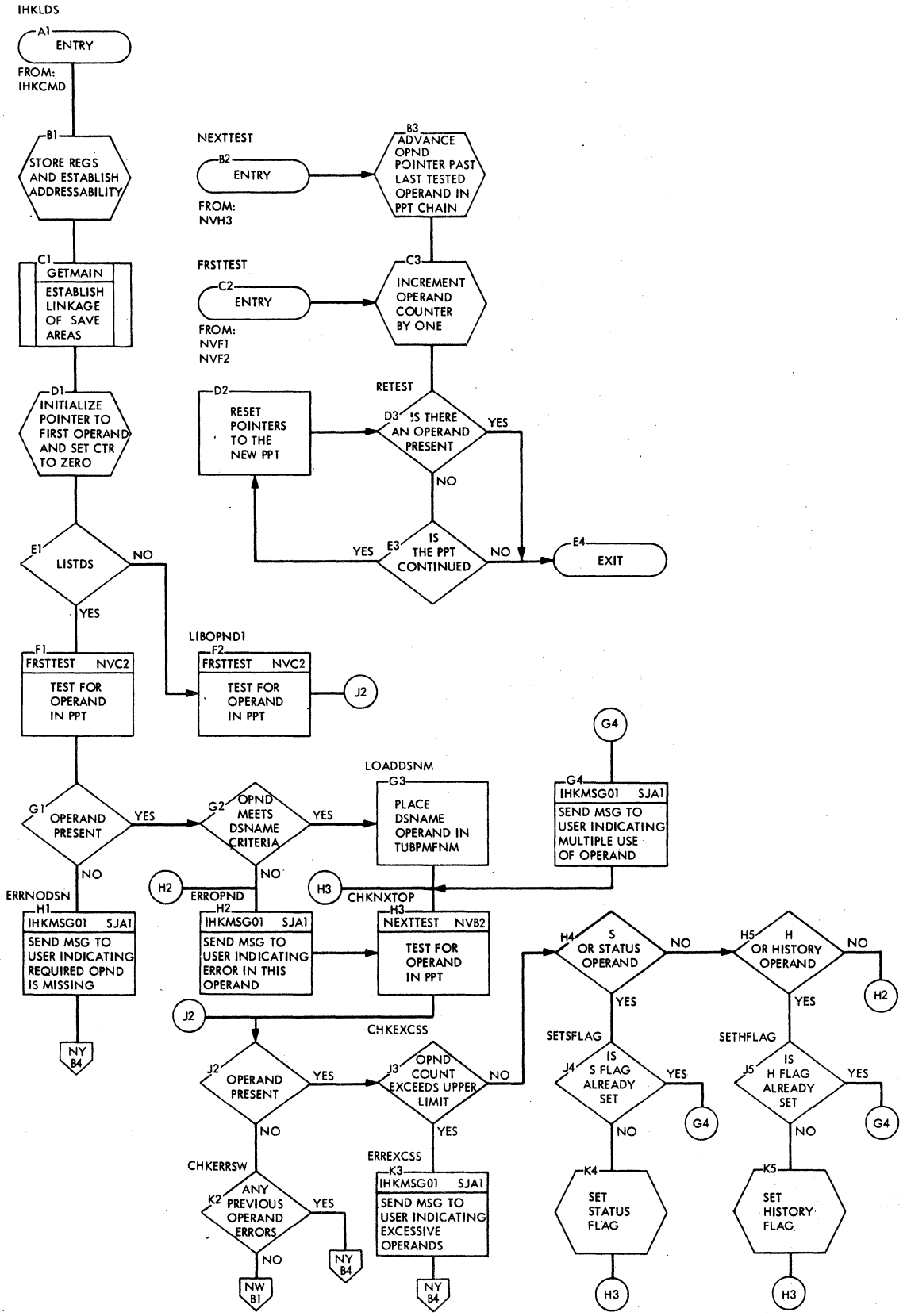


Chart NW. LISTDS and LISTLIB Command Processor (IHKLDS)

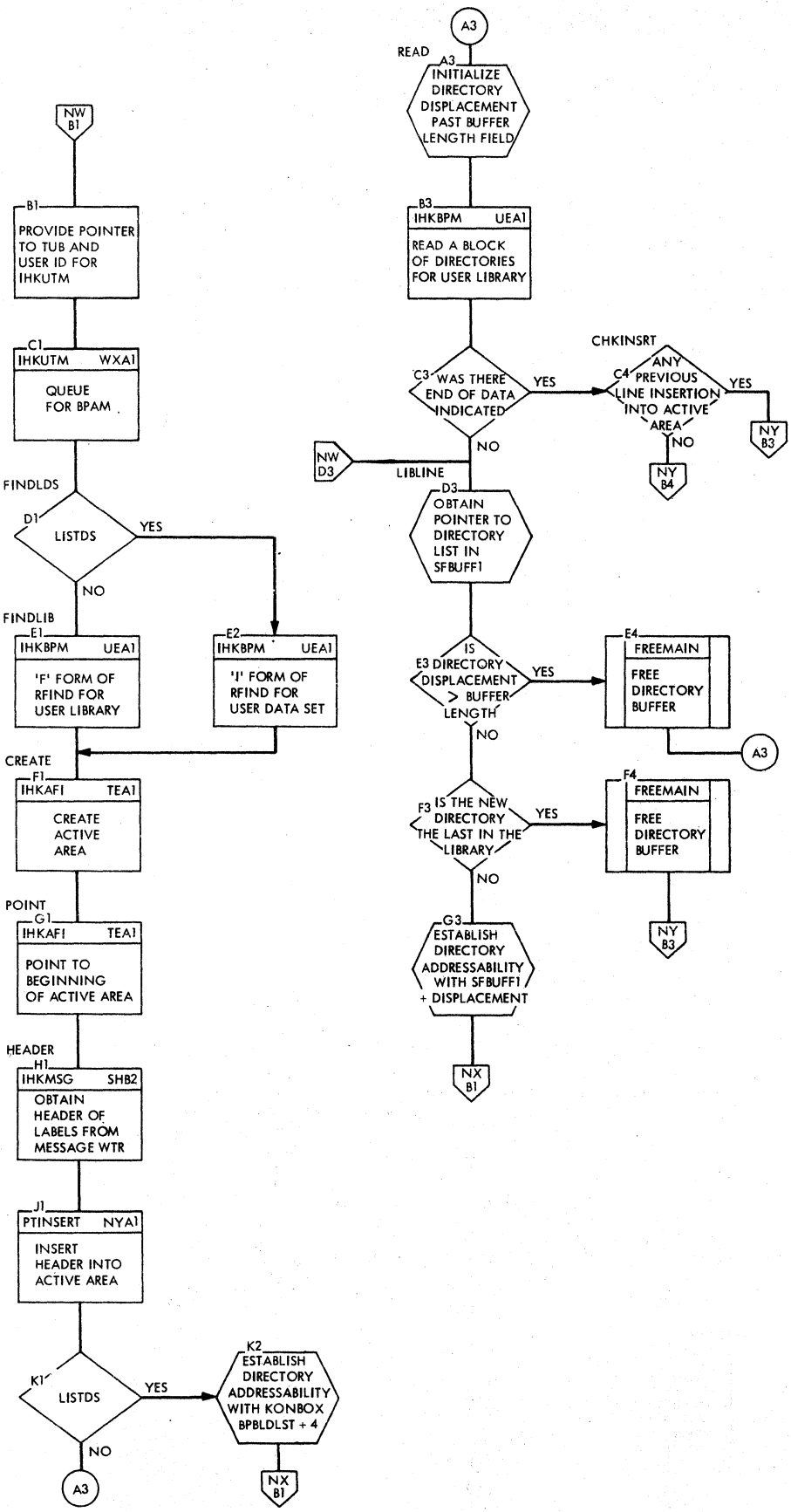


Chart NX. LISTDS and LISTLIB Command Processor (IHKLDS)

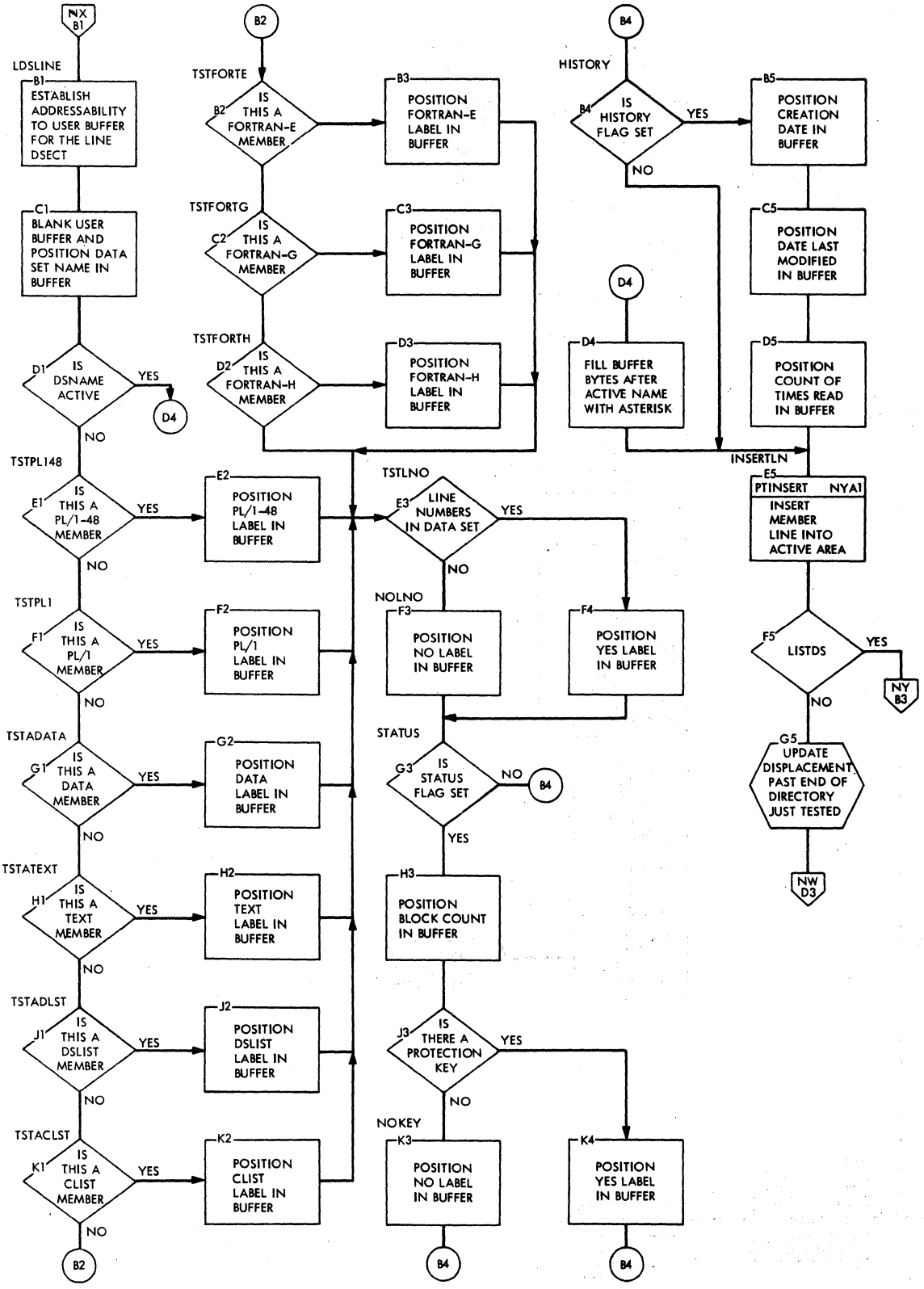


Chart NY. LISTDS and LISTLIB Command Processor (IHKLDS)

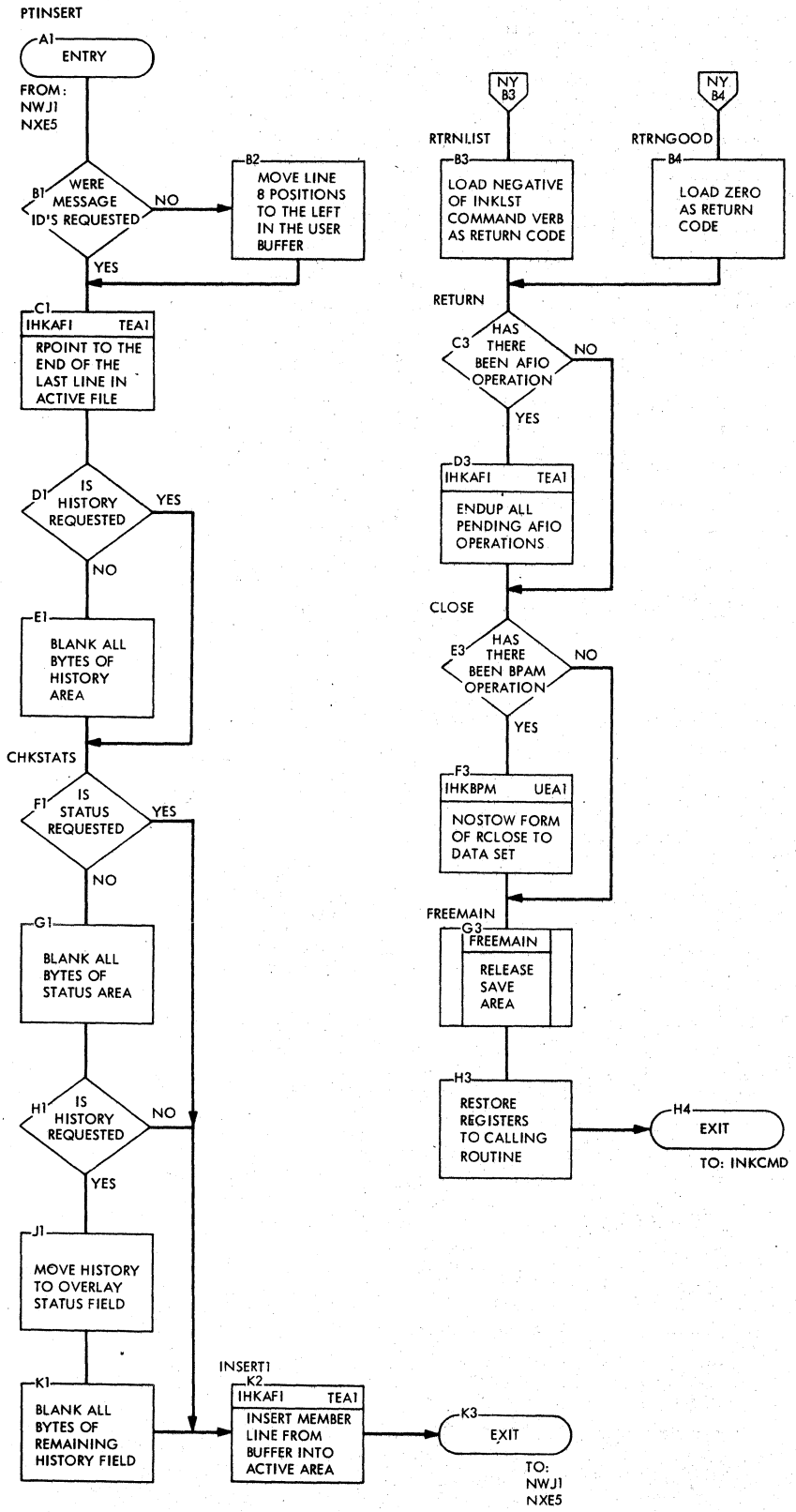


Chart PA. LOGOFF Command Processor (IHKLGF)

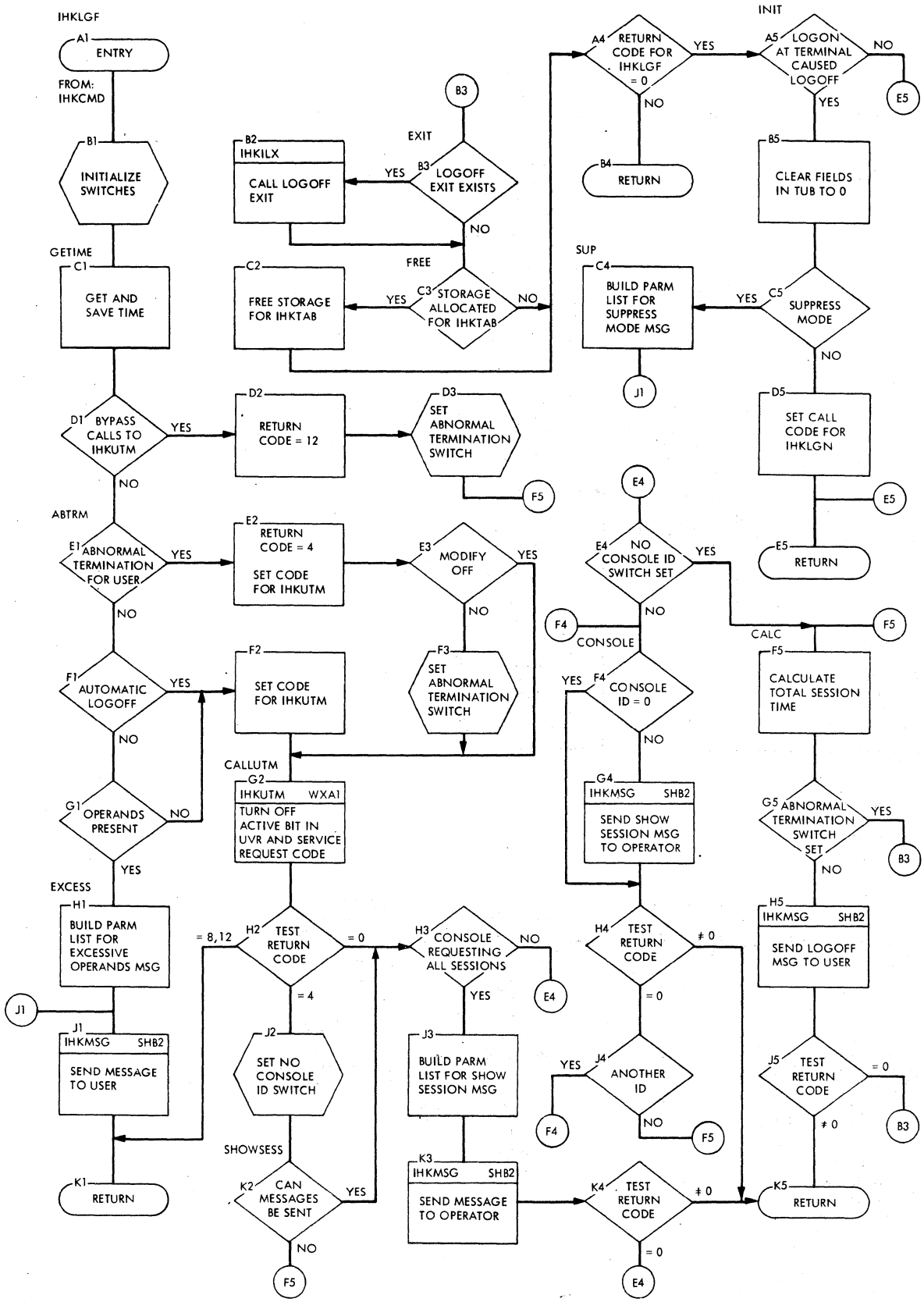


Chart PE. LOGON Command Processor (IHKLGN)

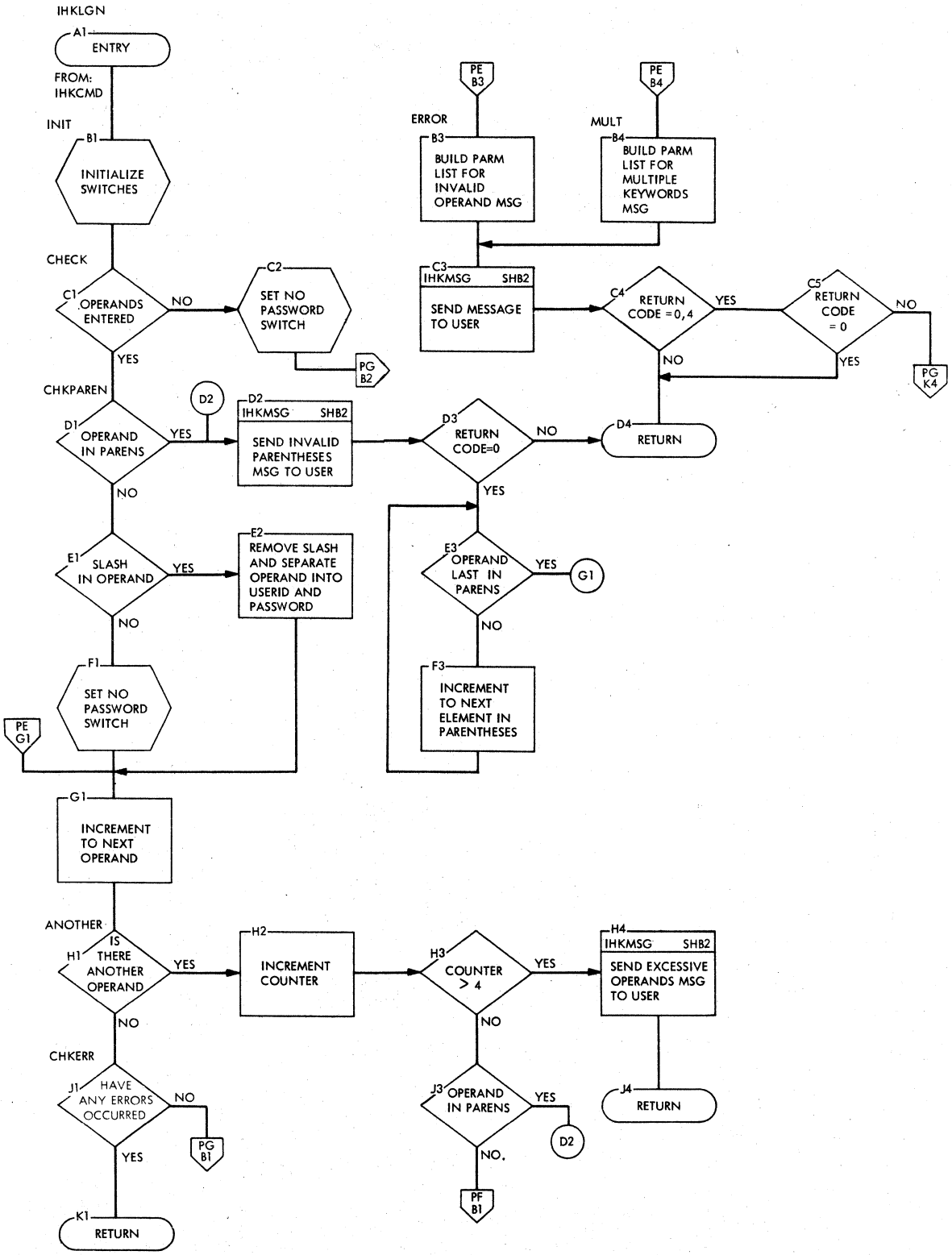
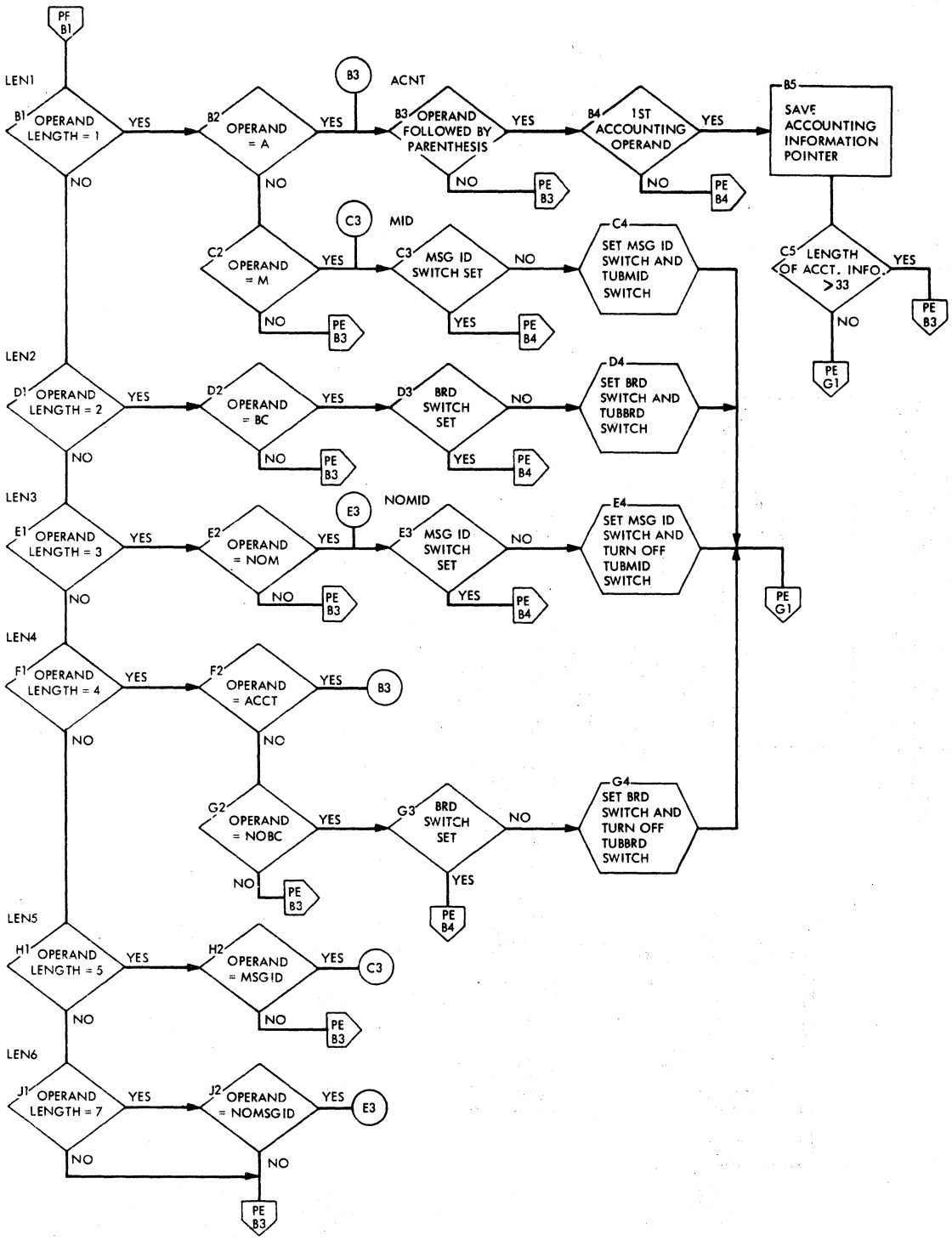
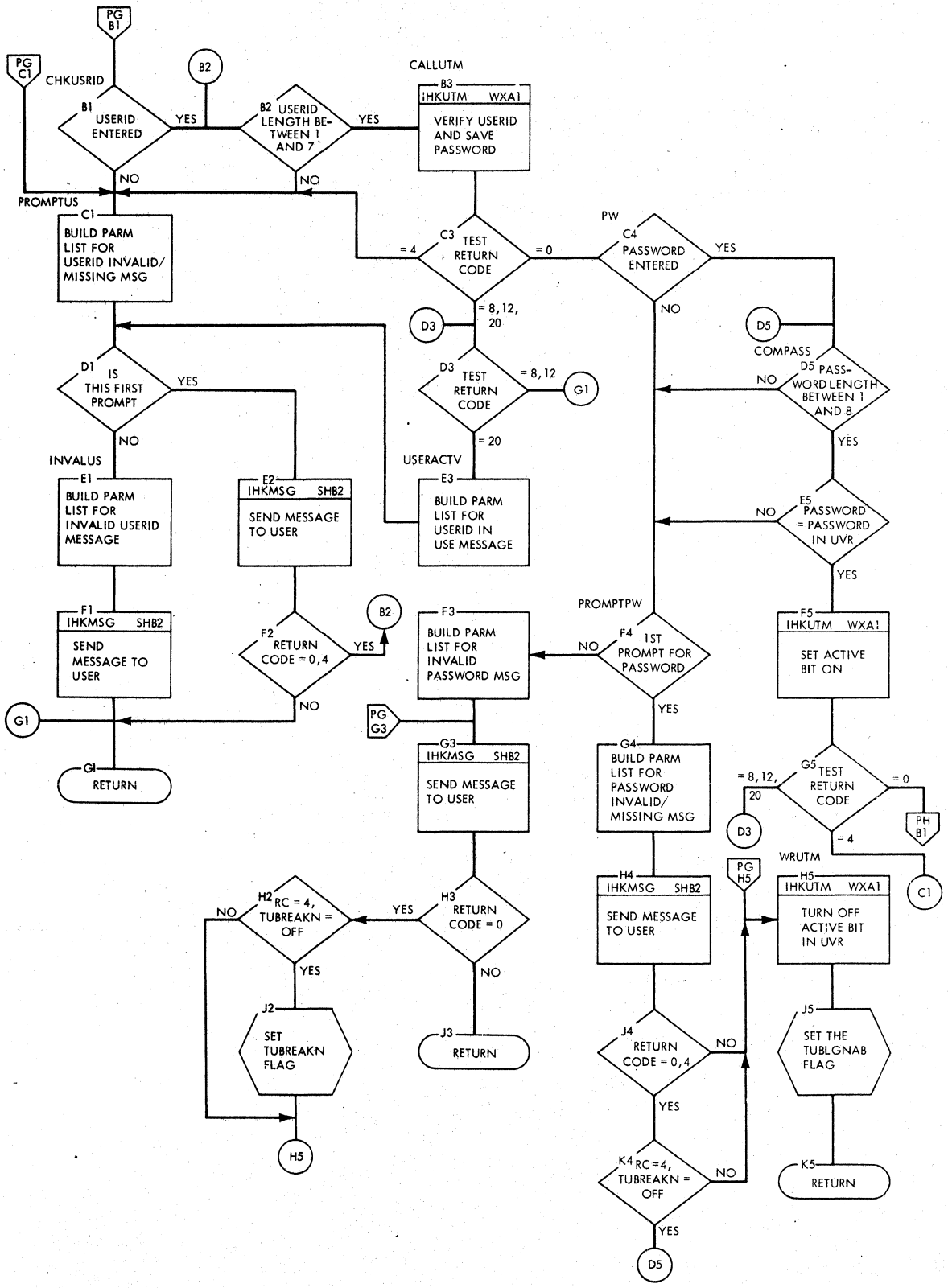


Chart PF. LOGON Command Processor (IHKLGN)



• **Chart PG. LOGON Command Processor (IHKLGN)**



• Chart PH. LOGON Command Processor (IHKLGN)

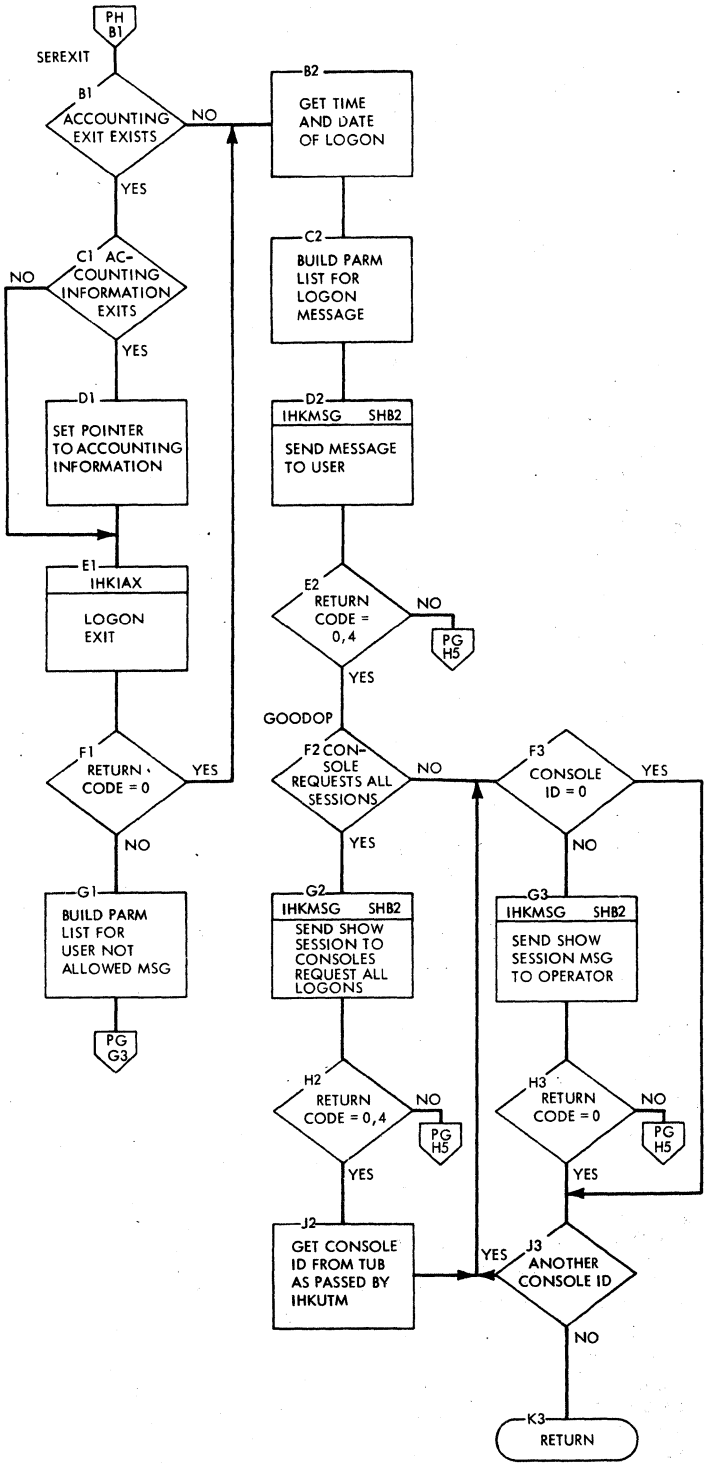


Chart PJ. MERGE Subcommand Processor (IHKMGE)

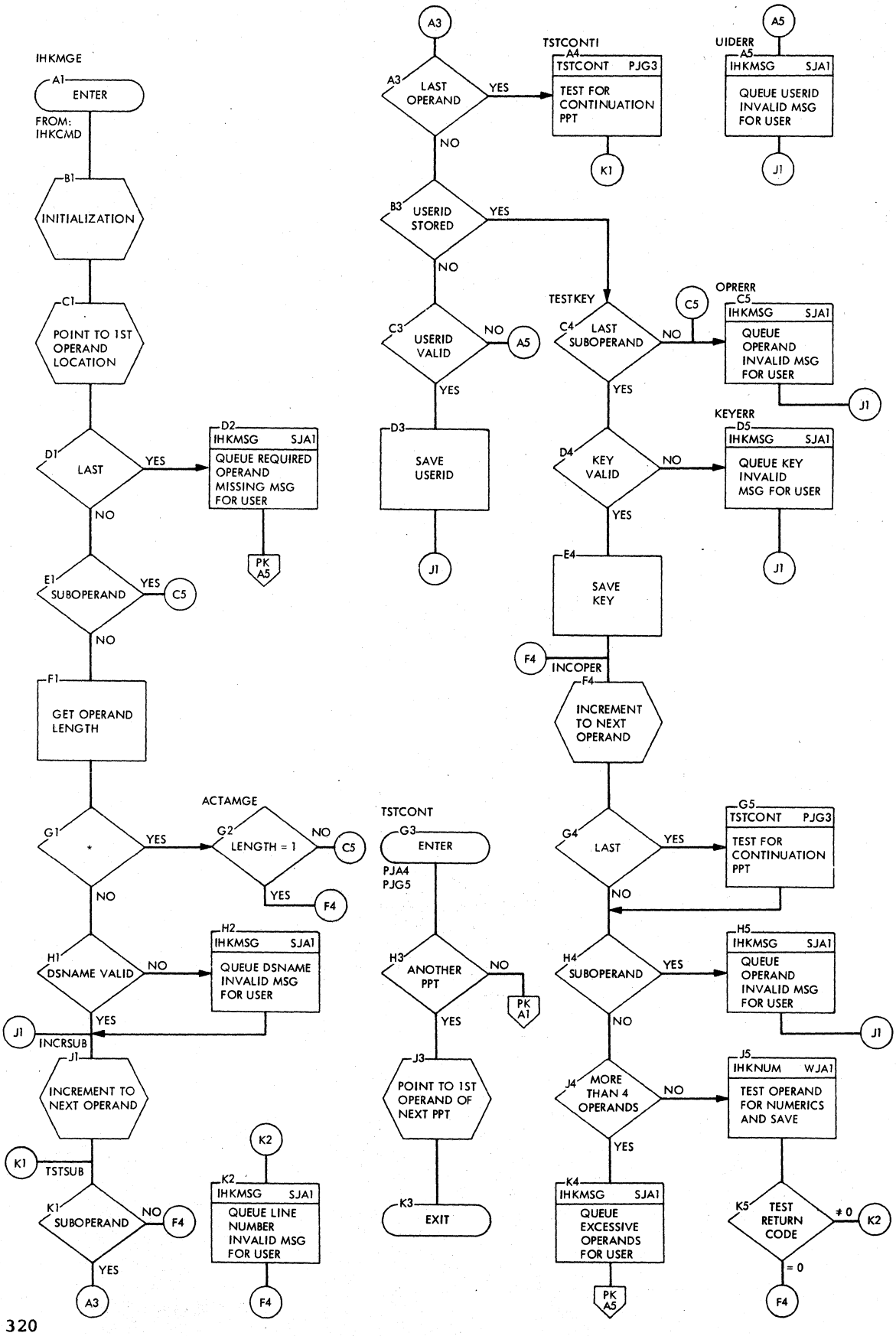


Chart PK. MERGE Subcommand Processor (IHKMGE)

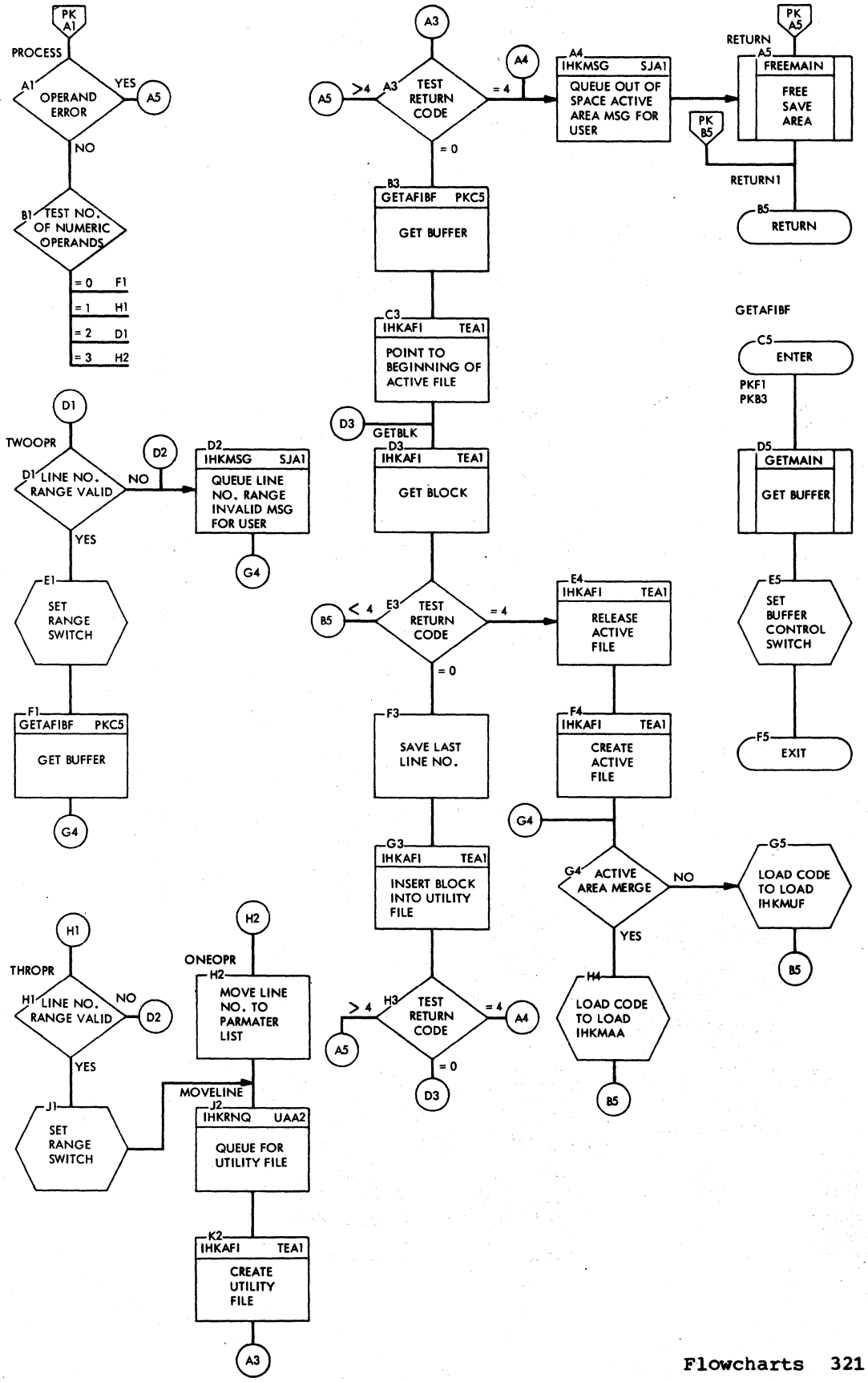


Chart PN. MERGE Subcommand Processor (IHKMAA)

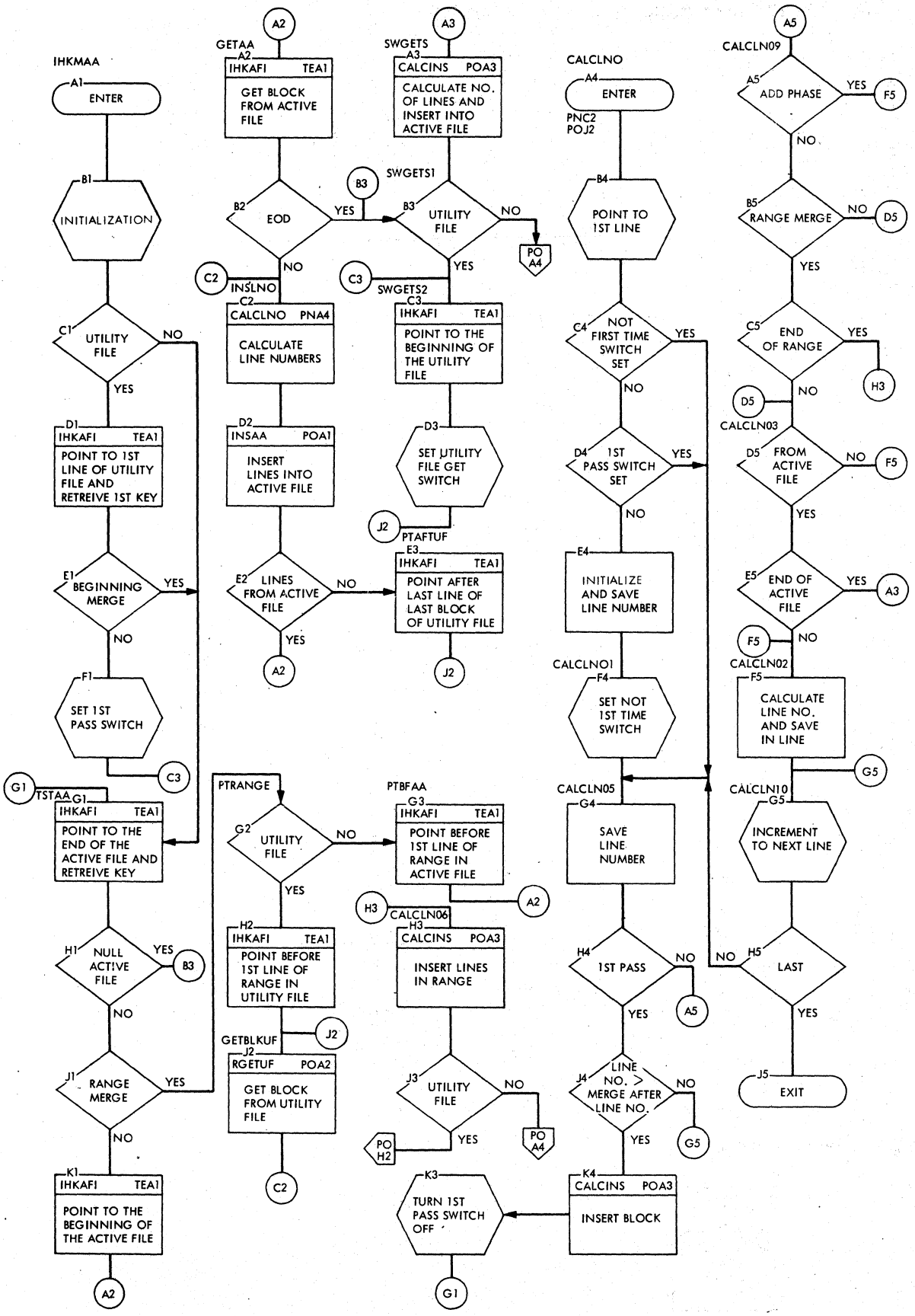


Chart PO. MERGE Subcommand Processor (IHKMAA)

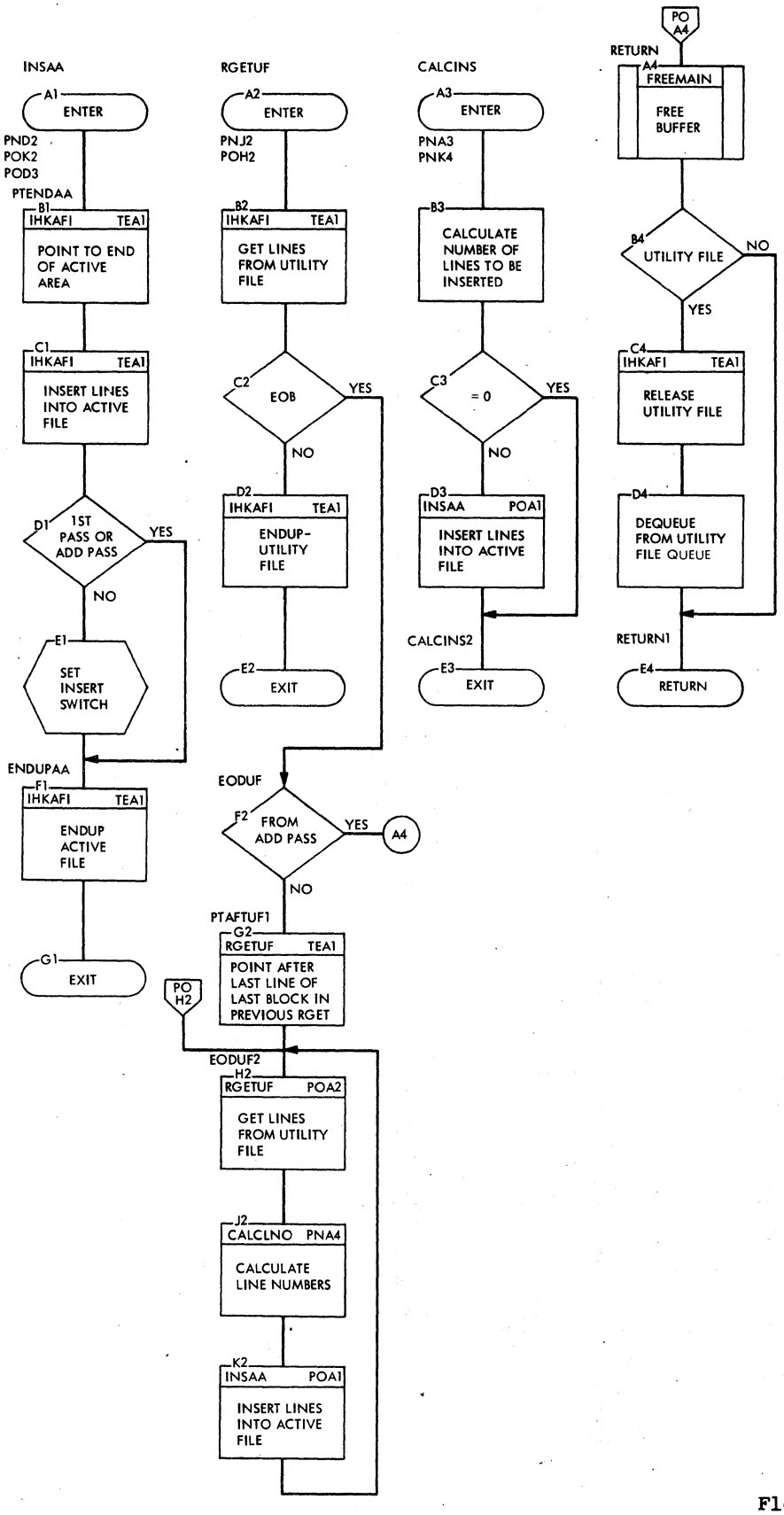


Chart PS. MERGE Subcommand Processor (IHKMUF)

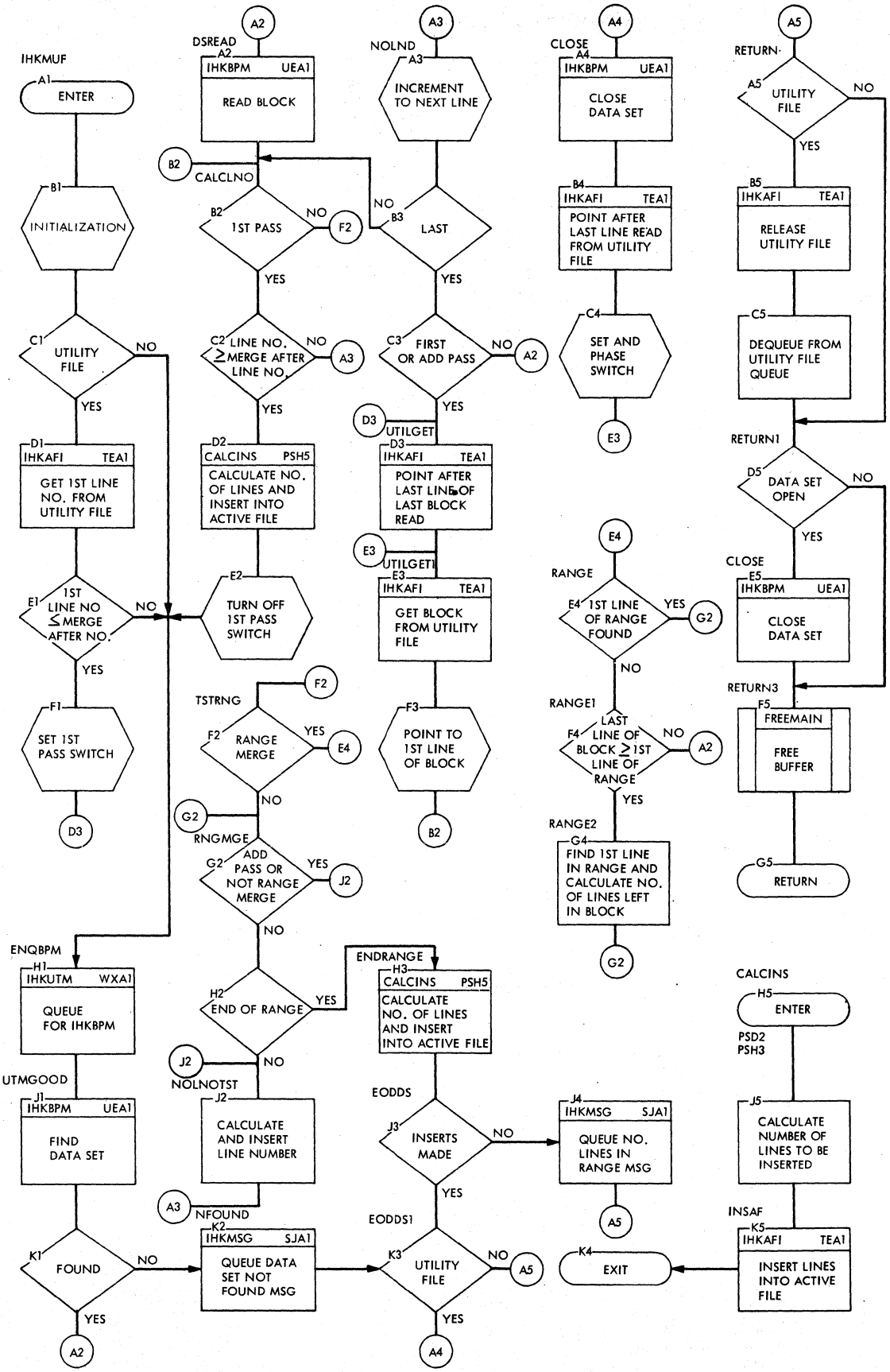
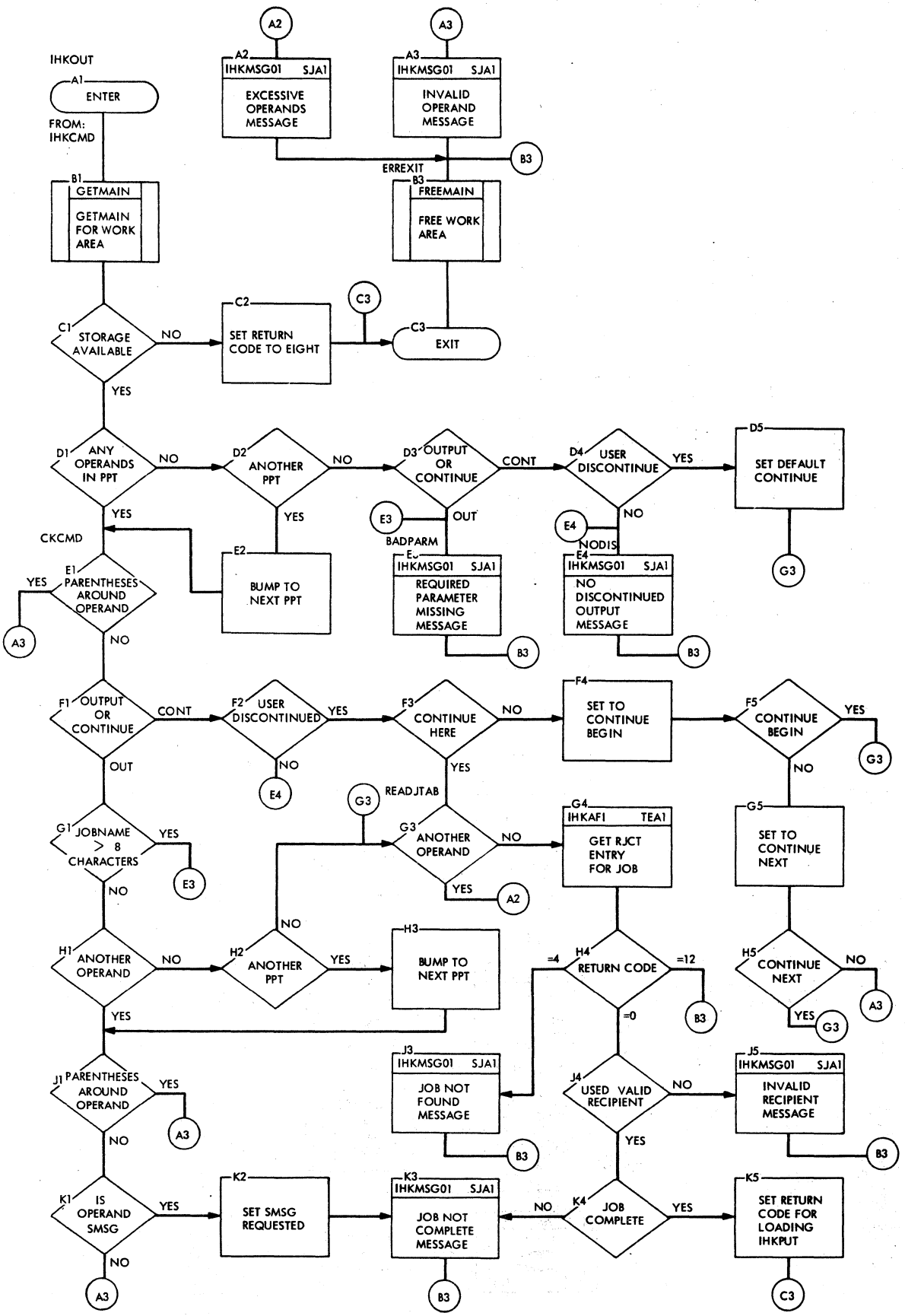


Chart QA. OUTPUT and CONTINUE Command Processor (IHKOUT)



• Chart QB. Transmit Output Module (IHKPUT)

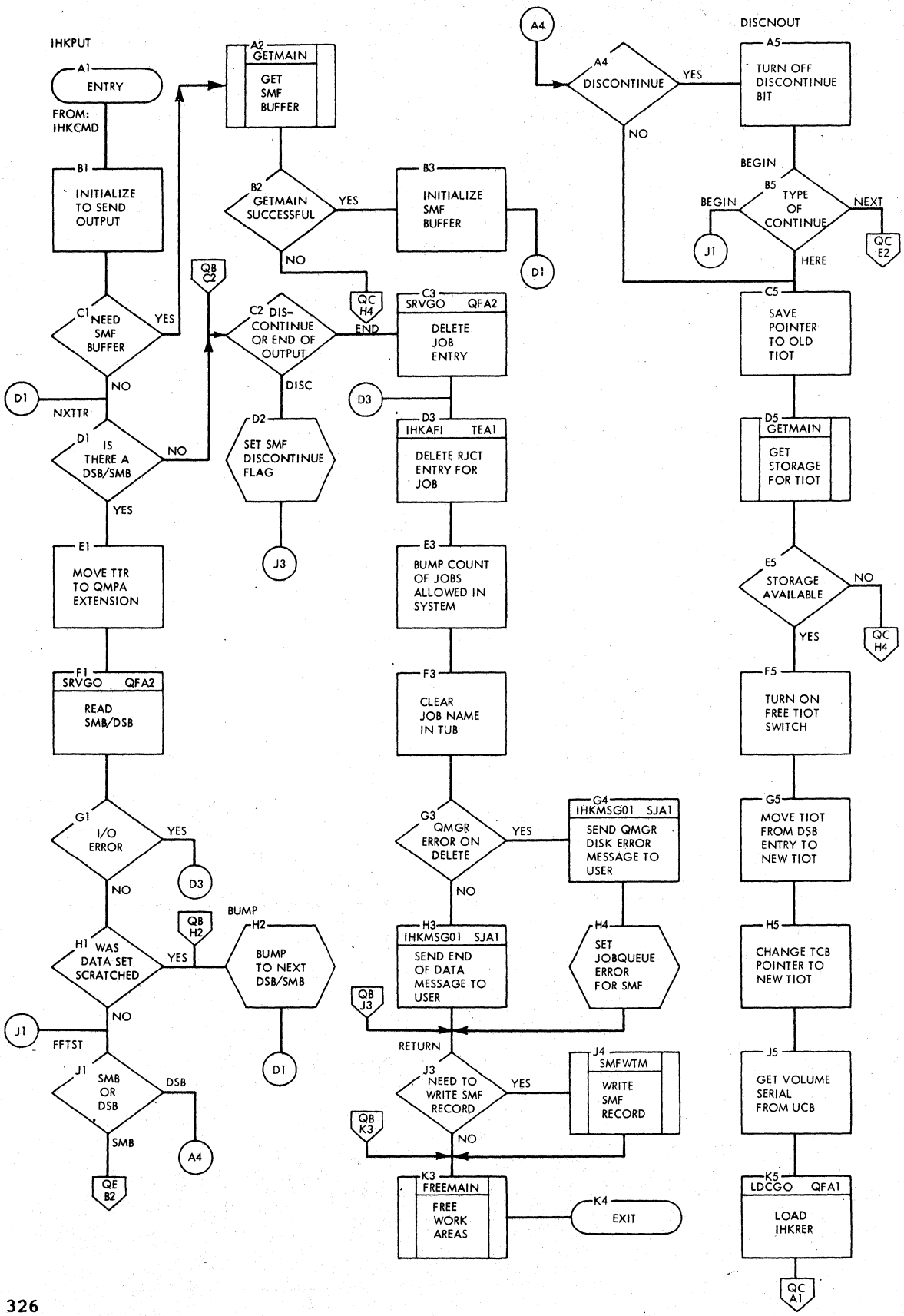
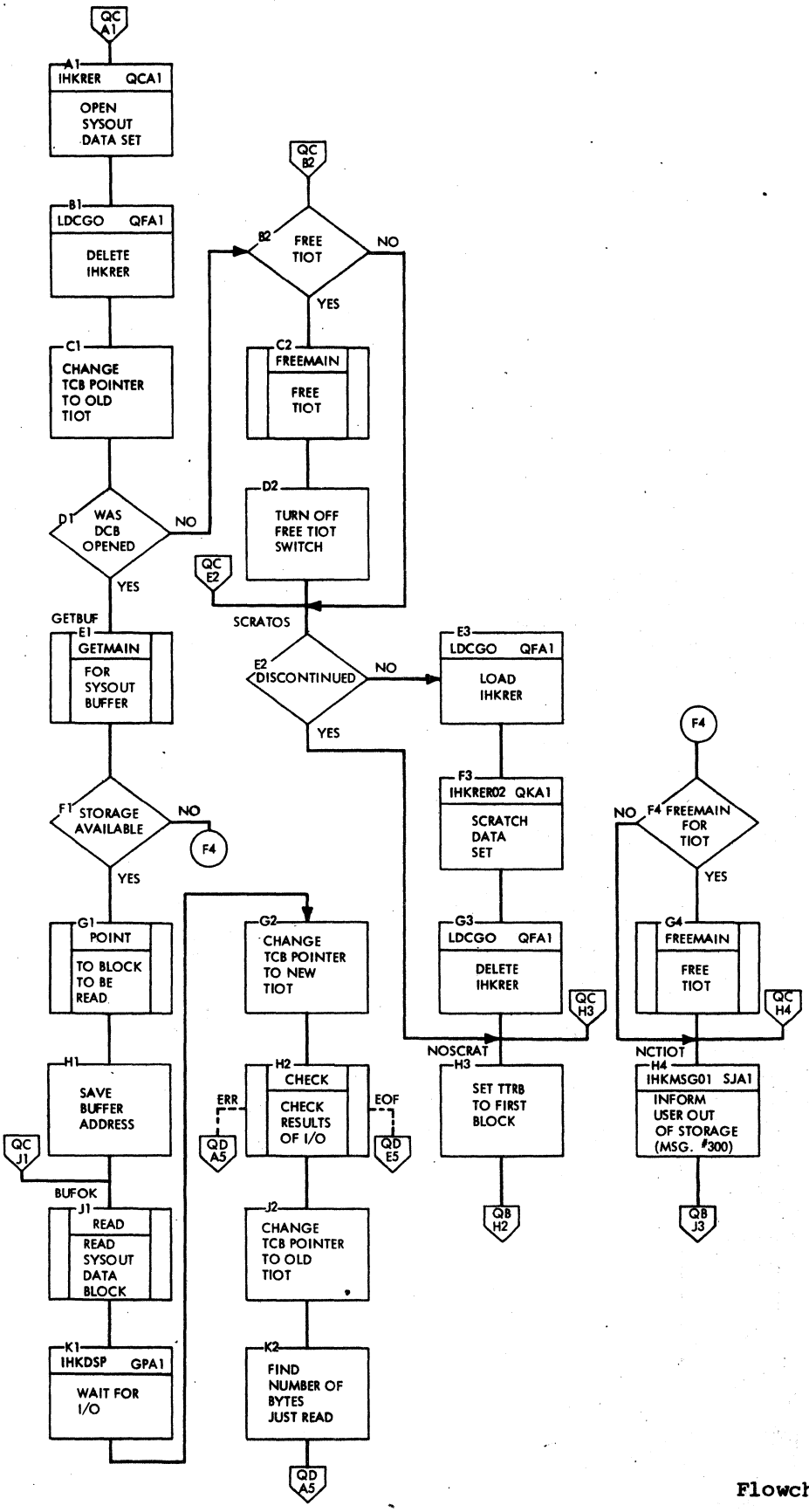


Chart QC. Transmit Output Module (IHKPUT)



• Chart QD. Transmit Output Module (IHKPUT)

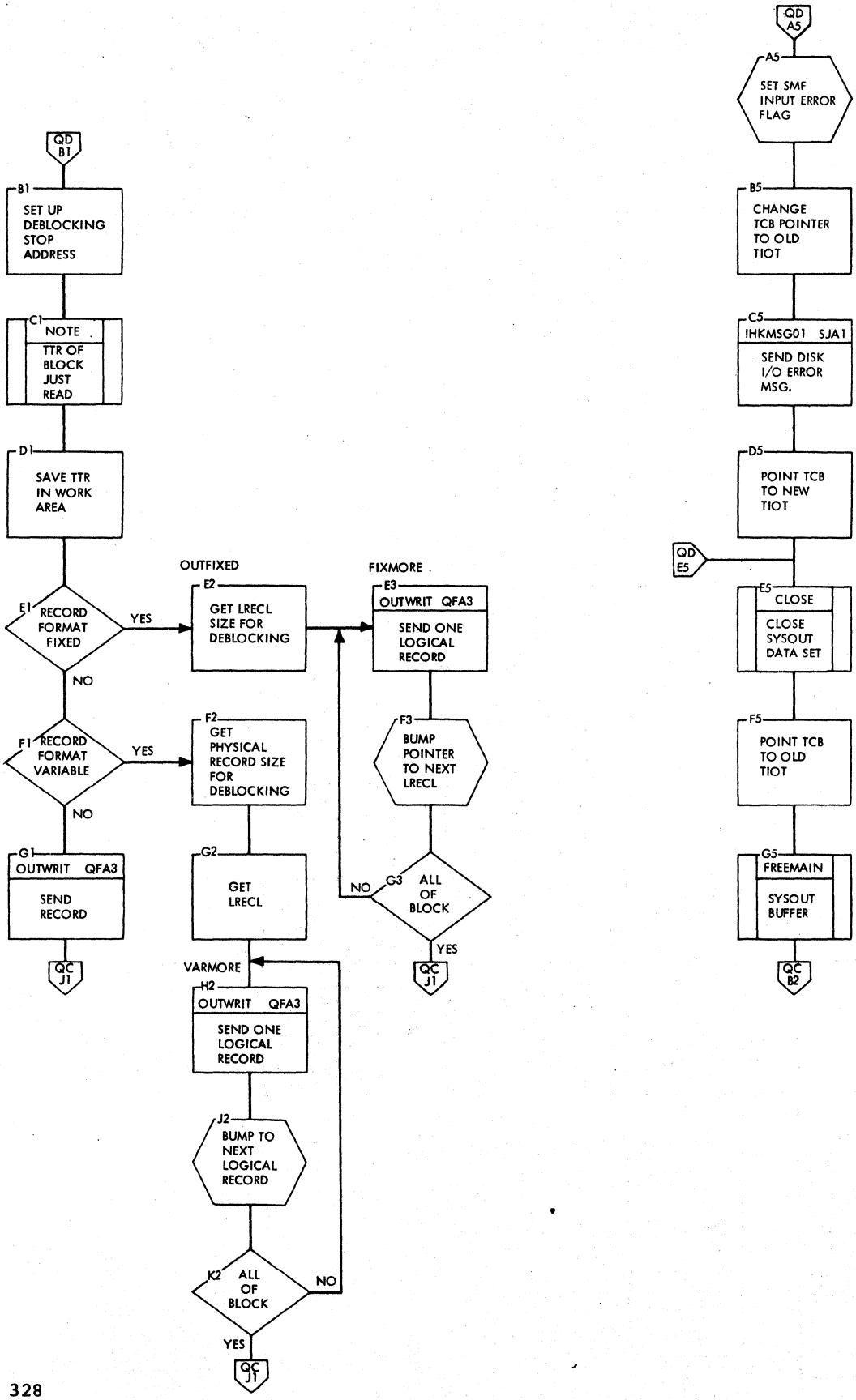
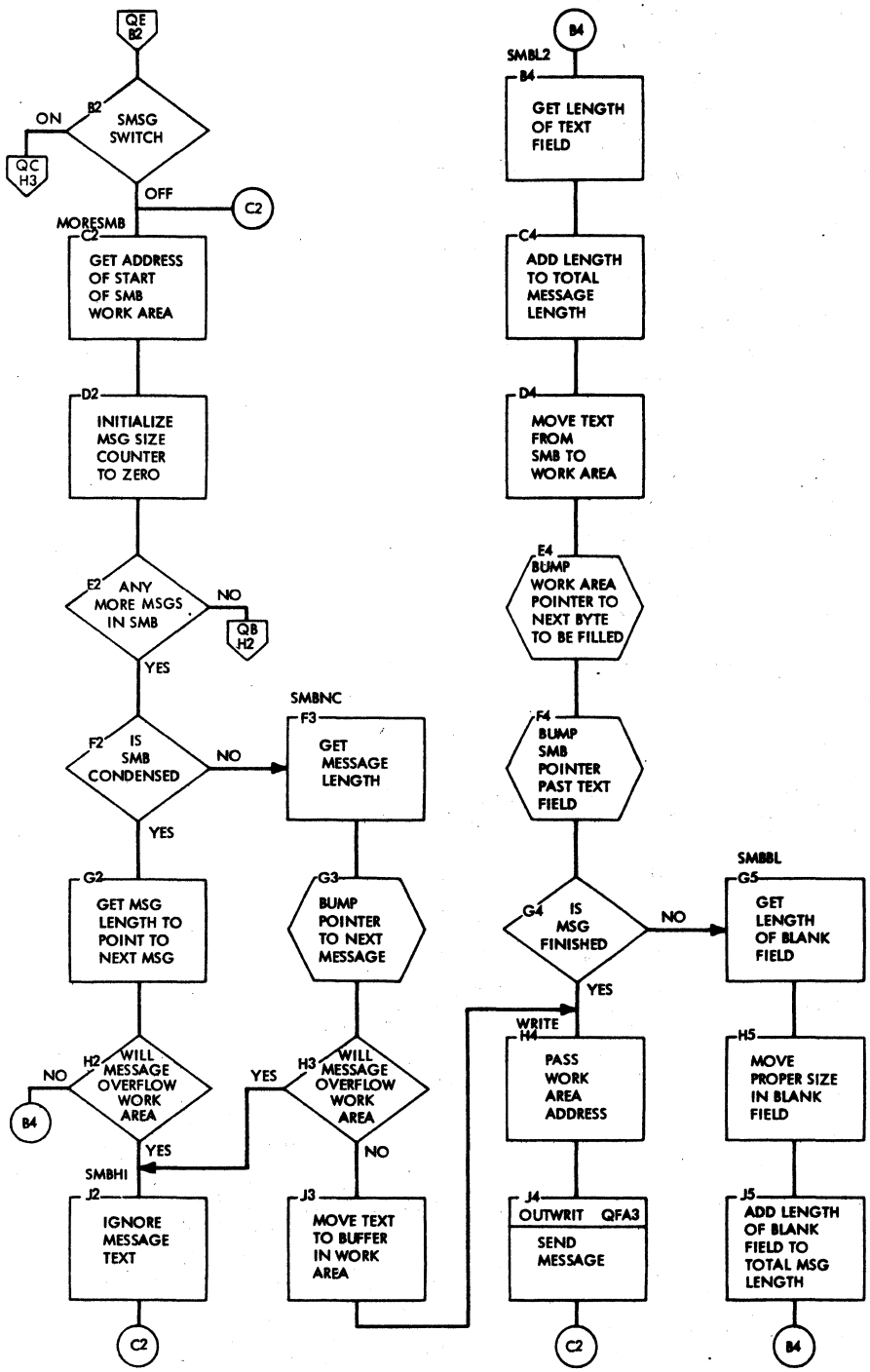
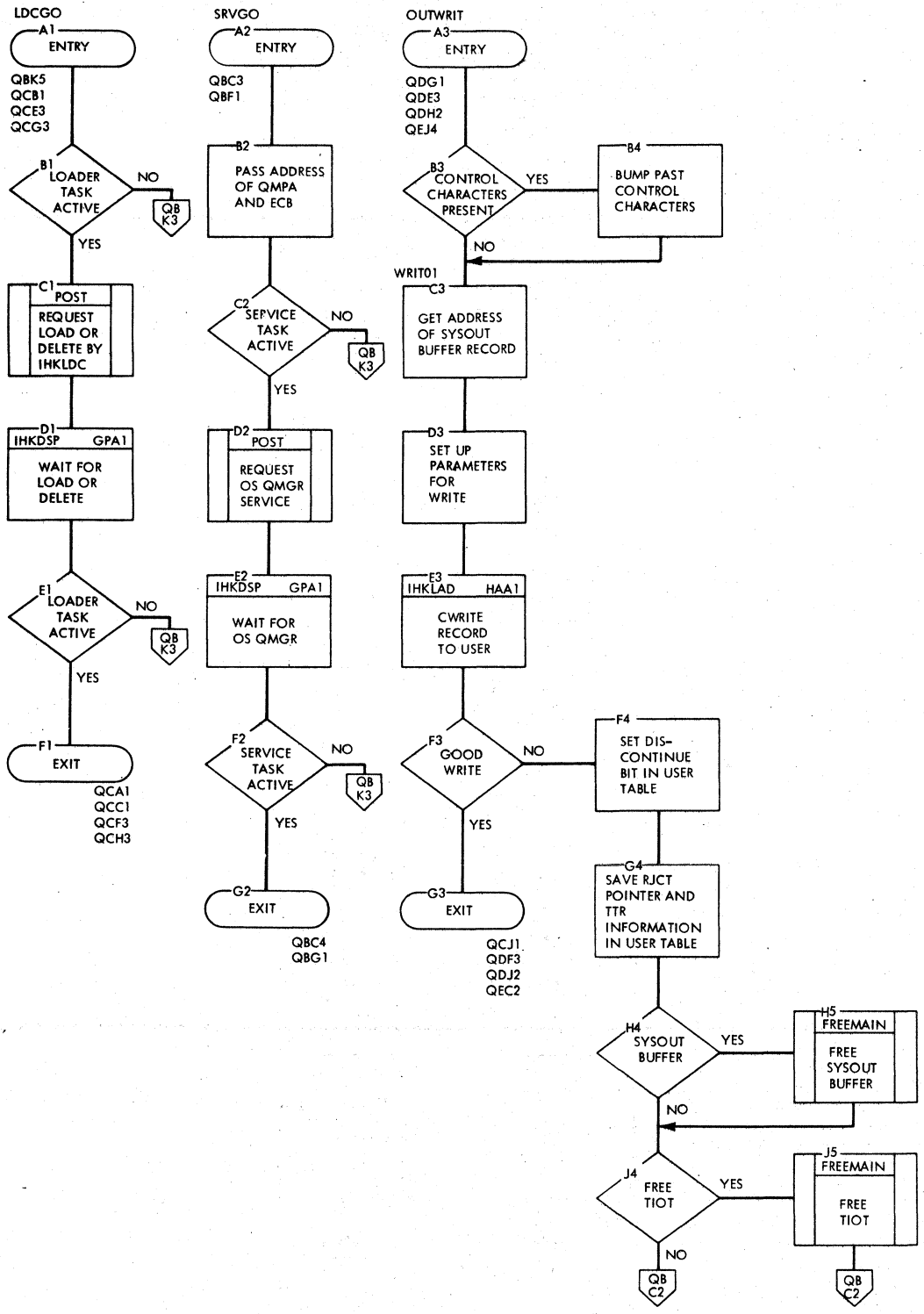


Chart QE. Transmit Output Module (IHKPUT)



• Chart QF. Transmit Output Module (IHKPUT)



- Chart QJ. SYSOUT Open, Job Delete, Data Set Scratch, and CANCEL Module (IHKRER)

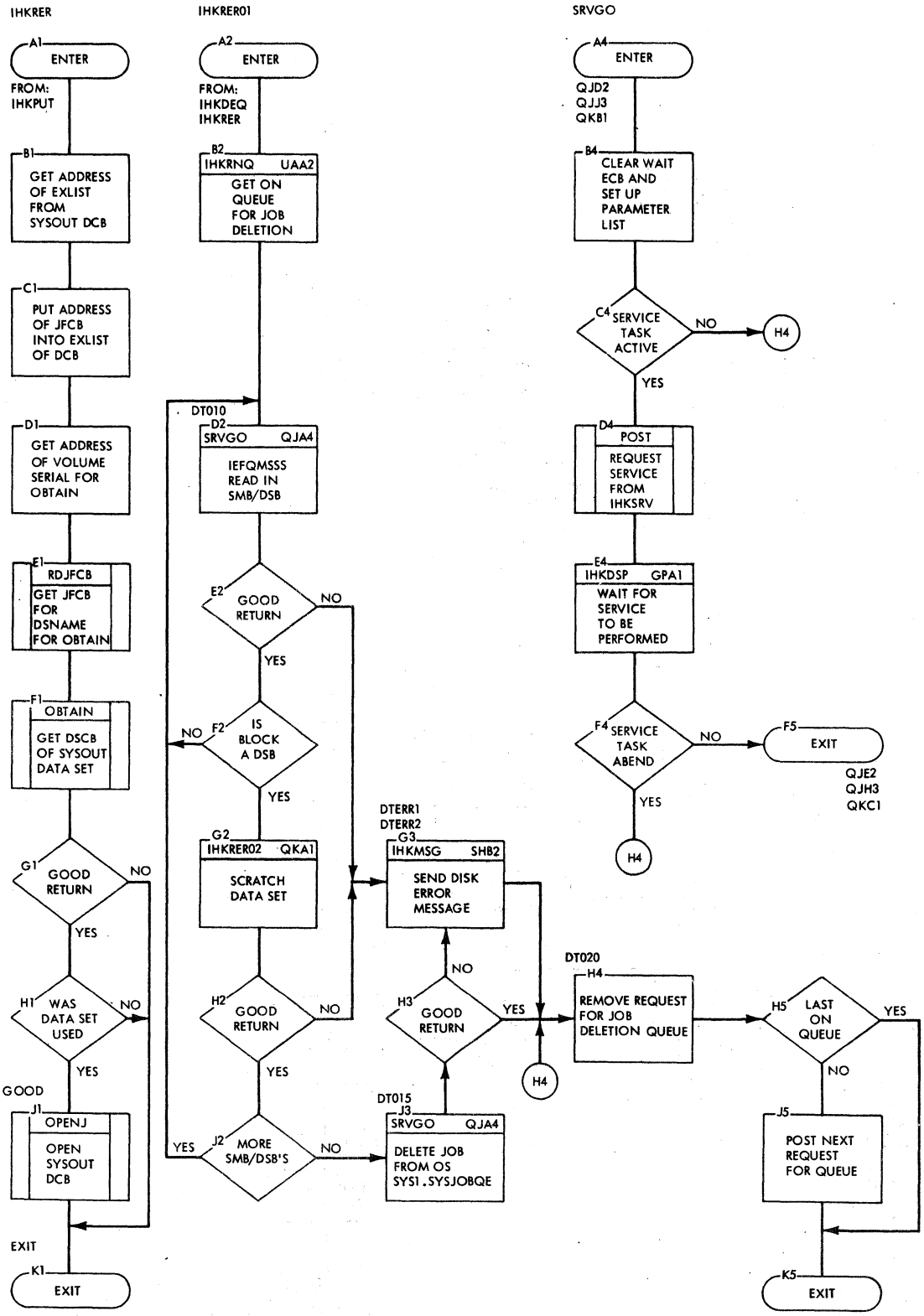


Chart QK. SYSOUT Open, Job Delete, Data Set Scratch, and CANCEL Module (IHKRER)

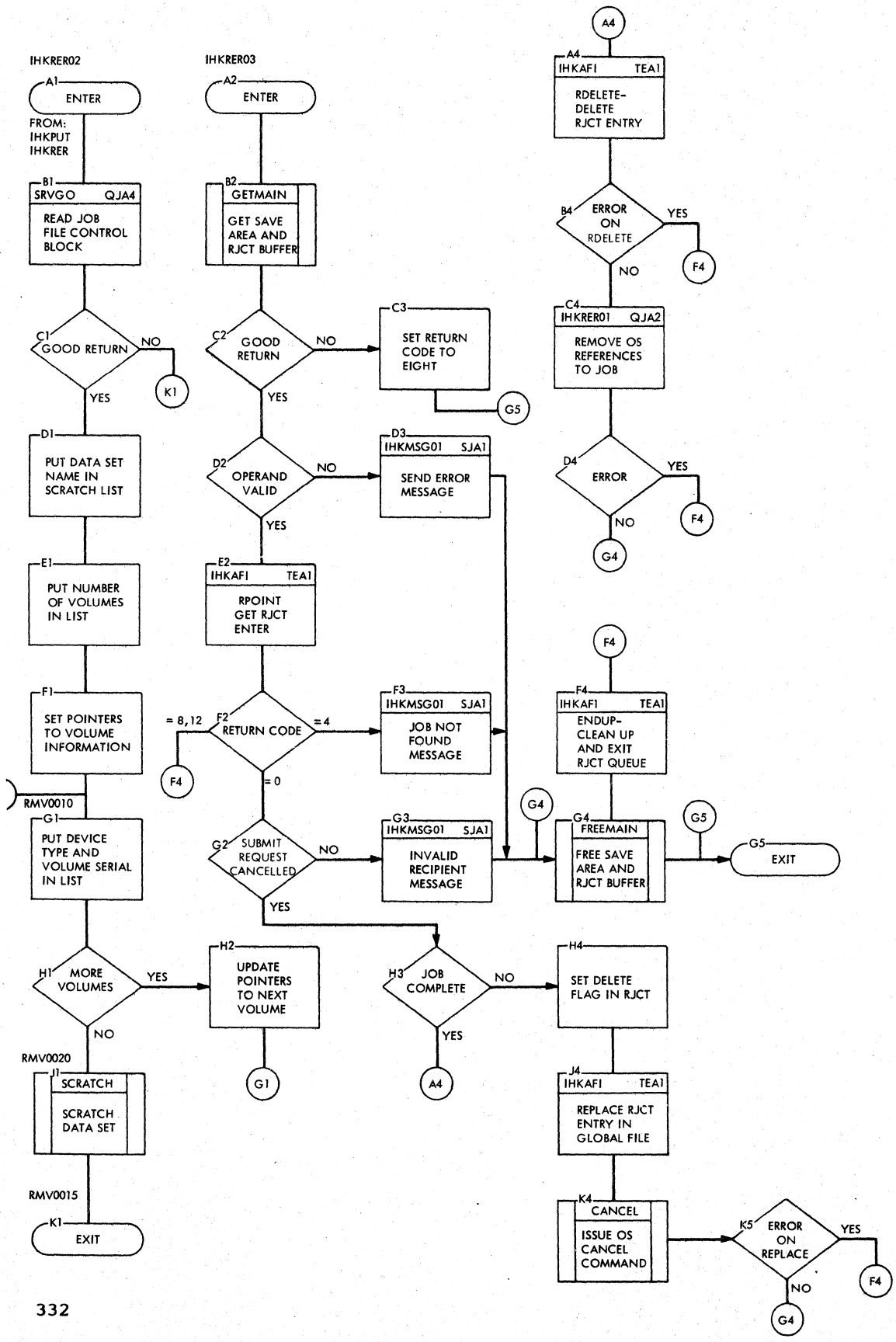


Chart QQ. RENUMBER Subcommand Processor (IHKRNR)

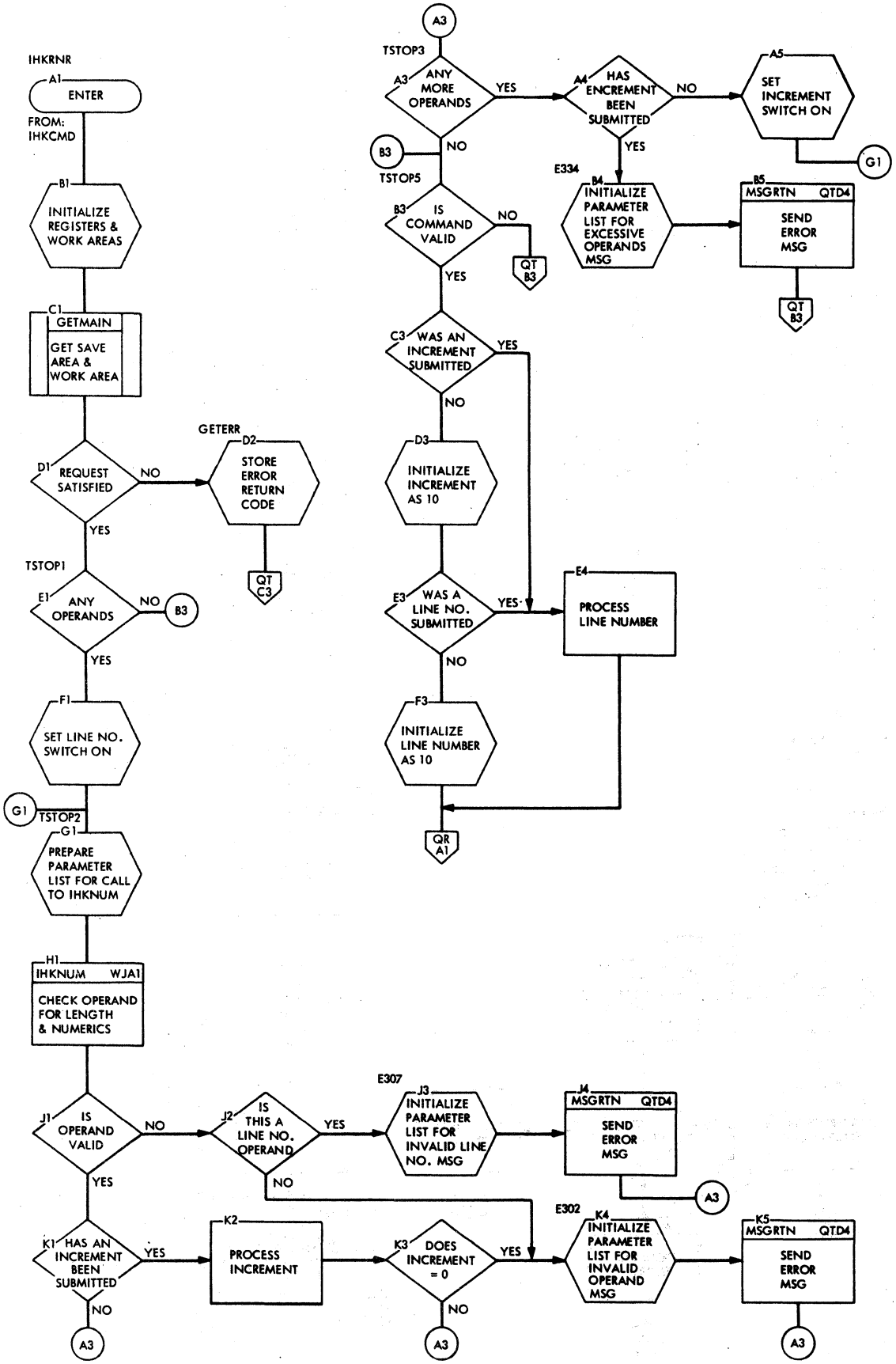


Chart QR. RENUMBER Subcommand Processor (IHKRNR)

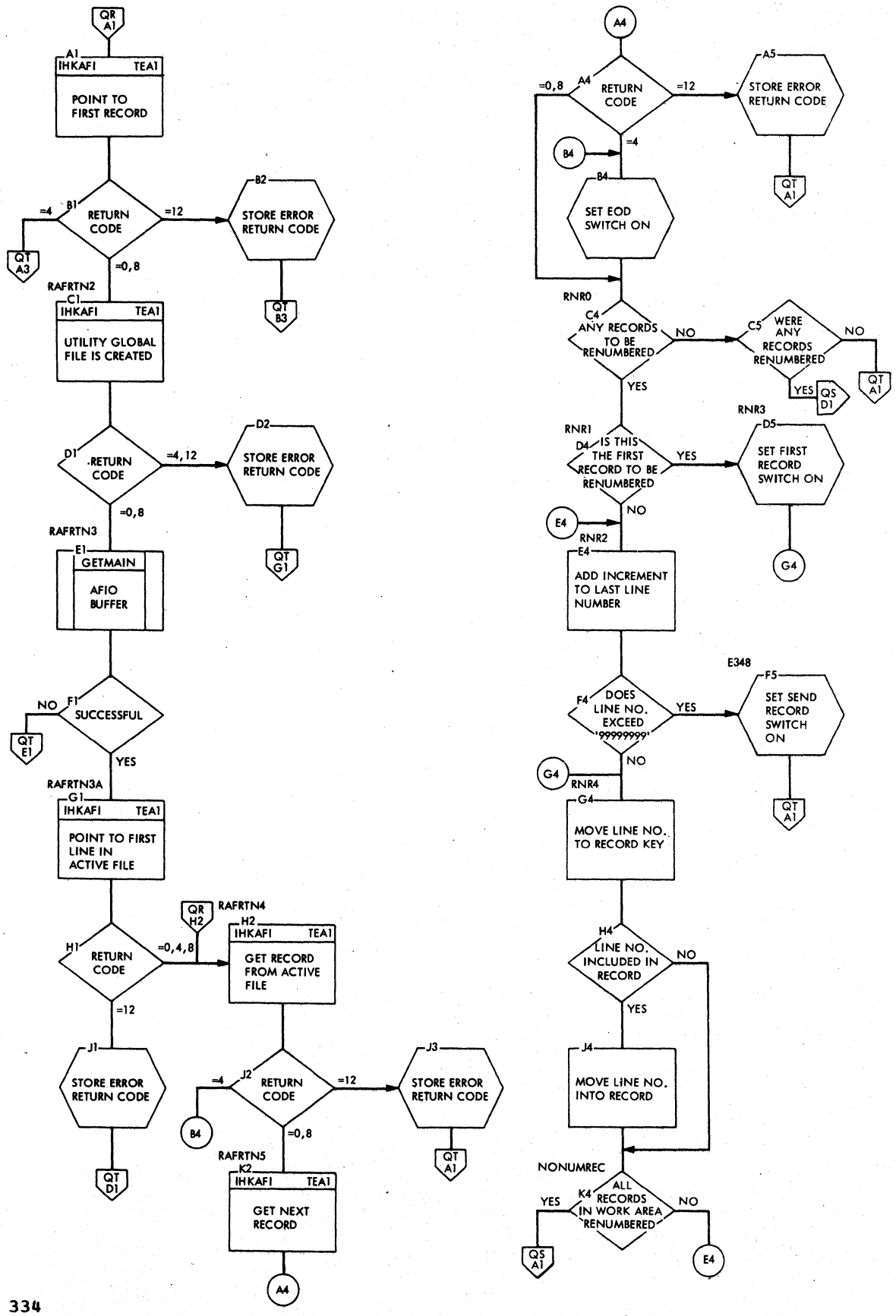


Chart QS. RENUMBER Subcommand Processor (IHKRNR)

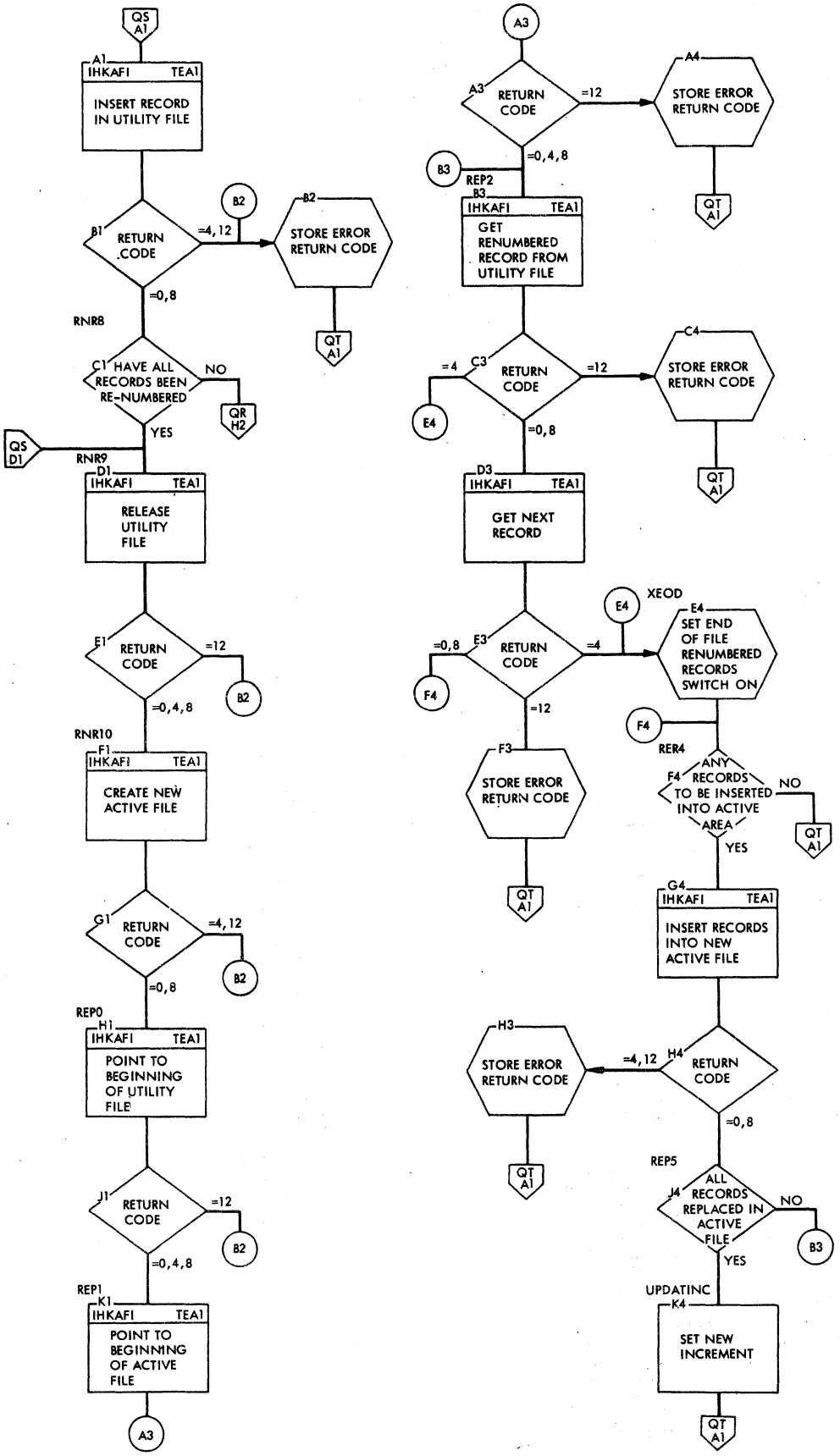


Chart QT. RENUMBER Subcommand Processor (IHKRNR)

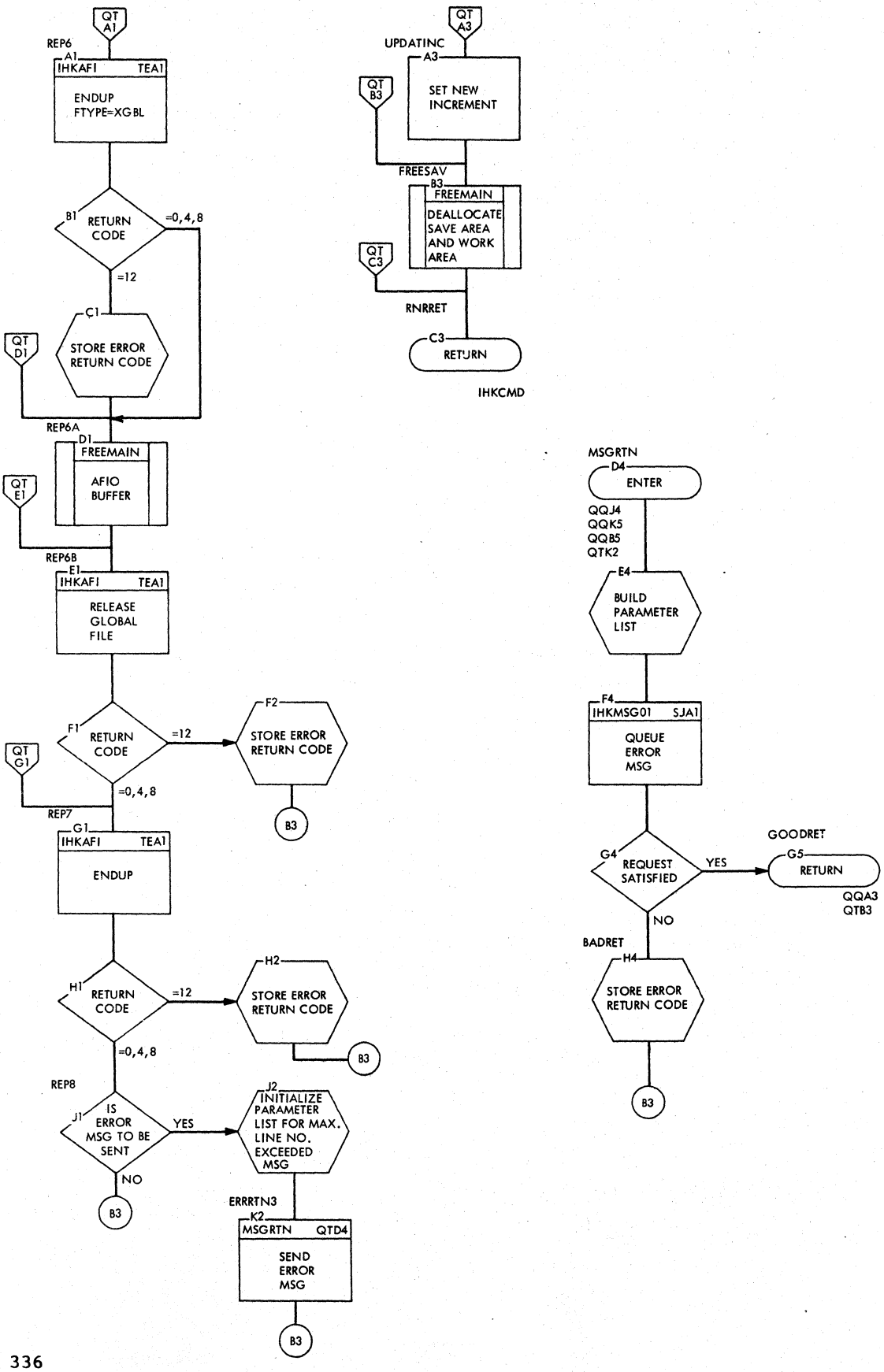


Chart QW. SAVE Subcommand Processor (IHKSAV)

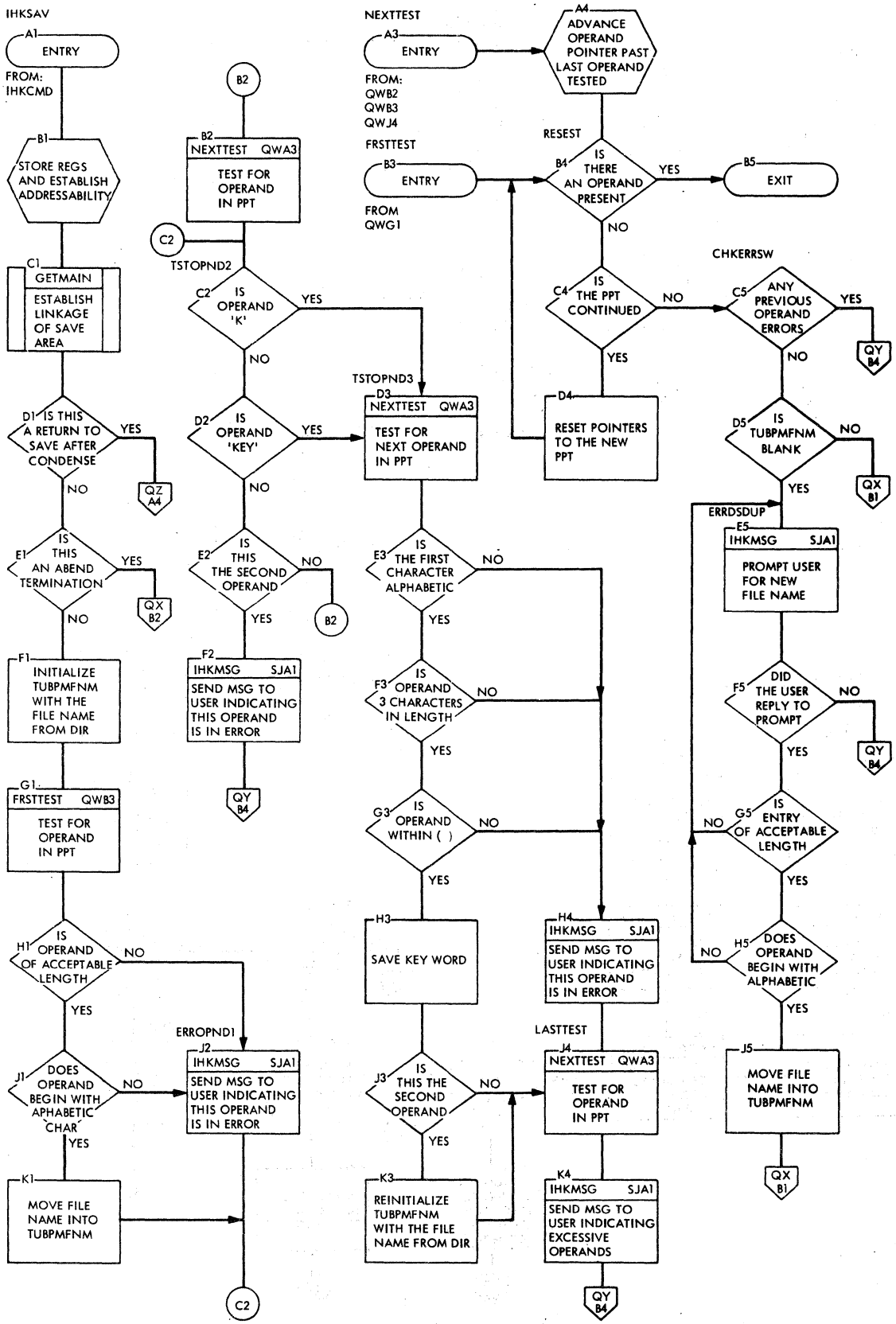


Chart QX. SAVE Subcommand Processor (IHKSAV)

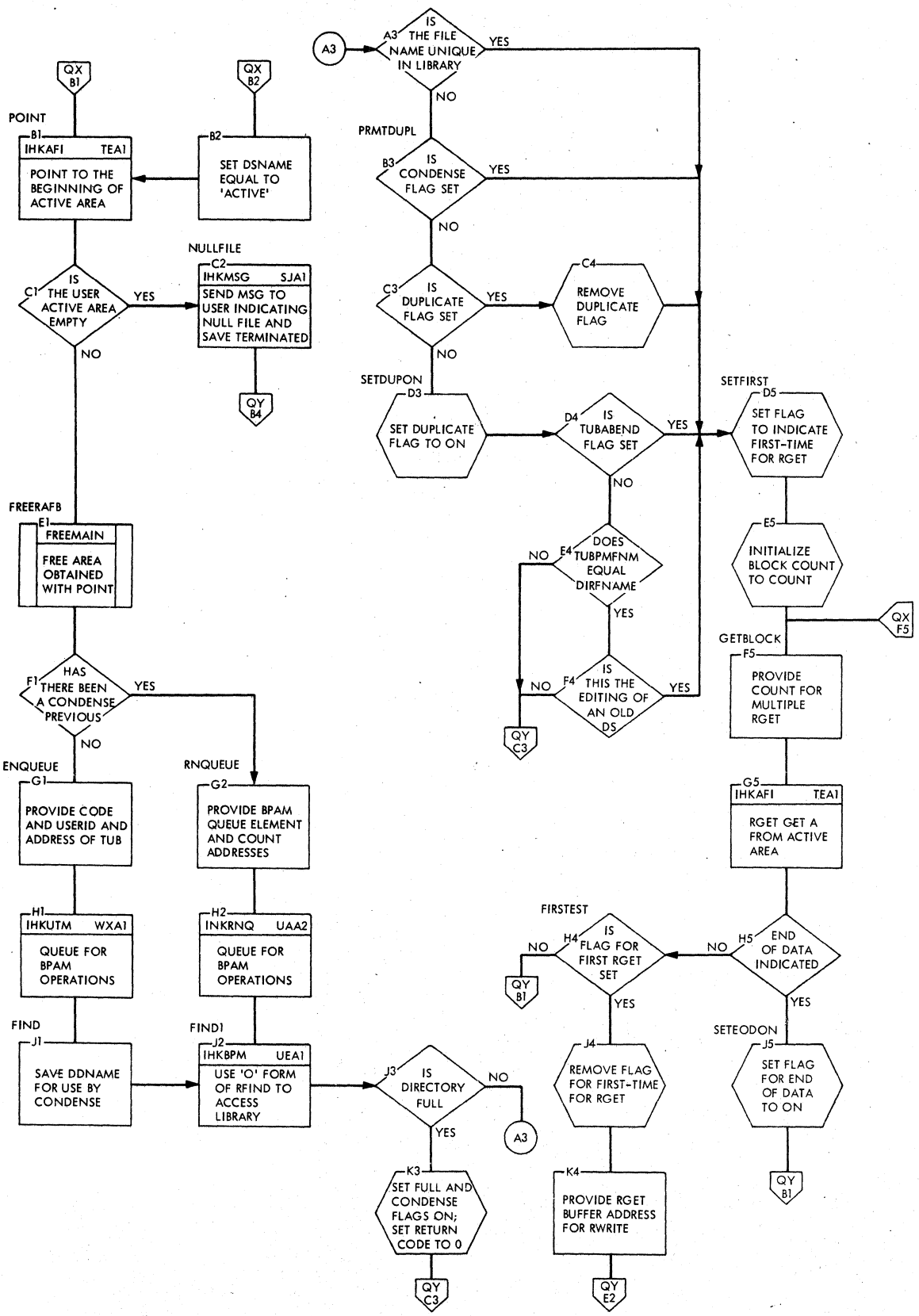


Chart QY. SAVE Subcommand Processor (IHKSAV)

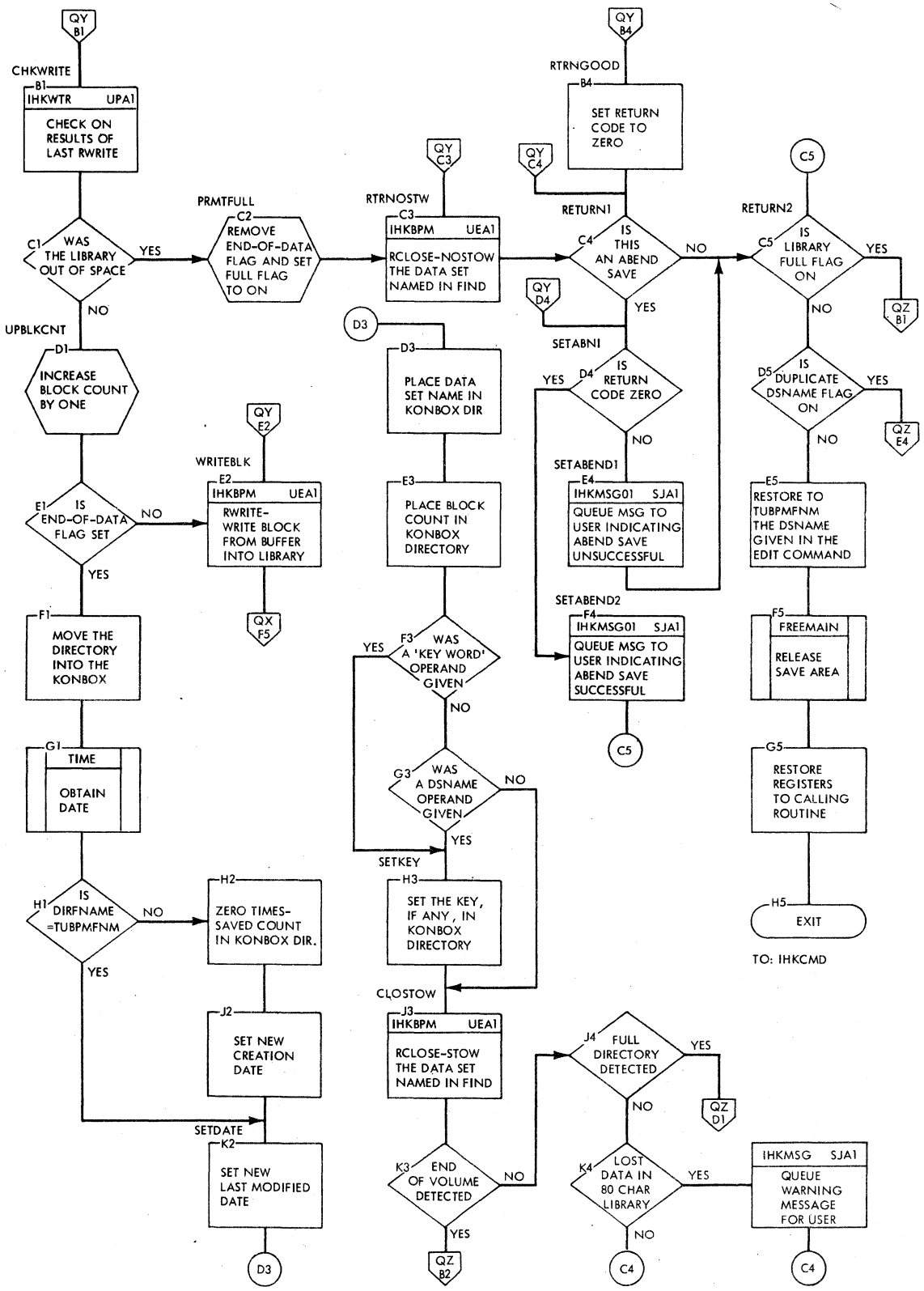


Chart QZ. SAVE Subcommand Processor (IHKSAV)

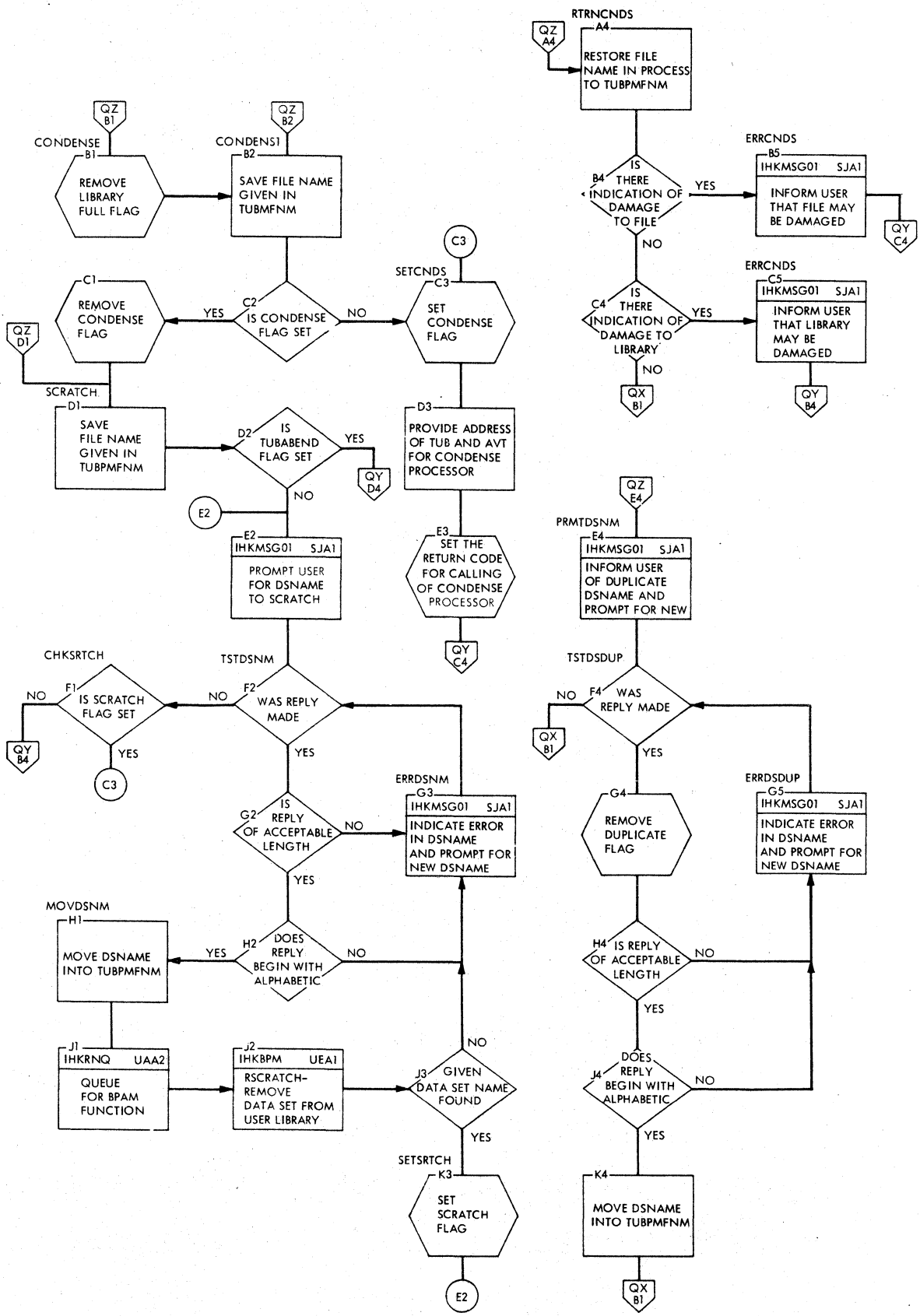


Chart RA. SCAN Subcommand Processor (IHKSCN)

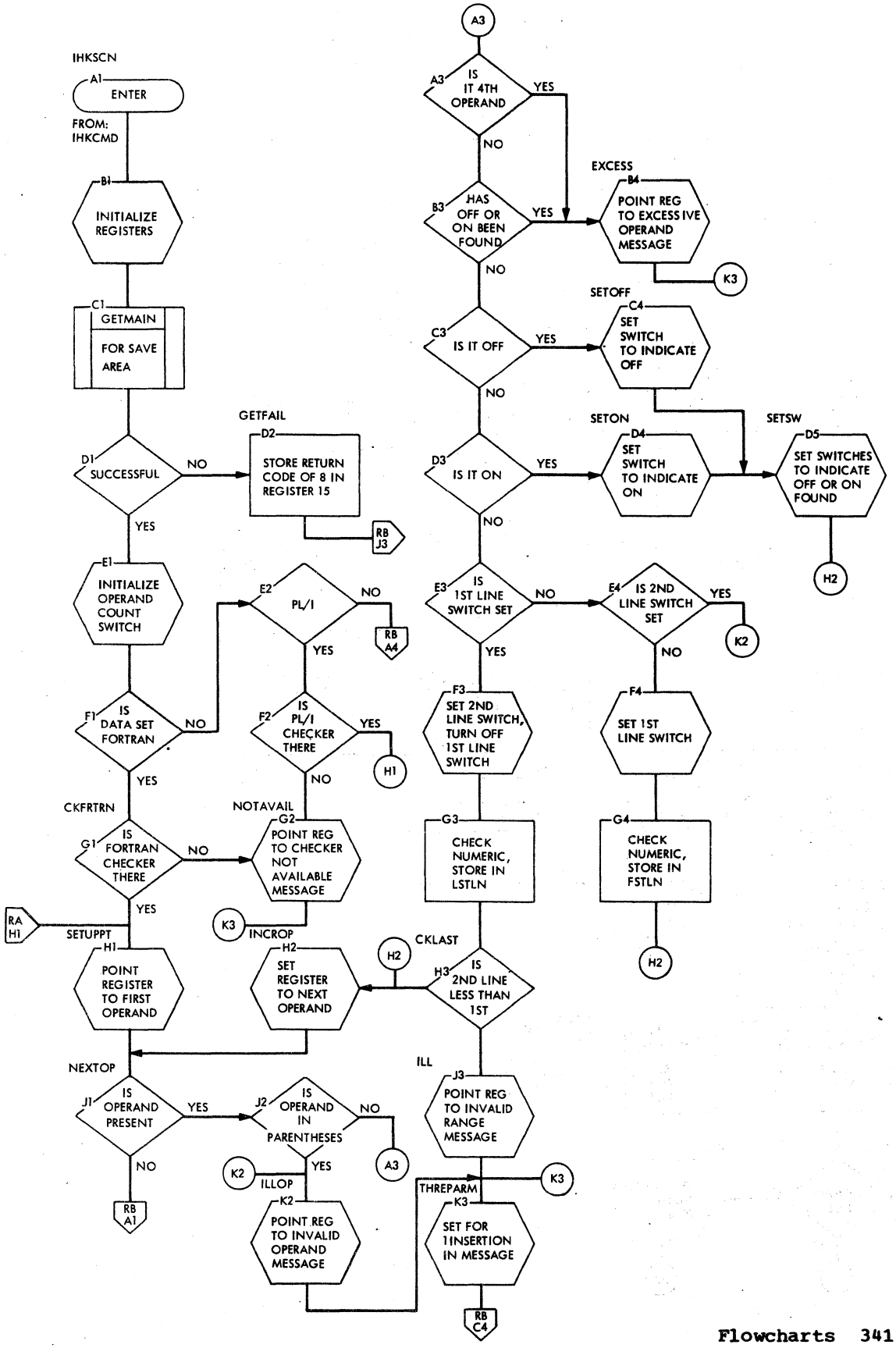


Chart RB. SCAN Subcommand Processor (IHKSCN)

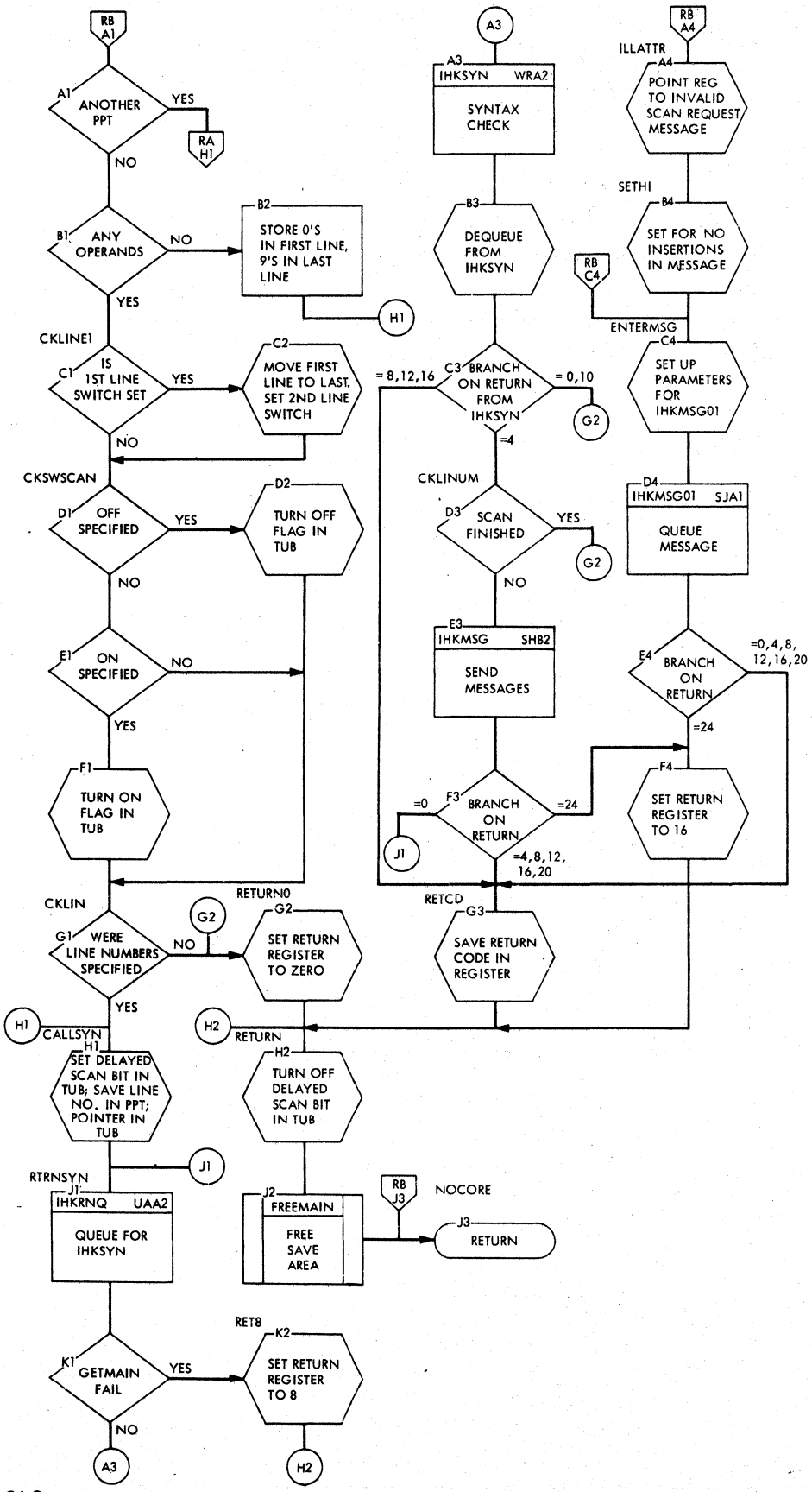


Chart RE. SEND Command Processor (IHKSND)

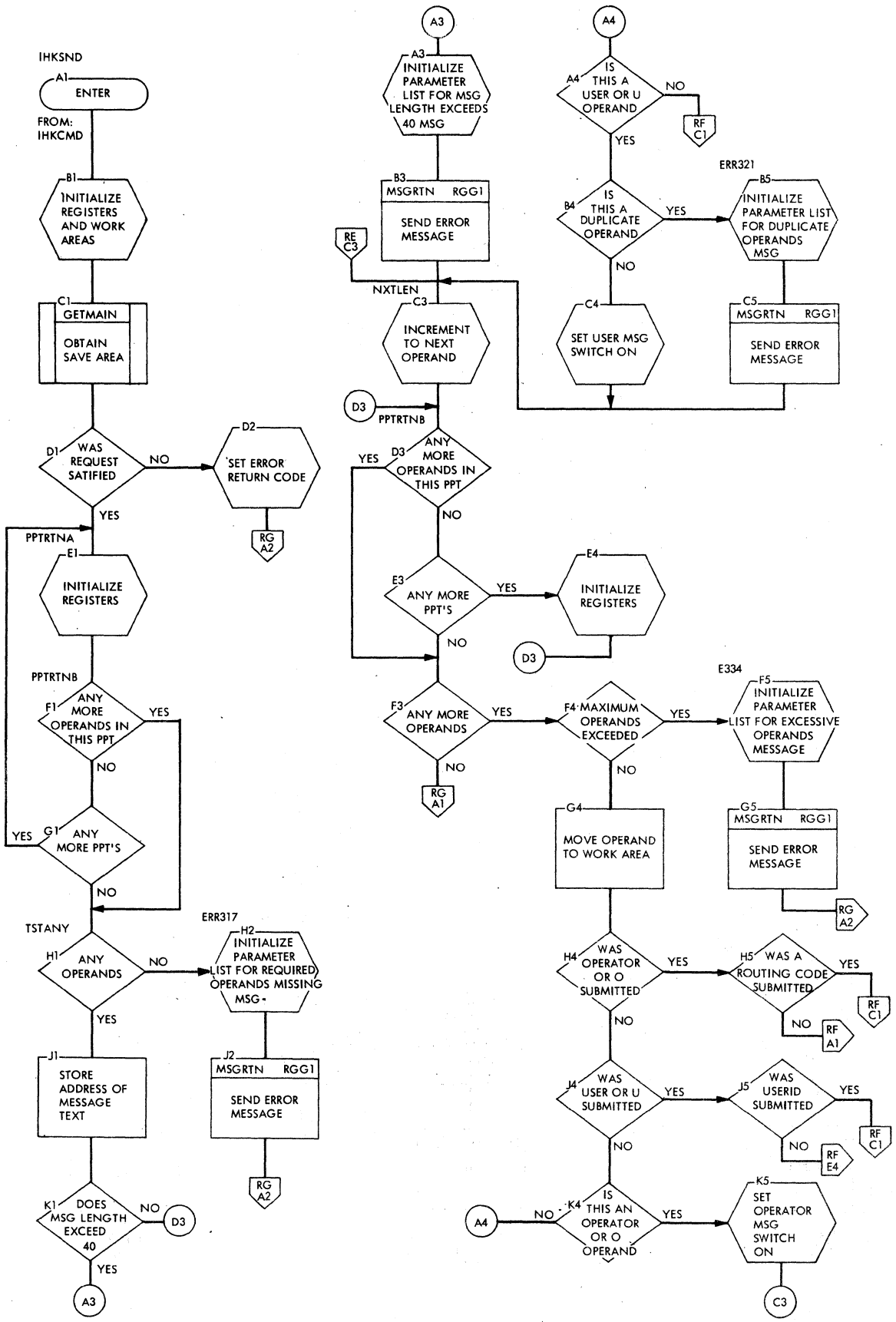


Chart RF. SEND Command Processor (IHKSND)

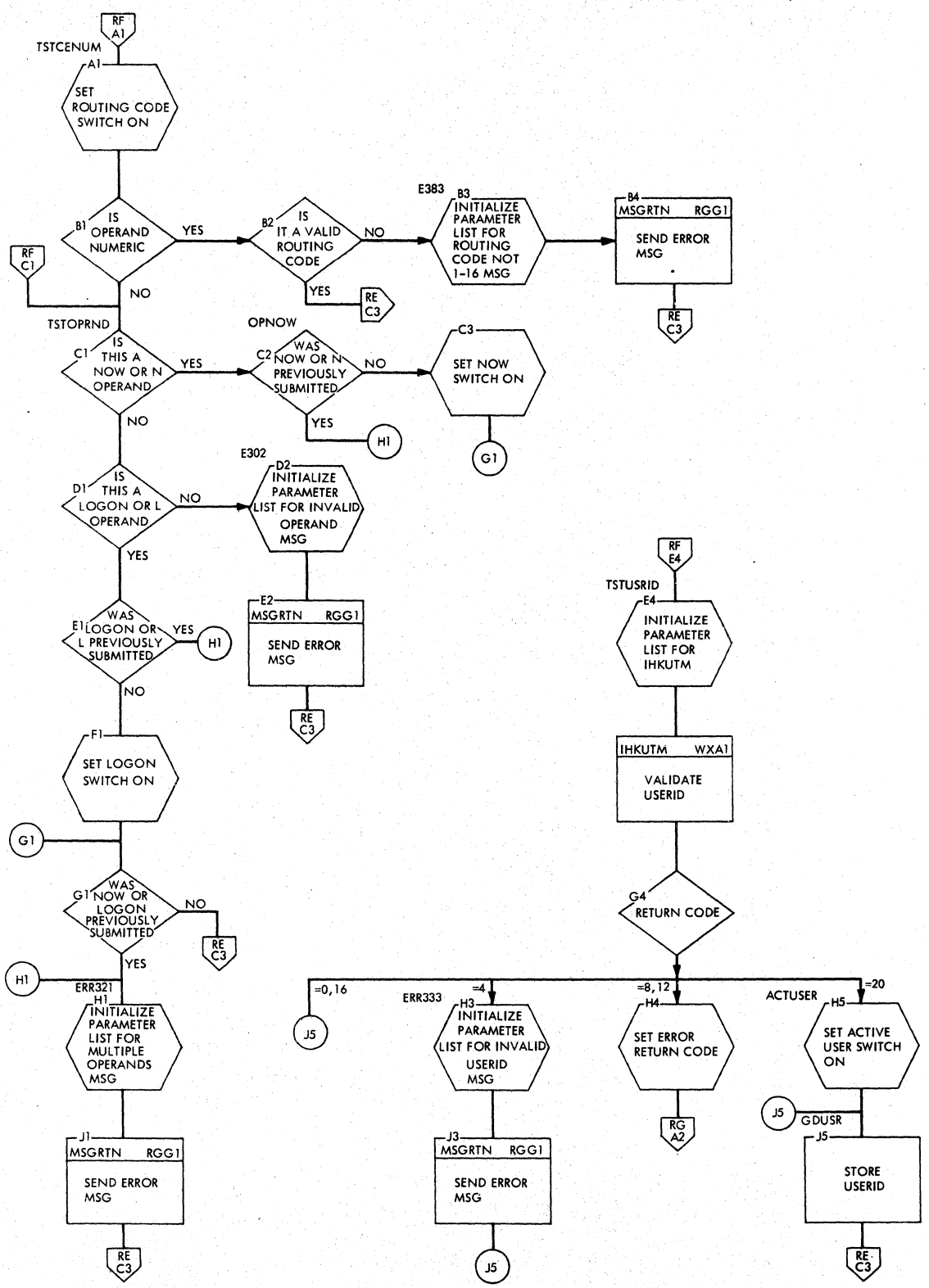


Chart RG. SEND Command Processor (IHKSND)

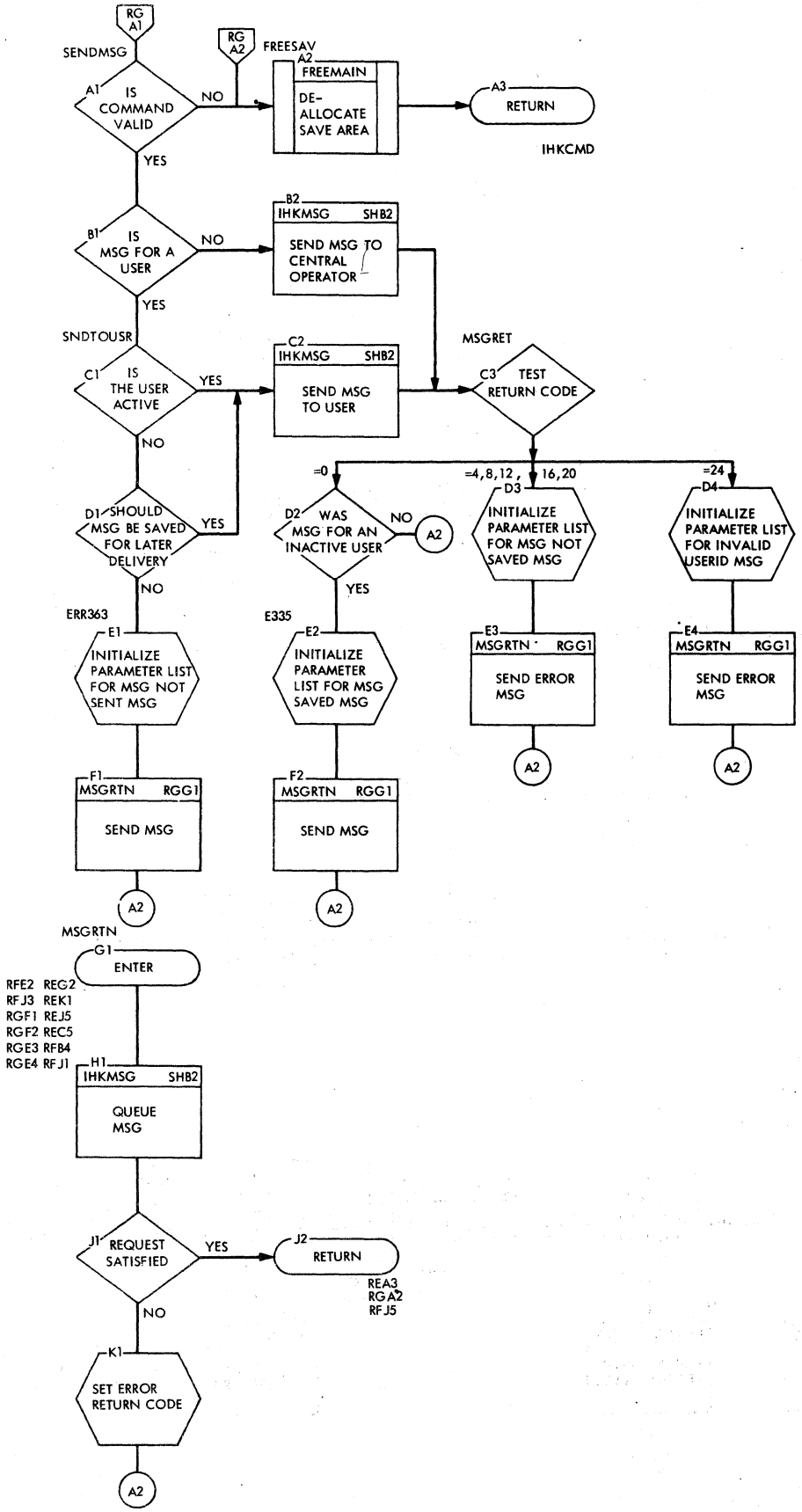
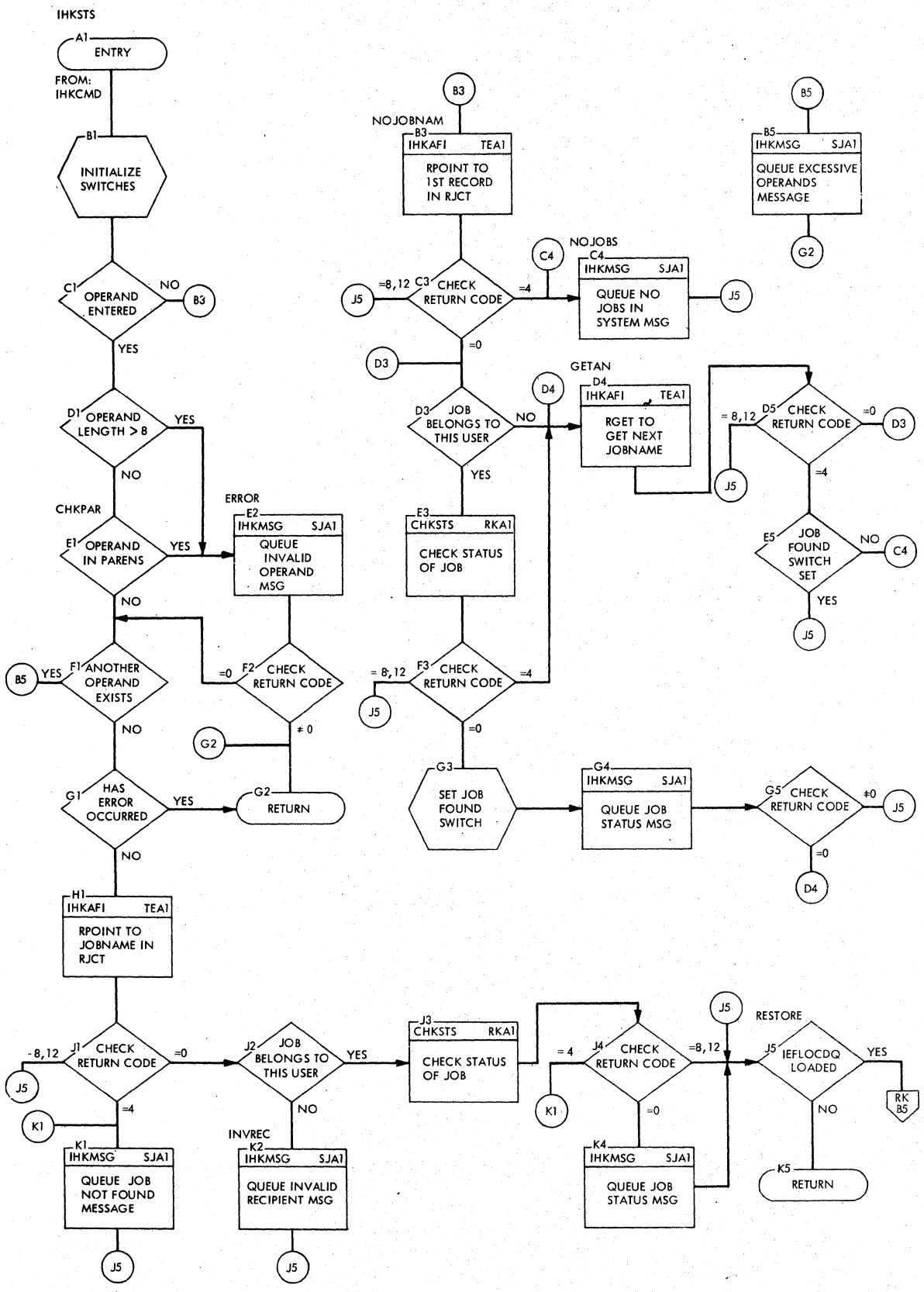


Chart RJ. STATUS Command Processor (IHKSTS)



• Chart RK. STATUS Command Processor (IHKSTS)

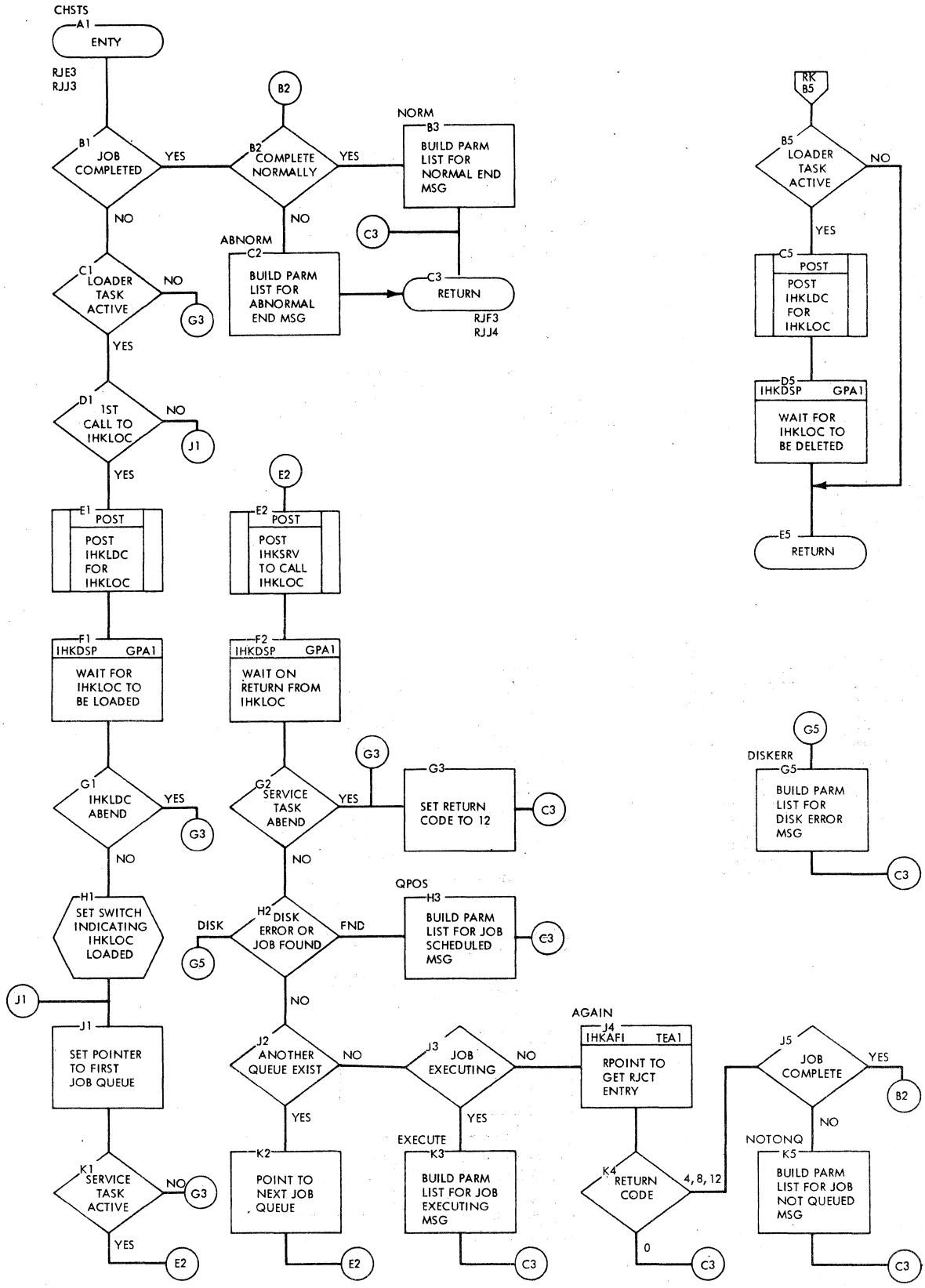


Chart RO. SUBMIT Command Processor (IHKSUB)

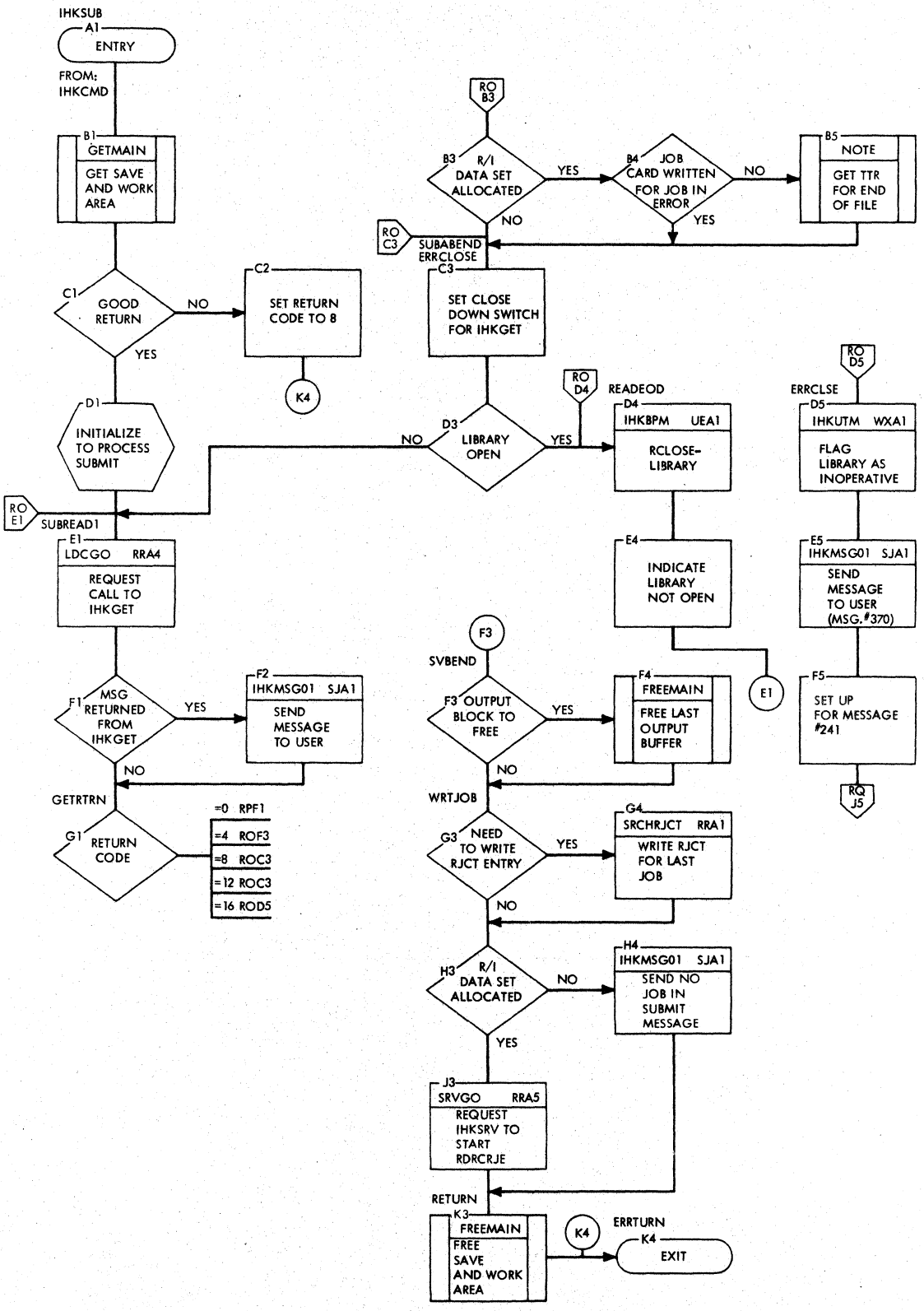


Chart RP. SUBMIT Command Processor (IHKSUB)

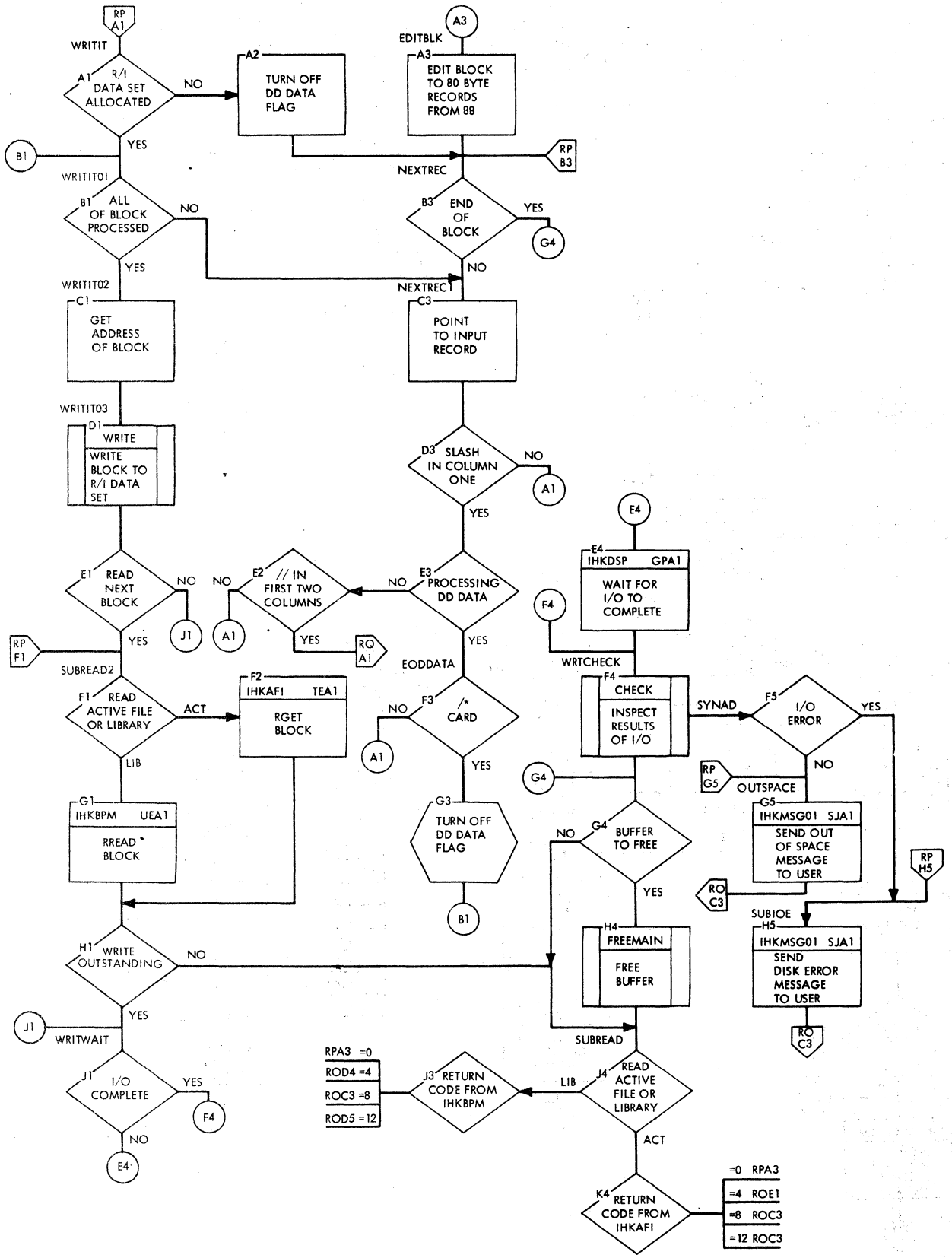
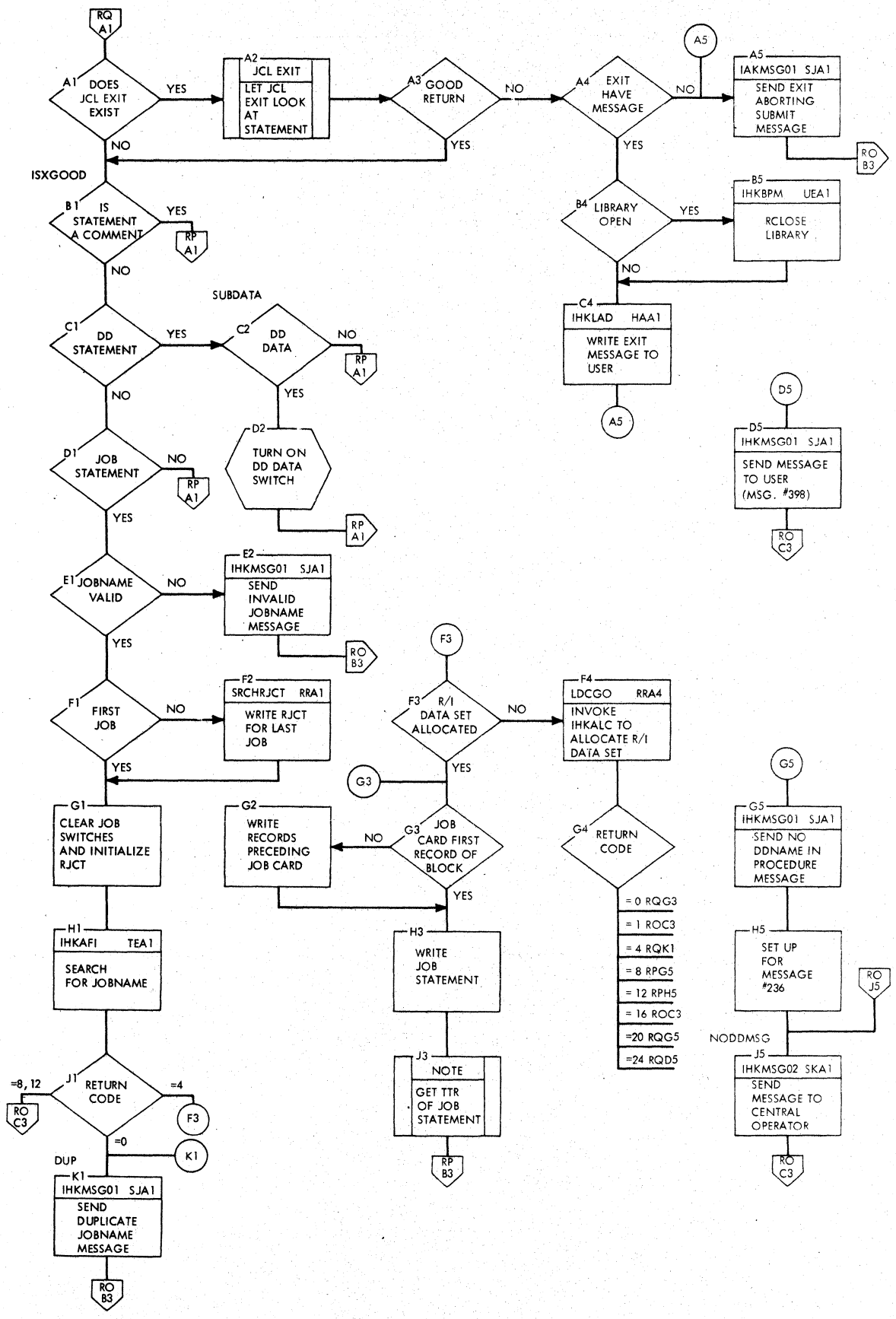


Chart RQ. SUBMIT Command Processor (IHKSUB)



• Chart RR. SUBMIT Command Processor (IHKSUB)

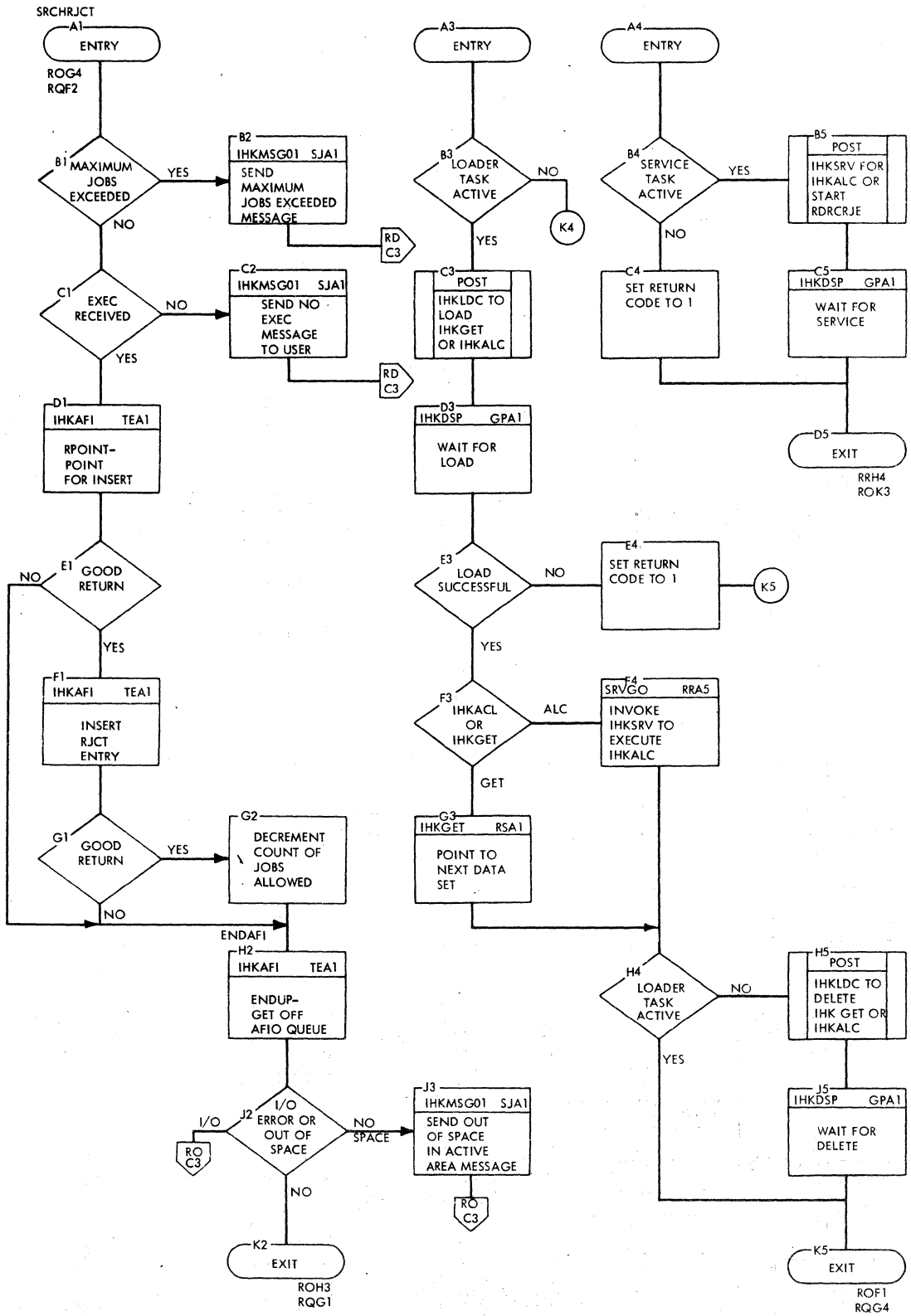


Chart RS. SUBMIT Input Record Processor (IHKGET)

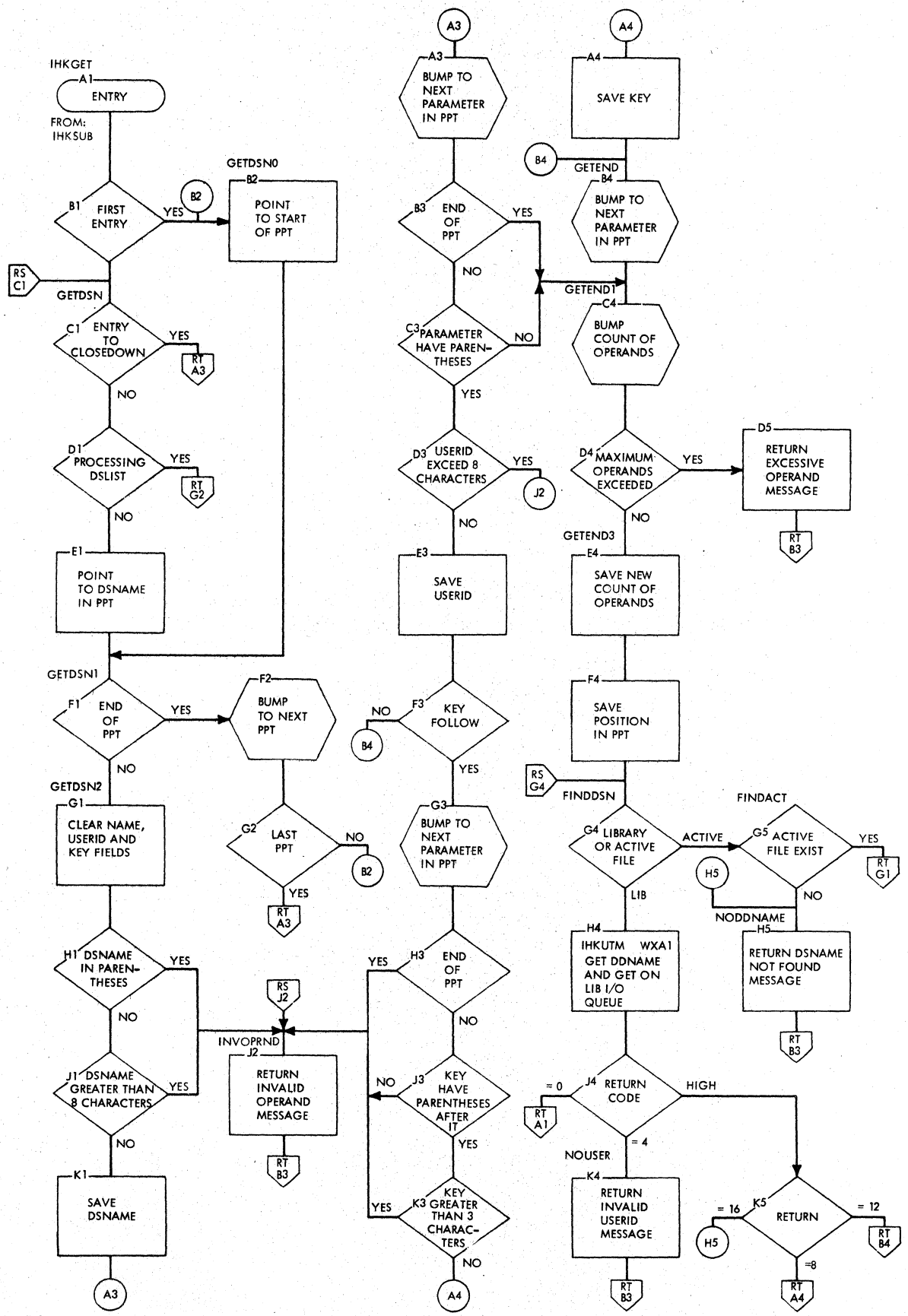


Chart RT. SUBMIT Input Record Processor (IHKGET)

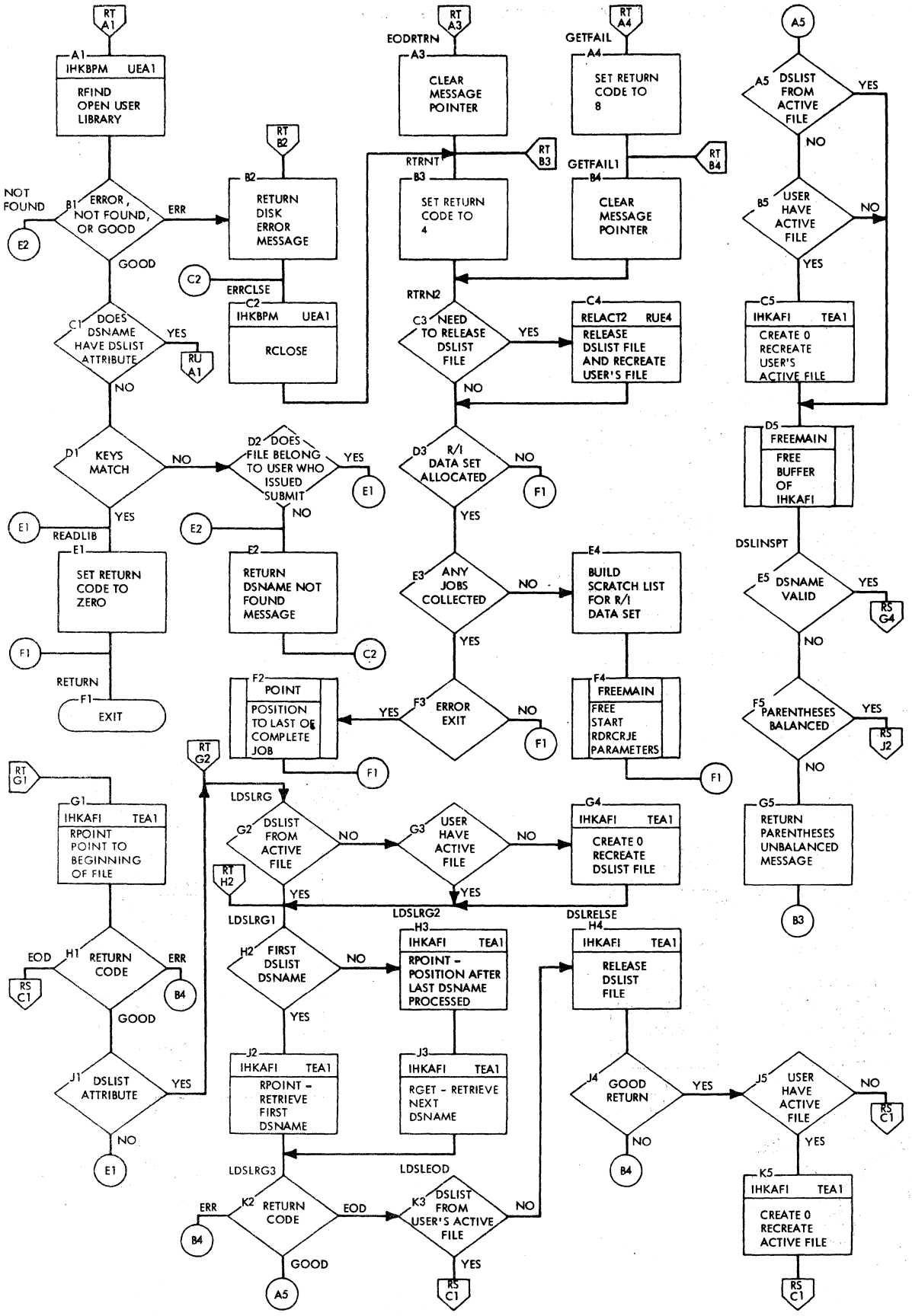


Chart RU. SUBMIT Input Record Processor (IHKGET)

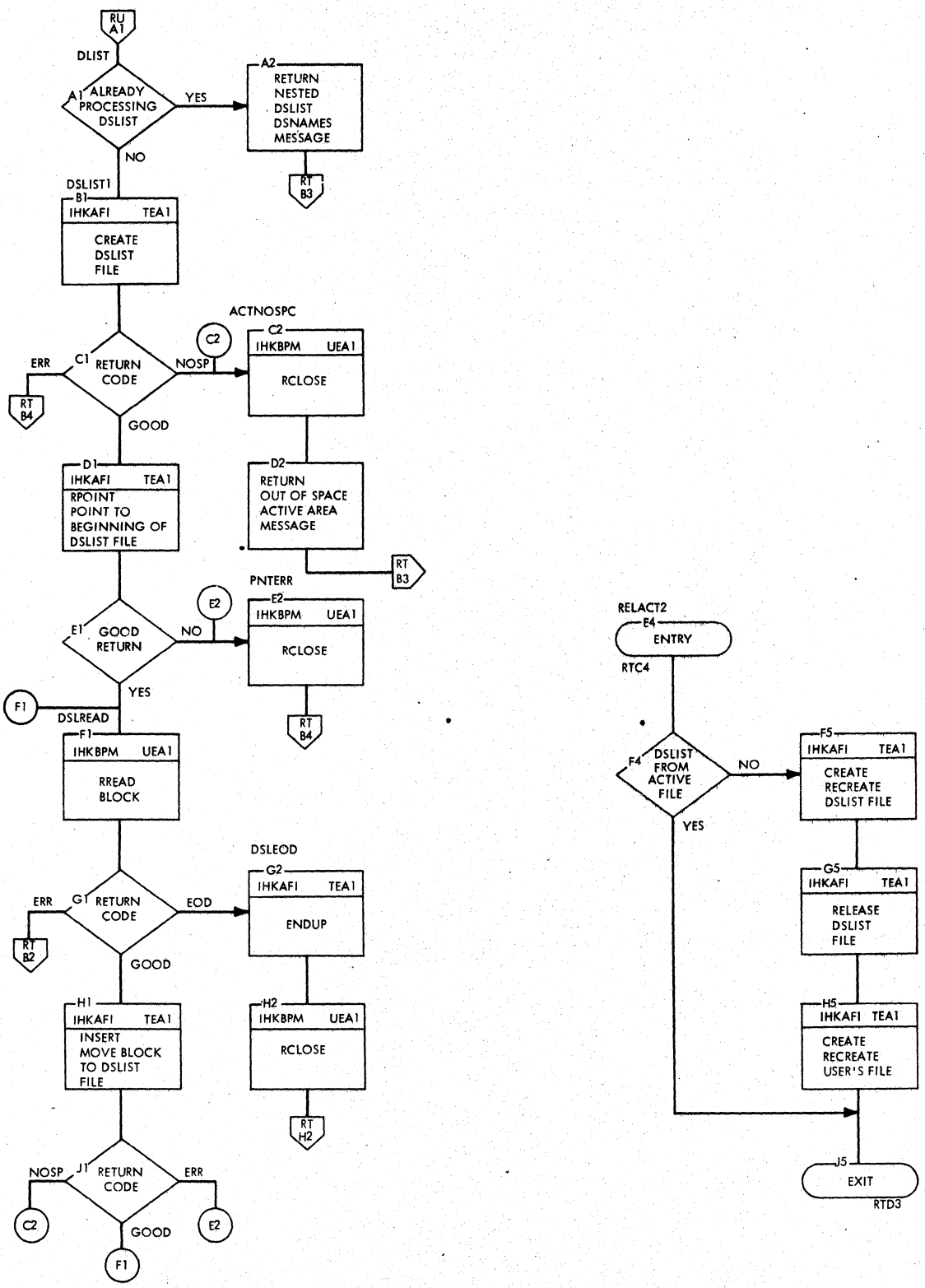


Chart RZ. Allocate Routine (IHKALC)

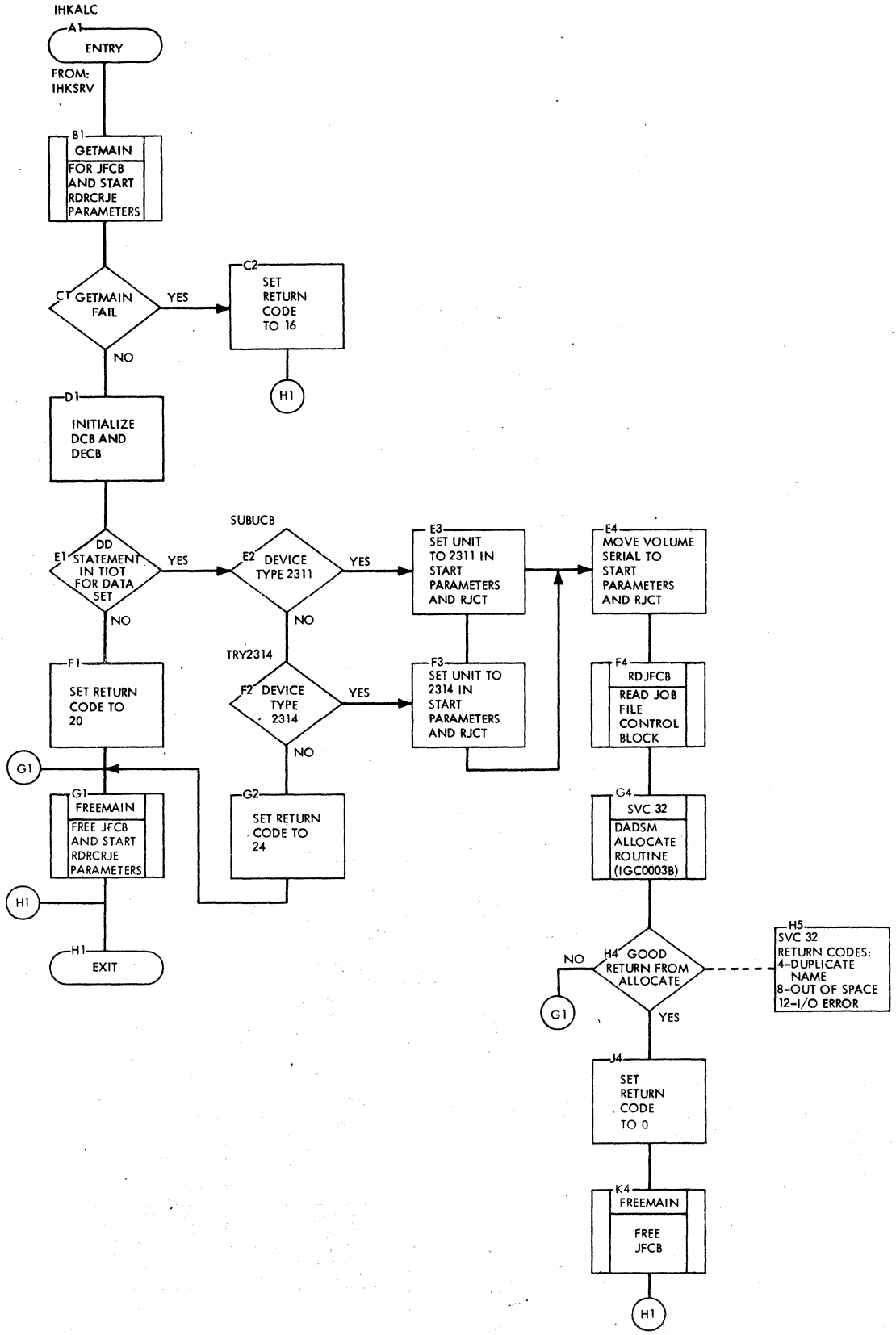


Chart SE. TABSET Command Processor (IHKTAB)

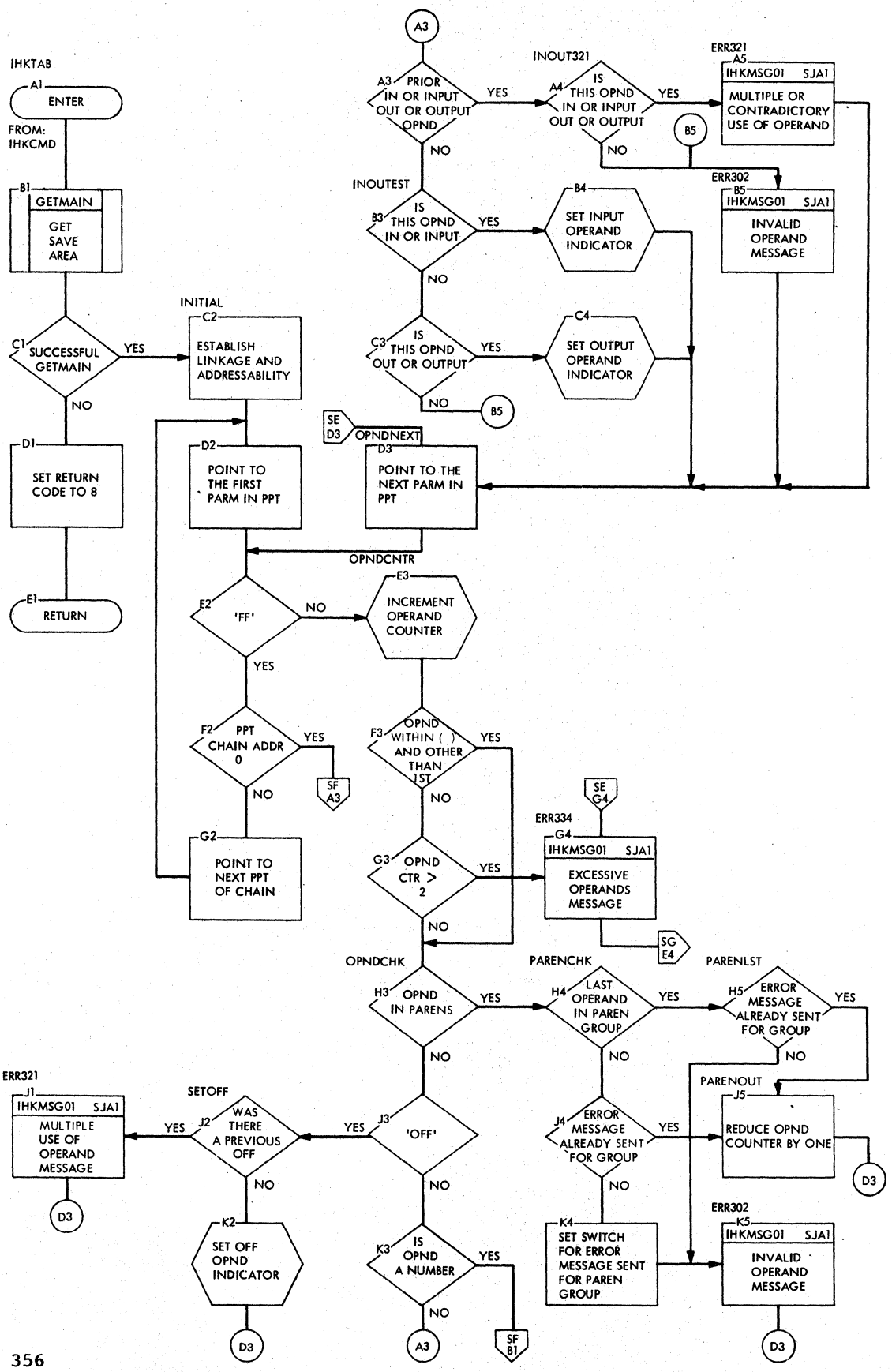


Chart SF. TABSET Command Processor (IHKTAB)

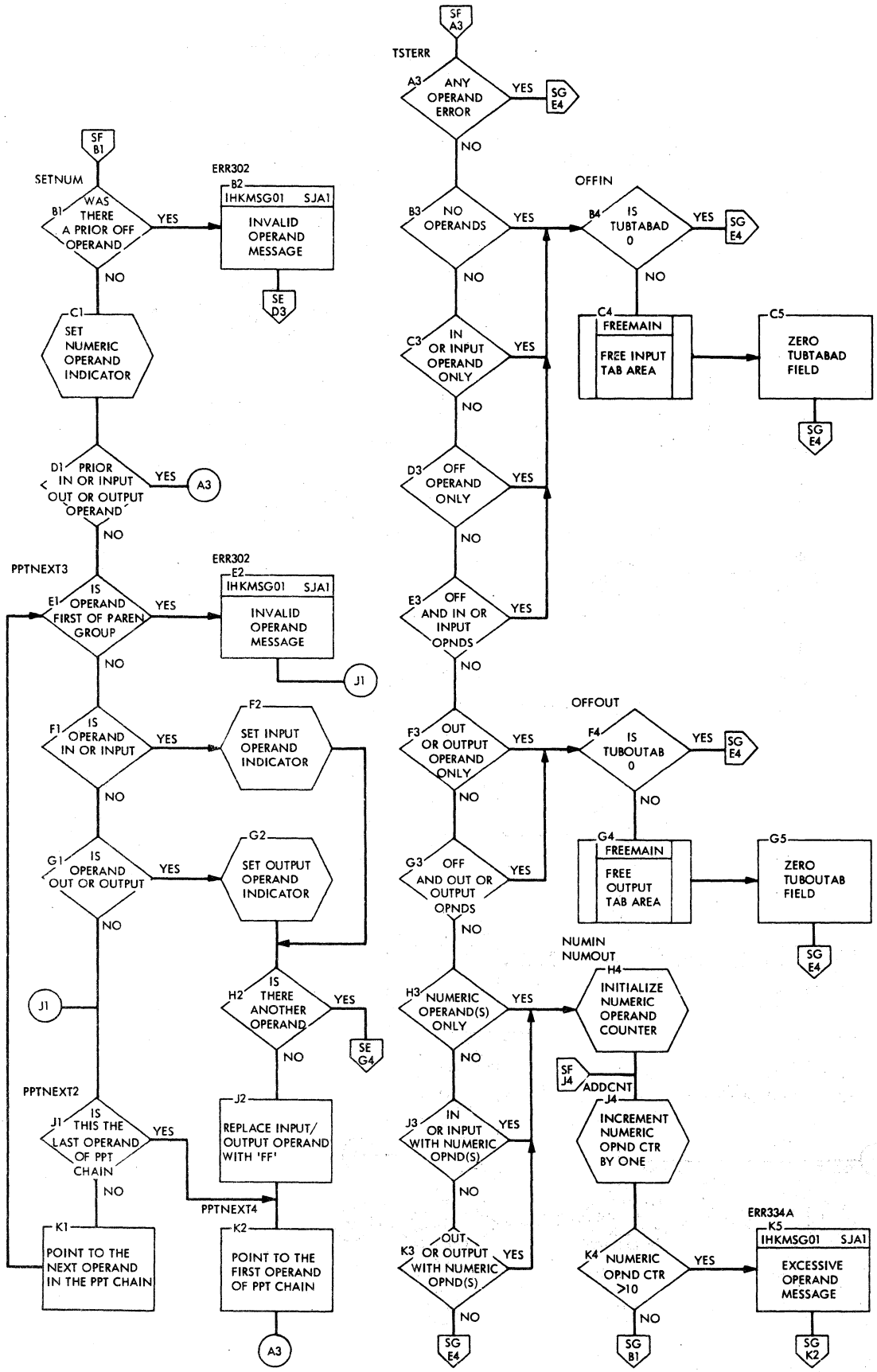
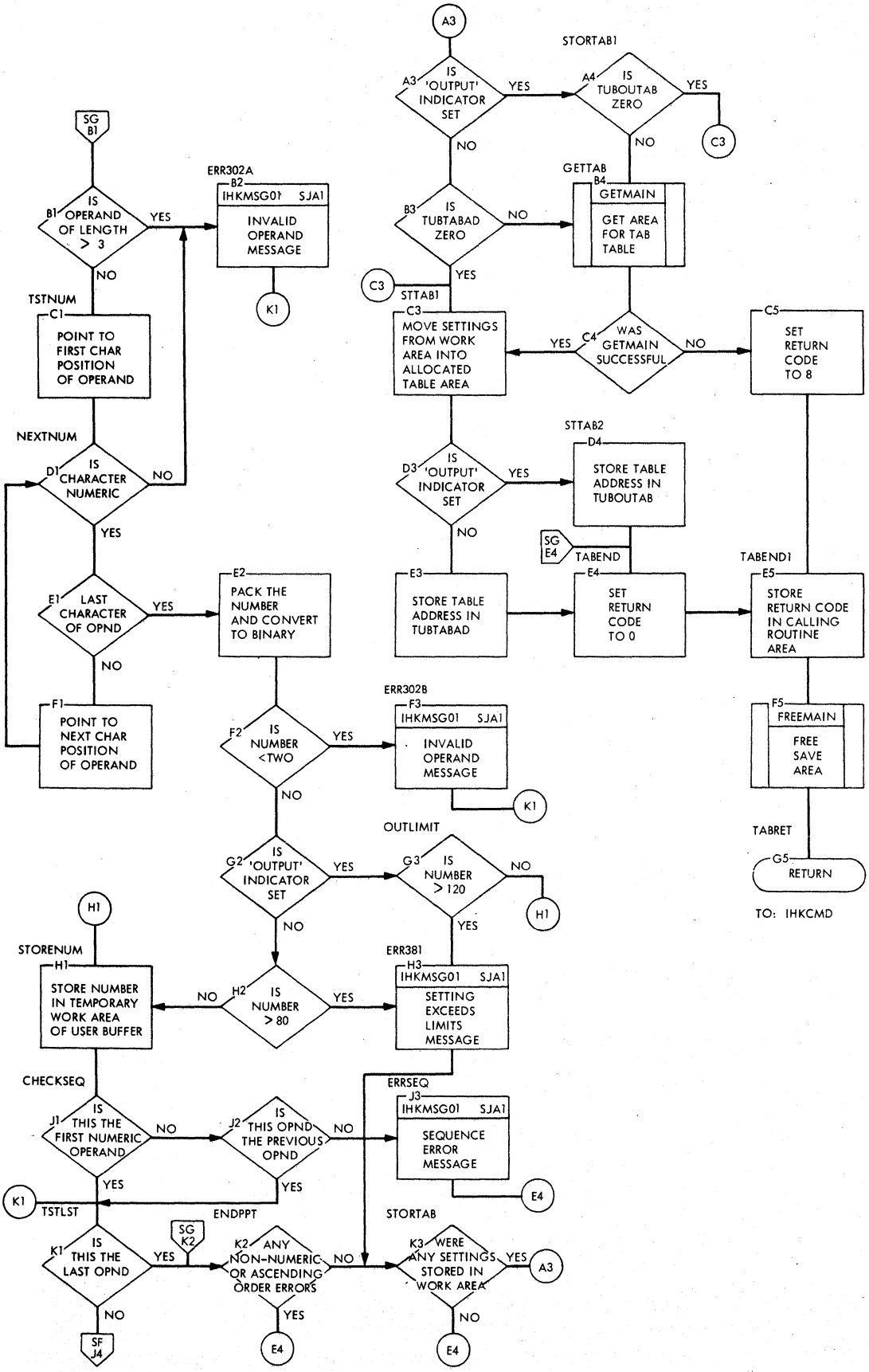


Chart SG. TABSET Command Processor (IHKTAB)



•Chart SH. Message Writer (IHKMSG)

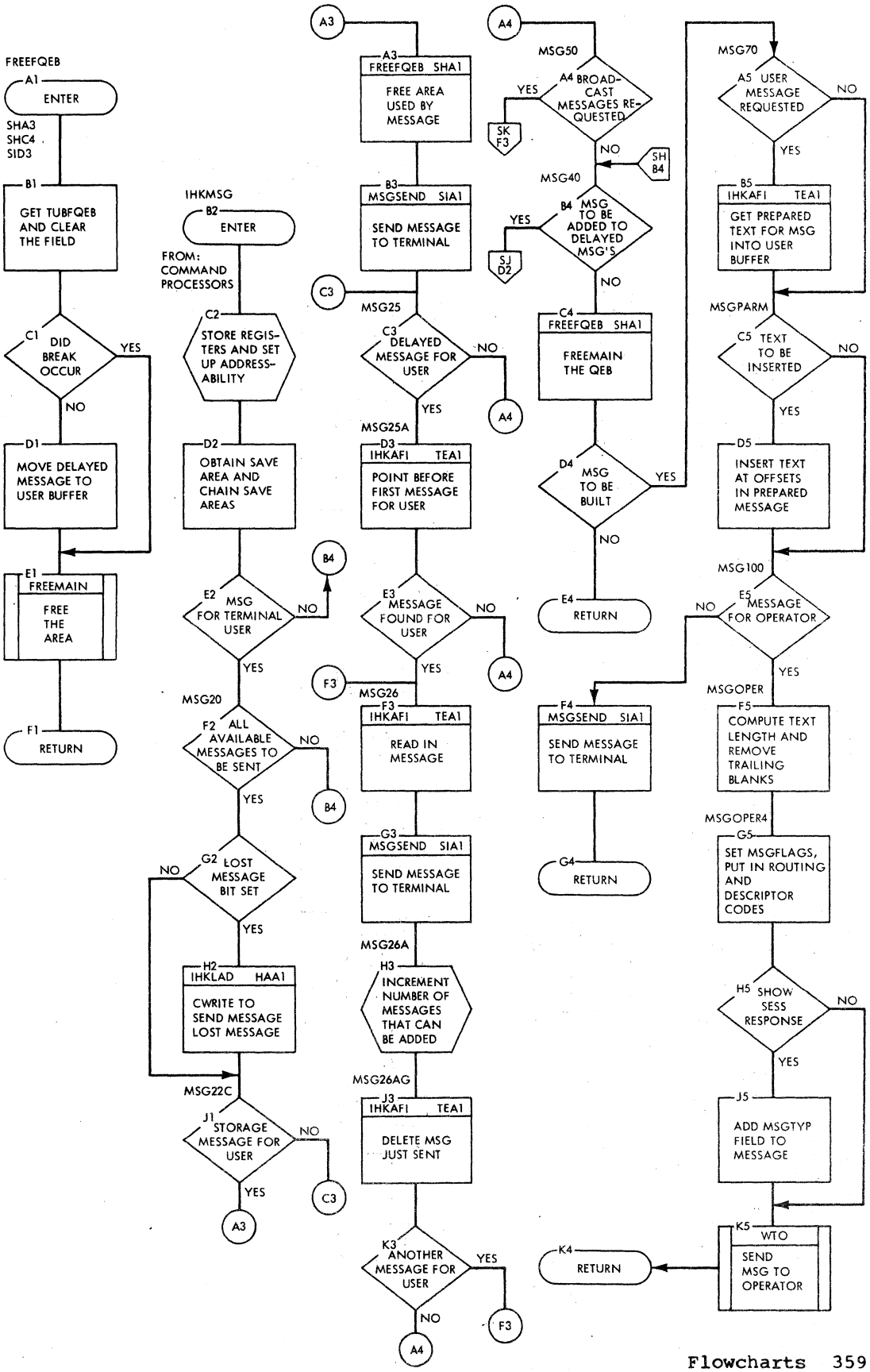


Chart SI. Message Writer (IHKMSG)

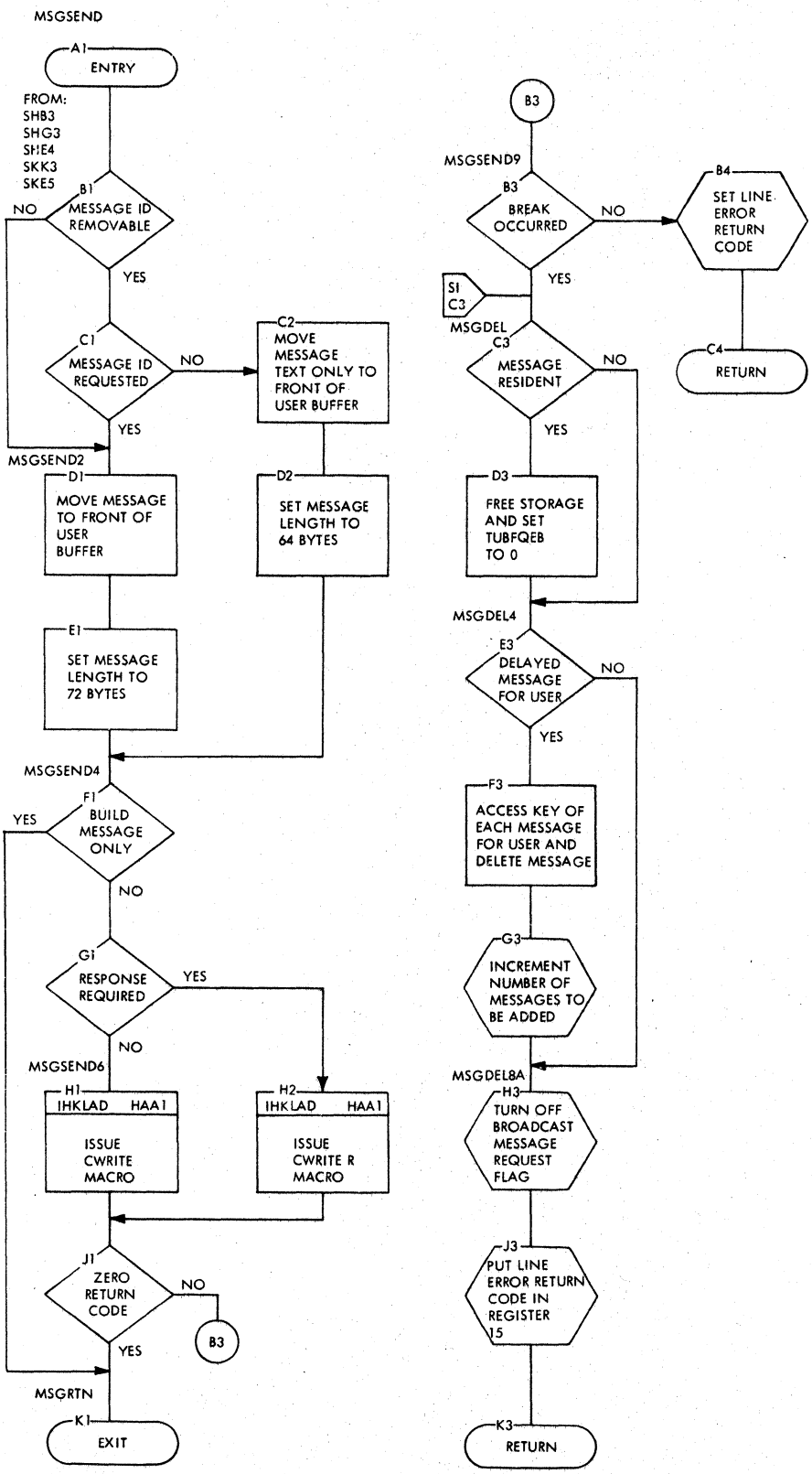


Chart SJ. Message Writer (IHKMSG)

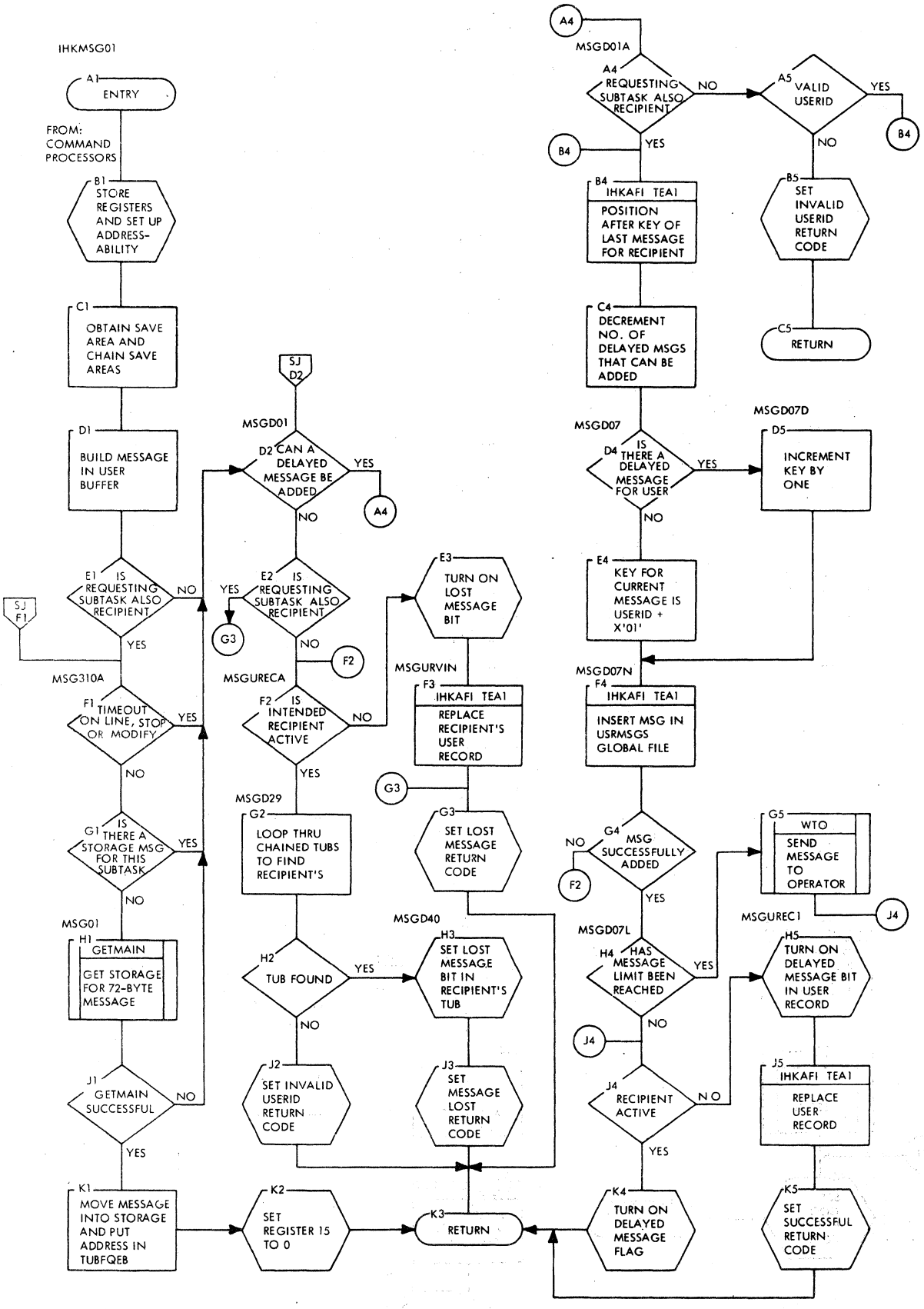


Chart SK. Message Writer (IHKMSG)

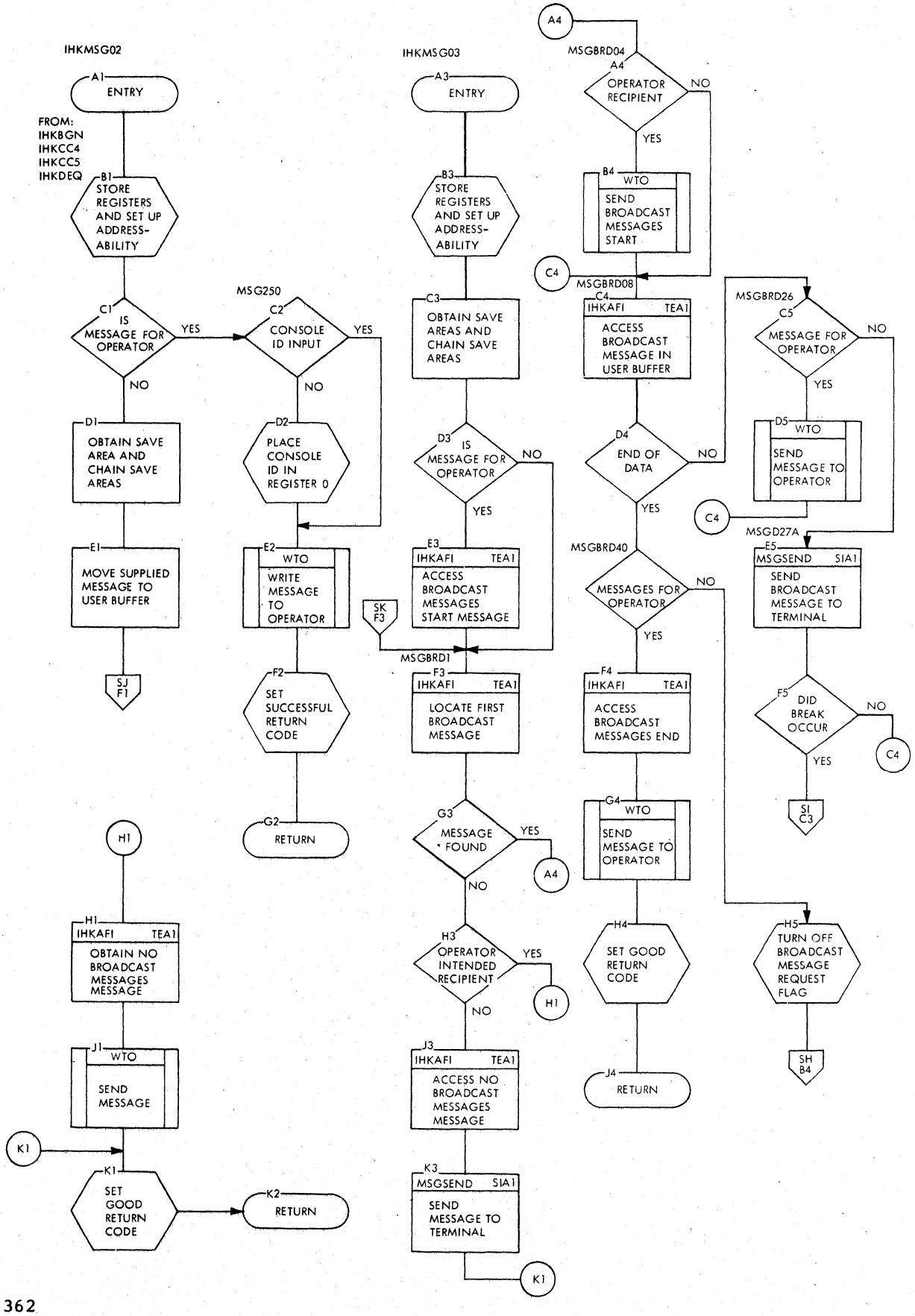


Chart TE. Active Area I/O Control/Command Interpreter (IHKAFI)

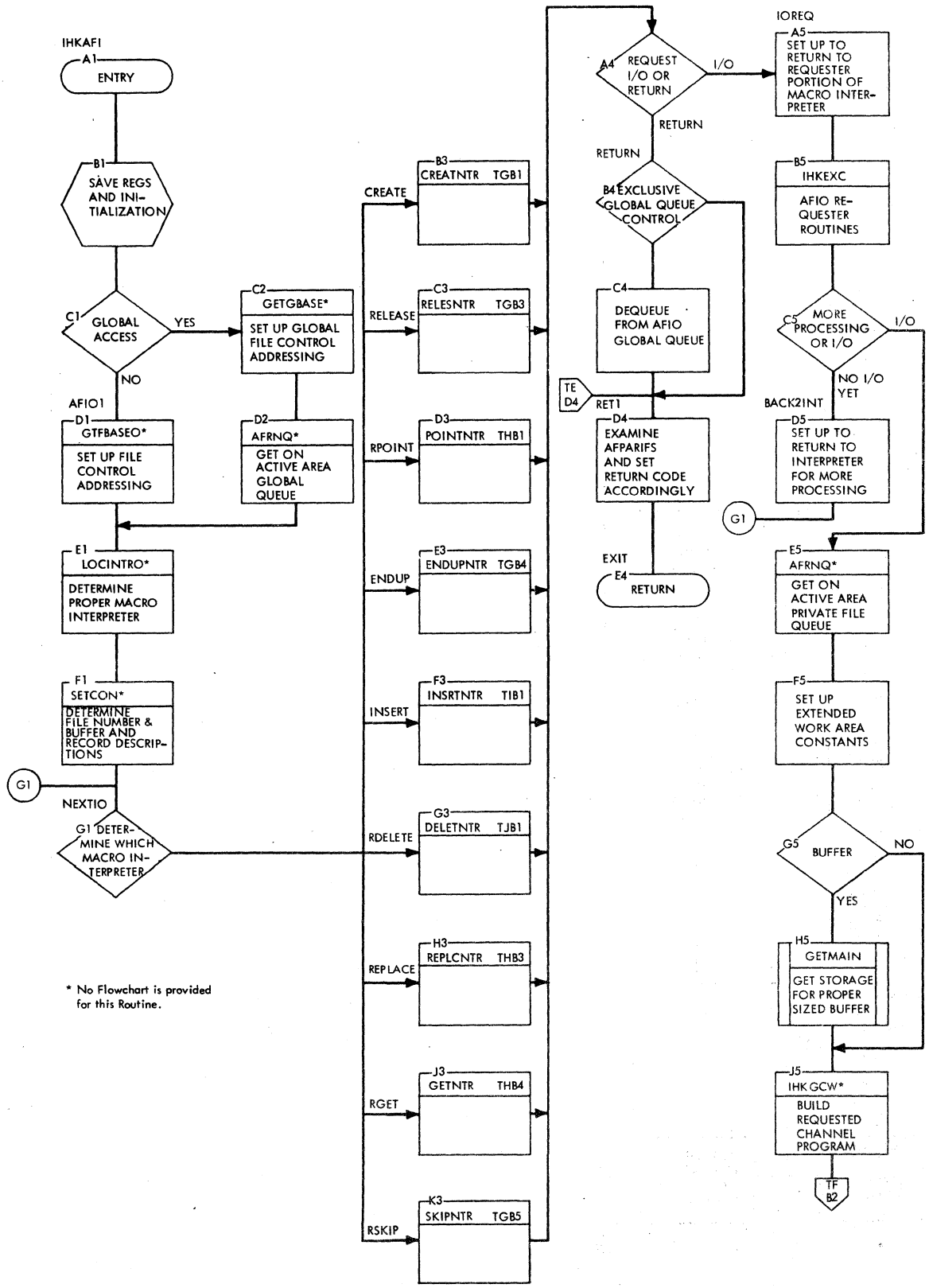


Chart TF. Active Area I/O Control/Command Interpreter (IHKAFI)

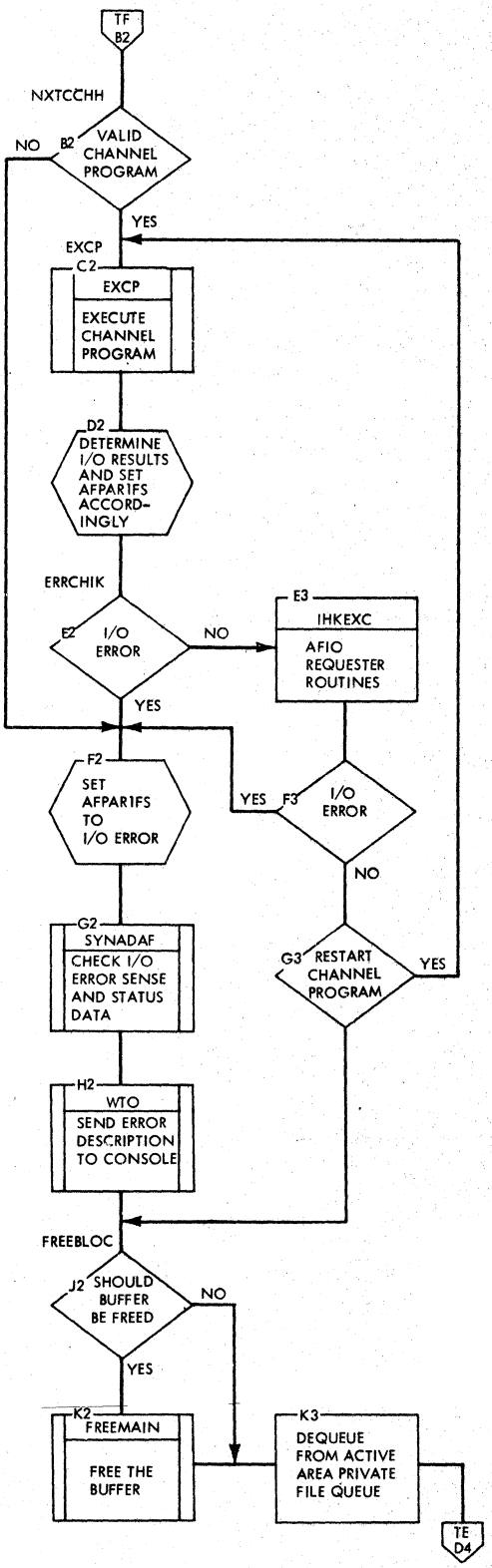
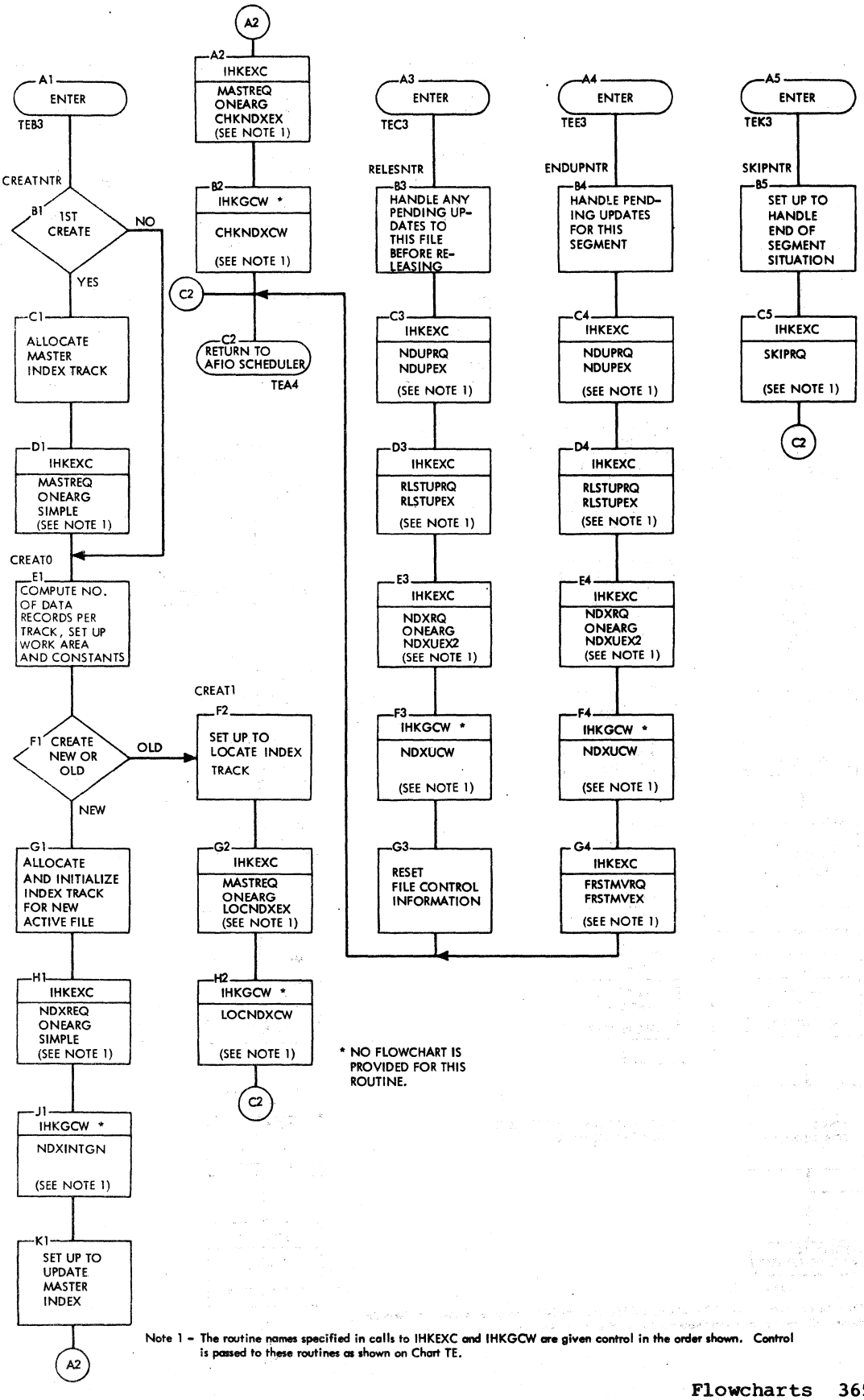
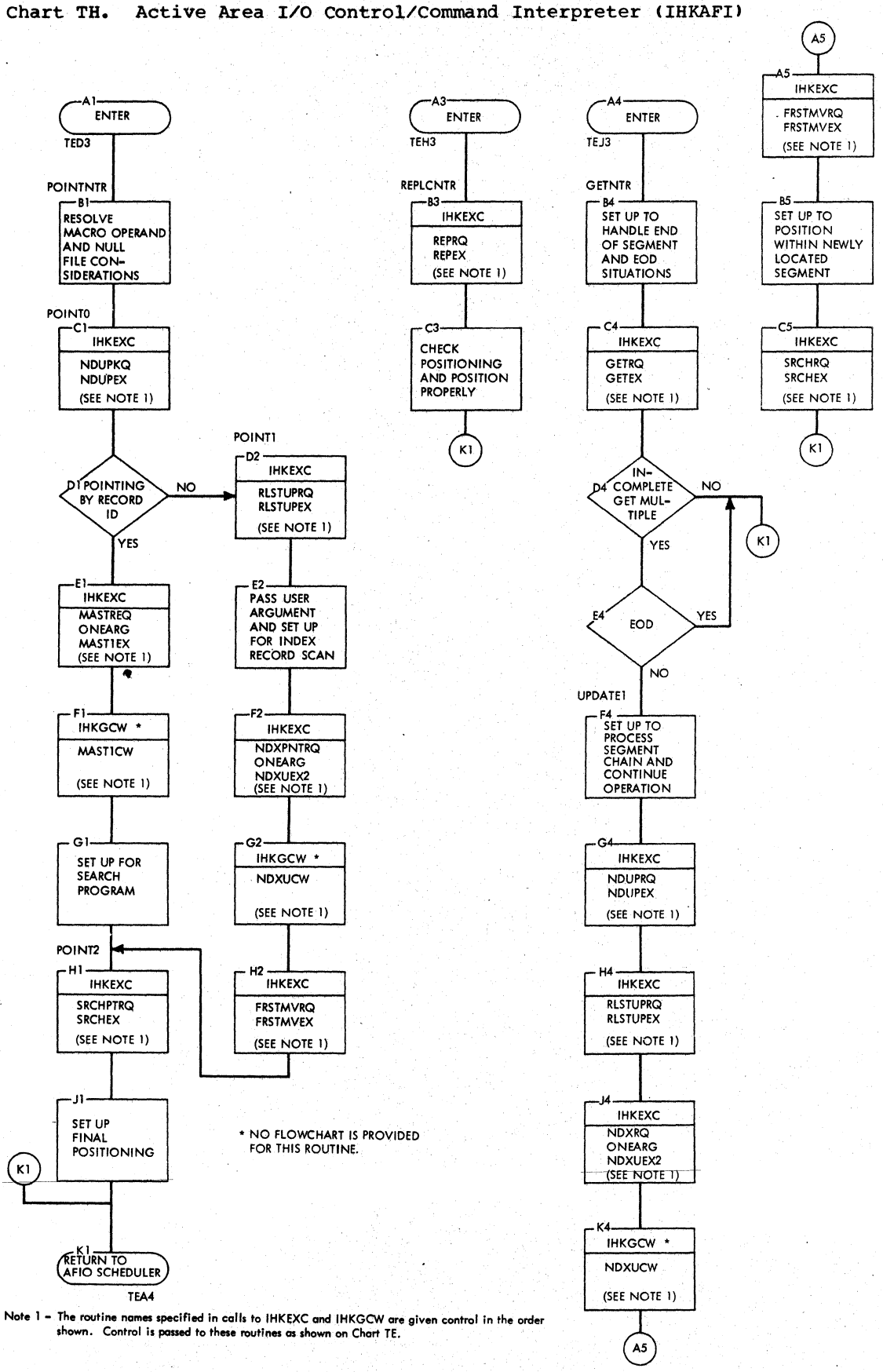


Chart TG. Active Area I/O Control/Command Interpreter (IHKAFI)



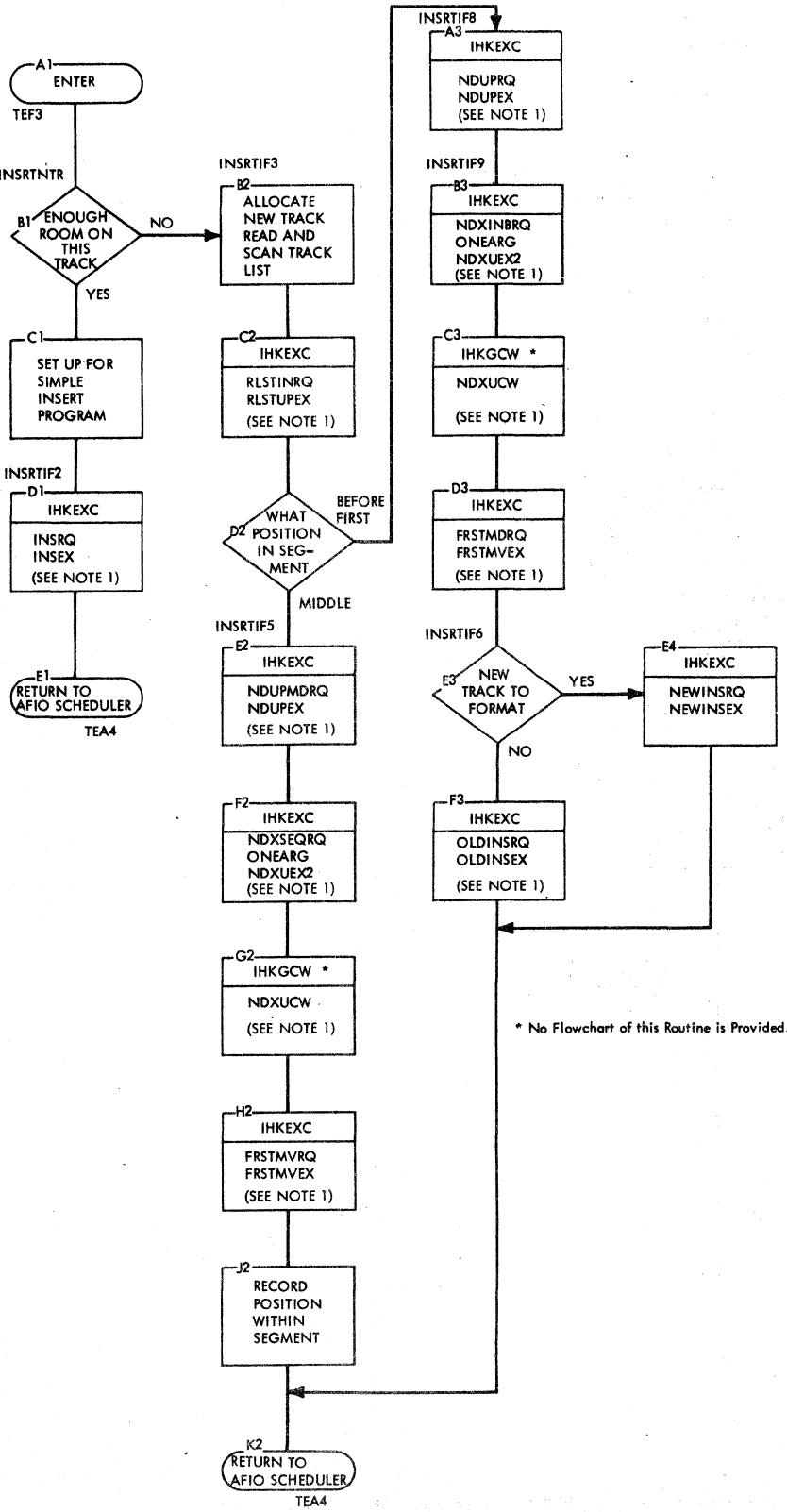
Note 1 - The routine names specified in calls to IHKEXC and IHKGCW are given control in the order shown. Control is passed to these routines as shown on Chart TE.

Chart TH. Active Area I/O Control/Command Interpreter (IHKAFI)



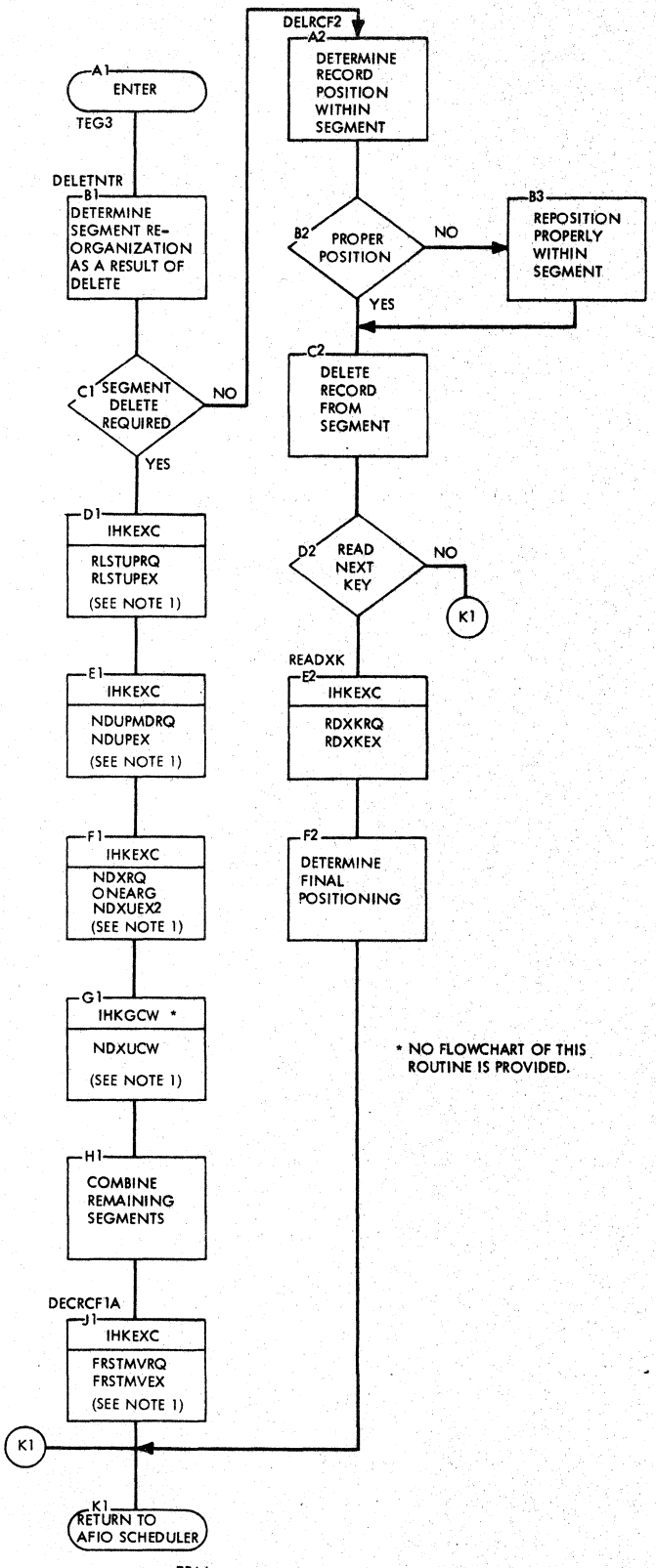
Note 1 - The routine names specified in calls to IHKEXC and IHKGCW are given control in the order shown. Control is passed to these routines as shown on Chart TE.

Chart TI. Active Area I/O Control/Command Interpreter (IHKAFI)



Note 1 – The routine names specified in calls to IHKEXC and IHKGCW are given control in the order shown. Control is passed to these routines as shown on Chart TE.

Chart TJ. Active Area I/O Control/Command Interpreter (IHKAFI)



Note 1 – The routine names specified in the calls to IHKEXC and IHKGCW are given control in the order shown. Control is passed to these routines as shown on Chart TE.

Chart TK. Active Area I/O Requester/Executor (IHKEXC)

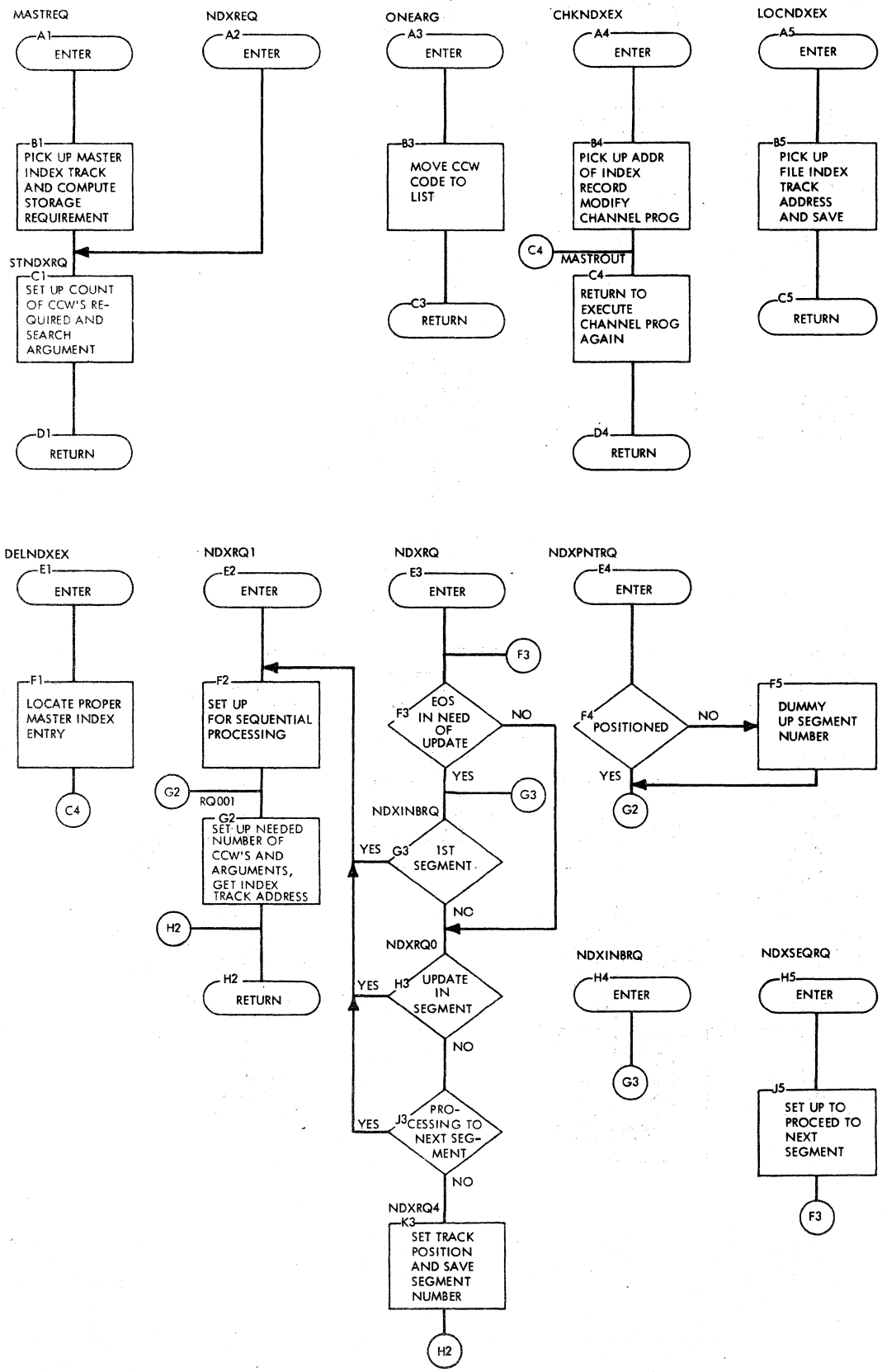


Chart TL. Active Area I/O Requester/Executor (IHKEXC)

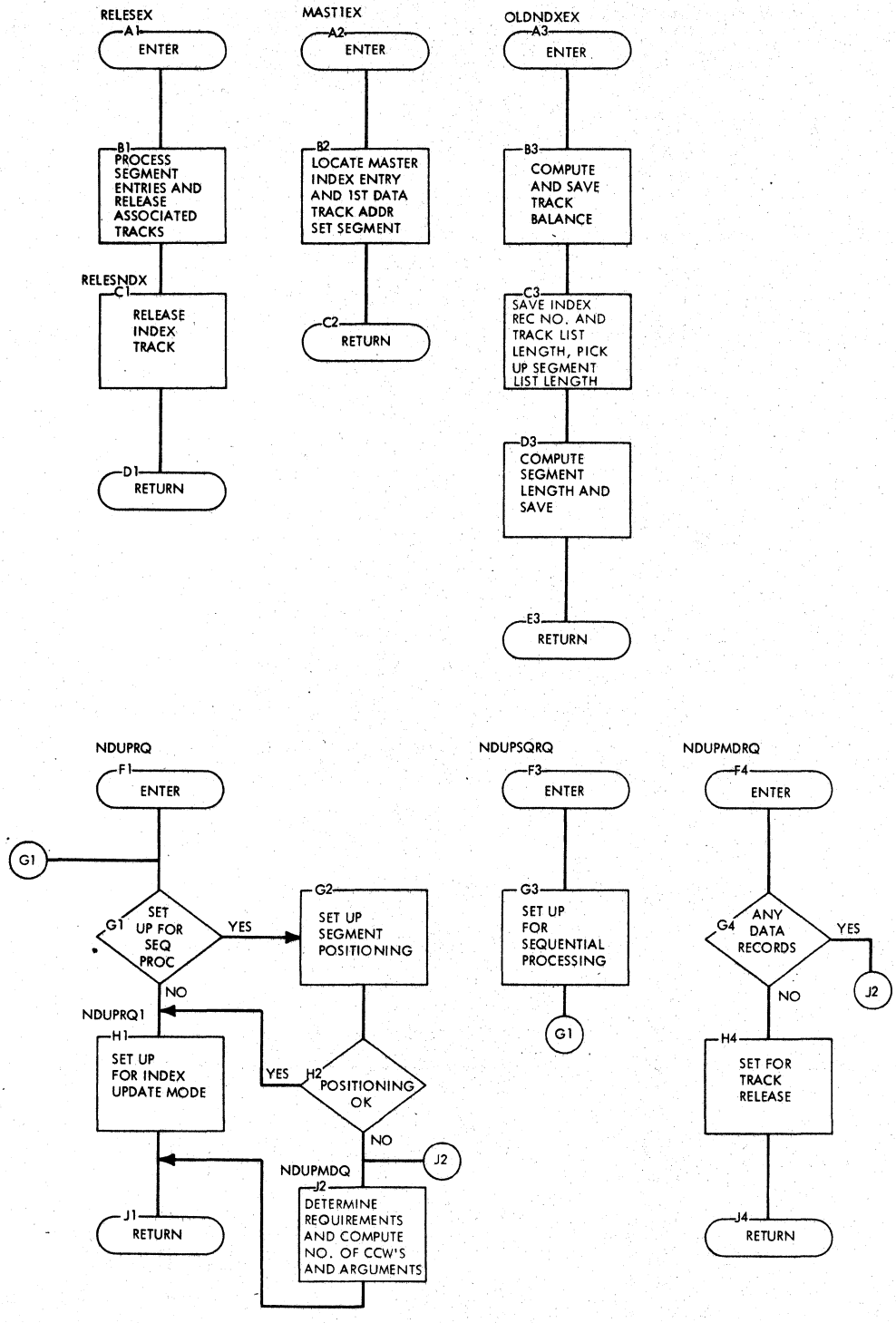


Chart TM. Active Area I/O Requester/Executor (IHKEXC)

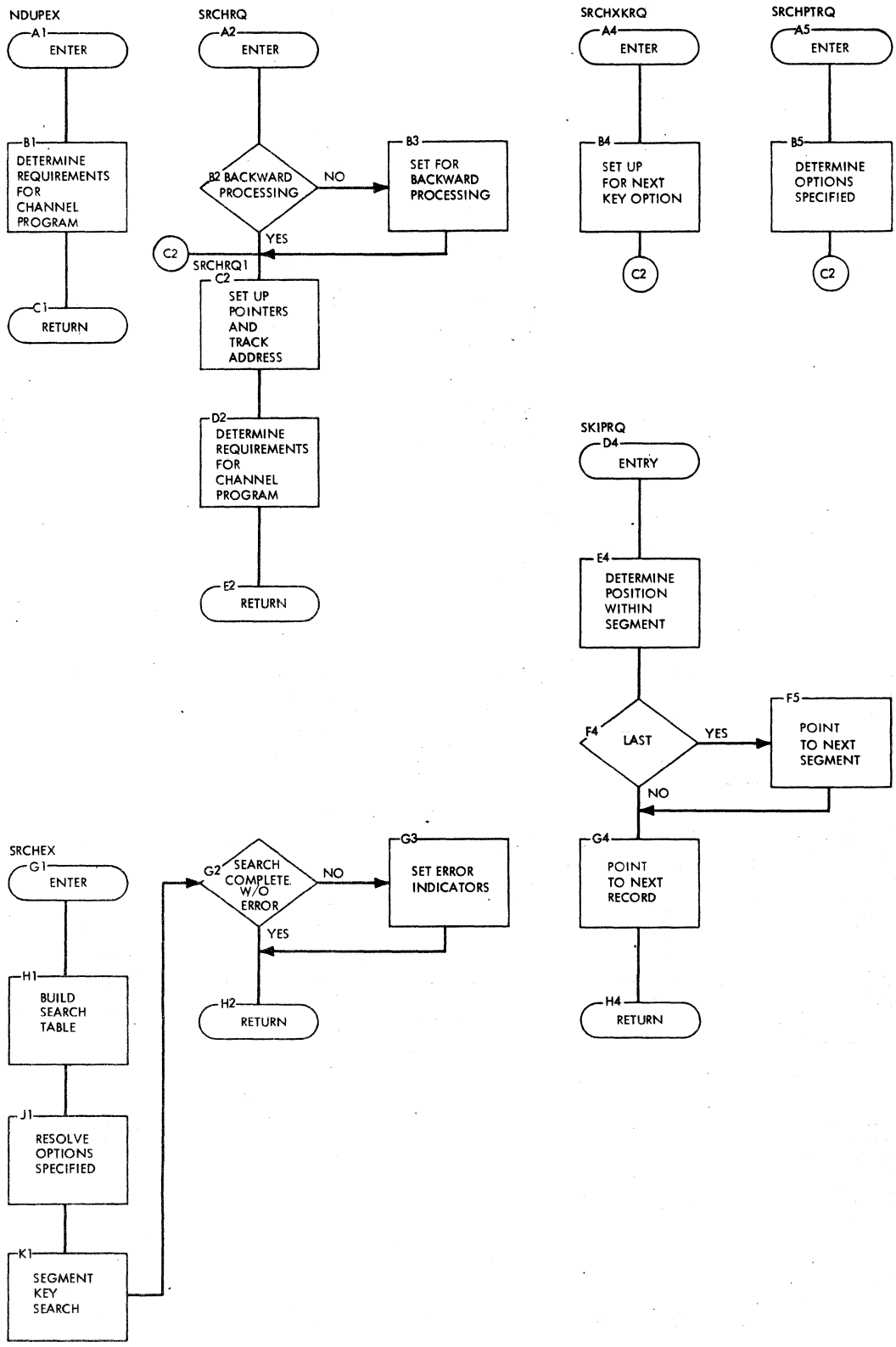


Chart TN. Active Area I/O Requester/Executor (IHKEXC)

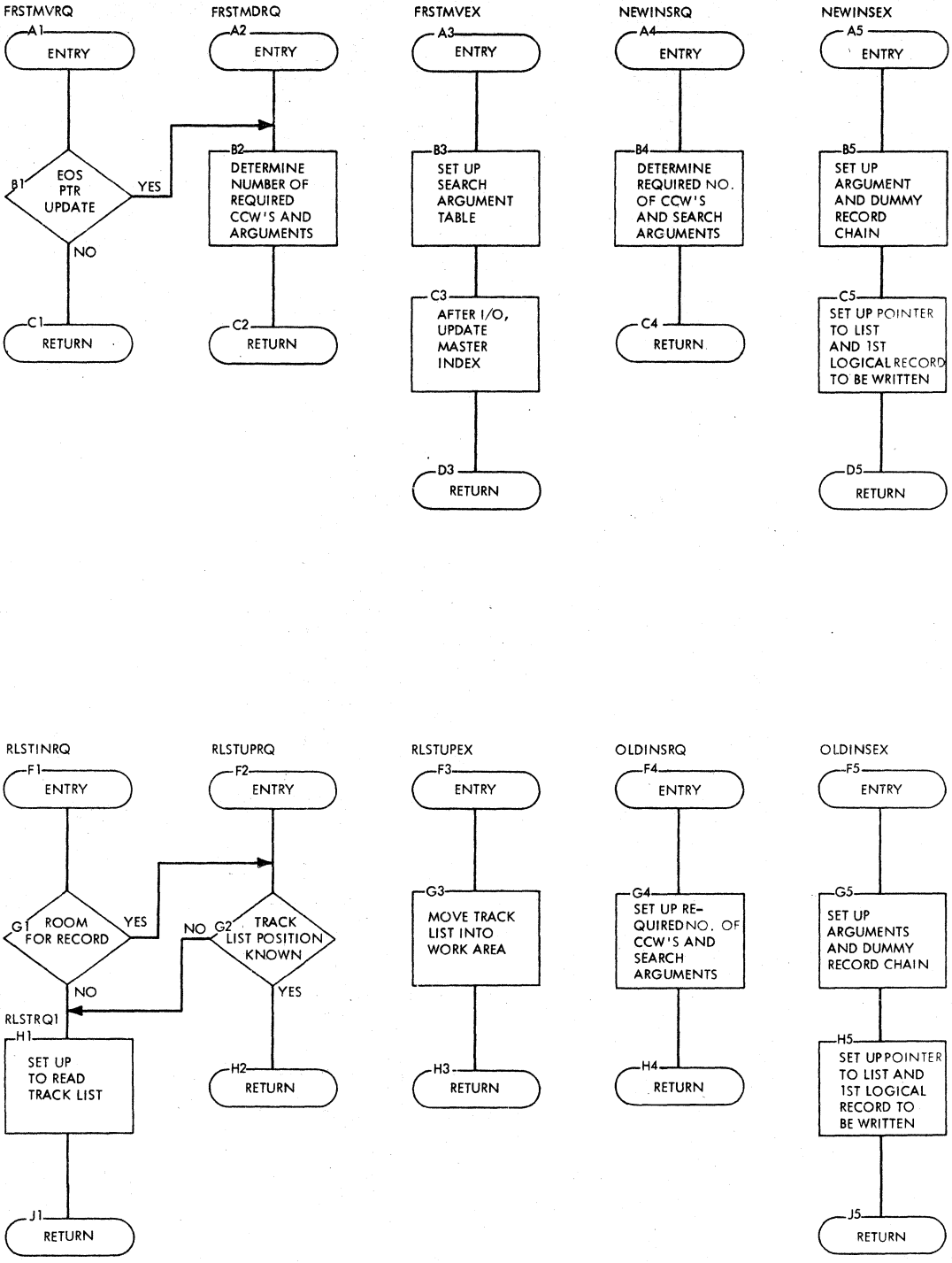


Chart TO. Active Area I/O Requester/Executor (IHKEXC)

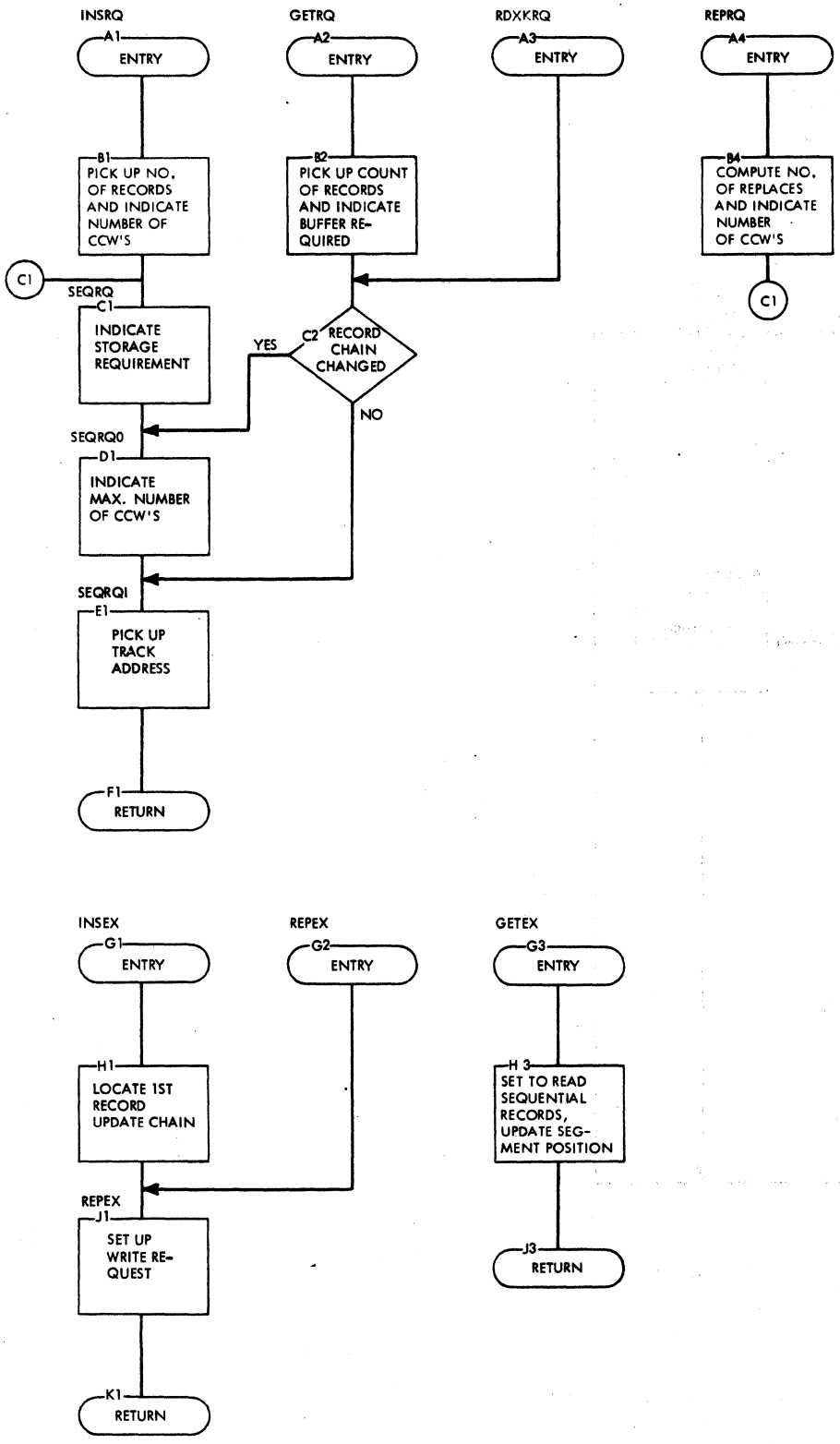


Chart UA. Librarian Queue Module (IHKRNQ)

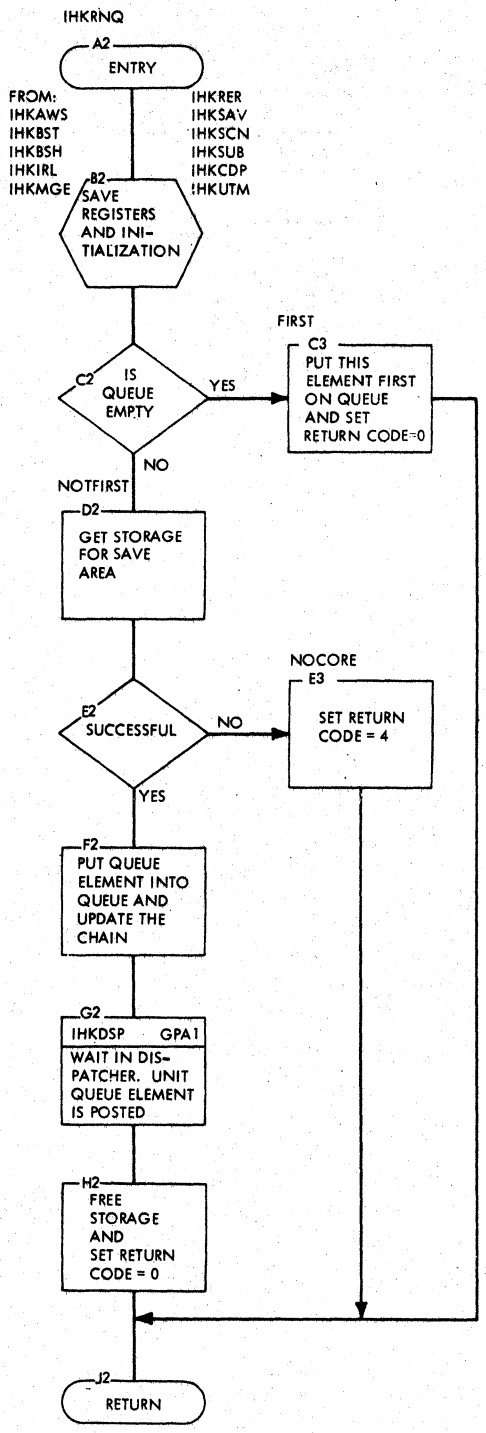


Chart UE. Library I/O Module (IHKBPM)

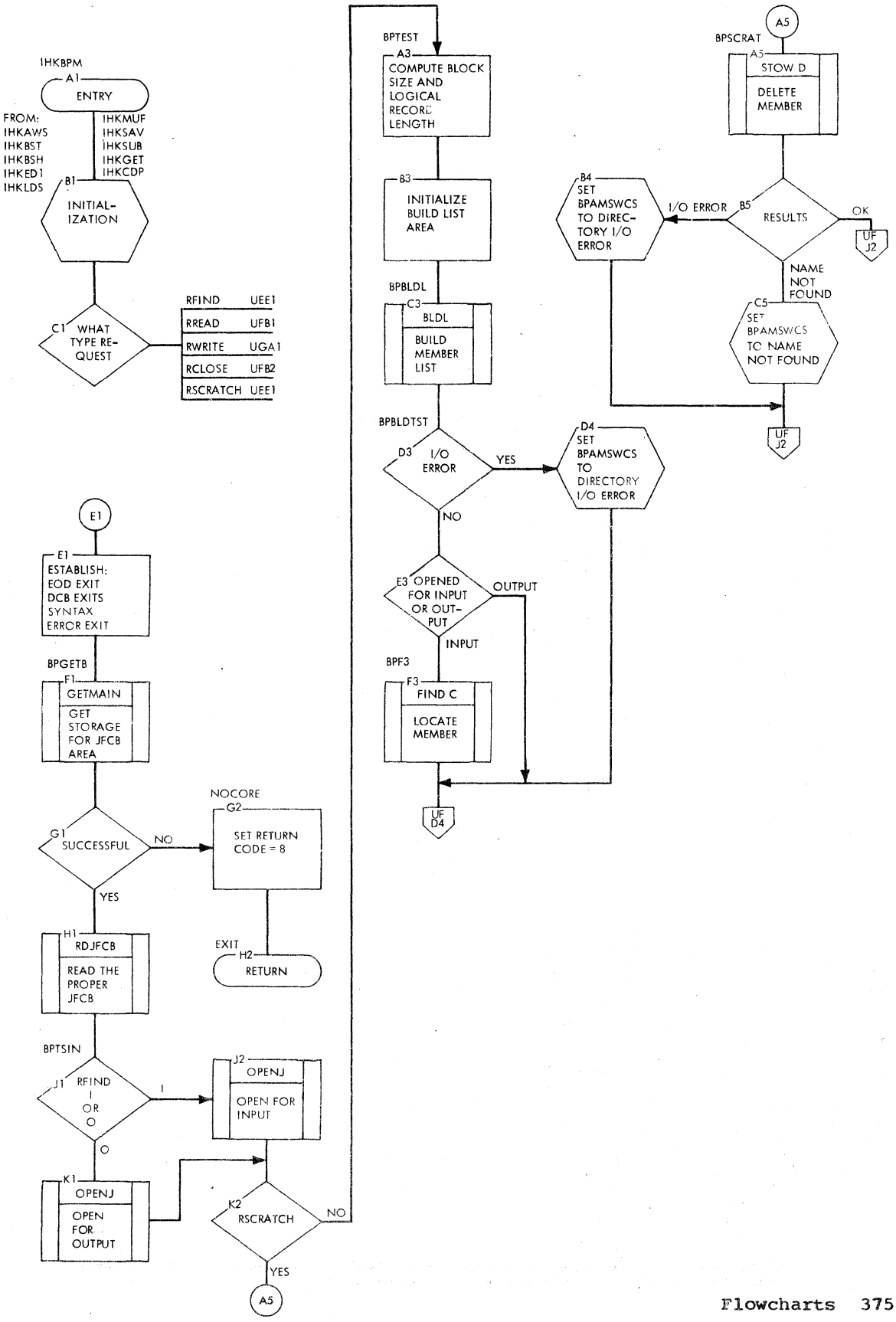
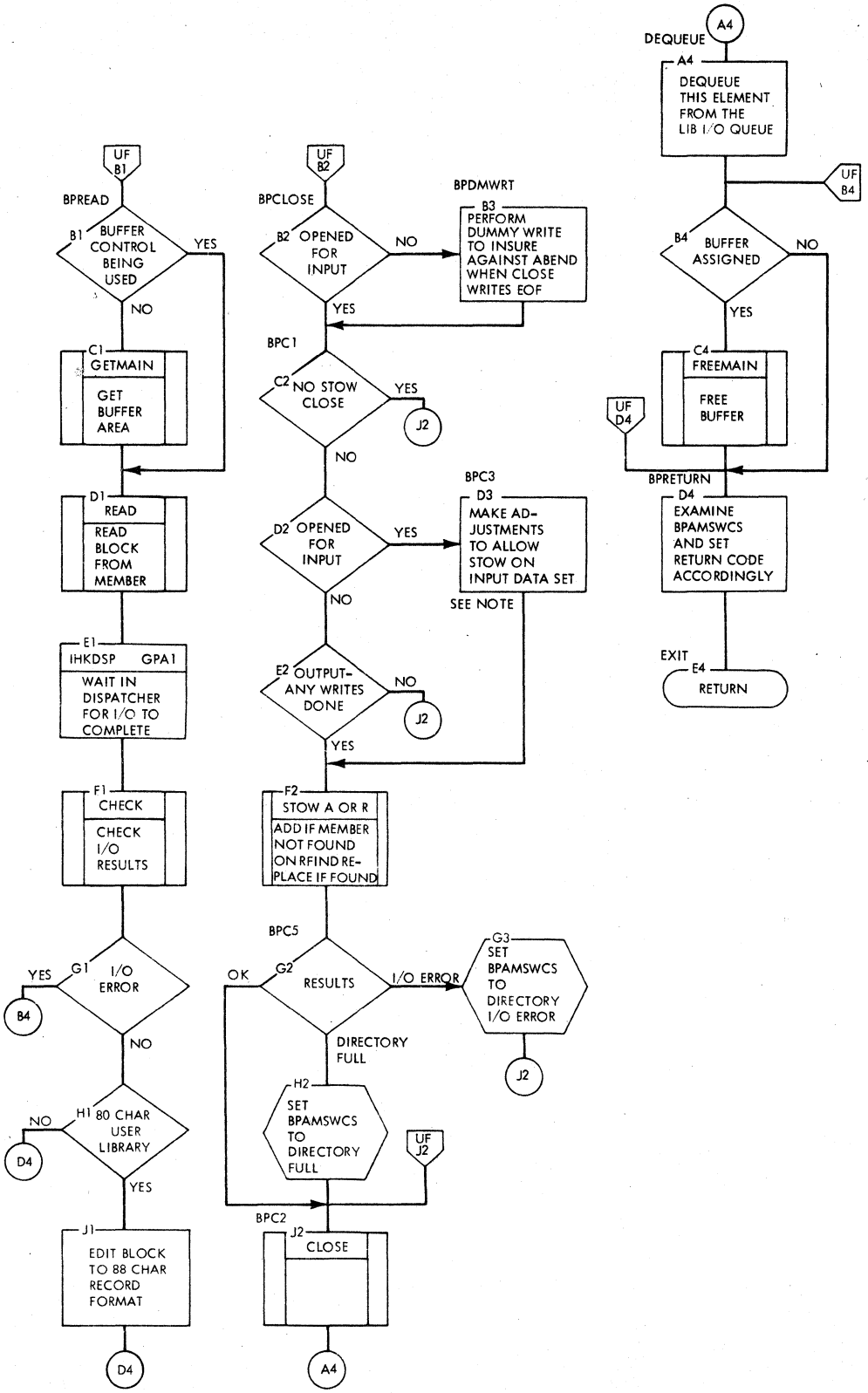
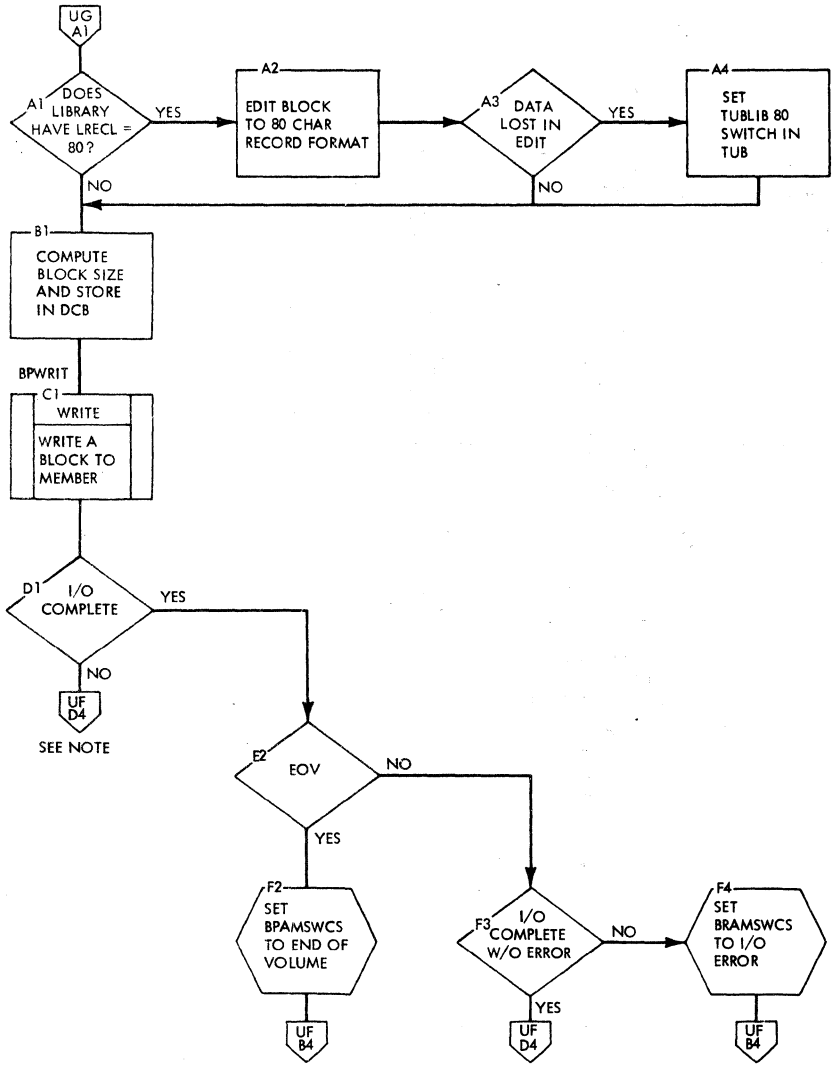


Chart UF. Library I/O Module (IHKBPM)



NOTE - Specifications require that directory user information be updated for every library member access. In order to do a STOW on a data set opened for INPUT, several fields and switches must be manipulated. Refer to listing.

• Chart UG. Library I/O Module (IHKBPM)



Note: Checking of BPAMSWCS at BPRETURN is of no significance after a WRITE because user must call IHKWTR to check results of WRITE.

Chart UP. Library I/O Wait Module (IHKWTR)

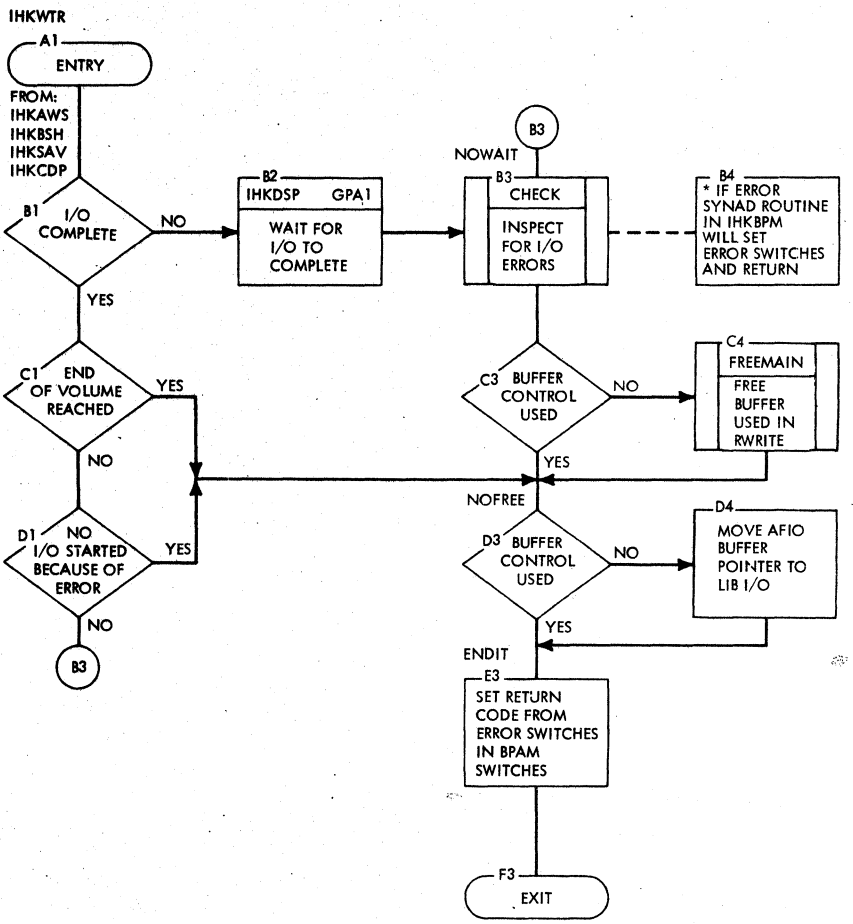


Chart UR. Library Condense Module (IHKCDP)

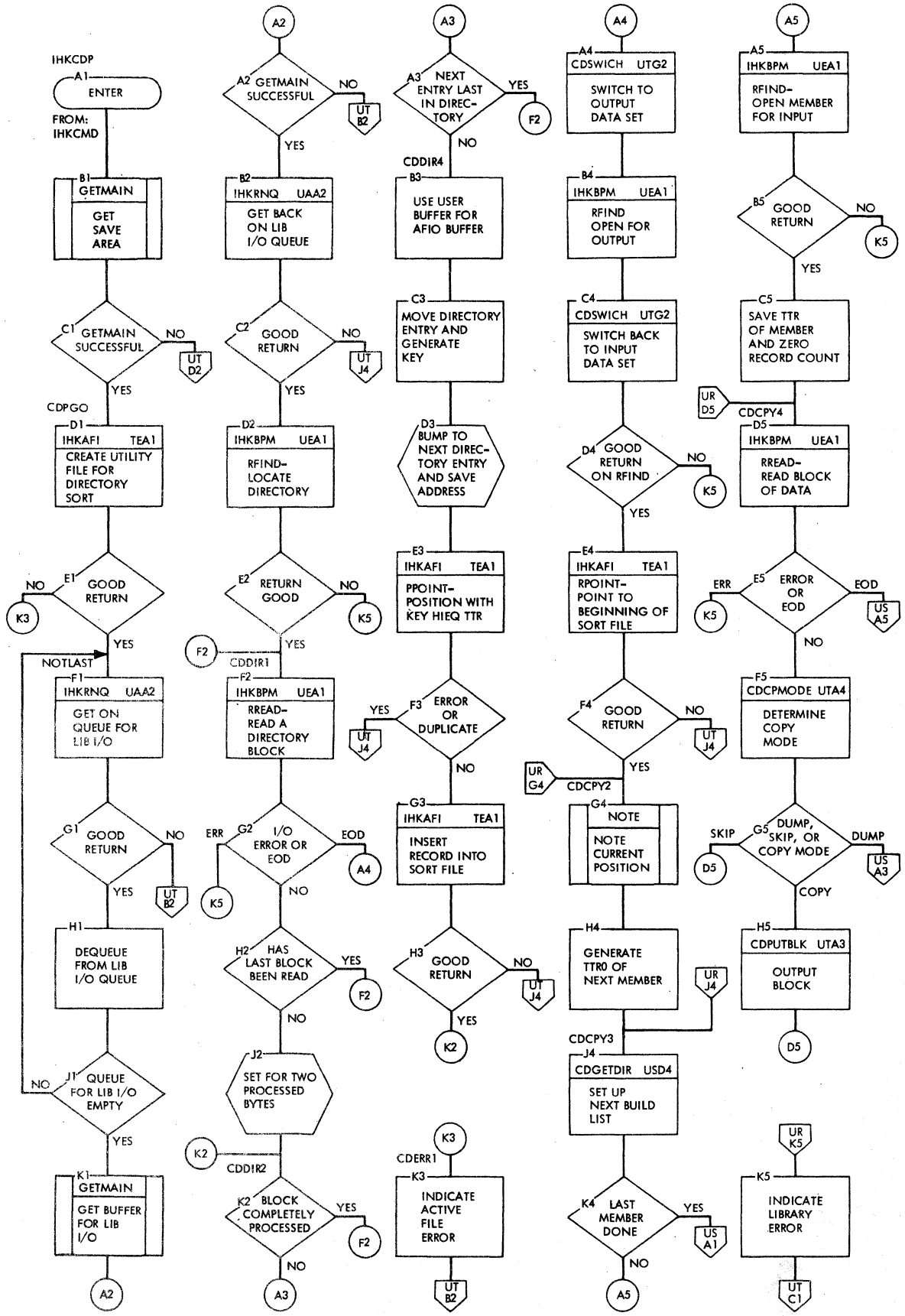


Chart US. Library Condense Module (IHKCDP)

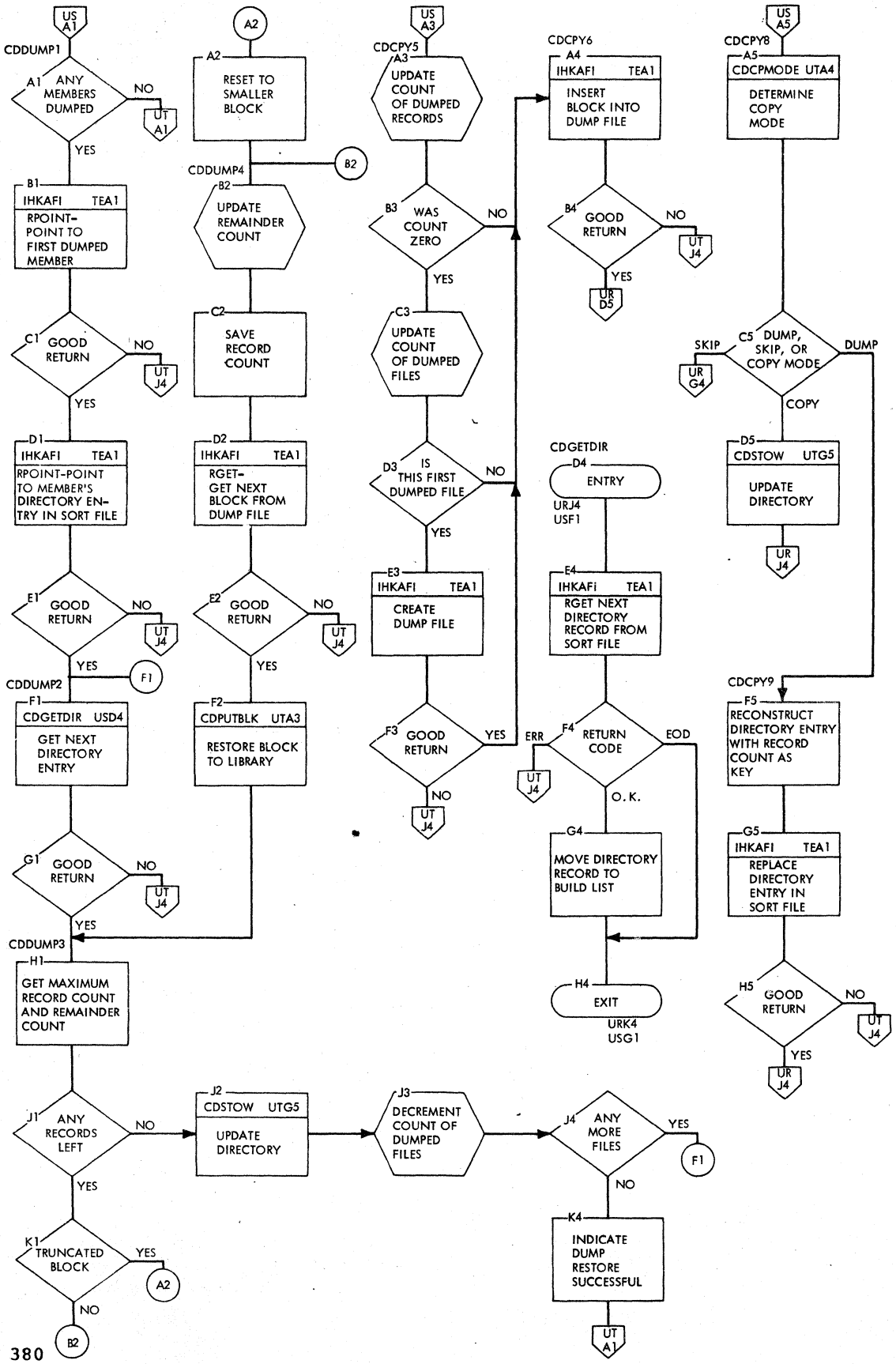


Chart UT. Library Condense Module (IHKCDP)

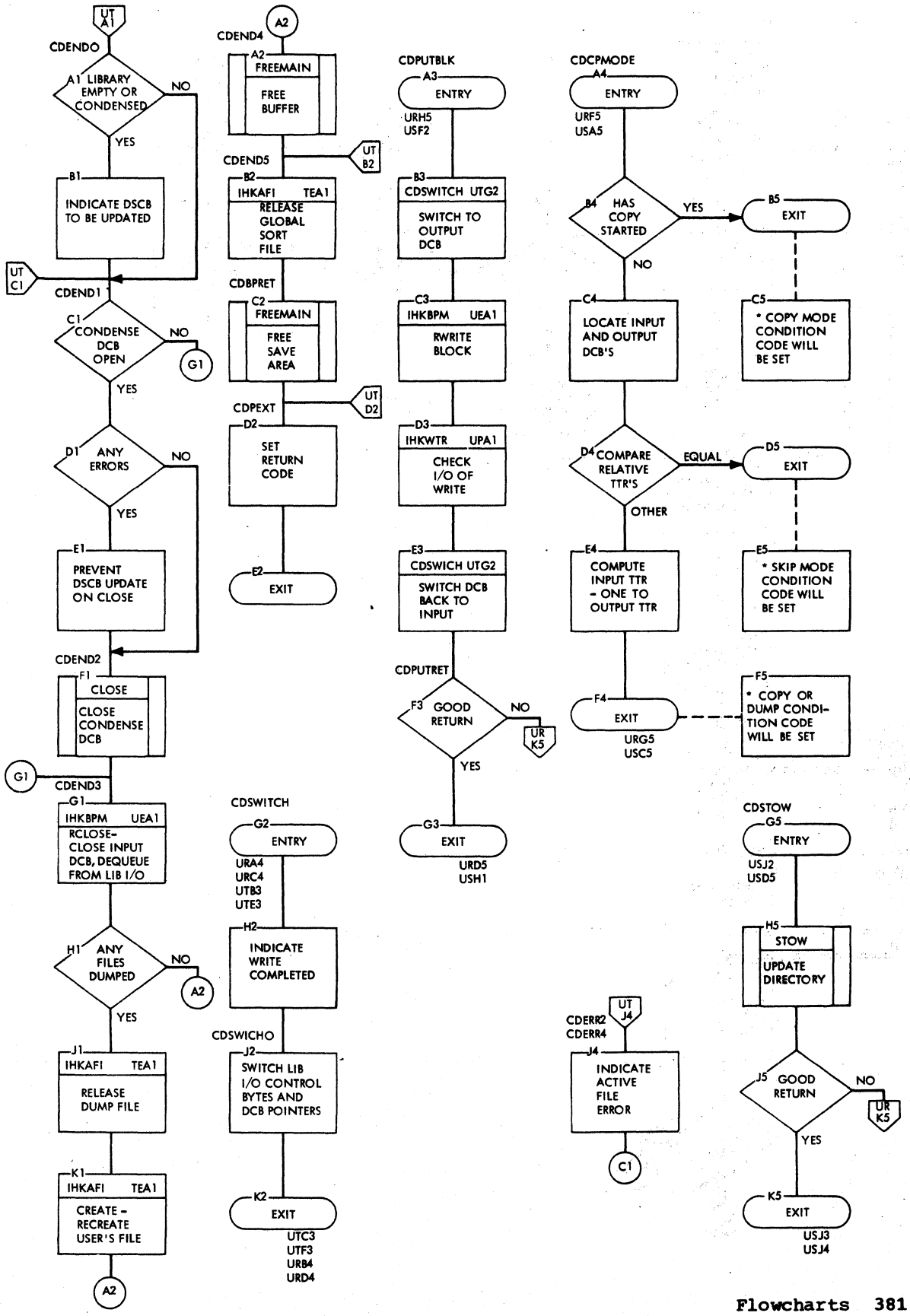


Chart WA. Scan Module (IHKCCS)

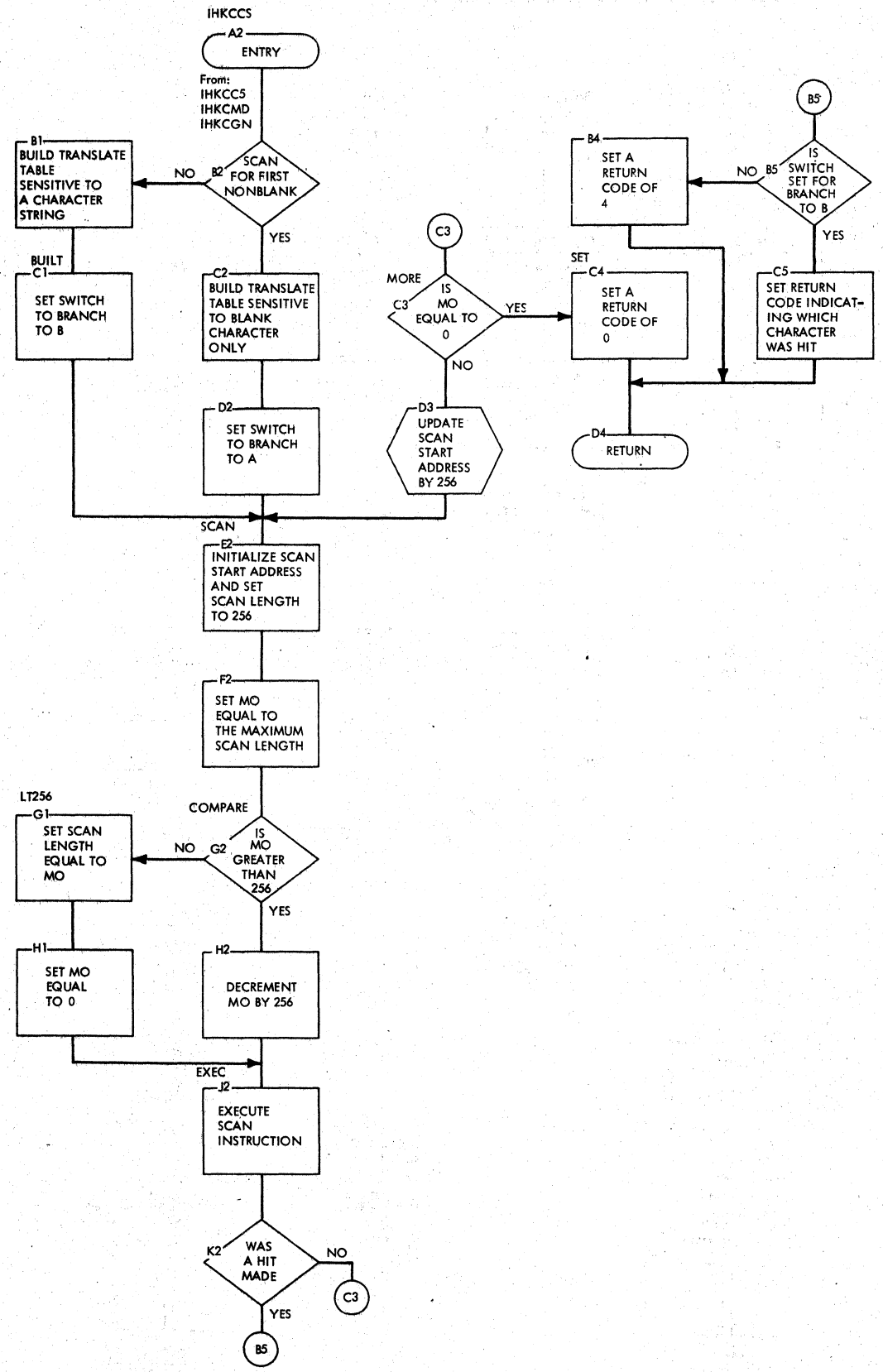


Chart WJ. Numeric Verification Module (IHKNUM)

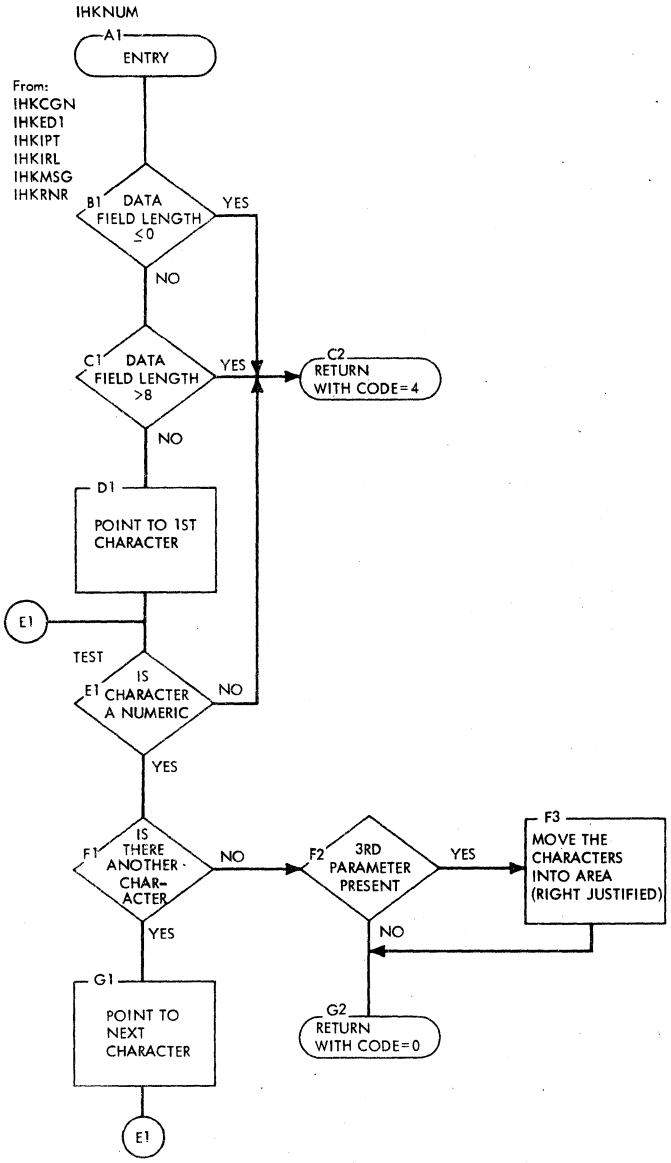


Chart WR. FORTRAN and PL/I Conversational Syntax Checker Interface (IHKSYN)

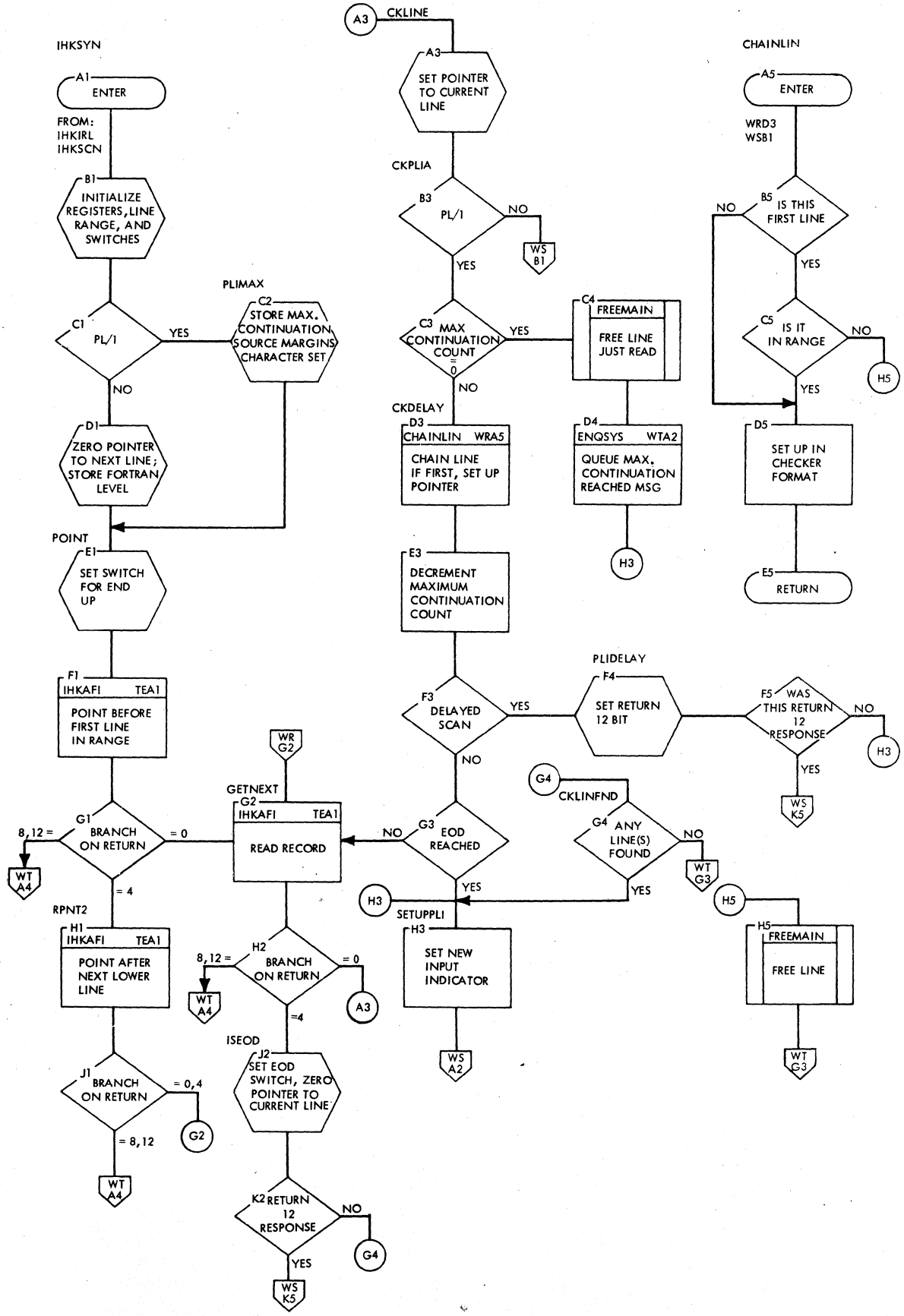


Chart WS. FORTRAN and PL/I Conversational Syntax Checker Interface (IHKSYN)

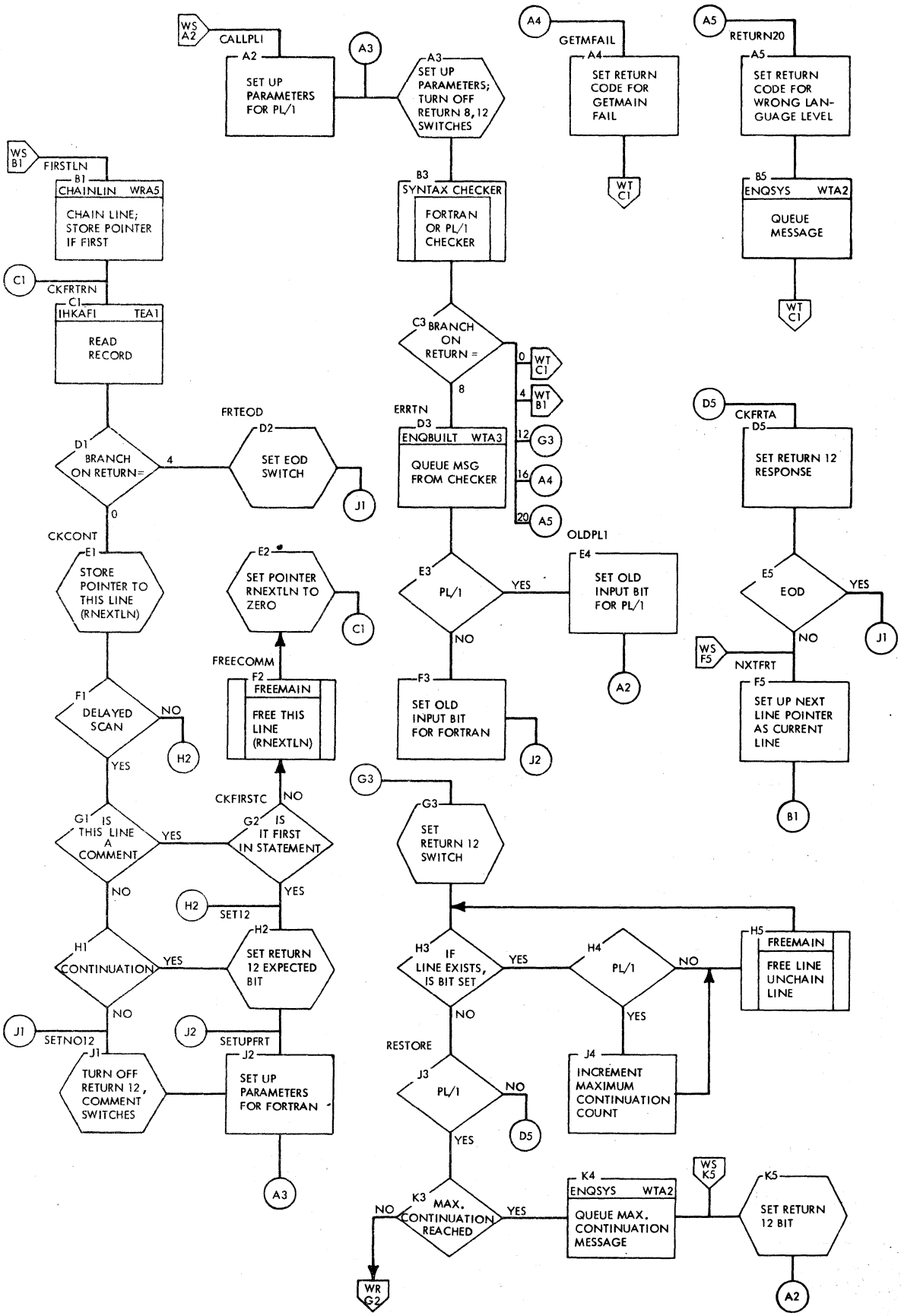


Chart WT. FORTRAN and PL/I Conversational Syntax Checker Interface (IHKSYN)

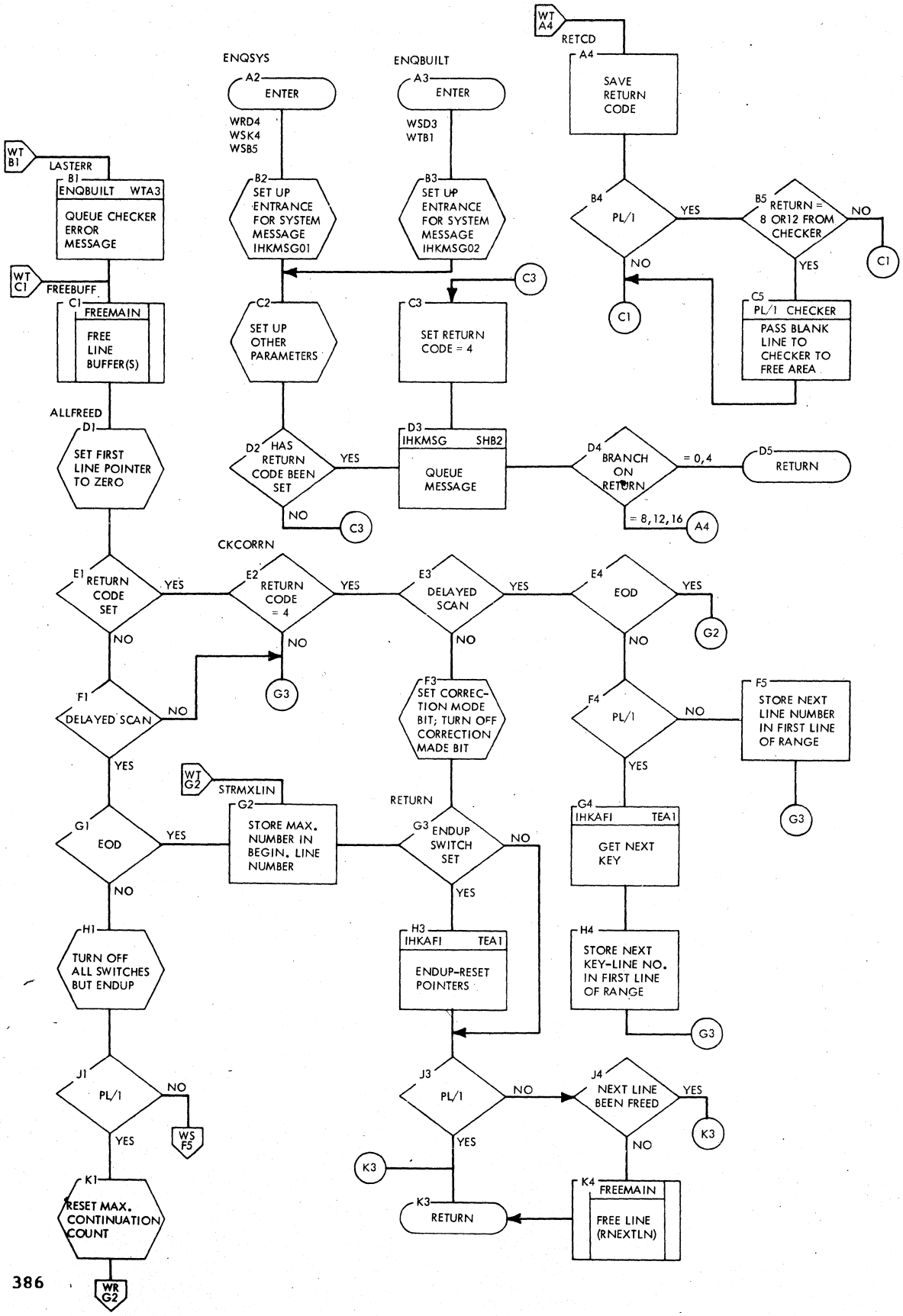


Chart WX. User File Manager (IHKUTM)

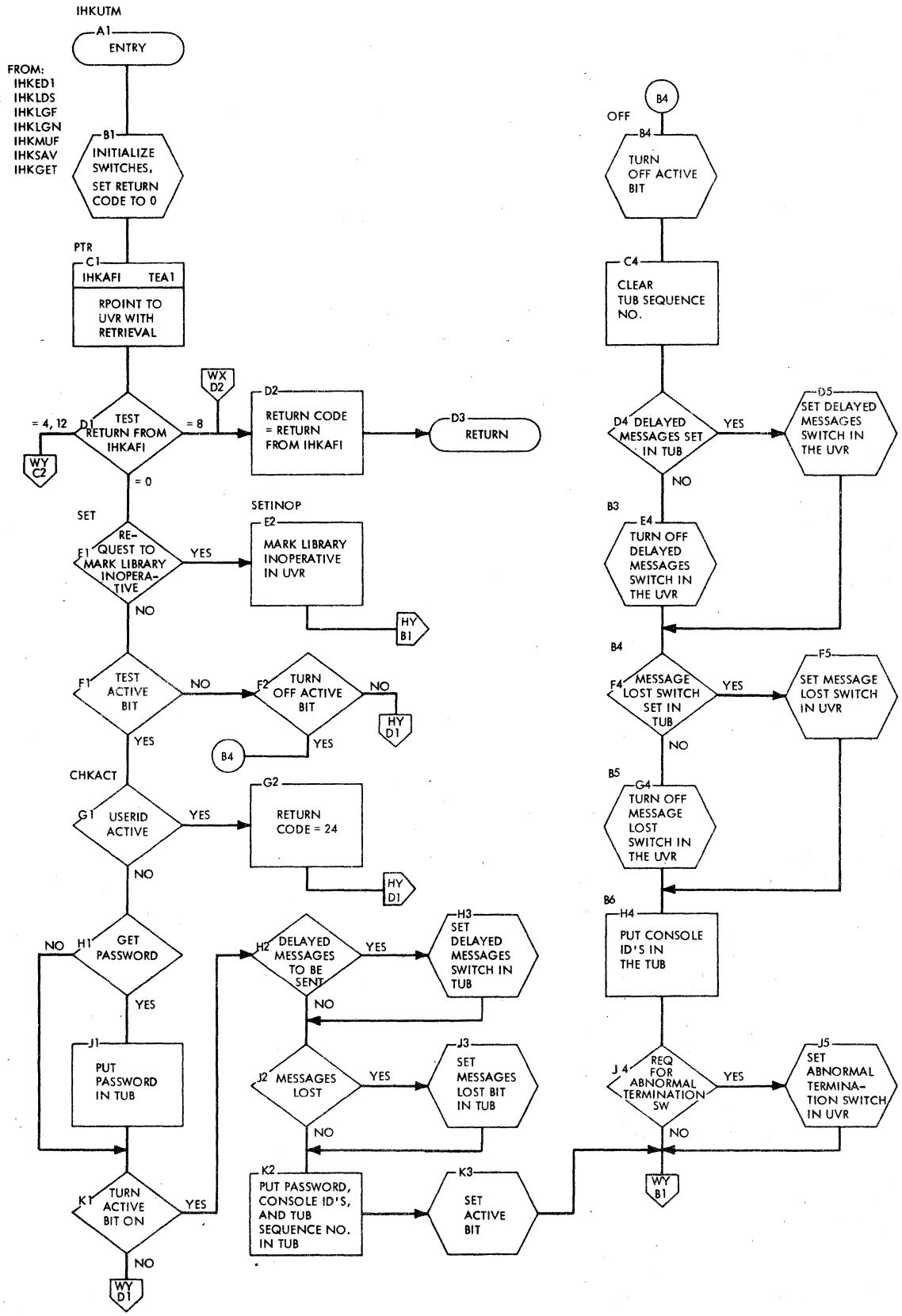
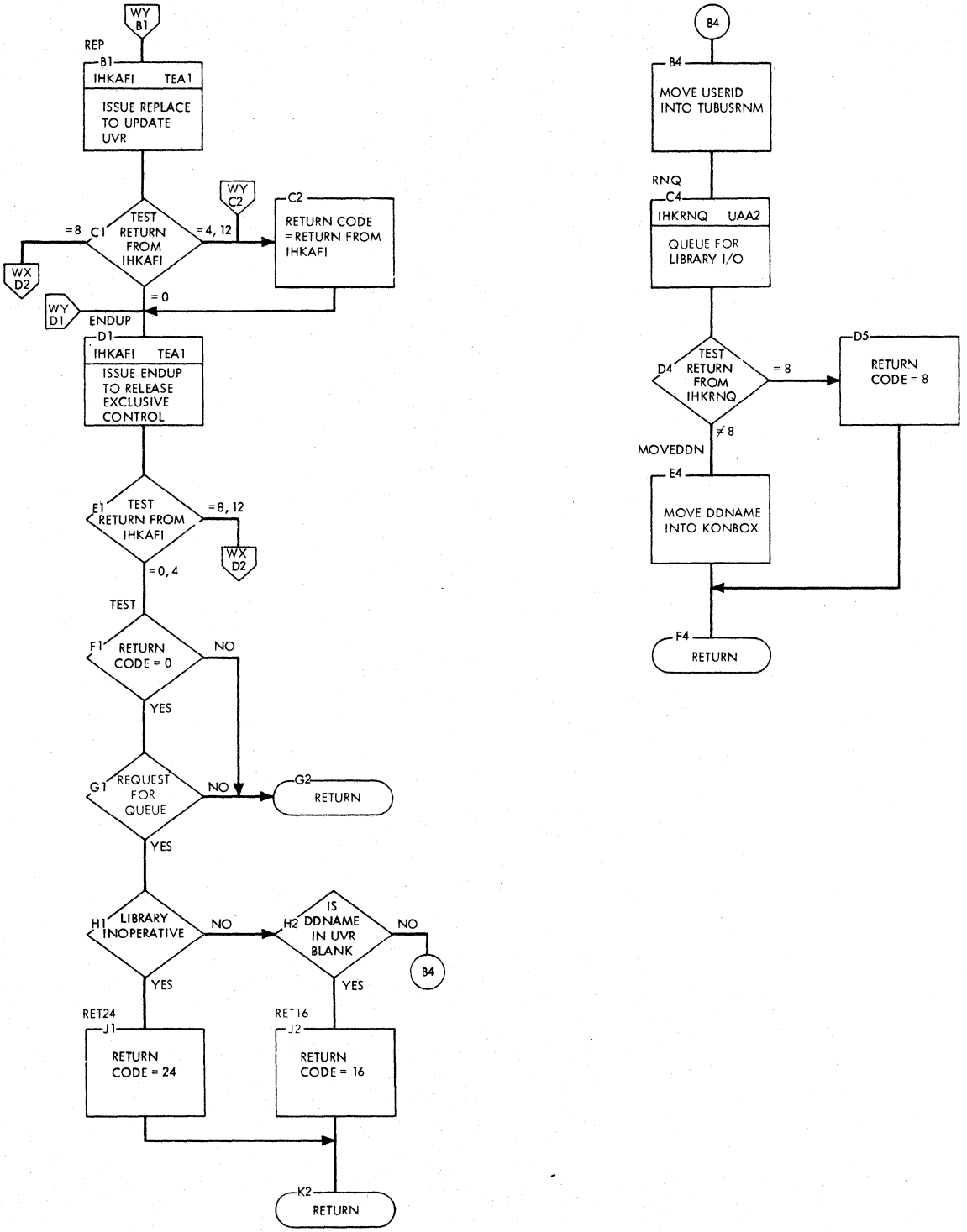


Chart WY. User File Manager (IHKUTM)



All modules, except one, in the CRJE system have object module names with the first ree letters the same as in the RJE system. Object module names in both systems begin th the letters IHK. Object module names in RJE are eight letters and object module mes in CRJE are only six letters, (aliases are eight letters). The one exception is e module that RJE and CRJE share, which is the RJE/CRJE central command

ECT ME	OBJECT MODULE NAME	GENERIC NAME	LOAD MODULE	ATTRIBUTES OF MODULE*	CHART IDs	ENTRY POINTS
C1503D	IGC1503D	RJE/CRJE Central Command Scheduling Routine	IGC1503D	R, N	EA	IGC1503D
KAFI	IHKAFI	Active Area I/O Control/COMMAND Interpreter	IHKBGN	SR, S	TE-TJ	IHKAFI
KALC	IHKALC	Allocate Routine	IHKALC	R, N	RZ	IHKALC
KAST	IHKAST	Active Area Start-up Initialization	IHKAST	SR, N	AP	IHKAST
KAWS	IHKAWS	Active Area Recovery Module	IHKAWS	SR, N	AR-AT	IHKAWS
KBGN	IHKBGN	START Command Processor	IHKBGN	SR, S	AF	IHKBGN
KBLKS	IHKMAC	CLBs, STCBs, DECBs, DCBs	IHKBGN			IHKGML
KBPM	IHKBPM	Library I/O Module	IHKBGN	SR, S	UE-UG	IHKBPM
KBSH	IHKBSH	Library I/O Shutdown Module	IHKBSH	SR, N	B1, BJ	IHKBSH
KBST	IHKBST	Library I/O Start-up Module	IHKBST	SR, N	AV, AW	IHKBST
KCC1	IHKCC1	SHOW USERS and SHOW JOBS Central Command Processor	IHKCC1	R, N	EG-EH	IHKCC1
KCC2	IHKCC2	SHOW LERB, SHOW BRDCST, AND MODIFY Central Command Processor	IHKCC2	R, N	EJ-EK	IHKCC2
KCC3	IHKCC3	BRDCST Central Command Processor	IHKCC3	R, N	EN-EO	IHKCC3
KCC4	IHKCC4	SHOW MSGS and MSG D= Central Command Processor	IHKCC4	R, N	EP, EQ	IHKCC4
KCC5	IHKCC5	CENOUT Central Command Processor	IHKCC5	R, N	ER-ET	IHKCC5

---

*SR=serially reusable;R=reentrant;F=refreshable;N=nonresident;S=resident

IHKCC6	IHKCC6	SHOW SESS and SHOW SESSREL Central Command Processor	IHKCC6	R, N	EUSZEV	IHKCC6
IHKCC7	IHKCC7	USERID Central Command Processor	IHKCC7	R, N	EW	IHKCC7
IHKCC8	IHKCC8	MSG and SHOW ACTIVE Central Command Processor	IHKCC8	R, N	EX-EY	IHKCC8
IHKCC1	IHKCC1	Central Command Interface	IHKBGN	SR,S	EE	IHKCC1
IHKCCS	IHKCCS	Scan Routine	IHKBGN	SR,S	WA	IHKCCS
IHKCDP	IHKCDP	Library Condense Module	IHKCDP	R,N	UR-UT	IHKCDP
IHKCGN	IHKCGN	CHANGE Subcommand Processor	IHKCGN	R,N	MA-MC	IHKCGN
IHKCIP	IHKCIP	CRJE Initialization Module	IHKCIP	SR,N	AK-AM	IHKCIP
IHKCLN	IHKCLN	CRJE Closedown Module	IHKCLN	SR,N	BF-BG	IHKCLN
IHKCMD	IHKCMD	Command Analyzer	IHKBGN	R,S	GA-GJ	IHKCMD
IHKDEQ	IHKDEQ	Dequeue/Job End Processor	IHKDEQ	R,N	FJ,FK	IHKDEQ
IHKDSP	IHKDSP	CRJE Dispatcher	IHKBGN	R,S	GP	IHKDSP
IHKED1	IHKED1	EDIT, DELETE, and EXEC Command Processor	IHKED1	R,N	MJ,MK	IHKED1
IHKECD	IHKMAC	ECB list for Dispatcher	IHKBGN			
IHKECL	IHKMAC	ECB list for IHKLDC	IHKBGN			
IHKECS	IHKMAC	ECB list for IHKSRV	IHKBGN			
IHKEDT	IHKEDT	EDIT, DELETE, and EXEC Command Processor	IHKEDT	R,N	MG-MI	IHKEDT
IHKEND	IHKEND	END Subcommand Processor	IHKLGF	R,N	MW	IHKEND
IHKEOS	IHKEOS	EDIT Command Processor	IHKEOS	R,N	MM-MN	IHKEOS
IHKERR	IHKERR	Line Error and Active Area I/O Error Recorvery Routine	IHKALC	R,F,N	GS,GT	IHKERR
IHKEXC	IHKEXC	Channel Program Initializer/Requester	IHKBGN	SR,S	TK-TO	IHKEXC
IHKGCW	IHKGCW	Channel Command Word List Generator	IHKEGN	SR,S	No	IHKGCW
IHKGET	IHKGFT	SUBMIT Input Record Processor	IHKGET	R,N	RS-RU	IHKGET
IHKIN1	IHKIN1	CRJE System Library Initialization Unity	IHKINT	SR	AA	IHKIN1

---

*SR=serially reusable;R=reentrant;F=refreshable;N=nonresident;S=resident

IHKIPT	IHKIPT	INPUT Subcommand Processor	IHKIPT	R,N	NA-NC	IHKIPT
IHKIRL	IHKIRL	Insert/Replace/Delete Processor	IHKBGN	R,S	NJ-NM	IHKIRL IHKIRL01 IHKIRL02
IHKLAB	IHKLAB	Output Text Formatting Module	IHKBGN	R,S	HP-HQ	IHKLAB
IHKLAD	IHKLAD	Communication Line Administrator Module	IHKBGN	R,S	HA-HE	IHKLAD
IHKLAP	IHKLAP	Input/Output Operation Initiation Module	IHKBGN	SR,S	HK-HM	IHKLAP
IHKLAT	IHKLAT	TABSET Edit Module	IHKBGN	SR,S	HU	IHKLAT
IHKLAY	IHKLAY	1050X Programmed Time-out Module	IHKLAY	R,N	HX-HZ	IHKLAY
IHKLDC	IHKLDC	Loader/Controller	IHKLDC	SR,S	DA-DD	IHKLDC
IHKLDS	IHKLDS	LISTDS and LISTLIB Command Processor	IHKLDS	R,N	NV-NY	IHKLDS
IHKLERB	IHKMAC	Line Error Control Blocks	IHKBGN			
IHKLEW	IHKLEW	Line Edit Write Routine	IHKBGN	R,S	HW	IHKLEW
IHKLGF	IHKLGF	LOGOFF Command Processor	IHKLGF	R,N	PA	IHKLGF
IHKLGN	IHKLGN	LOGON Command Processor	IHKLGN	R,N	PE-PG	IHKLGN
IHKLOC		Alias for IEFLOCDQ				
IHKLST	IHKLST	LIST Subcommand Processor	IHKBGN	R,S	NR-NT	IHKLST
IHKMAA	IHKMAA	MERGE Subcommand Processor	IHKMAA	R,N	PN-PO	IHKMAA
IHKMGE	IHKMGE	MERGE Subcommand Processor	IHKMGE	R,N	PJ,PK	IHKMGE
IHKMSG	IHKMSG	Message Writer	IHKBGN	R,S	SH-SK	IHKMSG, IHKMSG01, IHKMSG02, IHKMSG03
IHKMUF	IHKMUF	MERGE Subcommand Processor	IHKMUF	R,N	PS	IHKMUF
IHKNUM	IHKNUM	Numeric Verification Routine	IHKBGN	R,S	WJ	IHKNUM
IHKOPN	IHKOPN	OS Data Set Open Routine	IHKOPN	R,N	DH	IHKOPN
IHKOUT	IHKOUT	OUTPUT and CONTINUE Command Processor	IHKIPT	R,N	QA	IHKOUT
IHKPUT	IHKPUT	Transmit Output Module	IHKPUT	R,N	QB-QF	IHKPUT
IHKRER	IHKRER	SYSOUT Open, Job Delete, Data Set Scratch, and	IHKRER	R,N	QJ,QK	IHKRER, IHKRER01,

---

*SR=serially reusable;R=reentrant;F=refreshable;N=nonresident;S=resident

		CANCEL Module				IHKRER02, IHKRER03
IHKRNQ	IHKRNQ	Librarian Queue Module	IHKBGN	R,S	UA	IHKRNQ
IHKRNR	IHKRNR	RENUMBER Subcommand Processor	IHKRNR	R,N	QQ+QT	IHKRNR
IHKSAV	IHKSAV	SAVE Subcommand Processor	IHKSAV	R,N	QW-QZ	IHKSAV
IHKSCN	IHKSCN	SCAN Subcommand Processor	IHKBGN	R,S	RA,RB	IHKSCN
IHKSDQ	IHKSDQ	Job Termination Handling Routine	IHKBGN	SR,S	FA	IHKSDQ
IHKSMG	IHKSMG	System Messages	IHKINT			
IHKSND	IHKSND	SEND Command Processor	IHKSND	R,N	RE-RG	IHKSND
IHKSRV	IHKSRV	START RDRCRJE, Allocate, and Q Manager Service Routine	IHKSRV	SR,S	CA,CB	IHKSRV
IHKSTP	IHKSTP	CRJE Stop Module	IHKTAB	R,N	[illegible]	IHKSTP
IHKSTS	IHKSTS	STATUS Command Processor	IHKSTS	R,F,N	RJ,RK	IHKSTS
IHKSUB	IHKSUB	SUBMIT Command Processor	IHKSUB	R,N	RO-RR	IHKSUB
IHKSYN	IHKSYN	FORTRAN and PL/I Conversational Syntax Checker Interface	IHKBGN	SR,S	WR-WT	IHKSYN
IHKTAB	IHKTAB	TABSET Subcommand Processor	IHKTAB	R,N	SE-SG	IHKTAB
IHKUSR	IHKUSR	USERIDs/Passwords	IHKINT	S		
IHKUTM	IHKUTM	User File Manager	IHKBGN	R,S	WX,WY	IHKUTM
IHKWTR	IHKWTR	Library I/O Wait Module	IHKBGN	R,S	UP	IHKWTR

---

*SR=serially reusable;R=reentrant;F=refreshable;N=nonresident;S=resident

ABLES

AME	CSECT NAME	GENERIC NAME	MODULE CONTAINING TABLE	LOCATION	SYNOPSIS
VT	IHKAVT	Address Vector Table	IHKAVT	resident	Contains entry points of resident modules.
CT	IHKCCT	CRJE Control Table	IHKMAC	resident	Contains information pertaining to entire CRJE system.
LB		Conversational Line Block	IHKMAC	resident	Contains information about one line; a CLB exists for each line.
EF	IHKDEF	Command Default Table	IHKDEF	resident	Contains default options of operands on commands and subcommands.
HKNBX	IHKNBX	KONBOX	IHKNBX	resident	Contains information for AFIO and Library I/O.
AP		Block Table	Allocated at start-up	resident	Contains entry for each 2K block of main storage in the transient area; used by IHKLDC.
CL	IHKMCL	Major Command List	IHKMAC	resident	Contains all major commands and associates a code with each one.
OD	IHKMOD	Module Table	IHKMOD	resident	Contains an entry with a loading code for each nonresident module; used by IHKLDC.
PT		Parameter Position Table		Allocated dynamically as needed	Contains command and operands; one or more PPTs associated with each TUB.
JCT		Remote Job Control Table		Resides in system library on disk; resides in global file on disk when system is active.	Contains information about jobs submitted to OS for processing through CRJE.
CL	IHKSCL	Subcommand Table	IHKMAC	resident	Contains all subcommands and associates a code with each one.
TCB		Subtask Control Block	IHKMAC	resident	Contains pointer to entry in ECB list in dispatcher; one exists for each CRJE subtask; used by dispatcher for giving control to CRJE subtasks.

TRT	IHKMOD	Translate Table	IHKMOD	resident	Contains command and subcommand codes and a corresponding code for the module to be loaded or deleted; used by IHKLOC.
TUB		Terminal User Block		Allocated dynamically by line administrator.	Contains information about one user and his session; a TUB exists for each active user.
UVR		User Verification Record		Resides in system library and global file, both on disk.	Contains userid and password of all potential users.

This section contains a description of the eight control blocks used in CRJE. Whenever possible a map of the control block is provided. Displacements (in bytes) are provided in the left-hand corner of the box representing the field.

Following the map of the control block is a brief description of the fields. If possible, the field is broken down into flags or bits. The modules that set, check, or turn off the bits are also indicated here.

The user's verification record, which is part of the system library and resides in a global file, and the three forms of user library directory entries are also described in this section.

ADDRESS VECTOR TABLE

			Control Blocks and Tables	
IHKYYCCT	DS	A	IHKCCT	CRJE control tables
IHKYYSCB	DS	A	IHKSTCB1	STCB pointer for start-up
IHKYYSC1	DS	A	IHKSTCB2	Current STCB in first circle
IHKYYSC2	DS	A	IHKSTCB2	Current STCB in second circle
IHKYYSC3	DS	A	IHKSTCB3	Current STCB in third circle
IHKYYSC4	DS	A	0	Reserved
IHKYYCLB	DS	A	IHKCLB	Conversational line block list
IHKYYTUB	DS	A	0	Address of first TUB in chain
	DS	A	0	Address of last TUB in chain
IHKYYMCL	DS	A	IHKMCL	Major command code list
IHKYYMOD	DS	A	IHKMOD	Nonresident module table
IHKYYMAP	DS	A	0	Current map of transient area
IHKYYTRT	DS	A	IHKTRT	Command code translate table
IHKYYDEF	DS	A	IHKDEF	Option default table
IHKYYCIB	DS	A	IHKCIB	Central command input buffer
IHKYYGML	DS	A	IHKGML	GETMAIN list for refreshability
IHKYYDQE	DS	A	IHKDQE	SYS dequeue ECB
IHKYYDQS	DS	A	IHKDQS	Stop ECB for SYS dequeue
IHKYYLCE	DS	A	IHKLCE	Loader/Controller return ECB
IHKYYLCT	DS	A	0	Loader/Controller TCB address
IHKYYSRE	DS	A	IHKSRE	SRV ECB to return to OS
IHKYYSRT	DS	A	0	SRV TCB address
IHKYYSPL	DS	A	0	START parameter list
IHKYYECL	DS	A	IHKECL	Loader/Controller ECB list
IHKYYECD	DS	A	IHKECD	Dispatcher ECB list
IHKYYECS	DS	A	IHKECS	Utility TASK (IHKSRV) ECB list
			Operating System Modules	
IHKYYQMS	DS	A	0	Queue manager (queue)
IHKYYQMD	DS	A	0	Queue manager (dequeue)
IHKYYQML	DS	A	0	Queue manager (delete)
			CRJE Line Administrator	
IHKYYLAB	DS	V	IHKLAB	Output preparation
IHKYYLAP	DS	V	IHKLAP	Read/Write
IHKYYLAD	DS	V	IHKLAD	Entry, analysis, and end
IHKYYLAT	DS	V	IHKLAT	Tab character editing
IHKYYLAM	DS	A	0	Reserved

Message Writer Entry Points

IHKYYMSG	DS	V	IHKMSG	Build, or build and send
IHKYMSG1	DS	V	IHKMSG01	Build and queue
IHKYMSG2	DS	V	IHKMSG02	Queue existing message
IHKYMSG3	DS	V	IHKMSG03	List messages at central

AFIO/BPAM Modules

IHKYYAFI	DS	V	IHKAFI	AFIO
IHKYYEXC	DS	V	IHKEXC	EXCCW
IHKYYEXF	DS	V	IHKEXF	EXCPFW
IHKYYGCW	DS	V	IHKGCW	GENCCW
IHKYYBPM	DS	V	IHKBPM	BPAM
IHKYYIRP	DS	V	IHKIRP	INTRPFW
IHKYYRNQ	DS	V	IHKRNQ	Queue service requests
IHKYYKBX	DS	V	IHKNBX	KONBOX
IHKYYWTR	DS	V	IHKWTR	

Service Routines

IHKYYSYN	DS	V	IHKSYN	Syntax Checker Interface
IHKYYUTM	DS	V	IHKUTM	UVR Manager
IHKYYCCS	DS	V	IHKCCS	Character scan
IHKYYRJC	DS	V	IHKRJC	RJCT Manager
IHKYYNUM	DS	V	IHKNUM	Line number verification
IHKYYBGN	DS	V	IHKBGN	CRJE Being module
IHKYYSDQ	DS	V	IHKSDQ	Job termination
IHKYYDSP	DS	V	IHKDSP	Dispatcher

The Address of the Syntax Checkers are Inserted at the Operator's Discretion at Start-up Time.

IHKYYPL1	DS	V	IHKPL1	PL/1 Syntax Checker
IHKYYFRT	DS	V	IHKFRT	FORTRAN Syntax Checker

The Names of These Optional Installation-Defined Exits are Supplied in the CRJETABL Macro.

IHKYYIAX	DS	A	0	LOGON exit (accounting)
IHKYYISX	DS	A	0	SUBMIT exit (job card)
IHKYYILX	DS	A	0	LOGOFF exit

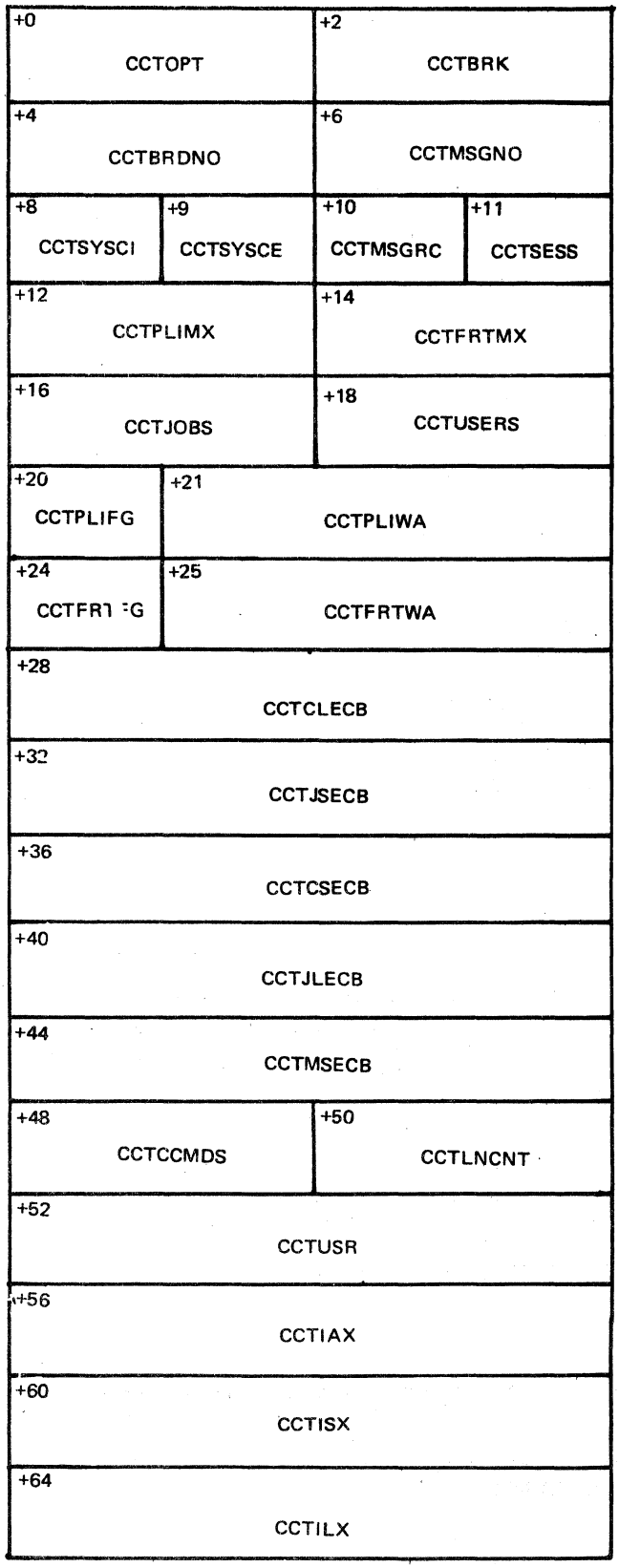
The Name of the Optional Installation-Defined User Command Processor is Supplied in the CRJETABL Macro.

IHKYYUSR	DS	A	0	

resident Command Processors

IHKYYCMD	DS	V	IHKLST	LIST
	DS	A	0	Unused
	DS	V	IHKSCN	SCAN
IHKYYIRL	DS	V	IHKIRL	Insert/Replace
	DS	V	IHKIRL01	Implicit subcommand
	DS	V	IHKIRL02	Correction mode
	DS	V	IHKIRL	DELETE subcommand
IHKYDSP1	DS	V	IHKDSP01	Current priority in dispatcher
IHKYYLPT	DS	V	IECTLERP	BTAM LERB print routine
IHKYYLEW	DS	A	0	Tab output formatting
IHKYYLAY	DS	A	0	1050X programmed time-out

CRJE CONTROL TABLE (CCT)



FIELD DESCRIPTIONS

CCTOPT - 2 bytes

Bits reflect either a parameter option or a condition of status within the system. If on, the condition exists; if off, the specified condition does not exist.

Bit Definitions

Bit 0 - Not used

Bit 1 - PL/1 syntax checker available
CCTPL1 Set by IHKCIP
Checked by IHKSCN and IHKIRL

Bit 2 - Not used

Bit 3 - Not used

Bit 4 - CRJE closedown in progress
CCTCLS Set by IHKSTP
Checked by IHKERR

Bit 5 - Suppress mode in effect
CCTSUP Set by IHKCC7
CCTSUPF Checked by IHKLGF and IHKCMD
Turned off by IHKCC7

Bit 6 - Not used

Bit 7 - Abnormal termination of the central system
CCTATERM Set by IHKDEQ
Checked by IHKCLN

Bit 8 - CRJE start-up in progress
CCTSRTUP Set by CRJETABL macro
Checked by IHKCCI
Turned off by IHKDEQ

Bit 9 - Job terminator busy
CCTJTBSY Set by IHKDEQ
Checked by IHKSTP
Turned off by IHKDEQ

Bit 10-13 - Not used

Bits 14-15 - These two flags indicate whether or not the
CCTFORT FORTRAN syntax checker is present in the
CCTFRTGH system and, if so, what level it is.
CCTFORTE Set by IHKCIP
Checked by IHKIRL and IHKSCN

CCTBRK - 2 bytes

Number of lines of text to be sent between simulated interrupts for those terminals without interrupt feature

Set by CRJETABL macro at assembly time
Checked by IHKLAP

CCTBRDNO - 2 bytes

Maximum number of broadcast messages that may be added to the already existing broadcast messages

Set by CRJETABL macro at assembly tine
Changed and checked by IHKCC3

CCTMSGNO - 2 bytes

Maximun number of delayed messages that may be added to the already existing delayed messages

Set by CRJETABL macro at assembly time
Changed and checked by IHKMSG

CCTSYSCI - 1 byte

Internal representation of SYSOUT code

Set by CRJETABL macro at assembly time
Checked by IHKOUT

CCTSYSCE - 1 byte

CCTMSGEC - 1 byte

External representation of SYSOUT code

Set by CRJETABL macro at assembly time
Checked by IHKOUT

CCTMSGRC - 1 byte

Not used

CCTSESS - 1 byte

Number of central consoles that have entered SHOW SESS commands

Set by IHKCC6
Checked by IHKLGN and IHKLGF

CCTPL1MX - 2 bytes

Maximum number of continuation lines (PL/1)

Set by CRJETABL macro
Used by IHKSYN and IHKIRL

CCTFRTMX - 2 bytes

Maximum number of continuation lines (FORTRAN)
Set by CRJETABL macro
Used by IHKIRL

CCTJOBS - 2 bytes

Maximum number of jobs that may be submitted by active users

Set by CRJETABL macro at assembly time
Checked by IHKSUB

CCTUSERS - 2 bytes

Maximum number of users that may be added to the system

Set by CRJETABL macro at assembly time
Checked and modified by IHKCC7

CCTPD1FG - 1 byte

Defines entry into PL/1 syntax checker
Set to zero by CRJETABL macro at assembly time

Bit Definitions

Bit 0 - Shows if work area exists zero indicates first entry checked by syntax checker If zero, syntax checker obtains and initializes work area (1K) set to 1 by IHKCIP

Bit 1 - 0 - New statement to be scanned 1 - Old input - checking to be continued from where last error occurred. Checked by syntax checker Set by IHKSYN when moved to TUB for linkage to the syntax checkers

Bit 2 - 0 - Normal syntax checking 1 - Last entry, no syntax checking Work area to be released

Bits 3-7 Not used

CCTPL1WA - 3 bytes

Address of work area obtained by PL/1 syntax checker on first entry for use in subsequent entries

CCTFRTFG - 1 byte

Same bit configuration as CCTPL1FG Defines entry into FORTRAN syntax checker

CCTFRTWA - 3 bytes

Address of work area for FORTRAN Stored by FORTRAN syntax checker on first entry

CCTCLECB - 4 bytes

ECB for passing control from IHKCCI to loader/controller

CCTJSECB - 4 bytes

ECB for passing control from IHKDEQ to IHKSRV

CCTCSECB - 4 bytes

ECB for passing control from central commands to IHKSRV

CCTJLECB - 4 bytes

ECB for passing control from IHKSDQ to loader/controller

CCTMSECB - 4 bytes

ECB for passing control from the IHKSRV timer routine to IHKSRV

CCTCCMDS - 2 bytes

Maximum number of central commands that may be queued

Set by CRJETABL macro at assembly line
Passed to OS by IHKCIP at start-up

CCTLNCNT - 2 bytes

Number of lines defined in the system
Set by CRJETABL macro at assembly time
Used by IHKBPM on warmstart

CCTUSR - 4 byte

Address of installation command processor
Set by CRJETABL macro at assembly time
Used by IHKCMD

CCTIAX - 4 bytes

Address of installation accounting exit
Set by CRJETABL macro at assembly time
Used by IHKLGN

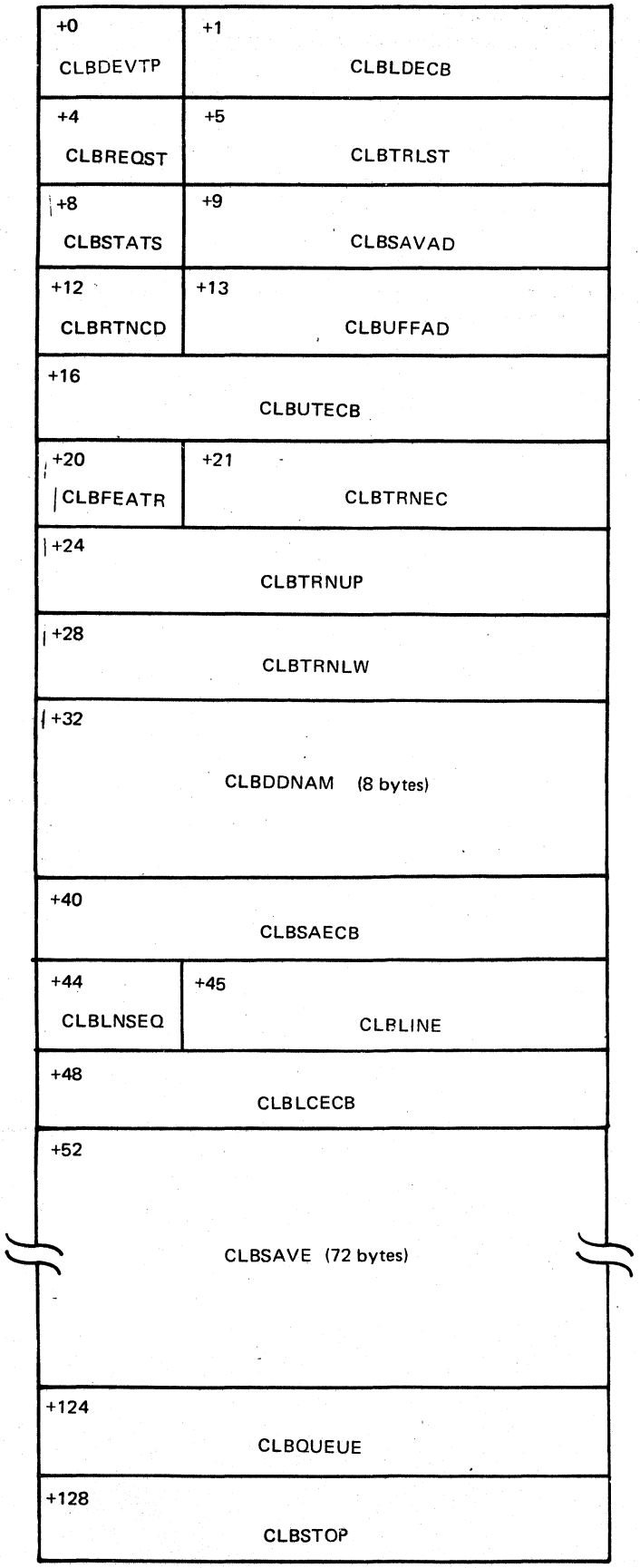
CCTISX - 4 bytes

Address of installation - defined JCL exit
Set by CRJETABL macro at assembly time
Used by IHKSUB

CCTILX - 4 bytes

Address of installation LOGOFF exit
Set by CRJETABL macro at assembly time
Used by IHKLGF

CONVERSATIONAL LINE BLOCK (CLB)



FIELD DESCRIPTIONS

CLBDEVTP 1 byte

The bit patterns of this byte reflect the line and terminal types associated with this CLB.

Bit Definitions

Bits 0 - Reserved for future expansion

Bits 1-3 - Not used

Bit 4 CLB2741 — If on, the terminal is a 2741
Set by CRJELINE macro at assembly time
Checked by IHKLAB, IHKLAP, and IHKLAD

Bit 5 CLB2740 — If on, the terminal is a 2740
Set by CRJELINE macro at assembly time
Checked by IHKLAB, IHKLAP, and IHKLAD

Bit 6 CLB1050 — If on, the terminal is a 1050
Set by CRJELINE macro at assembly time
Checked by IHKLAB, IHKLAP, and IHKLAD

Bit 7 CLBDIAL — If on, the line is a switched line
Set by CRJELINE macro at assembly time
Checked by IHKLAB, IHKLAP, and IHKLAD

CLBLDECB - 3 bytes

Contains the address of the BTAM DECB associated with the line.

Set by CRJELINE macro at assembly time
Used by IHKLAB, IHKLAP, and IHKLAD for BTAM I/O operations
Used by IHKCC2 for SHOW LERB and MODIFY

CLBREQST - 1 byte

Bit Definitions

Bit 0 - Not used

Bit 1 - TUB identifier, must be zero in CLB
Checked by IHKCMD and IHKERR.

Bits 2 - Not used

Bit 5 CLBSTOPN CLBSTOPF — If on, a MODIFY D or P CRJE was entered
Set by IHKCC2 or IHKSTP
Checked by IHKLAP and IHKCMD
Turned off by IHKERR

Bit 6 CLBRDTIN CLBRDTIF CLBREADN CLBREADF — If on, a CREAD I or a CREAD request has been received by line administrator.
Set by caller via CREAD macro
Checked by IHKLAD
Turned off by IHKLAD

Bit 7 CLBINITN CLBINITF — If on, a CREAD I request has been received by line administrator.
Set by caller via CREAD macro
Checked by IHKLAD and IHKLAP
Turned off by IHKLAD

CLBTRLST - 3 bytes

Contains the address of the BTAM DFTRMLST for initial READ.

Set by CRJELINE macro at assembly time
Used by IHKLAD and IHKLAP for initial
I/O operations on the line

CLBSTATS - 1 byte

The bits in this field reflect the line status and reasons for failing to complete caller's requests.

Bit Definitions

Bit 0 - If on, CLB/DCB is active
CLBACTVN Set by IHKLAD
CLBACTVF Checked by IHKCC2
Turned off by IHKLAD

Bit 1 - MODIFY has been requested for this line
CLBMODYN Set by IHKCC2
CLBMODYF Checked and reset by IHKCMD

Bit 2 - If on, CLB has been queued by programmed
CLBTQUEN time-out module (IHKLAY).
CLBTQUEF Set by IHKLAY
Checked by IHKLAD and IHKLAP
Turned off by IHKLAY

Bits 3-5 - Not used

Bits 6-7 - Not used

CLBSAVAD - 3 bytes

Contains the line save area address (used for saving registers during linkage between line administrator modules and between the IHKLAD module and the CRJE dispatcher).

Established by IHKLAD
Used by IHKLAD

CLBRTNCD - 1 byte

Contains the return code from BTAM after line administrator issues an I/O operation.

Set by IHKLAP
Checked by IHKLAD

CLBUFFAD - 3 bytes

Contains the address of the line buffer used for all communications between the CPU and the terminal.

Established by IHKLAD
Used by IHKLAB, IHKLAP, and IHKLAD

CLBUTECB - 4 bytes

ECB for passing control from command processors to IHKSRV

Posted by the requesting command processor
Cleared by IHKSRV

CLBFEATR - 1 byte

These fields reflect features possessed by all terminals that use this line.

Bit Definitions

Bits 0 - Reserved for future use

Bit 1 CLBSUPRS - If on, terminal is a leased 1050 with time-out suppression
Set by IHKCIP
Checked by IHKLAP

Bits 2-4 - Not used

Bit 5 CLBBCD - If on, terminal(s) on this line has PTTC/BCD code.

Bit 6 CLBCORRE - If on, terminal(s) on this line has correspondence code.

Bit 7 CLBREAK - If on, terminal has interrupt feature.
Set by CRJELINE macro at assembly time
Checked by IHKLAD

Note - If neither bit 5 nor 6 is on, the terminal on this line is assumed to have PTTC/EBCD code.

CLBTRNEC - 3 bytes

Contains the address of the translation table for translating EBCDIC text to be sent to the terminal.

Established by CRJELINE macro at assembly time
Used by IHKLAD

CLBTRNUP - 4 bytes

Contains the address of the translation table for translating text sent from the terminal into all-uppercase EBCDIC.

Established by CRJELINE macro at assembly time
Used by IHKLAD

CLBTRNLW - 4 bytes

Contains the address of the translation table for translating text sent from the terminal into EBCDIC, preserving upper and lower case letters.

Established by CRJELINE macro at assembly time
Used by IHKLAD

CLBDDNAM - 8 bytes

Contains EBCDIC representation of name specified in the DD statement defining the SYSIN data set.

Established by CRJELINE macro at assembly time
Used by IHKALC

CLBSAECB - 4 bytes

Stop acknowledgment ECB

Posted by IHKCIP and IHKCMD
Waited on by IHKSTP
Checked by IHKCC2 and IHKCLN

CLBLNSEQ - 1 byte

Line sequence number

Established by CRJELINE macro
Used by IHKLAD and AFIO

CLBLINE - 3 bytes

Contains the EBCDIC representation of physical line address.

Set at OPEN time by IHKCIP
Used by IHKLGN in LOGON message
Used by IHKCC2, IHKCC1, and IHKCC8

CLBLCECB - 4 bytes

ECB for passing control to IHKLDC

Posted by IHKCMD, IHKCLN, IHKPUT, IHKSUB,
IHKCCI, and IHKSTS
Cleared by IHKLDC

CLBMCSCD - 1 byte

ID of console issuing MODIFY command

CLBSAVE - 72 bytes

Save area for registers of IHKCMD on a CREAD I

Established by CRJELINE macro at system assembly time
Used by IHKCMD only. These registers are restored and
returned if there is no TUB.
The first byte is also used for passing the ID of the
console requesting a MODIFY from IHKCC2 to IHKCMD for use
when the request has been satisfied.

CLBQUEUE - 4 bytes

Contains chain pointer for queuing of CLBs for leased 1050 terminals
with time-out suppression.

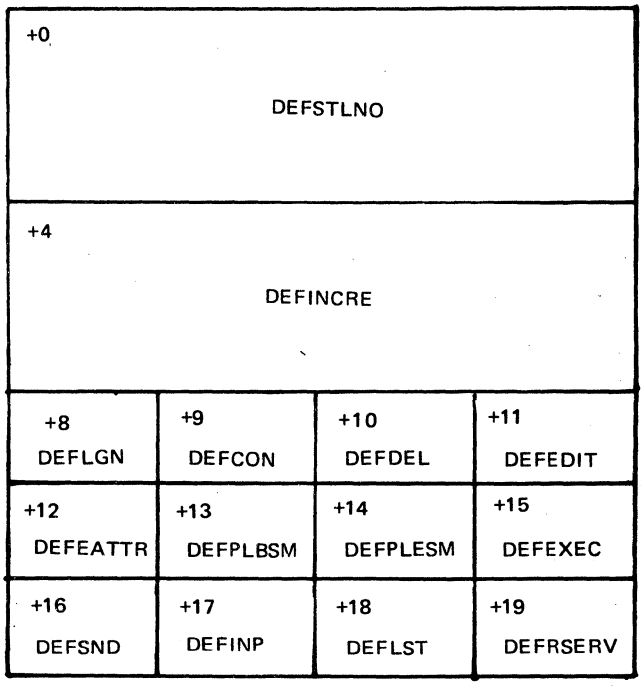
Used exclusively by IHKLAY for
queuing and dequeuing purposes.

CLBSTOP - 4 bytes

Contains time at which this CLB will be dequeued and an IOHALT will be
issued on the line if there is no response from the line.

Used exclusively by IHKLAY for
timing purposes

TERMINAL COMMAND DEFAULT TABLE (DEF)



FIELD DESCRIPTIONS

DEFSTLNO - 4 bytes

Default data set starting line number

DEFINCRE - 4 bytes

Default data set increment

DEFLGN - 1 byte

LOGON defaults

Bit 0 - Not used

Bit 1 - Message IDs requested
DEFLMID

Bit 7 - Broadcast messages requested
DEFLBC

DEFCON - 1 byte

CONTINUE defaults

Bit 6 - Continue at beginning
DEFCBGN

Bits 5-6 - Continue at next data set
DEFCNXT

Note: If bits 5 and 6 are both off, CONTINUE HERE is assumed.

DEFDEL - 1 byte

DELETE (major) defaults

Bit 0 - Purge
DEFDPURG

DEFEDIT - 1 byte

EDIT options

Bit 0 - New data set
DEFENEW
DEFENEWF

Bit 1 - Line numbers in line
DEFENUM
DEFENUMF

Bit 2 - Syntax scan requested
DEFESCAN
DEFESCNF

Bit 5 - PL/1 with 48 character set
DEFPL148

DEFEATTR - 1 byte

Content attribute of data set

Bit 1 & 7 - FORTRAN H
DEFFORTH

Bit 1 - FORTRAN G
DEFFORTG

Bit 2 - Data
DEFADATA

Bit 3 - Text
DEFATEXT

Bit 4 - Data set list
DEFADLST

Bit 5 - Command List

Bit 6 - PL/1
DEFPL1

Bit 7 - FORTRAN E
DEFFORTE

DEFPLBSM - 1 byte

PL/1 beginning source margin

DEFPLESM - 1 byte

PL/1 ending source margin

DEFEXEC - 1 byte

EXEC defaults

Bit 0 - List requested
DEFELST

DEFSND - 1 byte

Send defaults

Bit 0 - Do not queue messages
DEFSNOW

DEFINP - 1 byte

Input defaults

Bit 0 - line number prompts requested
DEFIREP

DEFLST - 1 byte

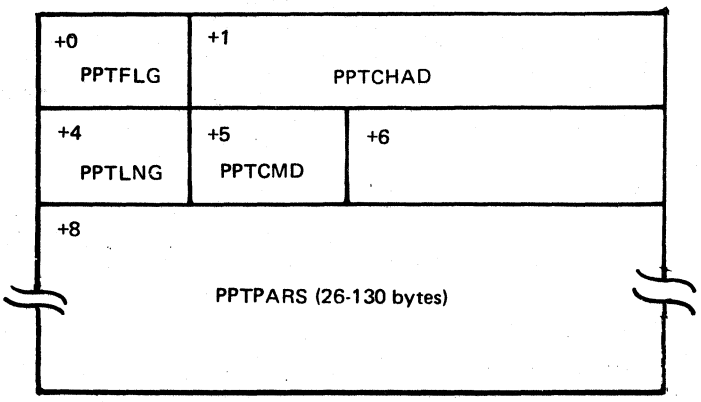
List defaults

Bit 0 - line numbers to be listed
DEFLNUM

DEFRSERV - 1 byte

Not used

PARAMETER POSITION TABLE (PPT)



FIELD DESCRIPTION

PPTFLG - 1 byte

Bits reflect a condition of status within the system, If on, the condition exists; if off, the condition does not exist.

Bit Definitions

Bits 0-7 - used internally in IHKCMD and in command processors

PPTCHAD - 3 bytes

Contains address of the next PPT.

Set by IHKCMD
Checked by command processors
Turned off by IHKCMD

PPTLNG - 1 byte

Contains length of this PPT.

Set by IHKCMD
Utilized by command processors and IHKCMD
Turned off by IHKCMD

PPTCMD - 1 byte

Contains the command code.

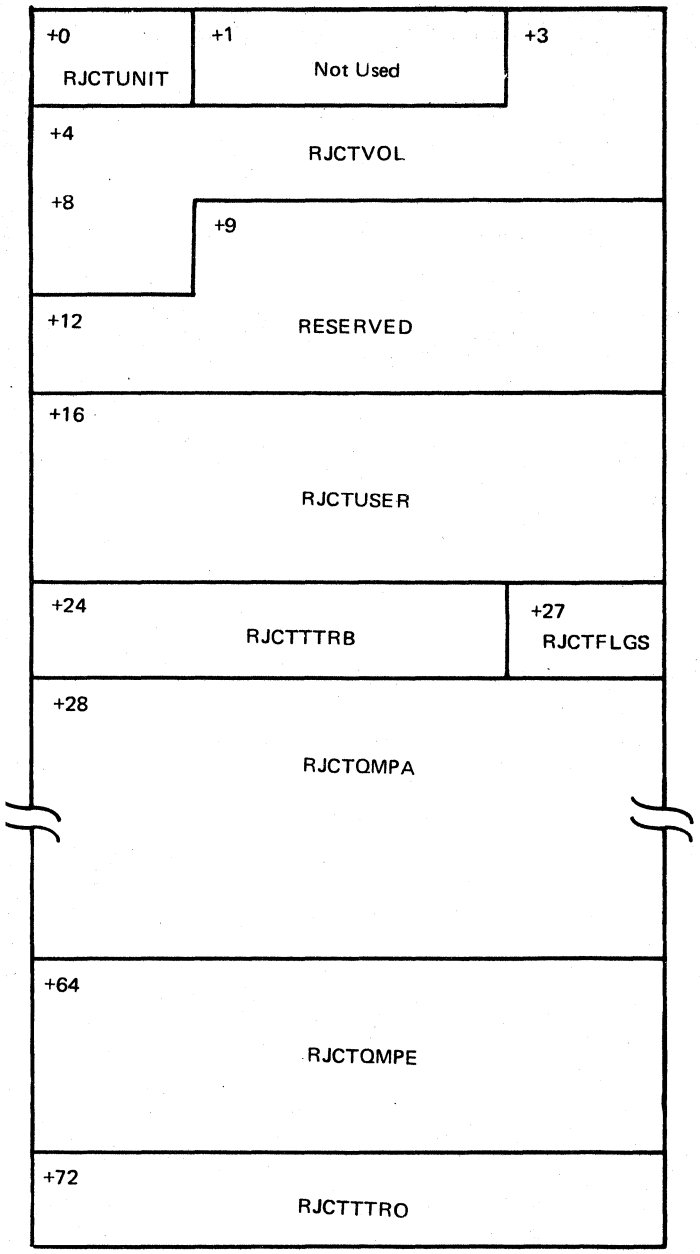
Set by IHKCMD
Checked by command processors
Turned off by IHKCMD

PPTPARS - Variable

Contains command parameters (delimiters are removed). Each parameter is preceded by one byte indicating the length of the parameter. The first

bit of the length indicator is set if the parameter was enclosed in parentheses, and was the last parameter within the parentheses. The second bit is set for a parameter which was in parentheses but was not the last. The last parameter in the PPT is succeeded by X'FF'.

REMOTE JOB CONTROL TABLE (RJCT)



FIELD DESCRIPTIONS

RJCTUNIT - 1 byte

Contains last character of unit for which 'S RDRCRJE' command was issued. Either 1 for 2311 or 4 for 2314.

Set by IHKALC
Used by IHKCIP

Not used - 2 bytes

RJCTVOL - 6 bytes

Contains volume serial of 'S RDR' unit.

Set by IHKSUB
Used by IHKCIP

Reserved - 7 bytes

RJCTUSER - 8 bytes

Contains identification of user submitting the job.

Set by IHKSUB
Checked by IHKOUT, IHKDEQ,
IHKCC1, IHKSTS, and IHKRER03

RJCTTTRB - 3 bytes

Contains relative track and record address of first block of SYSOUT data set to be sent upon receipt of a CONTINUE HERE command.

Set by IHKPUT
Checked by IHKOUT

RJCTFLGS - 1 byte

Each bit reflects a condition of status of this job. If on, the conditionn exists; if off, the specified condition does not exist. All bits are initialized to OFF by IHKSUB.

Bit Definitions

Bit 0 - Notification message sent
Set by IHKDEQ
Checked by IHKDEQ
Turned off by IHKDEQ

Bit 1 - Job complete
Set by IHKDEQ
Checked by IHKDEQ, IHKOUT, IHKCC5,
IHKCC1, and IHKSTS
Turned off by IHKOUT and IHKCIP (warmstart)

Bit 2 - Job was first job in a SUBMIT command
Set by IHKSUB
Checked by IHKCIP (to determine if any S RDR commands were issued by CRJE but not processed by OS).

Bit 3 - Job terminated abnormally
Set by IHKDEQ
Checked by IHKSTS and IHKCC1

Bit 4 - Job to be canceled
Set by IHKRER03
Checked by IHKDEQ

Bits 5-7 - Reserved for future use

RJCTQMPA - 36 bytes

Contains information to transmit output from this job

OS queue manager parameter area
Set by IHKDEQ

Checked by IHKPUT, IHKRER01
and IHKCC5

RJCTJOB - First 8 bytes of RJCTQMPA

Contains name of the job from the user's job card (also key for the record). It identifies the job within the CRJE system.

Set by IHKSUB
Checked by IHKSUB, IHKOUT, IHKCC5,
IHKDEQ, IHKSTS and IHKCC1
Turned off (when entry is deleted)
by IHKOUT, IHKCC5, IHKDEQ, and IHKCIP
(coldstart specified)

RJCTQMPE - 8 bytes

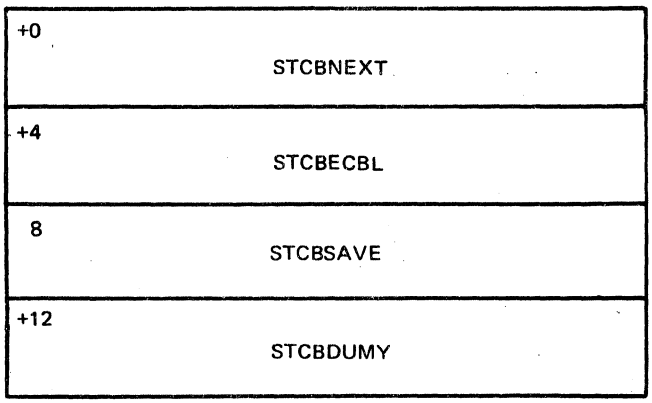
OS external queue manager parameter area is an extension of the RJCTQMPA.

Set by IHKDEQ
Checked by IHKOUT and IHKCC5

RJCTTTR0 - 4 bytes

Contains relative track and record address of the next SMB/DSB.
Modified by IHKPUT and IHKSUB

SUBTASK CONTROL BLOCK (STCB)



FIELD DESCRIPTIONS

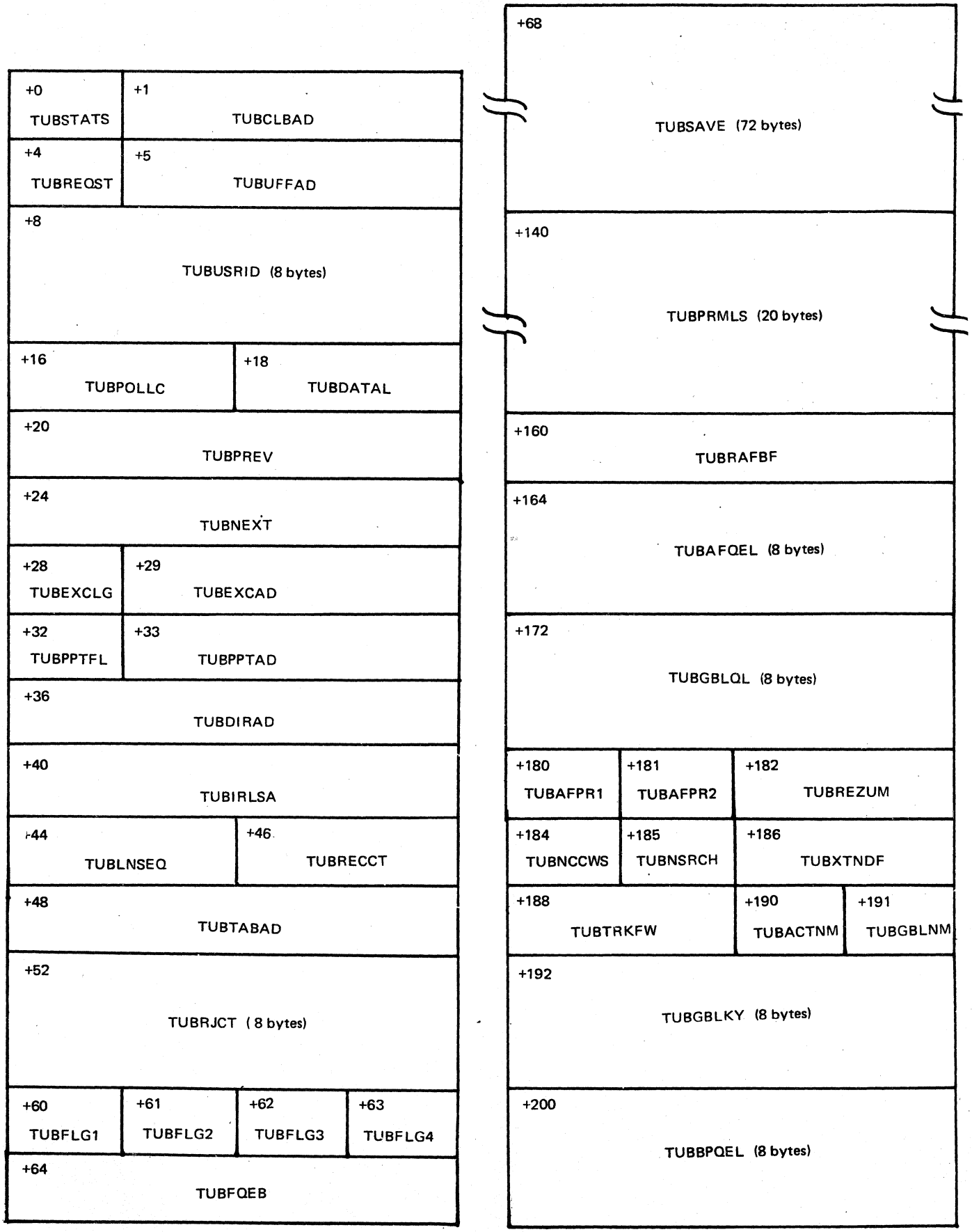
STCBNEXT - 4 bytes
Address of next STCB in chain

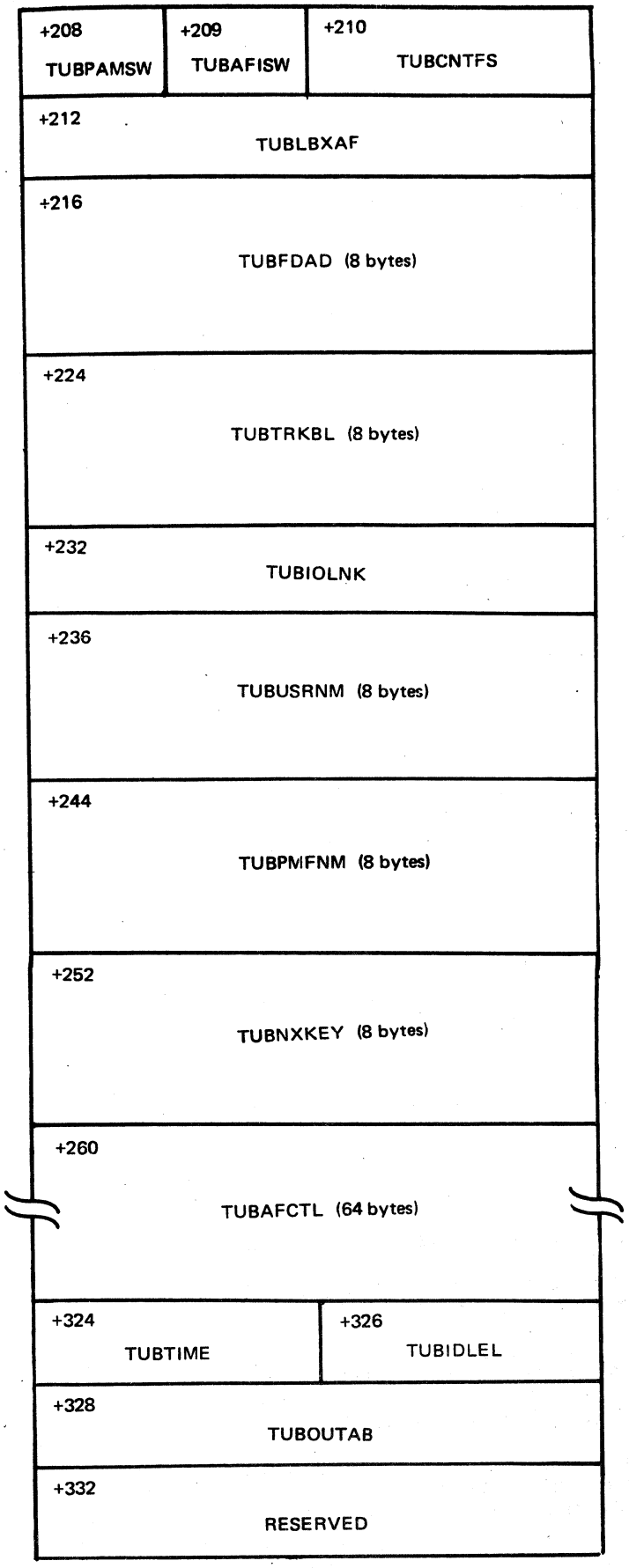
STCBECBL - 4 bytes
ECB list entry address

STCBSAVE - 4 bytes
Caller's save area address

STCBDUMY - 4 bytes
Dummy ECB for initialization

TERMINAL USER BLOCK (TUB)




+208
TUBPAMSW
+209
TUBAFISW
+210
TUBCNTFS
+212
TUBLBXAF
+216
TUBFDAD (8 bytes)
+224
TUBTRKBL (8 bytes)
+232
TUBIOLNK
+236
TUBUSRNM (8 bytes)
+244
TUBPMFNM (8 bytes)
+252
TUBNXKEY (8 bytes)
+260
TUBAFCTL (64 bytes)
+324
TUBTIME
+326
TUBIDLEL
+328
TUBOUTAB
+332
RESERVED

FIELD DESCRIPTIONS

TUBSTATS - 1 byte

Bits reflect status of terminal and/or user.

Bit Definitions

Bit 0 - Line administrator error recovery in progress
TUBERPN
TUBERPF
Set and checked by IHKLAD.

Bit 1 - Interrupt (real or simulated) has occurred.
TUBREAKN
TUBREAKF
Used within line administrator to indicate simulated interrupt should be performed, and to indicate to caller that interrupt has occurred.
Set by IHKLAP
Turned off by IHKERR
Checked by IHKMSG and IHKLAD

Bit 2 - Terminal has not responded to polling.
TUBTIMEN
TUBTIMEF
Set by IHKLAD
Checked by IHKERR

Bit 3 - Terminal off-line/on-hook
TUBOFFLN
TUBOFFLF
Set by IHKLAD

Bit 4 - User is queued for job deletion.
TUBRERNQ
Used only by IHKRER

Bit 5 - Not used.

Bit 6-7 - Line administrator ERP count flags
TUBERPN
TUBERPF
Used within IHKLAD to control retry counts on line errors
Set, checked, and reset by IHKLAD

TUBCLBAD - 3 bytes

Contains the address of the CLB.

Inserted by IHKLAD at allocation
Used by IHKLGN, IHKLGF, IHKCC1, IHKCC8, IHKLAB, IHKLAP, IHKCMD, and IHKERR

TUBREQST - 1 byte

The bits of this byte reflect I/O requests for the line.

Bit Definitions

Bit 0 - not used

Bit 1 - Identifies this as a TUB. Must be on at all times.
TUBIDENT
Set at allocation time by IHKLAD
Checked by IHKCMD and IHKERR

Bit 2 - Indicates next block of text written must be preceded by carrier return and idles.
TUBIDLEN
TUBIDLEF
Set by IHKLAD if text read has no ending CR
Checked and reset by IHKLAB

Bit 3 - Indicates occurrence of error on the line and that line must be reset.
TUBRSETN
TUBRSETF
Set by CWRITE R macro

Checked by IHKLAD
Reset by IHKLAD (after CREAD R)

Bit 4 - Upper/lower case preservation
TUBTEXTN Set by IHKIRL and IHKCMD
TUBTEXTF Turned off by IHKLAD

Bit 5 - Indicates a write request.
TUBWRITN Set by CRWRITE macro
TUBWRITF Checked by IHKLAP and IHKLAD
Reset by IHKLAD and IHKLAP

Bit 6 - Indicates a read request.
TUBREADN Set by CREAD or CWRITE macro
Checked by IHKLAP and IHKLAD
Reset by IHKLAD

Bit 6 & 7 - Indicates an initial operation.
TUBRDTIN Set by CREAD I macro
TUBRDTIF Checked by IHKLAP and IHKLAD
Reset by IHKLAD

TUBUFFAD - 3 bytes

Contains user buffer address.

Created and destroyed by IHKLAD
Used for passing data between the line administrator modules and between IHKLAD and its callers.

TUBUSRID - 8 bytes

Contains user's identification sequence.

Inserted by IHKLGN
Utilized by most command processors and IHKMSG
Checked by IHKLAP at request for on-line terminal test

TUBPOLLC - 2 bytes

Reserved

TUBDATAL - 2 bytes

Contains length of data being passed in the user buffer.

TUBPREV - 4 bytes

Contains address of the previous TUB in the chain.

Set by IHKLAD
Modified by IHKLAD
Used by IHKLAB

TUBNEXT - 4 bytes

Contains address of the next TUB in the chain.

Set by IHKLAD
Modified by IHKLAD
Used by IHKLAB, IHKMSG, IHKCC1, IHKCC5, and IHKCC8

TUBEXCLG - 1 byte

TUBEXCAD - 3 bytes

Contains length and address of EXEC work area.

Inserted by IHKEDT
Utilized by IHKED1 and IHKCMD
Reset by IHKCMD and IHKERR

TUBPPTFL - 1 byte

These flags reflect the status of the user. They are set, checked and reset only by IHKCMD, except for bit 0 which is turned off by IHKERR.

Bit 0 - Active area I/O error
TUBAARRN
TUBAARRF

Bit 1 - Null parameter (empty parentheses)
TUBNULPN
TUBNULPF

Bit 2 - Warning message to be sent (excessive commas).
TUBWARNN
TUBWARNF

Bit 3 - Comma was recognized as delimiter.
TUBCOMAN
TUBCOMAF

Bit 4 - User is prompted for LOGON once.
TUBLGNRN
TUBLGNRF

Bit 5 - Interrupt has occurred (real or simulated).
TUBRAKEN
TUBRAKEF

Bit 6 - This is a continuation line.
TUBCNTEN
TUBCNTEF

Bit 7 - This line is to be continued.
TUBCNTSN
TUBCNTSF

TUBPPTAD - 3 bytes

Contains the address of the parameter position table associated with this user.

Created by IHKCMD
Utilized by command processors
Cleared by IHKCMD

TUBDIRAD - 4 bytes

Contains address of the PDS directory entry for the data set currently in the active area for this user.

Set by IHKIRL, IHKED1, and IHKEOS
Used by most command processors and IHKLAD
Used by IHKLGN as temporary work area.
Cleared by IHKEND

TUBIRLSA - 4 bytes

Contains address of IHKIRL GETMAIN area saved during correction mode; or
Contains address of IHKIRL work area saved during input scan.

Set by IHKIRL or IHKSCN during scan
Used by IHKSYN, IHKCGN, and IHKSCN
Used by IHKLGN, IHKMUF, IHKOPN, IHKALC, IHKGET, and IHKMGE
IHKGET, and IHKMGE as temporary work area
Cleared by IHKIRL

TUBLNSEQ - 2 bytes

Contains unique identification number for this particular user.

Assigned by IHKLAD at TUB allocation time
Utilized by AFIO/BPAM and IHKUTM

TUBRECCT - 2 bytes

Contains number of blocks sent to this terminal without an intervening READ.

Used in interrupt simulation
Set by the CRJELINE macro at assembly time
Incremented, checked and reset by IHKLAP

TUBTABAD - 4 bytes

Contains address of tab setting information.

Inserted by IHKTAB
Utilized by IHKLAT and IHKLEW
Checked by IHKLAD
Reset by IHKLGF and IHKTAB

TUBRJCT - 8 bytes

Contains the jobname of a discontinued job.

Set by IHKOUT
Checked by IHKOUT and IHKCC5

TUBFLG1 - 1 byte

Bits reflect either a parameter option or a condition of status within the system. If on, the condition exists; if off, the specified condition does not exist.

Bit Definitions

Bit 0 - Line numbers are requested on the list.
TUBLNO
Set by IHKLST
Checked and reset by IHKLST

Bit 1 - Message identifiers requested.
TUBMID
Set by IHKLGN and by IHKLAD at time of TUB allocation

TUBMIDF
Checked by IHKMSG and IHKLDS

Bit 2 - Automatic syntax scan requested.
TUBSCN
Set by IHKEDT, IHKSCN, and IHKIRL
TUBSCNF
Checked by IHKIRL
Turned off by IHKSCN, IHKEND and IHKIRL

Bit 3 - Data set in active area has FORTRAN attribute.
TUBFOR
Set by IHKEDT, IHKED1, and IHKIRL
Checked by IHKSCN, IHKSYN and IHKIRL
Turned off by IHKEND

Bit 4 - Data set in active area has PL/1 attribute.
TUBPL1 Set by IHKEDT, IHKED1, and IHKIRL
Checked by IHKSCN, IHKSYN and IHKIRL
Turned off by IHKEND

Bit 5 - Delayed messages exist for user.
TUBCMSGN Set, checked and turned off
TUBCMSGF by IHKMSG

Bit 6 - Message is queued for user.
TUBMSG Set, checked and turned off by IHKUTM
TUBMSGF Checked by IHKMSG

Bit 7 - Broadcast messages requested.
TUBBRD Set by IHKLGN and IHKLST
TUBBRDF Checked and turned off by IHKMSG

TUBFLG2 - 1 byte

Bits reflect either a parameter option or a condition of status within the system. If on, the condition exists; if off, the specified condition does not exist.

Bit Definitions:

Bit 0 - List commands before processing.
TUBEXLST Set by IHKEDT
TUBXLSTF Checked by IHKCMD
Turned off by IHKCMD

Bit 1 - Message lost for this user.
TUBLMSGN Set and checked by IHKUTM and IHKMSG
TUBLMSGF Turned off by IHKMSG

Bit 2 - EDIT mode switch
TUBEDIT Set by IHKEDT
TUBEDITF Checked by IHKCMD, IHKERR, and IHKLAD
Turned off by IHKEND

Bit 3 - Abnormal termination for user
TUBABEND Set by IHKERR
Checked by IHKLGF and IHKEND

Bit 4 - Discontinued output.
TUBDISOP Set by IHKPUT
Checked by IHKOUT and IHKPUT
Turned off by IHKPUT

Bit 5 - Line nymbers to be included in data set line.
TUBLNUMN Set by IHKED1, IHKEDT, and IHKIRL
TUBLNUMF Checked by IHKLST, IHKCGN, IHKRNR, IHKIRL, IHKEOS, IHKMAA, IHKMUF, and IHKED1
Turned off by IHKEND

Bit 6 - IHKLGN did not complete processing and returned a bad return code.
TUBLGNAB Set by IHKLGN
Checked by IHKCMD and IHKLGF

Bit 7 - System messages specified in output.
TUBSMSG Set by IHKOUT
Checked by IHKOUT
Turned off by IHKOUT and IHKDEQ

TUBFLG3 - 1 byte

Bits reflect either a parameter option or a condition of status within the system. If on, the condition exists; if off, the specified condition does not exist.

Bit Definitions

Bit 0 - Line number prompts requested.
TUBLNPMT Set by IHKEDT, IHKED1, IHKIPT, and IHKIRL
TUBLNPMF Checked by IHKIRL
Turned off by IHKIPT and IHKEND

Bit 1 - No carrier return made at end of text
TUBNOCRN and trailing blanks not deleted.
TUBNOCRF Set by IHKMSG, IHKCMD, and IHKIRL
Checked by IHKLAP
Checked and turned off by IHKLAD

Bit 2 - Syntax error corrections to be made.
TUBCORRN Set by IHKSYN
TUBCORRF Checked by IHKCMD, IHKCGN, and IHKIRL
Turned off by IHKIRL

Bit 3 - Active area I/O error
TUBUTMN Set by IHKERR
TUBUTMF Checked by IHKLGF

Bit 4 - N-lines override (no simulated interrypt)
TUBOVERN Set by IHKLAD
TUBOVERF Checked by IHKLAP
Turned off by IHKLAD and IHKLAP

Bit 5 - Data set logical record length is 80-120
TUBLONGN Set by IHKEOS
TUBLONGF Checked by IHKLST
Turned off by IHKEND

Bit 6 - Correction has been entered.
TUBCORCN Set by IHKIRL and IHKCGN
TUBCORCF Checked and turned off by IHKIRL

Bit 7 - If bit is set, delayed scan is on.
TUBDLAYN Set by IHKSCN
TUBDLAYF Checked by IHKSYN
Turned off by IHKSCN

TUBFLG4 - 1 byte

Bit Definitions

Bit 0 - User logged off due to MODIFY or STOP.
TUBSTOP Set by IHKERR and IHKCC2
Checked by IHKLGF and IHKERR

Bit 1 - IHKLAB entered to append idles at
TUBIDAPN end of text.
TUBIDAPF Set, checked, and reset by IHKLAB

Bit 2 - Unused

Bit 3 - User has an 80-character library.
TUBLIB80 Set by IHKBPM
Checked and reset by IHKAWS and IHKSAV

Bits 4-7 - Unused

Reserved for future expansion

TUBFQEB - 4 bytes

Contains address of the storage message queued for this user.

Set, checked, modified, and cleared by IHKMSG
Checked by IHKLAD to determine if message area is to be freed prior to deallocation of TUB.

TUBSAVE - 72 bytes

Save area

Utilized by AFIO
Second word used by IHKLAD to store caller's save area address.

TUBPRMLS - 20 BYTES (TUBPARM1-5)

Parameter list area

First two words will be modified by AFIO whenever it is called.

TUBRAFBF - 4 bytes

Used as a pointer to the buffer for the INSERT macro and must be initialized by the processor issuing the INSERT macro. Also used by AFIO when an RGET macro is issued to point to the dynamically-allocated buffer that the record(s) was read into. Also used by RPOINT macro with R operand.

TUBAFQEL - 8 bytes

Active file queue element - used by AFIO and IHKRNQ for gaining access to the restricted work area.

TUBGBLQL - 8 bytes

Global file queue element
Used by AFIO to queue requests for access to the global files.

TUBAFPAR - 2 bytes (TUBAFPR1 and TUBAFPR2)

Switches used by the AFIO macro interpreter routines only.

TUBREZUM - 2 bytes

Return displacement used by AFIO macro interpreter only.

TUBNCCWS - 1 byte

This byte holds the number of CCWs generated for an I/O operation; used by AFIO only.

TUBNSRCH - 1 byte

Contains maximum number of search arguments. (Used by AFIO only)

TUBXTNDF - 2 bytes

Contains extended work area required indicator. (Used only by AFIO.)

TUBTRKFW - 2 bytes

Contains track address indicator. (Used by AFIO only.)

TUBACTNM - 1 byte

Contains number of the active file associated with this TUB.
Set by processor with CREATE macro to either 1 (default) or 2 (private utility)
Checked by AFIO

TUBGBLNM - 1 byte

Same as TUBACTNM except relation is to global files. This field (instead of TUBACTNM) is used when the FTYPE=parameter is coded in the CREATE macro.

TUBGBLKY - 8 bytes

This field contains the key of the record desired in a global file. It is used by AFIO and by processors when it is desired to POINT, REPLACE, DELETE, RGET, or SKIP a specific record or key.

TUBBPQEL - 8 bytes

Library I/O queue element - used by library I/O module and the IHKRNQ module to queue requests for access to the system or user libraries.

TUBPAMSW - 1 byte

This field is used as a byte of switches for the library I/O modules.

TUBAFISW - 1 byte

Contains internal switches for restart and buffer control for AFIO and library I/O.

Bit 0 - Use user buffer, not GETMIAN area
TUBFCTLN Set by callers to AFIO, BPAM
TUBFCTLF

Bit 1 - Warmstart in progress
Set by IHKAWS

Bit 2 - Warmstart completed
Set by IHKAWS

TUBCNTFS - 2 bytes

This field is used by the processors requesting multiple AFIO operations and by AFIO itself. If a multiple RGET, for instance, is desired, this field is initialized to reflect the number of records requested.

TUBLBXAF - 4 bytes

AFIO work area - used for user-oriented processing.

TUBFDAD - 8 bytes

Used by Lib I/O for track address

TUBTRKBL - 8 bytes

Track balance information for Lib I/O

TUBIOLNK - 4 bytes

AFIO linkage - used only by AFIO.

TUBUSRNM - 8 bytes

This field contains the userid of the user who owns the file.
Set by IHKUTM, IHKLDS, and IHKEDT

TUBPMFNM - 8 bytes

This field contains the name of the library or user library file being processed. It is initialized by IHKEDT and IHKLDS for the users of library I/O.

TUBNXKEY - 8 bytes

This field is used by processors and by AFIO. When,for instance, POINT is used to point to a particular record, the key for that record should be in TUBNXKEY. When AFIO retrieves a key upon request, it is retrieved into TUBNXKEY.

TUBAFCTL - 64 bytes

This area is used by AFIO only for control of this user's active file area.

TUBTIME - 2 bytes

Time of day of LOGON
Inserted by IHKLGN
Utilized by IHKLGN, IHKLGF, and IHKCC1

TUBIDLEL - 2 bytes

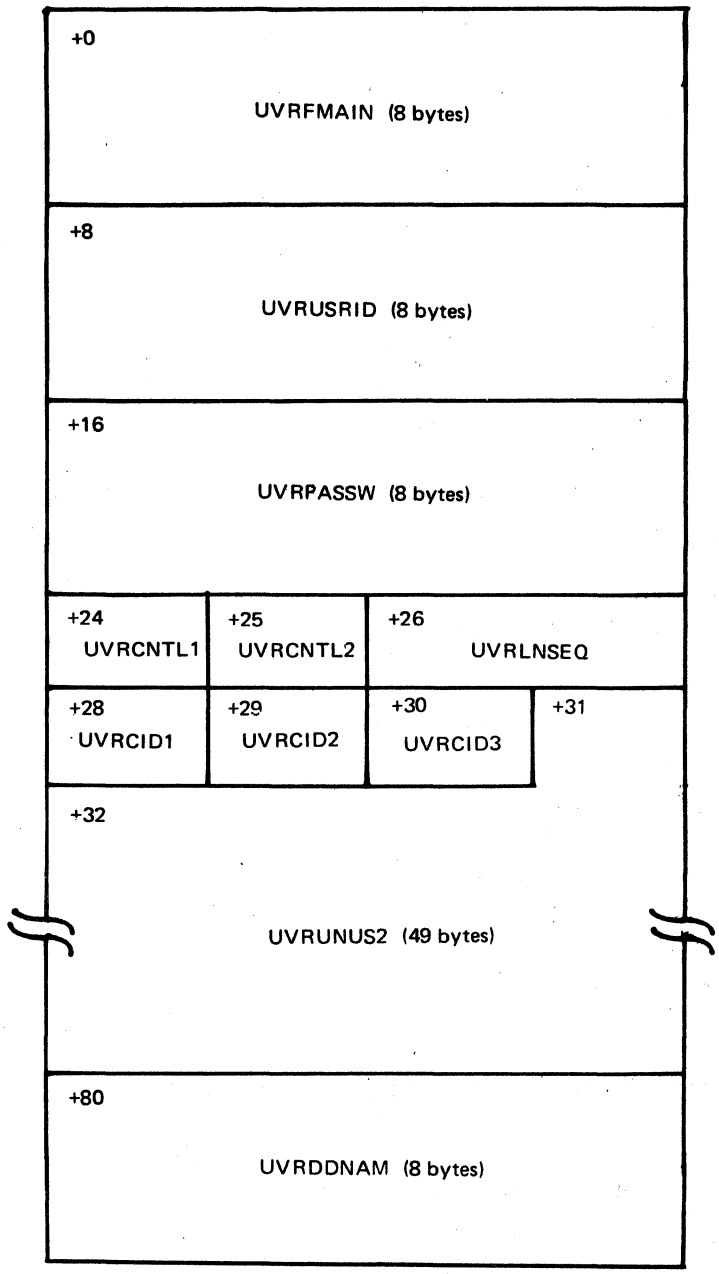
Logical data length of previous line operation
Inserted by IHKLAD
Utilized by IHKLAB

TUBOUTAB - 4 bytes

Address of output tab setting information
Set by IHKTAB
Used by IHKLEW
Checked by IHKLAD
Reset by IHKTAB

Reserved - 4 bytes

USER VERIFICATION RECORD (UVR)



FIELD DESCRIPTIONS

UVRFMAIN - 8 bytes

Reserved

UVRUSRID - 8 bytes

Contains userid by which record is located.

UVRPASSW - 8 bytes

Contains corresponding password.

UVRCNTL1 - 1 byte

Bits reflect control information.

Bit Definitions

Bit 0 UVRSESS UVRSESSF	-	Session is to be indicated to operator Set only if MCS is not in system
Bit 1 UVRDMSG UVRDMSGF	-	Delayed messages are available. Set, checked, and turned off by IHKMSG Turned off by IHKCC4
Bit 2 UVRABTM1	-	Indicates abnormal termination. Set by IHKUTM
Bit 3 UVRABTM2	-	Indicates abnormal termination.
Bit 4	-	Unused
Bit 5 UVRLMSGN UVRLMSGF	-	Delayed message for this user was lost. Set, checked and turned off by IHKUTM Set by IHKMSG
Bit 6 UVRLIBIN	-	User's library was inoperative last session. Set and checked by IHKUTM
Bit 7 UVRACTVN UVRACTVF	-	Indicates that user is active. Set, checked and turned off by IHKUTM Checked by IHKCC1 and IHKCC7

UVRCNTL2 - 1 byte

Used to facilitate user message handling.

UVRLNSEQ - 2 bytes

TUB identification field to facilitate warmstart following abnormal termination of CRJE.

Set by IHKUTM
Used by IHKAWS to retrieve user's active area
Cleared by IHKUTM

UVRCID1 - 1 byte

Console ID of an operator's console that requested notification whenever this user logs on or logs off.

Set by IHKCC6 only when MCS is in system.
Passed to IHKLGN and IHKLGF by IHKUTM
Cleared by IHKCC6

UVRCID2 - 1 byte

See UVRCID1

UVRCID3 - 1 byte

See UVRCID1

UVRUNUS2 - 49 bytes

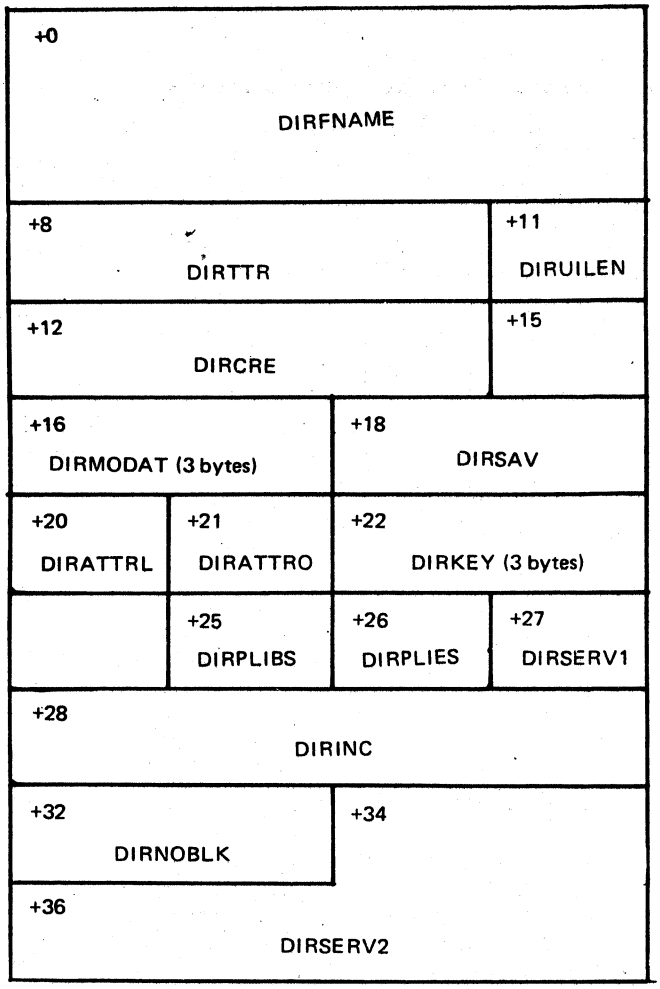
Not used

UVRDDNAM - 8 bytes

Contains the name of the DD card describing the user library.

Checked by IHKUTM

CRJE-CREATED USER LIBRARY DIRECTORY ENTRY (DIR)



FIELD DESCRIPTIONS

DIRFNAME - 8 bytes
Member name

DIRTTR - 3 bytes
TTR for beginning of member

DIRUILEN - 1 byte
Number of halfwords of user information

DIRCRE - 3 bytes
Creation date

DIRMODAT - 3 bytes
Date last modified

DIRSAV - 2 bytes
Number of times read

DIRATTRL - 1 byte
File type

Bit Definitions

Bit 0 - PL/1 with 48 character set
DIRPL148

Bit 1 - FORTRAN G
DIRFORTG

Bit 2 - Data
DIRADATA

Bit 3 - Text
DIRATEXT

Bit 4 - Data set list
DIRADLST

Bit 5 - Command list
DIRACLST

Bit 6 - PL/1
DIRPL1

Bit 7 - FORTRAN E
DIRFORTE

Bits 1 & 7- FORTRAN H
DIRFORTH

DIRATTRO - 1 byte

Other attributes

Bit 0 - line numbers are contained in line.
DIROLNO
DIROLNOF
Bit 1 - EDIT old data set
DIROLD
DIROLDF

DIRKEY - 3 bytes

Data set key

DIRPL1BS - 1 byte

PL/1 beginning source margin

DIRPL1ES - 1 byte

PL/1 ending source margin

DIRSERV1 - 1 byte

Reserved

DIRINC - 4 bytes

Line number increment
Set by IHKIRL
Reset by IHKRNR

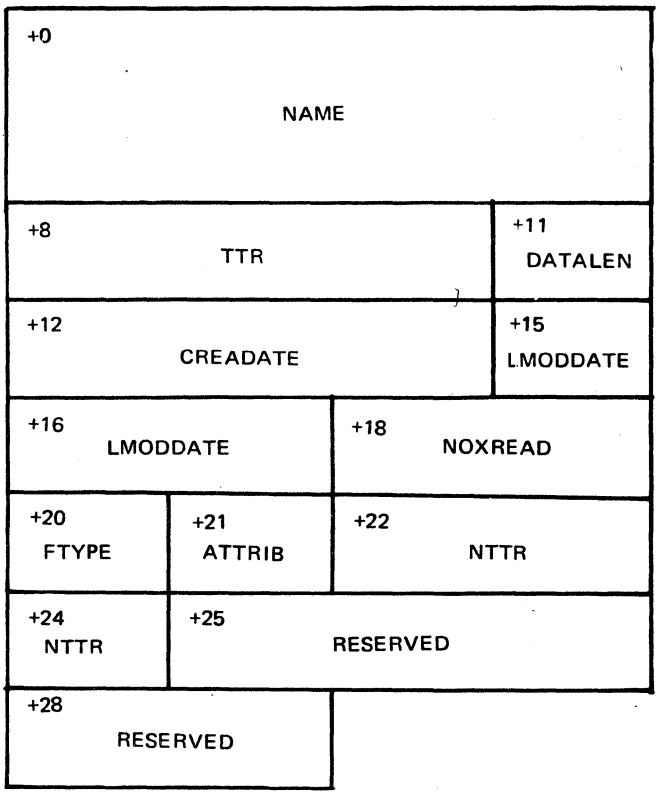
DIRNOBLK - 2 bytes

Number of blocks in data set

DIRSERV2 - 6 bytes

Reserved

CRBE-CREATED USER LIBRARY DIRECTORY ENTRY



FIELD DESCRIPTIONS

NAME - 8 bytes
Member name

TTR - 3 bytes
Pointer to beginning of member

DATALEN - 1 byte
Length of user information in halfwords

CREADATE - 3 bytes
Creation date

LMODDATE - 3 bytes
Date last modified

NOXREAD - 2 bytes
Number of times read

FTYPE - 1 byte
File type

Bit Definitions

Bit 0	-	not used
Bit 1	-	FORTRAN file
Bit 2	-	Non-FORTRAN
Bit 3	-	Object module
Bit 4	-	File list
Bit 5	-	Record length less than 120 and greater than 80
Bits 6-7	-	Not used

ATTRIB - 1 byte
Other file attributes

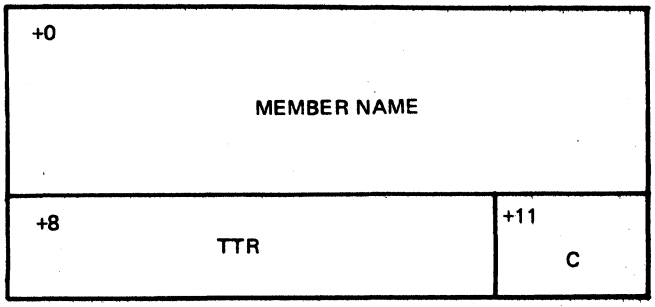
Bit Definitions

Bit 0	-	Scan requested
Bit 1	-	Line numbers contained in line
Bit 2	-	Lower to upper case conversion not desired (not implemented)
Bits 3-7	-	Not used

NTTR (PTRONAR)- 3 bytes
Pointer to narrative (TTR)

RESERVED - 5 bytes
Not used

UTILITY-CREATED USER LIBRARY DIRECTORY ENTRY



FIELD DESCRIPTIONS

MEMBER NAME - 8 bytes
Member name

TTR - 3 bytes
Pointer to beginning of member

C - 1 byte
Length of user data in halfwords

CHART OF GENERAL REGISTER USAGE BY MODULE

Module	Register		Contents
IHKALC	2	-	DECB base
	3	-	RJCT base
	4	-	DCB address
	5	-	TUB base
	6	-	AVT base
	12	-	base
IHKAST	8	-	KONBOX
	10	-	base
	12	-	AVT base
	14	-	active area DCB base
IHKAWS	7	-	AVT base
	8	-	KONBOX base
	9	-	TUB active file work area address
	10	-	base
	12	-	TUB base
IHKBGN	10	-	AVT base
	12	-	base
IHKBPM	3	-	DCB address
	8	-	KONBOX base
	10	-	base
	12	-	TUB base
IHKBSH	3	-	DCB address
	5	-	index to next file
	8	-	KONBOX base
	10	-	TUB base
	11	-	AVT base
	12	-	base
IHKBST	8	-	KONBOX base
	9	-	AVT base
	10	-	base
	12	-	TUB base
IHKCCI	4	-	return ECB address
	5	-	code of routine to be loaded or deleted
	8	-	communications ECB address
	9	-	CIB address
	10	-	base
	11	-	CCT base
	12	-	AVT base
IHKCCS	12	-	base
IHKCC1	4	-	0 return code from IHKAFI
	5	-	4 return code from IHKAFI
	6	-	UVR base
	7	-	internal linkage
	8	-	CIB address
	9	-	dummy TUB base
	10	-	internal linkage
	11	-	AVT base
	12	-	base
IHKCC2	5	-	CLB base
	9	-	dummy TUB address
	10	-	CIB address
	11	-	AVT base
	12	-	base
IHKCC3	3	-	CIB address
	4	-	0 return code from IHKAFI, CIB data pointer
	5	-	4 return code from IHKAFI
	7	-	internal linkage, CCT base
	8	-	TUB base
	11	-	base
	12	-	AVT base
IHKCC4	2	-	AVT base
	3	-	CIB address
	4	-	pointer to data in CIB
	5	-	length of data in CIB
	6	-	CCT base
	8	-	TUB base
	9	-	internal linkage
	11	-	base
IHKCC5	2	-	internal linkage
	4	-	operand counter
	8	-	RJCT base
	9	-	CIB address
	10	-	AVT base
	11	-	base
	12	-	pointer to dummy TUB
IHKCC6	2	-	UCM base
	6	-	UVR base
	7	-	internal linkage
	8	-	CIB base
	9	-	TUB base
	10	-	CCT base
	11	-	AVT base
	12	-	base
IHKCC7	6	-	UVR base
	7	-	internal linkage
	8	-	CIB base
	9	-	TUB base
	10	-	CCT base
	11	-	AVT base
	12	-	base

IHKCC8	5	-	last character in CIB
	7	-	internal linkage
	8	-	CIB base
	9	-	TUB base
	11	-	AVT base
	12	-	base
IHKCDP	4	-	internal linkage
	8	-	KONBOX base
	9	-	AVT base
	10	-	base
	12	-	TUB base
IHKCGN	2	-	AVT base
	3	-	TUB base
	4	-	PPT base
	6	-	user buffer address
	8	-	pointer to work area
	10	-	line retrieved by AFIO
	11	-	base
IHKCIP	11	-	AVT base
	12	-	base
IHKCLN	3	-	DCB address
	4	-	DECB base
	8	-	CLB base
	9	-	CCT base
	11	-	AVT base
	12	-	base
IHKCMD	4	-	CLB or TUB base
	10	-	base
	11	-	PPT base
	12	-	AVT base
IHKDEQ	3	-	SYSOUT class for CRJE
	7	-	CCT base
	8	-	AVT base
	9	-	ECB base
	10	-	QMPA address
	11	-	RJCT base
	12	-	base
IHKDSP	5	-	base
	9	-	current priority (0-top, 4-medium, 8-low)
	12	-	address of current STCB
IHKEDT	2	-	TUB base
	3	-	PPT base
	8	-	user buffer base
	9	-	address of current operand
	11	-	base
	12	-	AVT base
IHKED1	2	-	Tub base
	4	-	address of directory entry
	10	-	KONBOX base
	11	-	base
	12	-	AVT base
IHKEND	9	-	PPT base
	10	-	AVT base
	11	-	base
	12	-	TUB base
IHKEOS	2	-	TUB base
	11	-	base
	12	-	AVT base
IHKERR	4	-	TUB or CLB base
	8	-	CCT base
	10	-	base
	12	-	AVT base
IHKGET	2	-	address of DSLIST name buffer
	3	-	PPT base
	5	-	length of dsname and base for directory entry
	6	-	address of dsname
	8	-	KONBOX base
	10	-	TUB base
	11	-	AVT base
	12	-	base
IHKINI	10	-	DCB address
	12	-	base
IHKIPT	4	-	address of user buffer
	6	-	operand counter
	7	-	pointer to operands
	8	-	directory entry address
	9	-	PPT base
	10	-	AVT base
	11	-	base
	12	-	TUB base
IHKIRL	3	-	PPT base
	5	-	TUB base
	8	-	AVT base
	11	-	base
IHKLAB	12	-	base
IHKLAD	2	-	TUB base
	3	-	CLB base
	4	-	DECB base
	5	-	buffer address
	7	-	AVT base
	12	-	base
IHKLAP	2	-	TUB base
	3	-	CLB base
	12	-	base
IHKLAT	2	-	TUB base
	3	-	pointer to work area
	4	-	pointer to work area
	5	-	line buffer address
	6	-	tab buffer address
	7	-	tab pointer
	11	-	base
	12	-	length of unedited text
IHKLAY	2	-	CLB base
	3	-	DECB base
	4	-	address of input area

	10	-	internal linkage
	11	-	AVT base
	12	-	base
IHKLDC	1	-	module table entry
	2	-	ECB in CLB
	3	-	return ECB address
	7	-	ECB list entry getting first service
	8	-	current ECB list entry
	9	-	ECB list
	10	-	base
	12	-	block table entry
IHKLDS	1	-	address of parameter list
	4	-	KONBOX base
	6	-	directory entry displacement
	7	-	operand counter
	9	-	PPT base
	10	-	AVT base
	11	-	TUB base
	12	-	base
IHKLEW	0	-	number of tab settings
	2	-	address of user buffer
	7	-	number of characters in string
	10	-	internal linkage
	11	-	pointer to tab settings
	14	-	number of blanks in string
	15	-	base register
IHKLGF	8	-	user buffer address
	9	-	PPT base
	10	-	AVT base
	11	-	base
	12	-	TUB base
IHKLGN	6	-	operand counter
	7	-	pointer to operands
	8	-	user buffer address
	9	-	PPT base and CVT base
	10	-	AVT base
	11	-	base
	12	-	TUB base
IHKLST	-		TUB base
	3	-	DEF base
	4	-	PPT base
	6	-	parameter counter
	7	-	user buffer address
	8	-	AFIO buffer address
	10	-	AVT base
	12	-	base
IHKMAA	2	-	AVT base
	3	-	TUB base
	5	-	PPT base
	11	-	base
IHKMGE	2	-	TUB base
	3	-	PPT base
	4	-	AVT BASE
	11	-	BASE
IHKMSG	2	-	internal linkage
	6	-	TUB base of request
	7	-	CCT base
	9	-	address of user buffer
	10	-	AVT base
	11	-	address of recipient TUB or address of recipient userid or address of console ID followed by 3 bytes of zeros
IHKMUF	2	-	TUB base
	3	-	AVT base
	4	-	PPT base
	11	-	base
IHKNUM	4	-	pointer to length byte
	7	-	pointer to data characters
	11	-	base
IHKOPN	2	-	TUB base
	3	-	save and work area base
	9	-	DCB base
	11	-	base
	12	-	AVT base
IHKOUT	5	-	address of length of operand and operand
	7	-	PPT base
	9	-	RJCT base
	10	-	TUB base
	11	-	AVT base
	12	-	base
IHKPUT	9	-	QMPA address
	10	-	RJCT base
	11	-	TUB base
	12	-	base
IHKRER	6	-	JFCB address
	7	-	EXLST address
	8	-	DCB address
	9	-	DSCB address
	12	-	base
IHKRER02	3	-	QMPA address
	7	-	UCB address
	8	-	JFCB address
	9	-	QMPA extension address
	10	-	DSB address
	11	-	scratch list address
	12	-	base
IHKRER03	7	-	PPT base
	8	-	RJCT base
	10	-	TUB base
	11	-	AVT base
IHKRNQ	1	-	parameter list address
	3	-	queue element address
	4	-	queue control element address
	10	-	base

IHKRNR	2	-	AVT base
	3	-	TUB base
	6	-	user buffer address
	10	-	AFIO buffer address
	11	-	base
IHKSAV	4	-	KONBOX base
	5	-	directory entry base
	7	-	block count
	9	-	PPT base
	10	-	AVT base
	11	-	TUB base
	12	-	base
IHKSCN	2	-	TUB base
	3	-	address of current operand being processed
	4	-	KONBOX base
	5	-	count of operands
	8	-	base for switches and user buffer
	9	-	CCT base
	10	-	internal linkage
	11	-	AVT base
	12	-	base
IHKSDQ	7	-	CCT base
	8	-	AVT base
	12	-	base
IHKSND	3	-	TUB base
	4	-	PPT base
	6	-	user buffer address
	8	-	operand length
	10	-	AVT base
	11	-	base
IHKSRV	2	-	address of ECB in CLB
	3	-	addres of return ECB
	5	-	address of timer ECB in CCT
	7	-	ECB list entry getting first service
	8	-	current ECB list entry
	9	-	second entry in ECB list
	10	-	base
	11	-	AVT base
	12	-	ECB list
IHKSTP	3	-	DECB base
	5	-	CLB base
	9	-	CCT base
	11	-	AVT base
	12	-	base
IHKSTS	6	-	user buffer address
	8	-	RJCT buffer address
	9	-	PPT base
	10	-	AVT base
	11	-	base
	12	-	TUB base
IHKSUB	3	-	AVT base
	5	-	DCB address
	7	-	RJCT base
	10	-	CLB base
	11	-	TUB base
	12	-	base
IHKSYN	2	-	maximum number of continuation lines
	5	-	AVT base
	7	-	address of line number of first line to be scanned
	8	-	address of line number of last line to be scanned
	9	-	CCT base
	10	-	address of current line
	11	-	TUB base
	12	-	base
IHKTAB	2	-	user buffer address
	3	-	TUB base
	4	-	PPT base
	5	-	PPT length code pointer
	7	-	PPT operand pointer
	10	-	operand counter
	11	-	base
IHKUTM	7	-	UVR base
	8	-	request code address
	9	-	KONBOX base
	10	-	AVT base
	11	-	base
	12	-	TUB base
IHKWTR	-		DCB address
	8	-	KONBOX base
	9	-	TUB base
	11	-	AVT base
	12	-	base
IGC1503D	2	-	address of extended save area
	3	-	CSCB address
	10	-	base

APPENDIX A: CRJE COMMAND CODES

CODE	COMMAND and/or FUNCTION	ENTRY POINT
X'81'	LOGON	IHKLGN
X'82'	LOGOFF	IHKLGF
X'83'	END	IHKEND
X'84'	ERROR ANALYSIS and RECOVERY	IHKERR
X'85'	ALLOCATION (SUBMIT)	IHKALC
X'86'	PROVIDE INPUT (SUBMIT)	IHKGET
X'87'	CONDENSE USER LIBRARY	IHKCDP
X'88'	SUBMIT	IHKSUB
X'89'	CANCEL	IHKRER03
X'8A'	OPEN SYSOUT DATA SETS	IHKRER
X'8B'	DELETE A JOB	IHKRER01
X'8C'	SCRATCH SYSOUT DATA SETS	IHKRER02
X'8D'	OUTPUT	IHKOUT
X'8E'	CONTINUE	IHKOUT
X'8F'	TABSET	IHKTAB
X'90'	STATUS	IHKSTS
X'91'	EXEC	IHKEDT
X'92'	DELETE (MAJOR)	IHKEDT
X'93'	EDIT	IHKEDT
X'94'	EDIT and DELETE (MAJOR)	IHKED1
X'95'	EXEC	IHKED1
X'96'	EDIT OS/360 DATA SET	IHKEOS
X'97'	SEND	IHKSND
X'98'	INPUT	IHKIPT
X'99'	CHANGE	IHKCGN
X'9A'	MERGE	IHKMGE
X'9B'	MERGE ACTIVE AREA LINES	IHKMAA
X'9C'	MERGE FROM USER LIBRARY	IHKMUF
X'9D'	RENUMBER	IHKRNR
X'9E'	SAVE	IHKSAV
X'9F'	SHOW USERS and SHOW JOBS	IHKCC1
X'A0'	SHOW LERB, SHOW BRDCST, and MODIFY	IHKCC2
X'A1'	BRDCST	IHKCC3
X'A2'	MSG D=userid and SHOW MSGS	IHKCC4
X'A3'	CENOUT	IHKCC5
X'A4'	SHUTDOWN	IHKSTP
X'A5'	JOB TERMINATION	IHKDEQ
X'A6'	LISTDS	IHKLDS
X'A7'	LISTLIB	IHKLDS
X'A8'	OS/360 QUEUE MANAGER	IEFLOCDQ
X'A9'	SHOW SESS and SHOW SESSREL	IHKCC6
X'AA'	TRANSMIT OUTPUT	IHKPUT
X'AB'	USERID	IHKCC7
X'AC'	MSG and SHOW ACTIVE	IHKCC8
X'AD'	OPEN OS DATA SET (EDIT)	IHKOPN
X'01'	LIST	IHKLST
X'02'	LISTLIB and LISTDS	IHKLDS
X'03'	SCAN	IHKSCN
X'04'	INSERT/REPLACE	IHKIRL
X'05'	IMPLICIT SUBCOMMAND	IHKIRL01
X'06'	CORRECTION MODE	IHKIRL02
X'07'	DELETE (SUBCOMMAND)	IHKIRL
X'08'	LISTBC	IHKCMD

MESSAGE NUMBER	MESSAGE TEXT	SOURCE OF MESSAGE
IHK300	OUT OF MAIN STORAGE	IHKCMD, IHKSUB, IHKPUT
IHK301	LOGONS SUPPRESSED	IHKLGF, IHKCMD
IHK302	INVALID OPERAND [operand[line number]]	IHKRNR, IHKCGN, IHKTAB, IHKEDT, IHKSCN, IHKSTS, IHKLGN, IHKIPT, IHKCMD, IHKLST, IHKLDS, IHKSAV, IHKSUB, IHKOUT, IHKIRL, IHKMGE, IHKSND
IHK303	LOGON REJECTED BY INSTALLATION	IHKLGN
IHK304	VOLUME NOT MOUNTED volume serial	IHKEOS
IHK305	INVALID COMMAND command	IHKCMD
IHK306	I/O ERROR ON LIB DURING ACTIVE AREA RECOVERY	IHKAWS
IHK307	INVALID LINE NUMBER line number	IHKIRL, IHKMGE, IHKRNR, IHKCGN
IHK308	LOGON REQUIRED	IHKCMD
IHK309	INVALID CLIST COMMAND command	IHKCMD
IHK310	INVALID NESTING OF DSLIST DATA SETS dsname [userid] [key] lineum	IHKSUB
IHK311	QUOTES NOT PAIRED	IHKCMD
IHK312	NO BROADCAST MSGS	IHKMSG
IHK313	PARENTHESES NOT BALANCED [DSNAME[USERID][KEY] LINEUM]	IHKCMD, IHKSUB
IHK315	ILLEGAL DELIMITER delimiter	IHKCMD
IHK316	LINE TRUNCATED, LENGTH EXCEEDS [72 / 80] line number	IHKIRL, IHKCGN
IHK317	REQ'D OPERAND MISSING { DSNAME / JOBNAME / TEXT / LINE NUMBER }	IHKMSG, IHKSND, IHKCGN, IHKEDT, IHKLDS, IHKRER03, IHKMGE, IHKOUT
IHK318	CONDENSE ERROR, DATA SETS MAY BE LOST OR DAMAGED	IHKSAV
IHK319	TEXT [1 / 2] EXCEEDS 40 CHARACTERS	IHKSND, IHKCMD

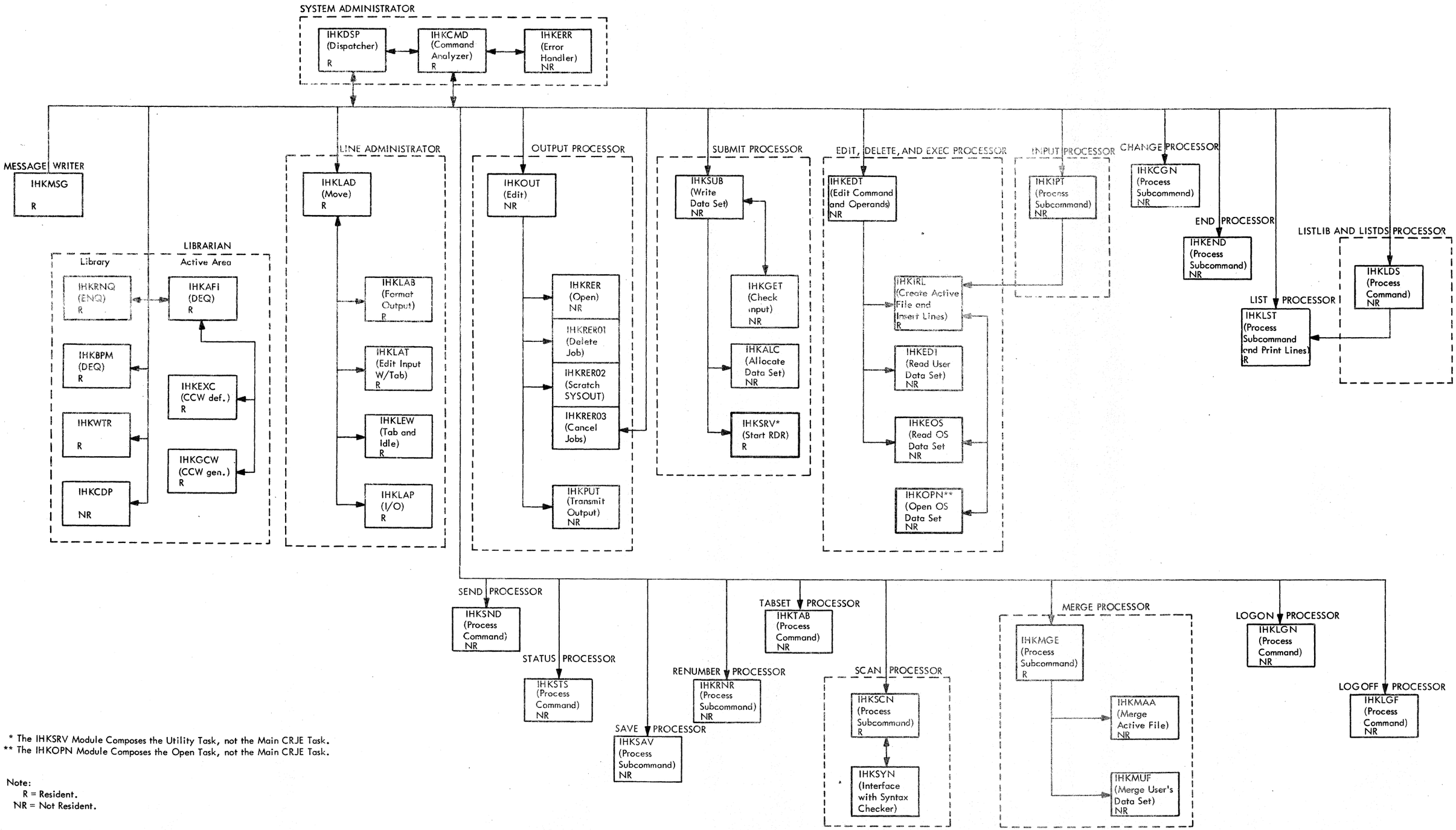
IHK320	DSNAME TYPE LNO [BLOCKS KEY] [CREATED LAST CHG XREAD]	IHKLDS
IHK321	MULTIPLE OR CONTRADICTORY USE OF OPERAND operand	IHKSND, IHKEDT, IHKLGN, IHKTAB, IHKIPT, IHKLDS
IHK322	ABNORMAL DRJE CLOSEDOWN	IHKERR
IHK323	INVALID JOB REFERENCE-JOB NOT SUBMITTED BY USER	IHKSTS, IHKRER03, IHKOUT
IHK324	JOB NOT COMPLETE	IHKOUT
IHK325	NO JOB(S) IN SYSTEM	IHKSTS, IHKRER03
IHK326	DUPLICATE JOBNAME jobname	IHKSUB
IHK327	***CRJE LINE nnn DATE yy.ddd TIME xx:xx***	IHKLGN
IHK328	MAX JOBS EXCEEDED jobname	IHKSUB
IHK329	JOB TERMINATED {NORMALLY / ABNORMALLY} jobname	IHKDEQ
IHK330	CRJE CLOSED DOWN	IHKERR
IHK331	JOB CANCELLED [jobname]	IHKRER03, IHKDEQ
IHK332	DISK ERROR {dsname / ddname}	IHKMUF, IHKSUB, IHKEOS, IHKPUT
IHK333	INVALID USERID	IHKMUF, IHKSND, IHKED1
IHK334	EXCESSIVE OPERANDS [{operand / dsname [userid][key]}]	IHKIRL, IHKMGE IHKSND, IHKRNR, IHKCGN, IHKTAB, IHKEDT, IHKSCN, IHKSTS, IHKLGF, IHKLGN, IHKEND, IHKIPT, IHKCMD, IHKLST, IHKLDS, IHKSAVE, IHKSUB, IHKOUT
IHK335	MSG SAVED FOR LOGON	IHKSND
IHK336	INVALID DSNAME dsname [userid][key][linenum]	IHKMSGE, IHKSUB
IHK337	MSG NOT SAVED	IHKSND
IHK338	DATA SET NOT FOUND [dsname][userid][key]linenum	IHKMUF, IHKED1, IHKLDS, IHKEOS, IHKSUB
IHK339	DATA SET NOT PDS OR SEQUENTIAL	IHKEOS
IHK340	NO LINES IN RANGE	IHKIRL, IHKMUF, IHKMAA, IHKCGN, IHKLST
IHK341	TEXT NOT FOUND	IHKCGN

IHK343	INVALID INCREMENT-0	IHKIPT
IHK344	JOB NOT FOUND jobname	IHKSTS, IHKOUT
IHK345	ORIGINAL NO. OF BLOCKS IN CLIST EXCEEDED	IHKED1
IHK346	LINE EXISTS, CANNOT INSERT line number	IHKIPT
IHK347	UNABLE TO OPEN dsname	IHKEOS
IHK348	MAX LINE NUMBER EXCEEDED	IHKIRL, IHKRNR
IHK349	LINE NOT FOUND line number	IHKCGN, IHKIPT, IHKLST
IHK350	ENTER USERID	IHKLGN
IHK351	ENTER PASSWORD ■■■■■■■■	IHKLGN
IHK352	INVALID USERID OR USERID IN USE, LOGON TERMINATED	IHKLGN
IHK353	INVALID PASSWORD, LOGON TERMINATED	IHKLGN
IHK354	LOGOFF time SESSION TIME xxxx MIN.	IHKLGF
IHK355	USERID IN USE, ENTER USERID	IHKLGN
IHK356	INVALID SOURCE MARGINS	IHKEDT
IHK357	ACTIVE AREA OUT OF SPACE	IHKIRL, IHKMGE, IHKMAA, IHKLDS, IHKMUF, IHKEOS
IHK358	MESSAGE LOST	IHKMSG
IHK359	INSERT TERMINATED. NEXT LINE NUMBER IS line number	IHKIRL
IHK360	INVALID SCAN REQUEST, DATA SET NOT PL/1 OR FORTRAN	IHKIRL, IHKED1, IHKSCN, IHKEOS
IHK361	END OF DATA	IHKIRL, IHKPUT
IHK362	TP LINE ERROR	IHKERR
IHK363	USER INACTIVE, MSG NOT SAVED	IHKSND
IHK364	LINE line number NOT IN RANGE OF LINES SCANNED	IHKIRL, IHKCGN
IHK365	INVALID JOBNAME jobname	IHKRER03
IHK366	DSNAME MUST BEGIN WITH USERID	IHKEDT
IHK367	INVALID LINE NUMBER RANGE	IHKIRL, IHKMGE, IHKCGN, IHKSCN, IHKLST
IHK368	TOO MANY CONTINUATION LINES. SCAN line number line number	IHKSYN, IHKIRL
IHK369	INVALID KEY {key / dsname [userid] [key] [linenum]}	IHKMUF, IHKED1, IHKMGE, IHKSAV

IHK370	LIBRARY INOPERATIVE	IHKMUF, IHKED1, IHKLGN, IHKSAV, IHKLDS
IHK371	INVALID EXEC, DATA SET IS NOT CLIST	IHKED1
IHK372	EXTRA COMMAS IGNORED	IHKCMD, IHKSUB
IHK373	LINE TOO LONG, REENTER LINE	IHKLAD
IHK374	NO DISCONTINUED OUTPUT AVAILABLE	IHKOUT
IHK375	NO DD STATEMENT IN PROCEDURE OR LIBRARY UNAVAILABLE [{ddname / dsname}] [userid] [key] [linenum]	IHKMUF, IHKED1, IHKSUB, IHKSAV, IHKLDS
IHK376	SUBMIT CANCELLED BY INSTALLATION xxxxxxxx	IHKSUB
IHK378	NO JOBS SUBMITTED	IHKSUB
IHK379	INVALID CORRECTION COMMAND command	IHKCMD
IHK380	EXEC TERMINATED	IHKCMD
IHK381	TABS EXCEED ALLOWED LIMIT	IHKTAB
IHK382	TABS NOT IN ASCENDING ORDER	IHKTAB
IHK383	ROUTING CODE NOT 1-16	IHKSND
IHK384	DATA SET RESIDES ON MULTIPLE VOLUMES	IHKEOS
IHK385	STATUS jobname { SCHED (n) / EXECUTING / NORMAL END / ABNORM END / NOT QUEUED / DISK ERROR }	IHKSTS
IHK386	LEVEL OF SYNTAX CHECKER NOT AVAILABLE	IHKSYN
IHK387	SYNTAX CHECKER NOT IN SYSTEM	IHKIRL, IHKSCN
IHK388	TIMEOUT	IHKLAD
IHK389	SESSION TERMINATED-ACTIVE AREA SAVED-DSNAME=ACTIVE	IHKAWS, IHKSAV
IHK390	OUT OF SPACE ON LIBRARY DURING ACTIVE AREA RECOVERY	IHKAWS
IHK391	INVALID EDIT SUBCOMMAND command	IHKCMD
IHK392	INVALID SAVE, NO LINES IN DATA SET	IHKSAV
IHK393	EDIT OF DATA SET REQ'D dsname	IHKMUF, IHKED1
IHK394	DUPLICATE DSNAME, ENTER DSNAME	IHKSAV
IHK395	LIBRARY FULL, ENTER DSNAME FOR DELETION	IHKSAV

IHK396	ENTER DSNAME FOR DELETION	IHKSAV
IHK397	DSNAME INVALID/MISSING, ENTER DSNAME	IHKSAV
IHK398	DATA SET NOT ON 2311 OR 2314 {ddname / dsname}	IHKSUB
IHK399	OPERAND NOT IN PARENTHESES operand	IHKSND
IHK400	TP LINE DEACTIVATED BY OPERATOR	IHKERR
IHK401	NO MEMBER NAME FOR PDS	IHKEOS
IHK402	LINE NUMBER NOT XXXXXXXX OR NUMERICS line number	IHKED1
IHK403	SESSION TERMINATED-ACTIVE AREA LOST	IHKSAV
IHK404	EXPIRATION DATE NOT REACHED-DATA SET NOT DELETED	IHKEOS
IHK405	LIBRARY EMPTY	IHKLDS
IHK406	TP LINE ERROR, REENTER LINE	IHKLAD
IHK407	OPERATOR FAILED TO SPECIFY CORRECT PASSWORD	IHKEOS
IHK408	INVALID EDIT REQUEST	IHKEDT
IHK409	DIRECTORY FULL, ENTER DSNAME FOR DELETION	IHKSAV
IHK410	DATA LOST IN TRUNCATION	IHKAWS, IHKSAV
IHK413	DIRECTORY FULL	IHKED1

APPENDIX C: COMPONENT BREAKDOWN OF MODULES



* The IHKSRV Module Composes the Utility Task, not the Main CRJE Task.
** The IHKOPN Module Composes the Open Task, not the Main CRJE Task.

Note:
R = Resident.
NR = Not Resident.

COMMANDS

1. CANCEL jobname
2. CONTINUE [H[ERE] / B[EGIN] / N[EXT]]
3. DELETE dsname
4. EDIT dsname [NEW / OLD] [NUM / NONUM] [S[CAN] / NOS[CAN]] [PL1[(parameters)] / FORT {E / G / H} / DSLIST / VLIST / DATA / TEXT]
5. EXEC dsname [L[IST] / NO[LIST]]
6. LISTBC
7. LISTDS dsname [S[TATUS]][H[ISTORY]]
8. LISTLIB [S[TATUS]][H[ISTORY]]
9. LOGOFF
10. LOGON userid/password[ACCT](accounting information)
    [BC / NOBC] [MISGID / NOM[SGID]]
11. OUTPUT jobname [SMSG]
12. SEND 'text' [U[SER] 'userid) [N[OW] / L[OGON]] / O[PERATOR] (integer)]
13. STATUS [jobname]
14. SUBMIT dsname...
15. TABSET [num... / OFF] [IN[PUT] / OUT[PUT]]

EDIT SUBCOMMANDS

1. linenum [Δtext]
2. CA[NCEL] jobname
3. C[HANGE] linenum [linenum] text 1 text2 [A[LL]]

4. D[ELETE] [linenum [linenum]]

5. END

6. I[NPUT] [linenum [[INCREMENT] [I] / R]] [P[ROMPT] / NOP[ROMPT]]

7. L[IST] [linenum [linenum]] [NUM / NONUM]

8. M[ERGE] {dsname / *} [linenum linenum] [linenum]

9. REN[UMBER] [linenum 10 [increment 10]]

10. S[AVE] [dsname][K[EY] (key)]

11. SC[AN] [linenum [linenum]] [ON / OFF]

12. SEND 'text' [U[SER] (userid) [N[OW] / L[OGON]] / O[PERATOR] (integer)]

13. SUB[MIT] {dsname / *} ...

14. TAB[SET] [num... / OFF] [IN[PUT] / OUT[PUT]]

APPENDIX E: CENTRAL OPERATOR MESSAGES

MESSAGE CODE	MESSAGE TEXT	DESCRIPTOR CODE	ROUTING CODE
IEE301I	jjj CANCEL COMMAND ACCEPTED	ICR	RC
IEE30I1	{cm / NO CORE / blanks / CSCB USE} COMMAND INVALID	ICR	RC
IEE326I	{CRJE / RJE/CRJE} NOT SUPPORTED	ICR	RC
IEE341I	ttt NOT ACTIVE	ICR	RC
IHK200I	LOGOFF userid	SS	RC
IHK201I	ACTIVE CRJE USER userid lineaddress time	ICR	RC
IHK202I	USERID INVALID/NOT FOUND {SHOW / MSG / USERID} userid	ICR	RC
IHK203I	START OF CRJE MESSAGES	ICR	RC
IHK204I	USERID PREVIOUSLY ASSIGNED userid	ICR	RC
IHK205I	INVALID PASSWORD password	ICR	RC
IHK206I	TEXT MUST BE 1 TO 40 CHARACTERS LONG {BRDCST / MSG} chars	ICR	RC
IHK207I	DELETED FROM USER LIST--userid	ICR	RC
IHK208I	ADDED TO USER LIST--userid	ICR	RC
IHK209I	MSG NOT SAVED BRDCST chars	ICR	RC
IHK210I	LINEADDRESS INVALID/NOT FOUND lineaddress	ICR	RC
IHK211I	JOB NOT IN SYSTEM jobname	ICR	RC
IHK212I	LOGON userid	SS	RC
IHK213I	END OF CRJE MESSAGES	ICR	RC
IHK214I	CENOUT jobname	ICR	RC
IHK215I	START CRJE REJECTED	ICR	RC,COI
IHK216I	DISK ERROR {ACTIVE AREA / CRJE.SYSLIB} {START / (dsname)}	SS	DP,ER
IHK217I	DRJE BROADCAST MESSAGES START	ICR	RC
IHK218I	CRJE BROADCAST MESSAGES END	ICR	RC
IHK219I	OUT OF MAIN STORAGE [START]	SS	COI
IHK220I	NNN LINE NOT OPERATIONAL	SS	TC,ER

IHK221I	LOGONS SUPPRESSED	ICR	RC
IHK222I	ABNORMAL CRJE CLOSEDOWN	SS	COI
IHK223I	CRJE NOW ACTIVE	SS	COI
IHK224I	NO ACTIVE AREA FOR RESTART	SS	COI
IHK225I	CRJE BROADCAST MESSAGES(S)DELETED [nnnn]	|CR	RC
IHK226I	OPERAND MISSING/INVALID command operand	ICR	RC
IHK227I	NO JOBS IN SYSTEM	ICR	RC
IHK228I	UNABLE TO OPEN {ACTIVE / ddname [RLN=xxxx]}	SS	ER,TC,COI
IHK229I	LOGONS RESUMED	ICR	RC
IHK230I	CRJE CLOSED DOWN	SS	COI
IHK231I	JOB NOT COMPLETE jobname [userid]	ICR	RC
IHK232I	NO CRJE BROADCAST MESSAGES	ICR	RC
IHK233I	NO CRJE DELAYED MESSAGES [userid]	ICR	RC
IHK234I	ACTIVE AREA NOT ON 2311 or 2314	SS	COI
IHK235I	DD CARD NOT IN PROCEDURE {dsname / ddname}	SS	COI
IHK236I	MAXIMUM NO. OF CRJE USER MESSAGES REACHED	SS	COI
IHK237I	MAXIMUM BRDCST IDENTIFIER EXCEEDED--LAST. =nnnn	ICR	RC
IHK238I	SPECIFIC SHOW SESS MAXIMUM EXCEEDED --userid	ICR	RC
IHK239I	SHOW SESS NOT IN EFFECT [userid]	ICR	RC
IHK240I	ACTIVE AREA OUT OF SPACE	SS	COI
IHK241I	LIBRARY I/O ERROR ddname CRJE.LIB.userid	SS	ER,DP
IHK242I	ACTIVE AREA I/O Error--ABNORMAL CRJE CLOSEDOWN	SS	ER,DP,COI
IHK243I	LINE BEING ACTIVATED lineaddress	ICR	RC
IHK244I	LINE DEACTIVATED lineaddress	ICR	RC
IHK245I	{nnn / NO} ACTIVE CRJE USERS	ICR	RC
IHK246I	QUEUE MANAGER DISK ERROR [jobname]	SS	ER,DP,COI
IHK247I	NO CRJE USERS	ICR	RC
IHK248I	INACTIVE CRJE USER userid	ICR	RC
IHK249I	SHOW SESS IN EFFECT [userid]	ICR	RC
IHK250I	SHOW SESS RELEASED [userid]	ICR	RC

IHK251I	LINE NOT ACTIVE lineaddress	ICR	RC
IHK252I	USER LIST FULL [userid]	ICR	RC
IHK253I	JOB WAITING DELIVERY jobname	ICR	RC
IHK254I	ILLEGAL DELIMITER command operand	ICR	RC
IHK255I	MESSAGES DELETED FOR userid	ICR	RC
IHK256I	MSG QUEUED FOR DELIVERY userid	ICR	RC
IHK257I	JOB COMPLETE jobname userid	ICR	RC
IHK258I	MODIFY BEING PROCESSED lineaddress	ICR	RC
IHK259I	INVALID BRDCST IDENTIFIER operand	ICR	RC
IHK260I	NO SPACE AVAILABLE FOR SYNTAX CHECKER WORK AREA	SS	COI
IHK261I	ACTIVE AREA CONTAINS MULTIPLE EXTENTS	00	COI
IHK263I	REQUIRED PARAMETER MISSING {START / EXEC}	ICR	RC, COI
IHK264I	LINE NOT OPEN lineaddress	ICR	RC
IHK265I	LINE ALREADY ACTIVE lineaddress	ICR	RC
IHK267I	MAXIMUM NO. OF CRJE BROADCAST MESSAGES REACHED	SS	COI
IHK268I	CRJE BRDCST MESSAGE ADDED nnnn	ICR	RC
IHK269I	CRJE BRADCST MESSAGE REPLACED nnnn	ICR	RC
IHK270I	ACTIVE AREA I/O ERROR	SS	COI, ER, DP
IHK271I	MODIFY DEACTIVATE UNSUCCESSFUL lineaddress	ICR	RC
IHK272I	JOB DELETED jobname	ICR	RC
IHK273I	OUT OF SPACE CRJE.SYSLIB (dsname)	SS	DOI, ER, DP
IHK274I	CRJE SUBTASK ABEND xxx	SS	COI

Code Explanations

Descriptor Codes:

ICR - Immediate Command Response (Code 5) Immediate response to an operator command.

SS - System Status (Code 4) Indicates status of system or gives indication of uncorrectable I/O errors.

Routing Codes:

COI - Chief Operator Information (code X'4000') Indicates a change in system status and alerts chief operator to a condition that may require his attention.

DP - Direct Access Pool (code X'1000') Indicates status of direct access device or disposition of disk.

ER - System Maintenance Error (code X'0040') Indicates system errors or uncorrectable I/O errors (not program errors).

RC - Requesting Console (Code) Indicates information requested by a central command

TC - Teleprocessing Control (code X'0100') Indicates status of or disposition of teleprocessing equipment.

AFIO MACROS

```
CREATE   {IS}  [,ACTUM=value]  [,FTYPE= {GBL }]
         {O }                          {XGBL}

ENDUP    [FTYPE= {GBL }]
                 {XGBL}

INSERT   {S}  {,B}  [,FTYPE= {GBL }]
         {M}  {,A}           {XGBL}

RDELETE  {NXK}  {,B}  [,REC=PREVIOUS]  [,FTYPE= {GBL }]
         {XK }  {,A}                           {XGBL}

RELEASE  A  [,FTYPE= {GBL }]
                     {XGBL}

REPLACE  {NXK}  [,B]  [,REC=PREVIOUS]  [,FTYPE= {GBL }]
         {XK }  [,A]                           {XGBL}

RGET     {R}  {,S}[,B]  [,REC=PREVIOUS]  [,FTYPE= {GBL }]
         {XK} {,M}[,A]                            {XGBL}

RPOINT   {B}  {,R }  [,KEY= {FIRST}]  [,FTYPE= {GBL }]
         {A}  {,NR}         {LAST }           {XGBL}
                            {HIEQ }

RSKIP    {NXK}  {,B}  [,REC=PREVIOUS]  [,FTYPE= {GBL }]
         {XK }  {,A}                           {XGBL}
```

LIBRARY I/O MACROS

```
RCLOSE   [STOW  ]
         [NOSTOW]

RFIND    [O]  [,S ]
         [I]  [,NS]
         [F]

RREAD    none

RSCRATCH none

RWRITE   none
```

APPENDIX G: ORIGIN OF CENTRAL OPERATOR MESSAGES

MESSAGE CODE	MESSAGE TEXT	SOURCE OF MESSAGE
IEE301I	jjj CANCEL COMMAND ACCEPTED	
IEE305I	{ cm / NO CORE / blanks / CSCB USE } COMMAND INVALID	
IEE326I	{ CRJE / RJE/CRJE } NOT SUPPORTED	Master SCHEDULER Messages
IEE341I	ttt NOT ACTIVE	
IHK200I	LOGOFF userid	IHKLGF
IHK201I	ACTIVE CRJE USER userid lineaddress time	IHKCC1, IHKCC7, IHKCC8
IHK202I	USERID INVALID/NOT FOUND { SHOW / MSG / USERID } userid	IHKCC1, IHKCC4, IHKCC6, IHKCC7, IHKCC8
IHK203I	START OF CRJE MESSAGES	IHKCC4
IHK204I	USERID PREVIOUSLY ASSIGNED userid	IHKCC7
IHK205I	INVALID PASSWORD password	IHKCC7
IHK206I	TEXT MUST BE 1 TO 40 CHARACTERS LONG { BRDCST / MSG } chars	IHKCC3,IHKCC8
IHK207I	DELETED FROM USER LIST--userid	IHKCC7
IHK208I	ADDED TO USER LIST--userid	IHKCC7
IHK209I	MSG NOT SAVED BRDCST chars	IHKCC3
IHK210I	LINE ADDRESS INVALID/NOT FOUND lineaddress	IHKCC2
IHK211I	JOB NOT IN SYSTEM jobname	IHKCC1, IHKCC5
IHK212I	LOGON userid	IHKLGN
IHK213I	END OF CRJE MESSAGES	IHKCC4
IHK214I	CENOUT jobname	IHKCC5
IHK215I	START CRJE REJECTED	IHKBGN
IHK216I	DISK ERROR { ACTIVE AREA / CRJE SYSLIB } { START / (dsname) }	IHKBSH, IHKCIP, IHKINI
IHK217I	CRJE BROADCAST MESSAGES START	IHKMSG
IHK218I	CRJE BROADCAST MESSAGES END	IHKMSG

IHK219I	OUT OF MAIN STORAGE [START]	IHKCC1, IHKCC2, IHKCC3, IHKCC4, IHKCC5, IHKCC6, IHKCC7, IHKCC8, IHKCIP, IHKCMD, IHKERR
IHK220I	nnn LINE NOT OPERATIONAL	IHKERR
IHK221I	LOGONS SUPPRESSED	IHKCC7
IHK222I	ABNORMAL CRJE CLOSEDOWN	IHKCLN
IHK223I	CRJE NOW ACTIVE	IHKCIP
IHK225I	CRJE BROADCAST MESSAGE(S) DELETED [nnnn]	IHKCC3
IHK226I	OPERAND MISSING/INVALID command operand	IHKCC1, IHKCC2, IHKCC3, IHKCC4, IHKCC5, IHKCC6, IHKCC7, IHKCC8
IHK227I	NO JOBS IN SYSTEM	IHKCC1
IHK228I	UNABLE TO OPEN {ACTIVE / ddname [RLN=xxxx]}	IHKBSH, IHKCIP, IHKINI
IHK229I	LOGONS RESUMED	IHKCC7
IHK230I	CRJE CLOSED DOWN	IHKCLN
IHK231I	JOB NOT COMPLETE jobname [userid]	IHKCC1, IHKCC5
IHK232I	NO CRJE BROADCAST MESSAGES	IHKMSG
IHK233I	NO CRJE DELAYED MESSAGES [userid]	IHKCC4
IHK234I	ACTIVE AREA NOT ON 2311, 2314, or 2319	IHKCIP
IHK235I	DD CARD NOT IN PROCEDURE {dsname / ddname}	IHKSUB
IHK236I	MAXIMUM NO. OF CRJE USER MESSAGES REACHED	IHKCC8, IHKMSG
IHK237I	MAXIMUM BRDCST IDENTIFIER EXCEEDED--LAST=nnnn	IHKCC3
IHK238I	SPECIFIC SHOW SESS MAXIMUM EXCEEDED--userid	IHKCC6
IHK239I	SHOW SESS NOT IN EFFECT [userid]	IHKCC6
IHK240I	ACTIVE AREA OUT OF SPACE	IHKCC3, IHKCC7, IHKCC8, IHKCMD, IHKMSG
IHK241I	LIBRARY I/O ERROR ddname CRJE,LIB.userid	IHKED1, IHKLDS, IHKMUF, IHKSAV, IHKSUB
IHK242I	ACTIVE AREA I/O ERROR--ABNORMAL CRJE CLOSEDOWN	IHKERR
IHK243I	LINE BEING ACTIVATED lineaddress	IHKCC2
IHK244I	LINE DEACTIVATED lineaddress	IHKCC2, IHKERR
IHK245I	{nnn / NO} ACTIVE CRJE USERS	IHKCC8

IHK246I	QUEUE MANAGER DISK ERROR [jobname]	IHKCC5, IHKDEQ
IHK247I	NO CRJE USERS	IHKCC1
IHK248I	INACTIVE CRJE USER userid	IHKCC1, IHKCC8
IHK249I	SHOW SESS IN EFFECT [userid]	IHKCC6
IHK250I	SHOW SESS RELEASED [userid]	IHKCC6
IHK251I	LINE NOT ACTIVE lineaddress	IHKCC2
IHK252I	USER LIST FULL [userid]	IHKBST, IHKCC7
IHK253I	JOB WAITING DELIVERY jobname	IHKCC5
IHK254I	ILLEGAL DELIMITER command operand	IHKCC2, IHKCC3,
IHK255I	MESSAGES DELETED FOR userid	IHKCC4
IHK256I	MSG QUEUED FOR DELIVERY userid	IHKCC8
IHK257I	JOB COMPLETE jobname userid	IHKCC1
IHK258I	MODIFY BEING PROCESSD lineaddress	IHKCC2
IHK259I	INVALID BRDCST IDENTIFIER operand	IHKCC3
IHK260I	CRJE,aaa,bb,cccccccc,dddddd, eeeeeeeeeeeeeee,fffffffffffff, ggggggg,hhh	IHKAFI
IHK261I	ACTIVE AREA CONTAINS MULTIPLE EXTENTS	IHKCIP
IHK263I	REQUIRED PARAMETER MISSING {START / EXEC}	IHKCIP
IHK265I	LINE ALREADY ACTIVE lineaddress	IHKCC2
IHK267I	MAXIMUM NO. OF CRJE BROADCAST	IHKCC3
IHK268I	CRJE BRDCST MESSAGE ADDED nnnn	IHKCC3
IHK269I	CRJE BRDCST MESSAGE REPLACED nnnn	IHKCC3
IHK270I	ACTIVE AREA I/O ERROR	IHKCC1, IHKCC2, IHKCC3, IHKCC4, IHKCC5, IHKCC6, IHKCC7, IHKCC8
IHK272I	JOB DELETED jobname	IHKDEQ
IHK273I	OUT OF SPACE CRJE.SYSLIB (dsname)	IHKBSH
IHK274I	CRJE SUBTASK ABEND xxx	IHKCIP

Indexes to program logic manuals are consolidated in the publication IBM System/360 Operating System: Program Logic Manual Master Index, Y28-6717. For additional information about any subject listed below, refer to other publications listed for the same subject in the Master Index.

READER'S COMMENTS

TITLE: IBM System/360 Operating System
Conversational Remote Job Entry
Program Logic Manual

ORDER NO. GY30-2011-1

Your comments assist us in improving the usefulness of our publications; they are an important part of the input used in preparing updates to the publications. All comments and suggestions become the property of IBM.

Please do not use this form for technical questions about the system or for requests for additional publications; this only delays the response. Instead, direct your inquiries or requests to your IBM representative or to the IBM Branch Office serving your locality.

Corrections or clarifications needed:

Page	*Comment*

Please include your name and address in the space below if you wish a reply.

Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

GY30-2011-1

cut along this line

fold fold

FIRST CLASS
PERMIT NO. 33504
NEW YORK, N.Y.

BUSINESS REPLY MAIL

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM CORPORATION
1271 Avenue of the Americas
New York, New York 10020

Attention: PUBLICATIONS

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 S/360 OS CRJE PLM Printed in U.S.A. GY30-2011-1

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)