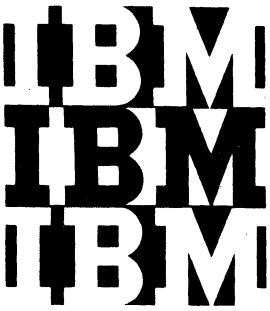


File No. S360-36
Form Y28-6613-1

Program Logic

IBM System/360 Operating System

Job Management

Program Number 360S-CI-505

This publication describes the internal logic within the job management portion of the IBM System/360 Operating System Primary Control Program. Job management prepares jobs for execution, and directs the disposition of data sets created during job execution. It also handles all communication between the operator and the primary control program. Included in the publication are descriptions of tables and work areas used by the job management routines and a directory of names and purposes of control sections, assembly modules, and load modules.

This publication is intended for use by persons involved in program maintenance, and system programmers who are altering the program design. Program logic information is not necessary for the use and operation of the program; therefore, distribution of this publication is limited to persons with program maintenance or modification responsibilities.

The information contained in this publication applies only to the primary control program.

Restricted Distribution

PREFACE

This publication describes the structure of the sequential scheduler level of job management, its functions, and the control flow between its major routines. It is divided into an introduction in which job management is briefly described and three major sections, master scheduler, reader/interpreter, and initiator/terminator, in which the corresponding components are described in greater detail. Included are three appendixes. Appendix A describes two subroutines used frequently by job management routines. Appendix B shows job management tables and work areas that are not described in the body of the publication. Appendix C lists job management load modules and the assembly modules that each contains. Further information on job management may be obtained from the program listings.

Readers should have a thorough understanding of IBM System/360 programming and should be familiar with the following publications:

IBM System/360 Operating System: Introduction, Form C28-6534

IBM System/360 Operating System: Concepts and Facilities, Form C28-6535

IBM System/360 Operating System: Operator's Guide, Form C28-6540

IBM System/360 Operating System: Job Control Language, Form C28-6539

IBM System/360 Operating System: Introduction to Control Program Logic, Program Logic Manual, Form Y28-6605

IBM System/360 Operating System: System Control Blocks, Form C28-6628

Second Edition (March 1967)

This publication is a major revision of Form Y28-6613-0 and obsoletes it. In addition to incorporating information released in Technical Newsletters Y28-2184 and Y28-2191, this edition also describes the changes made to job management in release 11 of the operating system. These changes are primarily concerned with automatic volume recognition (AVR). Vertical bars at the left of affected text and bullets beside illustration captions indicate changes and additions.

Specifications contained herein are subject to change from time to time. Any such change will be reported in subsequent revisions or technical newsletters.

This publication was prepared for production using an IBM computer to update the text and to control the page and line format. Page impressions for photo-offset printing were obtained from an IBM 1403 Printer using a special print chain.

Copies of this and other IBM publications can be obtained through IBM Branch Offices.

A form for readers' comments appears at the back of this publication. It may be mailed directly to IBM. Address any additional comments concerning this publication to the IBM Corporation, Programming Systems Publications, Department D58, PO Box 390, Poughkeepsie, N. Y. 12602

FIGURES

CHARTS

Job management (Chart 1) is the first and last portion of the control program that a job encounters. Its primary function is to prepare job steps for execution and, when they have been executed, to direct the disposition of data sets created during execution. Prior to step execution, job management:

- Reads control statements from the input job stream.
- Places information contained in the statements into a series of tables.
- Analyzes input/output (I/O) requirements.
- Assigns I/O devices.
- Passes control to the job step.

Following step execution, job management:

- Releases main storage space occupied by the tables.
- Frees I/O devices assigned to the step.
- Disposes of data sets referred to or created during execution.

Job management also performs all processing required for communication between the operator and the control program. Major components of job management are the job scheduler, which introduces each job step to System/360, and the master scheduler, which handles all operator-system-operator communication.

JOB SCHEDULER FUNCTIONS

The job scheduler includes two programs: the reader/interpreter and the initiator/terminator. The reader/interpreter is given control whenever a job step is to be obtained from the input job stream and processed. It directs the reading of control statements and from them constructs:

- A job control table (JCT) to describe the job.
- A step control table (SCT) to describe the job step.
- An account control table (ACT) to describe accounting information related to the job.
- Job file control blocks (JFCB) (one for each DD statement) to describe the data sets to be used by the job.
- Step input/output tables (SIOT) (one for each DD statement) to describe the I/O requirements of the job step.
- Volume tables (VOLT) (one for each step, with an entry for each DD statement) containing serial numbers of volumes to be used by the job step.
- Data set name (DSNAME) tables (one for each step, with an entry for each DD statement) containing names of previously defined data sets to be used by the job step.

In addition to the above, the reader/interpreter creates system message blocks, in which diagnostic messages to the programmer are stored before they are written onto the system output data set.

After all control statements for a job have been processed, or when data is encountered in the input job stream, the reader/interpreter gives control to the initiator/terminator. The latter analyzes the I/O requirements of the job step and, upon considering such factors as requests for specific units, volumes, and channels and their current employment, it assigns devices in such a way as to achieve maximum overlap of I/O activity during step execution.

When all devices requested for the step have been assigned, the initiator/terminator issues mounting messages (if any are required) and verifies for direct-access requests that the operator has mounted volumes on the correct units. Control is then passed to the job step. When the step has been executed, control is again given to the initiator/terminator, which performs data set dispositions and releases I/O resources.

MASTER SCHEDULER FUNCTIONS

The routines of the master scheduler process any communication between the operator and the system. The master scheduler processes:

- Operator commands, whether they are issued through the console or through the input job stream.
- Write-to-operator (WTO) and write-to-operator with reply (WTOR) macro-instructions.
- Interruptions caused when the INTERRUPT key is pressed.

JOB PROCESSING

Figure 1 shows the major components of job management and illustrates the general flow of control.

Control is passed to job management whenever the supervisor finds that there are no program request blocks in the request block queue. This can occur for two reasons: either the initial program loading (IPL) procedure has just been completed or a job step has just been executed.

ENTRY TO JOB MANAGEMENT FOLLOWING INITIAL PROGRAM LOADING

Following IPL, certain actions must be taken by the operator before job processing can begin. Therefore, control passes to the master scheduler, which issues a message to the operator instructing him to enter commands. These "initialization" commands include a SET command, a start writer (START WTR) command, and a start reader (START RDR) command. When a START command with a blank operand is issued, control is passed to the reader/interpreter.

ENTRY TO JOB MANAGEMENT FOLLOWING STEP EXECUTION

Following step execution, control is routed to the step termination routine of the initiator/terminator. If the job had been completed, control is also passed to the job termination routine of the initiator/terminator. Both routines are described under "Job and Step Termination."

CONTROL STATEMENT PROCESSING

After completion of the processing that immediately follows IPL, or after termination of a job or of a step containing data in the input job stream, control is passed to the reader/interpreter. The reader/interpreter reads and processes control statements until one of the following conditions is encountered:

- A DD * or DD DATA statement.
- Another JOB statement.
- A null statement.
- An end-of-data set (EOF) on the system input device.

Meanwhile, if the operator has pressed the REQUEST key and has entered a request (REQ) command during execution of the job step or any of the above processing, the master scheduler sets a command-pending indicator in the nucleus during the ensuing interruption. The indicator is now checked and, if found to be on, control is passed to the master scheduler, which issues a message instructing the operator to enter commands, and then processes the commands.

STEP INITIATION

Control next passes to the initiator/terminator, which examines I/O device requirements, assigns (allocates) I/O devices to the job step, issues mounting instructions, and verifies that direct-access volumes have been mounted on the correct units. Finally, the initiator/terminator passes control to the job step.

JOB AND STEP TERMINATION

When processing program execution is completed, the supervisor, finding no program request blocks in its request block queue, passes control to the job management routines. Entry is first made to the step termination routine.

The step termination routine performs end-of-step housekeeping and passes control to the user's accounting routine, if one was provided. When the accounting routine has been executed, the supervisor returns control to the step termination routine. Control is then passed to the job termination routine if there are no more steps in the job; to the reader/interpreter if the next step of this job has not been read yet (i.e., the step just terminated had data in the input stream); or to the step initiation routine if the next step of this job has been read.

The job termination routine performs end-of-job housekeeping. It exits to the user's accounting routine, if one was provided. After the accounting routine is executed, the supervisor returns control to the job termination routine, which passes control to the reader/interpreter.

OPERATOR-SYSTEM COMMUNICATION PROCESSING

The routines that handle operator-system communication are contained in the master scheduler. Communication may take one of two forms: commands, which allow the operator to change the status of the system or of a job or job step; and the WTO or WTOR macro-instructions, which allow processing programs or system components to issue messages to the operator. The master scheduler also switches functions from the primary console device to an alternate console device when the INTERRUPT key is depressed.

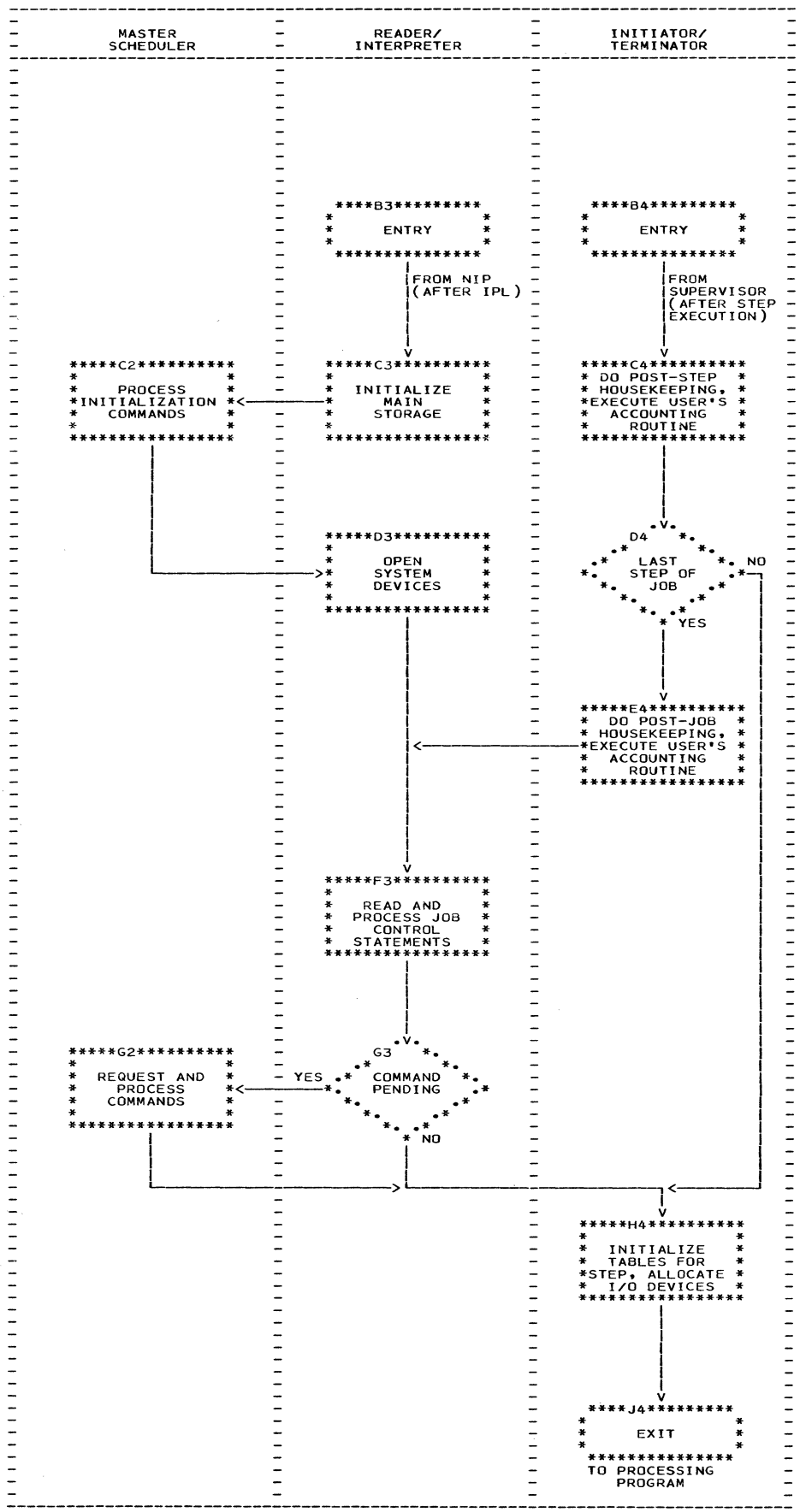


Figure 1. Job Management Control Flow

Command Processing

Commands may be issued by the operator in two ways: he may insert command statements between job steps in the input job stream, or he may issue commands through the console input device. Commands encountered in the input job stream cause control to be passed to the master scheduler, which processes them. Before entering commands through the console, however, the operator must press the REQUEST key to cause an attention interruption. Figure 2 shows the actions taken after the key is pressed.

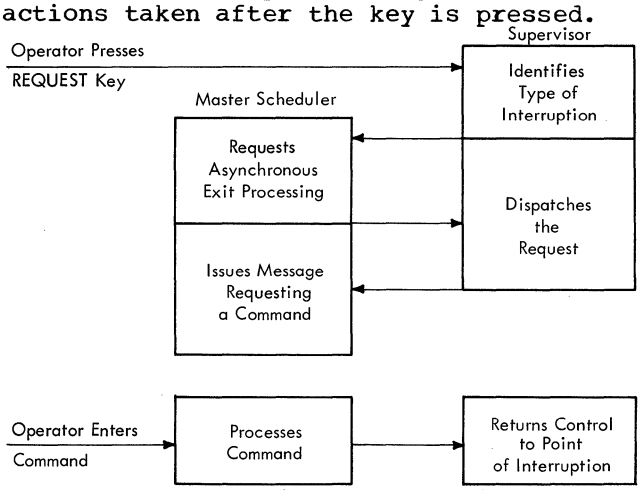


Figure 2. Attention Interruption Processing Flow

WTO/WTOR Macro-Instruction Processing

Whenever the WTO or WTOR macro-instruction is issued, an SVC interruption occurs. (See Figure 3.)

External Interruption Processing

When the operator presses the INTERRUPT key, an external interruption occurs, following which the master scheduler switches functions from the primary to the alternate console I/O device. (See Figure 4.)

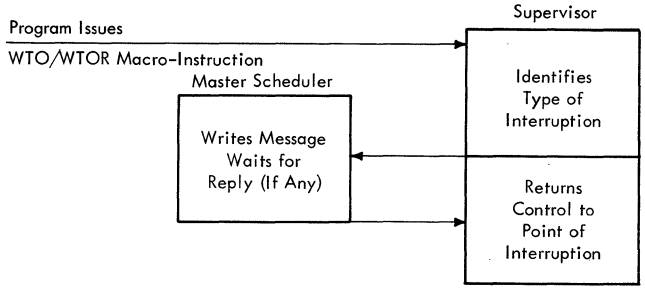


Figure 3. WTO/WTOR Macro-Instruction Processing Flow

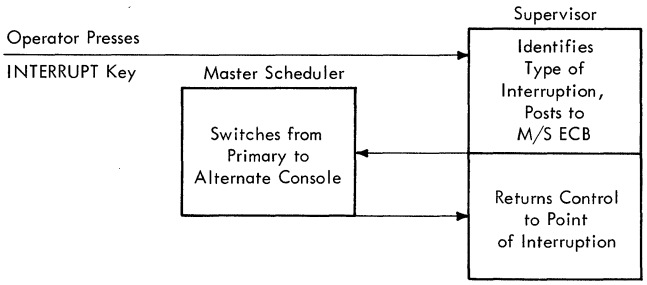


Figure 4. External Interruption Processing Flow

LOAD MODULES

Most job management routines exist as a series of load modules that reside in the link library (SYS1.LINKLIB). The exceptions are the interruption-handling routines of the master scheduler, which reside in the nucleus and the master command EXCP routine which is in the SVC library (SYS1.SVCLIB). Appendix C contains a list of the routines that make up each job management load module.

The master scheduler (Chart 2) processes all operator commands and messages directed to the operator through use of the WTO and WTOR macro-instructions. It also performs console switching when the secondary console is to be used in place of the primary console.

The five major routines of the master scheduler are:

- Console interrupt routine, which provides the supervisor with the information necessary to queue a request for processing an attention interruption.
- Master command EXCP routine, which reads commands from the console input device and processes all commands except SET, START RDR, and START WTR.
- Master command routine, which analyzes command verbs and routes control to appropriate command execution routines.
- Write-to-operator routine, which processes messages to the operator and all operator replies to these messages.
- External interrupt routine, which switches to the alternate console device when an external interruption occurs.

MASTER SCHEDULER CONTROL FLOW

Commands are issued through either the console I/O device or the input reader. (See Figure 5.) Before entering commands through the console I/O device, the operator must cause an I/O interruption by pressing the REQUEST key. When he does, control is given to the supervisor. The supervisor determines that an I/O interruption has occurred and passes control to the I/O supervisor. The I/O supervisor determines that an attention interruption has occurred and passes control to the master scheduler console interrupt routine.

The console interrupt routine resides in the nucleus. It passes to the supervisor the address of an interruption queue element to be added to an asynchronous exit queue. The interruption queue element contains the address of an interruption request block that points to the master scheduler interrupt request block routine. Control is passed to the interrupt request block routine when the request is honored by the supervisor. A description of the asynchronous exit queue and the manner in which it is used is contained in the publication IBM System/360 Operating System: Fixed-Task Supervisor, Program Logic Manual, Form Y28-6612. The format of the master scheduler interruption queue element is given in the section entitled "Console Interrupt Routine."

The interrupt request block routine causes the master command EXCP routine to be brought into the supervisor call (SVC) transient area of the nucleus, where control is passed to it.

The master command EXCP routine uses an EXCP macro-instruction to read the command. (The PROCEED light on the 1052 Printer-Keyboard is turned on at this time.) Eight commands, the REQ, START (blank), CANCEL, DISPLAY, MOUNT, STOP, UNLOAD, and VARY commands, are always accepted and processed. All other commands are ignored (control is returned to the supervisor) if issued at any time other than in response to a message issued by the master command routine. If the command is acceptable, it is moved from the buffer into which it was read to a local buffer, and control is passed to the master command routine.

The master command routine analyzes commands and routes control to appropriate command execution routines. If a command is issued through the input job stream, control is passed directly to the master command routine by the reader/interpreter. When all commands have been entered and processed, control returns to the reader/interpreter.

The write-to-operator routine is entered from the SVC handler when a WTO or WTOR macro-instruction is issued. When either macro-instruction is issued, an SVC interruption occurs and the write-to-operator routine is brought into the SVC transient area of the nucleus. Basically, the write-to-operator routine uses an EXCP macro-instruction to write the message on the console output device and, if a reply is expected, to read the reply, which is placed into an area designated by the requester. Control is returned to the supervisor.

The external interrupt routine assigns the functions performed by the primary console device to the alternate console device. When the operator presses the INTERRUPT key on the console, an external interruption occurs and control is given to

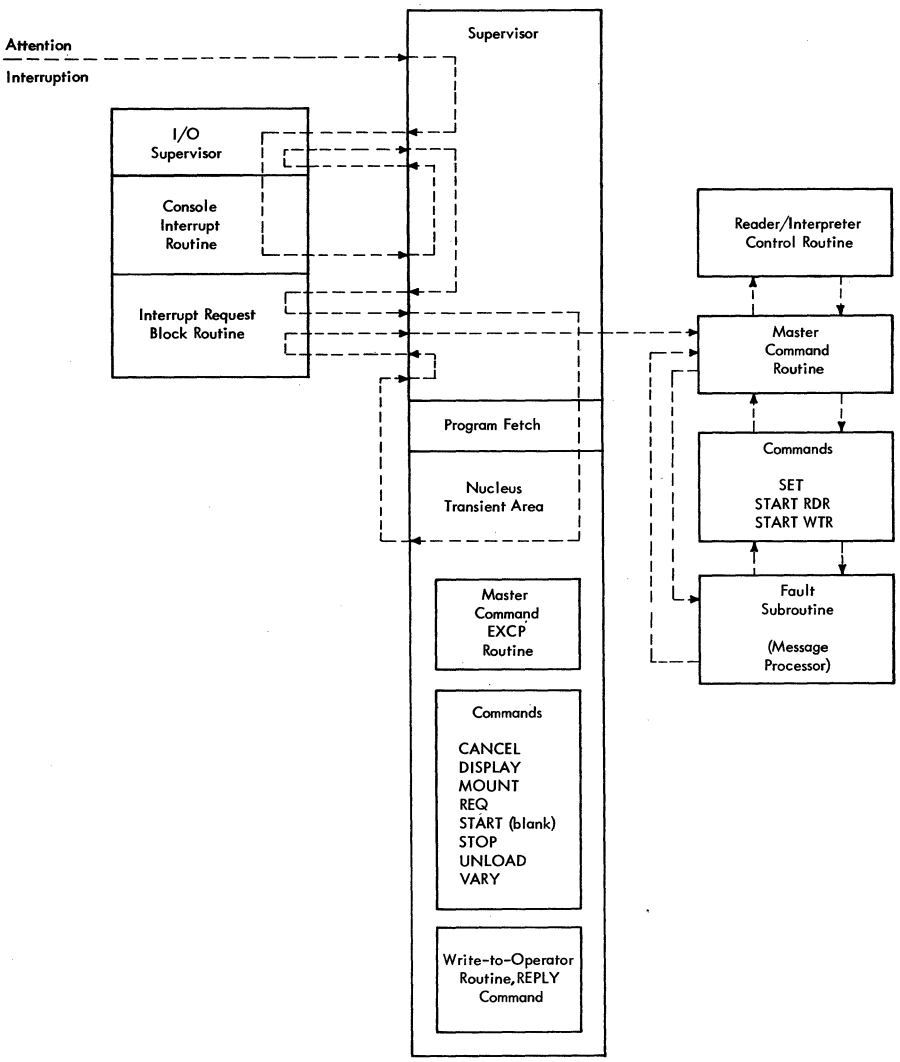


Figure 5. Master Scheduler - Command Processing Network

the supervisor, which identifies the interruption and passes control to the external interrupt routine. The external interrupt routine then switches consoles and returns control to the supervisor. Console functions may later be reassigned to the primary console device if the operator causes another external interruption (the external interrupt routine will again switch functions).

CONSOLE INTERRUPT ROUTINE

The console interrupt routine (Chart 3) provides the supervisor with the address of the routine to be given control when the supervisor processes an attention interruption. The console interrupt routine is part of the nucleus and is entered from the I/O supervisor each time an attention interruption occurs.

Upon entry to the console interrupt routine, the console flag switch is checked. If this switch is on, either the master command routine or the console interrupt routine is processing a prior request, and a RETURN is made to the I/O supervisor.

When an interruption is not being processed by either routine, the console flag switch is turned on, the address of the master scheduler interruption queue element is placed into general register 1, and control is passed to the supervisor. The interruption queue element is shown in Figure 6.

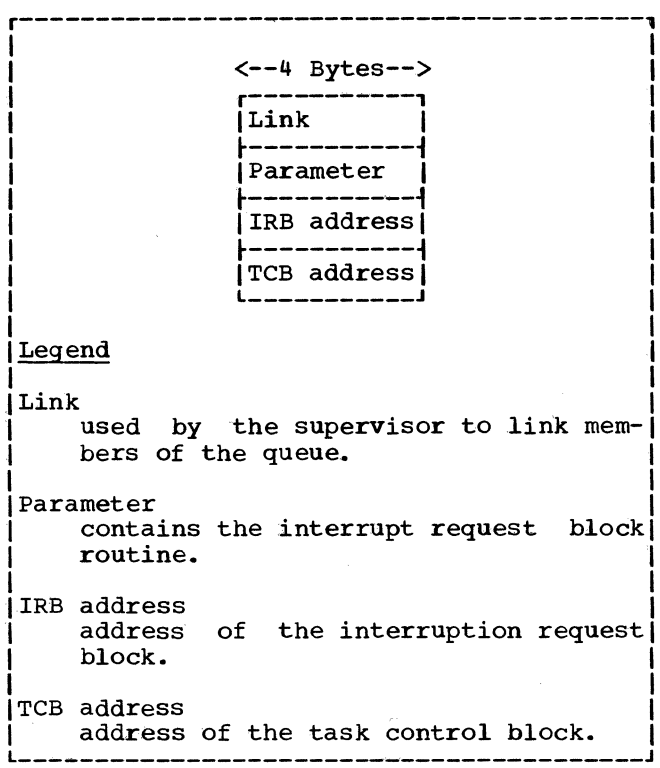


Figure 6. Master Scheduler Interruption Queue Element

The interruption request block contains the address of the interrupt request block (IRB) routine to which control is passed by the supervisor when it dispatches the request. The IRB routine uses an SVC 34 instruction to cause the master command EXCP routine to be brought into the transient area of the nucleus.

MASTER COMMAND EXCP ROUTINE

The master command EXCP routine (Chart 4) processes the CANCEL, DISPLAY, MOUNT, REQ, START (blank), STOP, UNLOAD, and VARY commands. It resides in SYS1.SVCLIB, and is brought into the transient area of the nucleus by the supervisor when an SVC 34 instruction is issued by the master scheduler interrupt request block routine or the master command routine.

If entry to this routine was from the interrupt request block routine, an EXCP macro-instruction is used to read the command from the console and place it into the command buffer. If the command is one of the eight previously mentioned commands, it is processed.

SET, START RDR, and START WTR commands are ignored unless they were issued in response to a message from the master command routine. If so, control is passed to the master command routine, which processes them.

Following return from the master command routine, or after execution of the REQ or START (blank) commands, the console flag switch is turned off to indicate to the console interrupt routine that another attention interruption can be processed.

If entry to the master command EXCP routine was from the master command routine, the command is available in a buffer (placed there by the master command routine). The command is processed.

The master command EXCP routine returns control to the supervisor.

MASTER COMMAND ROUTINE

The master command routine (Chart 5) analyzes command verbs and routes control to appropriate command execution routines. It also issues a message to the operator, informing him that commands will be accepted from the console. The master command routine is brought into main storage and entered when:

- The reader/interpreter encounters a command in the input job stream.
- The reader/interpreter is performing the initialization procedures that follow IPL.
- The reader/interpreter finds the command pending switch on. (The command pending switch is turned on by the routine that processes the REQ command.)
- The reader/interpreter encounters an end-of-data set condition in the input job stream, indicating the end of a job step or job. Control is passed to the master command routine after the job step has been processed.

Upon entry, general register 0 is examined. If it contains zeros, entry was made because the reader/interpreter encountered a command in the input job stream. The command is moved to the master command routine buffer and is written out on the console output device for the operator's records. The command verb is then analyzed, and if it is a SET, START RDR, or START WTR command, control is passed to an appropriate command execution routine. Otherwise, an SVC 34 instruction is used to pass control to the master command EXCP routine.

If general register 0 does not contain zeros upon entry to the master command

routine, the IPL pending, new reader pending, and new writer pending switches are checked. If any of these switches are on, the command pending switch is turned on and a message is issued requesting the operator to enter commands. Control is then passed to the initialization command routine, which provides certain commands, specified by the installation during system generation (SYSGEN), to relieve the operator of entering initialization commands. Each of these commands, if there are any, is moved to the master command routine buffer, written on the console output device for the operator's records, and executed.

If general register 0 does not contain zeros and none of the previously mentioned pending switches are on, entry to this routine was made because the reader/interpreter found the command pending switch on, or encountered an end-of-data set condition in the input job stream. A message is issued requesting commands from the operator. After the operator has issued commands and they have been processed, control is returned to the reader/interpreter.

WRITE-TO-OPERATOR ROUTINE

The write-to-operator routine (Chart 6) writes messages to the operator on the console output device when a WTO or WTOR macro-instruction is issued. These macro-instructions may be issued by system component programs and processing programs. Issuance of either macro-instruction causes an SVC interruption to occur. When the interruption is handled, the supervisor has the routine read into the transient area of the nucleus and passes control to it.

The message is written by using the EXCP macro-instruction. Replies (if any) are processed in the following manner. After an attention interruption, the EXCP macro-instruction is issued and a WAIT occurs until the operator responds with a REPLY command. When the reply is received, it is moved to the storage location whose address was supplied as a parameter in the WTOR macro-instruction, and the requester's event control block (ECB) is posted.

The write-to-operator routine returns control to the supervisor.

EXTERNAL INTERRUPT ROUTINE

The external interrupt routine (Chart 7) switches to an alternate console device when the operator presses the INTERRUPT key on the console. This routine resides in the nucleus.

The primary function of the reader/interpreter (Chart 8) is to read job control statements, analyze their contents, and build tables that are used during initiation and execution of job steps.

Control is passed to the reader/interpreter following:

- The IPL procedure.
- Execution and termination of a job step that was followed by data in the input job stream.
- Execution and termination of the last step of a job.

In each case, the reader/interpreter begins reading and processing control statements.

The reader/interpreter consists of four routines: A control routine that reads statements from the input job stream, and three routines that process the control statements. Entry to the reader/interpreter is always to the control routine.

The four routines of the reader/interpreter are:

- Control routine, which begins reading and processing statements, determines which type of statement was read, and passes control to the appropriate routine.

- Job routine, which analyzes the JOB statement and constructs a job control table (JCT) from the information in the statement.

- Execute routine, which analyzes the EXEC statement and constructs a step control table (SCT) from the information in the statement.

- DD routine which analyzes the DD statement and constructs a job file control block (JFCB), a step input/output table (SIOT) and, if necessary, a volume table (VOLT) and a dsname table from the information in the statement.

Major subroutines include:

- Breakout Routine, which scans statements and isolates each field as requested by the calling routine.

- Qualified name routine, which separates the elements of fully qualified data set names.

- Message routine, which builds system message blocks in which job management error messages are stored.

READER/INTERPRETER CONTROL ROUTINE

The reader/interpreter control routine (Chart 9) reads statements from the input job stream or procedure library, identifies the verbs, checks their validity, and for JOB, EXEC, and DD statements transfers control to the appropriate routine to process the statement. When the control routine encounters a second JOB statement, a DD *, DD DATA or null statement, or an EOF in the input job stream it passes control to the initiator/terminator.

The reader/interpreter control routine is entered from:

- The nucleus initialization program, when the master scheduler is to perform the actions immediately following IPL. The control routine begins reading statements from the input job stream after the actions are completed.

- The initiator/terminator, when termination procedures have been completed for a job step that was followed by data in the input job stream, or for the last step of a job. The control routine begins reading and processing statements for the next job step or job.

When a statement has been read from the input device (or procedure library when a procedure has been specified), it is examined. If it is a control statement, the control routine determines the length and starting address of the name, operation, and operand fields. This information is passed to the routine that processes the statement. The control routine then identifies the statement. For JOB, EXEC, and DD statements, exit is made to the appropriate statement processing routine. For commands, exit is made to the master scheduler's master command routine. When a second JOB or EXEC statement is read, or when a DD *, DD DATA, or null statement is read, or when an EOF is encountered in the input job stream, the step control table (SCT) for the completed step is stored by the table store subroutine. (It is either stored in auxiliary storage or, if the resident job queue option was specified during system generation, into a job queue area of main storage if the area is not full.) If a volume table (VOLT) or dsname

table exists for the step, it is also stored by the table store subroutine.

If either an EOF condition or a JOB, DD *, DD DATA, or null statement caused the SCT to be stored, control is passed to the initiator/terminator. Otherwise, the control routine continues reading and processing statements until it encounters such a statement or an EOF condition.

When control returns from the initiator/terminator, the control routine restores pointers to the previous statement, or reads the next statement and processes it.

Exits are to the job, execute, and DD routines when these control statement operation fields are recognized; to the initiator control function of the initiator/terminator after an SCT is stored when a second JOB, DD *, DD DATA, or null statement has been encountered; and to the master scheduler (a) when a command is recognized in the input job stream, (b) after return from job termination if an EOF occurred in the input job stream, and (c) before an SCT is stored if a command is pending.

JOB ROUTINE

The job routine (Chart 10) analyzes the JOB statement and, from the information it contains, constructs a job control table (JCT) and an account control table (ACT).

The job routine is entered from the reader/interpreter control routine when a JOB statement is recognized.

When it receives control, the job routine initializes main storage space for the JCT. Using the length and starting address parameters passed to the job routine by the control routine, the jobname is inserted in the JCT. Assumed JOB statement values are also inserted in the table at this time.

The breakout routine separates fields so that the list portion of the job statement can be examined. If accounting information is specified, it is put into the ACT for later use by an accounting routine.

As each keyword parameter is broken out, it is matched with the JOB keyword table. This table contains an entry for each keyword parameter which may appear in the JOB statement. The table also indicates if the parameter is supported in a given environment and gives the branch address for the routine which processes that parameter.

When the statement has been completely scanned, the job routine ascertains that all required fields are present and returns to the control routine which reads another statement. If any required fields are missing, an error message is written via the message routine, the job-failed bit in the JCT is set to indicate that the job has failed, and exit is made to the control routine. (Remaining control statements for the failed job are read and interpreted, but the job steps are not executed.)

Parameters in the JOB statement result in table entries shown in Figure 7.

JOB Statement Parameter	Table	Table Item
jobname	JCT	Jobname
account number	ACT	Account number, length of account number
programmer's name	ACT	Programmer's name
TYPRUN	Ignored in primary control program	
PRTY	Ignored in primary control program	
COND	JCT	Code, operator
MSGLEVEL	JCT	Message level
MGSCLASS	Ignored in primary control program	
REGION	Ignored in primary control program	

Figure 7. JOB Statement Parameter Dispositions

EXECUTE ROUTINE

The execute routine (Chart 11) analyzes the execute statement and, from the information it contains, constructs a step control table (SCT).

The routine includes a control subroutine that breaks out the fields and identifies the parameters, subroutines that process each keyword parameter, a subroutine that processes the statement if it indicates a cataloged procedure, and a subroutine that handles a reference to a previous step.

Entry to the execute routine is from the reader/interpreter control routine when an

EXEC statement is recognized. When the execute routine (control subroutine) receives control, it initializes main storage space for the SCT, regardless of whether the SCT was stored into the job queue area of main storage or into auxiliary storage, and then checks for the presence of parameters in the operand field. If none have been specified, an error message is issued, the job is failed, and control returns to the reader/interpreter control routine to continue processing the input job stream. If there are parameters in the operand field, the SCT is initialized and the stepname is inserted into the SCT.

The first parameter in the operand is then broken out. If it identifies a cataloged procedure, the execute routine finds the specified procedure in the procedure library and sets a switch to indicate that statements should be read from the procedure library as well as from the input job stream. Control is returned to the control routine to read and analyze the cataloged statements.

If the first parameter in the operand identifies a program, the execute routine checks first for a reference to a previous step (*.stepname.ddname) and, if such a reference is present, the refer-back subroutine is called. This subroutine scans the SCTs and SIOTs belonging to the current job to find the specified step and ddname. The auxiliary storage address of the SIOT is inserted into the programname field of the current SCT and processing of the statement continues. If there is no backward reference, the control subroutine inserts the specified program name into the SCT.

Before scanning of the statement continues, condition subparameters are placed into the SCT, and the parameter fields are set to zero. Scanning continues to an equal sign (=). When one is found, the preceding parameter is matched with the EXEC key field table. This table indicates if a parameter is supported in a given system and gives the branch address for the routine which processes that parameter. If a parameter is invalid, a message is issued, the parameter is ignored, the job-failed indicator is turned on, and scanning of the rest of the statement continues. (Remaining control statements for the failed job are read and interpreted, but the job steps are not executed.) If a parameter is not supported by the system, a message is issued but the job-failed indicator is not turned on. Scanning of the rest of the statement continues.

When the end of the statement is reached, control is returned to the reader/interpreter control routine, which continues processing the input job stream.

The parameters in the EXEC statement result in table entries shown in Figure 8.

EXEC Statement Parameter	Table	Table Item
stepname	SCT	Stepname
PGM	SCT	Programname
PROC		Cataloged control statements are interpreted and merged with input statements.
TIME		Ignored in the primary control program
COND	SCT	Code, operator, auxiliary storage address of referenced SCT
PARM	SCT	Initializing parameter values
ACCT	ACT	Step accounting fields
REGION		Ignored in primary control program

Figure 8. EXEC Statement Parameter Dispositions

DD ROUTINE

The DD routine (Chart 12) analyzes the DD statement and, from the information it contains, constructs a JFCB, SIOT, and, if necessary, a VOLT and dsname table.

The DD routine includes several subroutines that process the various fields and delimiters on a DD statement. The header and scan subroutines control the operation of the DD routine. They break out the fields in the statement and pass control to an appropriate subroutine. There are subroutines to process fields bounded by each type of delimiter which may appear, to convert the contents of each field to internal format where necessary, and to put the result in the tables. When the entire statement has been analyzed, the output subroutine uses the table store subroutine to store the completed tables. The tables are stored either in auxiliary storage or, if the resident job queue option was specified during system generation, into a job queue area of main storage if the area is not full.

Entry to the DD routine is made from the reader/interpreter control routine through the DD header subroutine. The subroutine checks entry conditions and processes any special ones that may exist (continuation received, DD statement expected but not received, JOBLIB DD statement). It also checks for a DD sequence error. If such an error is found, the job-failed indicator is turned on, a message is written, and the entire statement is ignored. (Remaining control statements for the failed job are read and interpreted, but the job steps are not executed.) If no syntactical or sequence errors are found, normal processing of the statement continues.

The name field is interpreted and broken down into its levels of qualification. The breakout subroutine scans the list portion of the statement for delimiters; i.e., comma, blank, equal sign, right parenthesis and left parenthesis. When a delimiter is encountered, control goes to the subroutine that processes the delimiter. These subroutines break out the contents of each field and insert them into the appropriate table. The DD major field table and the DD parameter list table provide the information necessary to treat each field.

The DD routine places flags into the step input/output table (SIOT, Figure 42), to denote requests for nonshareable and private volumes. (A device allocated to satisfy a request for a nonshareable volume may not be used to satisfy any other request for nonshareable volumes.)

A private volume is requested with a DD statement containing the PRIVATE subparameter.

A nonshareable volume is requested with a DD statement that:

- Specifies a private volume for the request.
- Requests a specific volume by serial number (VOLUME=SER=x), by reference (VOLUME=REF=x), or by referring to a cataloged or passed data set by data set name (DSNAME=x).
- Specifies a multivolume data set by giving a volume count subparameter greater than one.

The DD routine also places a flag into the job file control block (JFCB, shown in the publication System Control Blocks) to denote each request for temporary data set space. (The demand allocation routine later transfers the nonshareable and private flags from the step input/output table into the allocate work table entry for the request.)

When the statement has been completely scanned, an output subroutine completes the JFCB and SIOT, and uses the table store subroutine's write and assign functions to store the JFCB either in the job queue area of main storage or onto auxiliary storage. (If a job queue area was specified during system generation and is not full, the I/O supervisor causes the table to be stored in main storage.) If all interpreting for a job step is complete, or if data is encountered in the input job stream, the output routine also causes the SCT of the current step to be stored.

If information in the DD statement is to override information in a cataloged DD statement, a composite DD statement is created.

The DD routine exits to the reader/interpreter control routine when interpretation of a statement is complete and when an SCT is to be written out.

The parameters and subparameters of the DD statement result in table entries shown in Figure 9.

DD Statement Parameter	Table	Table Item	Bits
*, DATA	SIOT	SCTUTYPE	
	SIOT	SCTSBYT1	1
	SIOT	SCTSDISP	4
	SIOT	SCTSBYT3	7
	SCT	SCTSTYPE	
	JFCB	JFCBTSDM	2
	JFCB	JFCBIND2	1
	JFCB	JFCBDSNM	
DUMMY,DDNAME=	SIOT	SCTSBYT1	0
	JFCB	JFCBDSNM	
DSNAME=	SIOT	SCTSBYT4	0
	JFCB	JFCBDSNM	
	JFCB	JFCBELM	
	JFCB	JFCBIND1	6
	JFCB	JFCBIND1	7
	JFCB	JFCBIND2	7
DCB=			
dsname	SIOT	SIOTDCBR	
BFALN=D	JFCB	JFCBFALN	6
BFALN=F	JFCB	JFCBFALN	1
BFTEK=D	JFCB	JFCBFTEK	4
BFTEK=E	JFCB	JFCBFTEK	3
BFTEK=S	JFCB	JFCBFTEK	1
BLKSIZE	JFCB	JFCBLKSI	
BUFL	JFCB	JFCBUFL	
BUFNO	JFCB	JFCBUFNO	
BUFRQ	JFCB	JFCBUFRQ	
CODE=A	JFCB	JFCCODE	5
CODE=B	JFCB	JFCCODE	3
CODE=C	JFCB	JFCCODE	4
CODE=F	JFCB	JFCCODE	2
CODE=I	JFCB	JFCCODE	1
CODE=N	JFCB	JFCCODE	0
CODE=T	JFCB	JFCCODE	6
CPRI	JFCB	JFCCPRI	
CYLOFL	JFCB	JFCCYOFL	
DBUFNO	JFCB	JFCDBUFN	
DEN=0	JFCB	JFCDEN	67
DEN=1	JFCB	JFCDEN	167
DEN=2	JFCB	JFCDEN	067
DSORG=CQ	JFCB	JFCDSORG	4
DSORG=CX	JFCB	JFCDSORG	3
DSORG=DA	JFCB	JFCDSORG	2
DSORG=IS	JFCB	JFCDSORG	0
DSORG=MQ	JFCB	JFCDSORG	5
DSORG=PO	JFCB	JFCDSORG	6
DSORG=PS	JFCB	JFCDSORG	1
DSORG=U	JFCB	JFCDSORG	7
EROPT=ABE	JFCB	JFCEROPT	2
EROPT=ACC	JFCB	JFCEROPT	0
EROPT=SKP	JFCB	JFCEROPT	1
INTVL	JFCB	JFCINTVL	
KEYLEN	JFCB	JFCKEYLE	
LIMCT	JFCB	JFCLIMCT	
LRECL	JFCB	JFCLRECL	
MODE=C	JFCB	JFCMODE	0
MODE=E	JFCB	JFCMODE	1
NCP	JFCB	JFCNCP	
NTM	JFCB	JFCNTM	
OPTCD=A	JFCB	JFCOPTCD	4
OPTCD=C	JFCB	JFCOPTCD	4

DD Statement Parameter	Table	Table Item	Bits
DCB= (cont.)			
OPTCD=E	JFCB	JFCOPTCD	2
OPTCD=F	JFCB	JFCOPTCD	3
OPTCD=I	JFCB	JFCOPTCD	3
OPTCD=L	JFCB	JFCOPTCD	6
OPTCD=M	JFCB	JFCOPTCD	2
OPTCD=P	JFCB	JFCOPTCD	2
OPTCD=R	JFCB	JFCOPTCD	7
OPTCD=W	JFCB	JFCOPTCD	0
PRTSP	JFCB	JFCPRTSP	
RECFM=A	JFCB	JFCRECFM	5
RECFM=B	JFCB	JFCRECFM	3
RECFM=F	JFCB	JFCRECFM	0
RECFM=G	JFCB	JFCRECFM	5
RECFM=K	JFCB	JFCRECFM	7
RECFM=M	JFCB	JFCRECFM	6
RECFM=R	JFCB	JFCRECFM	6
RECFM=S	JFCB	JFCRECFM	4
RECFM=T	JFCB	JFCRECFM	2
RECFM=U	JFCB	JFCRECFM	01
RECFM=V	JFCB	JFCRECFM	1
RKP	JFCB	JFCRKP	
SOWA	JFCB	JFCSOWA	
STACK	JFCB	JFCSTACK	
TRTCH=C	JFCB	JFCTRTCH	367
TRTCH=E	JFCB	JFCTRTCH	267
TRTCH=ET	JFCB	JFCTRTCH	2467
TRTCH=T	JFCB	JFCTRTCH	2346
			7
TRTCH=TE	JFCB	JFCTRTCH	2467
SEP=	SIOT	SCTCSADD	
	SIOT	SCTSBYT2	1
AFF=	SIOT	SCTCSADD	
	SIOT	SCTSBYT2	0
UNIT=			
name	SIOT	SCTUTYPE	
n	SIOT	SCTNMBUT	
P	SIOT	SCTSBYT1	5
DEFER	SIOT	SCTSBYT2	6
SEP=	SIOT	SCTUSADD	
	SIOT	SCTSBYT1	7
POOL	NO Tables Affected		
poolname	SIOT	SCTSPOOL	
0	SIOT	SCTNMBUT	
1	SIOT	SCTNMBUT	
AFF=	SIOT	SCTUSADD	
	SIOT	SCTSBYT1	6
SPACE=			
TRK	JFCB	JFCBCTRI	0
CYL	JFCB	JFCBCTRI	01
average rec. length	JFCB	JFCBCTRI	1
	JFCB	JFCBDRLH	
primary qty.	JFCB	JFCBPQTY	
secondary qty.	JFCB	JFCBSQTY	
directory qty.	JFCB	JFCBDQTY	
RLSE	JFCB	JFCBIND1	1
MXIG	JFCB	JFCBCTRI	5
ALX	JFCB	JFCBCTRI	6
CONTIG	JFCB	JFCBCTRI	4
ROUND	JFCB	JFCBCTRI	7

• Figure 9. DD Statement Parameter Dispositions

(Continued)

DD Statement Parameter	Table	Table Item	Bits
SPACE= (cont.)			
ABSTR	No Table Affected		
primary qty.	JFCB	JFCBQTY	
beginning address	JFCB	JFCBABST	
directory qty.	JFCB	JFCBDQTY	
SPLIT=	SIOT	SCTSBYT1	23
n	JFCB	JFCBSPTN	
CYL	JFCB	JFCBCTRI	01
average rec. length	JFCB	JFCBCTRI	1
	JFCB	JFCBDRLH	
primary qty.	JFCB	JFCBPQTY	
secondary qty.	JFCB	JFCBSQTY	
SUBALLOC=	SIOT	SCTSBYT1	4
TRK	JFCB	JFCBCTRI	0
CYL	JFCB	JFCBCTRI	01
average rec. length	JFCB	JFCBCTRI	1
	JFCB	JFCBDRLH	
primary qty.	JFCB	JFCBPQTY	
secondary qty.	JFCB	JFCBSQTY	
directory qty.	JFCB	JFCBDQTY	
stepname.ddname	SIOT	SIOTVRSB	
ddname	SIOT	SIOTVRSB	
	SIOT	SCTSBYT3	3
VOLUME=			
PRIVATE	SIOT	SCTSDISP	02
RETAIN	SIOT	SCTSDISP	1
vol. seg. no.	JFCB	JFCBVLSQ	
vol. count	JFCB	JFCVOLCT	
SER=	SCT	SCTVOLTB	
	SCT	SCTVOLTL	
	SIOT	SCTVOLCT	
	SIOT	SCTVLTPR	
	SIOT	SCTSDISP	0
	JFCB	JFCBNVOL	
	JFCB	JFCBEXAD	
	JFCB	JFCBVOLS	
	VOLT	INDMVOLT	
REF=	dsname	INDMDSNT	
	SCT	SCTVLPTR	
	SCT	SCTVOLCT	
	SCT	SCTADSTB	
	SCT	SCTLDSTB	
	SIOT	SIOTVRSB	
	SIOT	SCTSBYT2	2
	SIOT	SCTSBYT3	0
	SIOT	SCTSDISP	0
LABEL=			
data set seg. no.	JFCB	JFCBFLSQ	
NL	SIOT	SCTSBYT2	4
	JFCB	JFCBLTYP	7
SL	JFCB	JFCBLTYP	6
NSL	SIOT	SCTSBYT2	4
	JFCB	JFCBLTYP	5
SUL	JFCB	JFCBLTYP	4
EXPDT	JFCB	JFCBCRDT	
	JFCB	JFCBXPDT	
RETPD	JFCB	JFCBCRDT	
	JFCB	JFCBXPDT	
DISP=			
NEW	JFCB	JFCBIND2	01
	SIOT	SCTSBYT3	5
OLD	JFCB	JFCBIND2	1
	SIOT	SCTSBYT3	7
DISP= (cont.)			
MOD	JFCB	JFCBIND2	0
	SIOT	SCTSBYT3	6
DELETE	SIOT	SCTSDISP	5
KEEP	SIOT	SCTSDISP	4
PASS	SIOT	SCTSDISP	3
CATLG	SIOT	SCTSDISP	6
UNCATLG	SIOT	SCTSDISP	7
SHR	JFCB	JFCBIND2	1
	SIOT	SCTSBYT3	7
SHARE	JFCB	JFCBIND2	1
	SIOT	SCTSBYT3	7
SYSOUT=	JFCB	JFCBDSNM	
	JFCB	JFCBTSDM	2
	JFCB	JFCBLTYP	4
	SIOT	SCTSBYT3	4
	SIOT	SCTSBYT1	0
classname	SIOT	SCTOUTPN	
progname	SIOT	SCTOUTNM	
form number	SIOT	SCTOUTNO	

Figure 9. DD Statement Parameter Dispositions

The initiator/terminator (Chart 13) ensures that all I/O resources needed by a job step are available before control is passed to the step. The initiator/terminator analyzes the I/O device requirements of job steps and allocates devices to them. If necessary, it issues mounting instructions and verifies that volumes were mounted on the correct units.

Control is passed to the initiator/terminator from:

- The reader/interpreter, when the reader/interpreter encounters a second JOB statement, a DD *, DD DATA, or null statement, or an EOF in the input job stream.
- The supervisor, following step execution.

The initiator/terminator passes control to:

- The job step, when all I/O devices needed by the step have been assigned and the step is ready for execution.
- The reader/interpreter, when termination procedures have been completed for a step or job.

Initiator/terminator routines are arranged into four groupings:

- Initiator control.
- Allocation and setup.
- Step initiation.
- Termination.

Initiator control routines perform housekeeping functions, analyze condition codes specified by the programmer in the EXEC statement, and update JFCBs and other tables associated with the step.

Allocation and setup routines analyze a step's I/O requirements (taking into consideration, for example, requests for absolute assignments and unit and volume affinity). They then allocate devices and issue messages instructing the operator to mount required volumes.

Step initiation routines open the job library data set for each step if a JOBLIB DD statement is included with the job. Also, if the step being initiated consists of a program that was created by a previous step (commonly known as "compile, load, and go"), a step initiation routine opens the data set containing the program. Before passing control to the job step, a step initiation routine takes two preparatory steps. It loads control information that followed the PARM keyword of the EXEC statement into main storage. It also uses the table store subroutine to store all tables associated with the job step, thereby protecting them for use by the termination routines.

Termination routines are entered after each job step is executed. They supervise entry to the user's accounting routine (if one exists) and, upon return, dispose of data sets referenced by the step during execution and release devices allocated to the step.

Information is passed between initiator/terminator routines by means of the linkage control table (LCT) (see Figure 10). The LCT is built and initialized during IPL. It is stored before processing program execution and, following execution, is retrieved by initiator/terminator termination routines. The beginning address of the LCT is maintained in general register 12 during execution of the initiator/terminator.

INITIATOR CONTROL

Initiator control (Chart 14) performs certain housekeeping functions for the initiator/terminator, and also checks EXEC statement condition codes (if any). Condition codes appearing in EXEC statements determine whether or not a job step is to be executed.

Routines that comprise initiator control are:

- System control routine, which is the entry point for the initiator/terminator. Control is passed to the initiator/terminator when a step is ready for initiation and also after one has been executed and terminated, if another step is to be initiated. Housekeeping is performed and control is passed to the execute statement condition code routine.
- Execute statement condition code routine, which checks any dependencies encountered in EXEC statements.

- JFCB housekeeping routines, which complete portions of JFCBs and SIOTs that describe the volumes to be used during step execution. These routines also construct a passed data set queue (PDQ) to describe data sets being passed and update the PDQ for data sets being received by the step being processed.

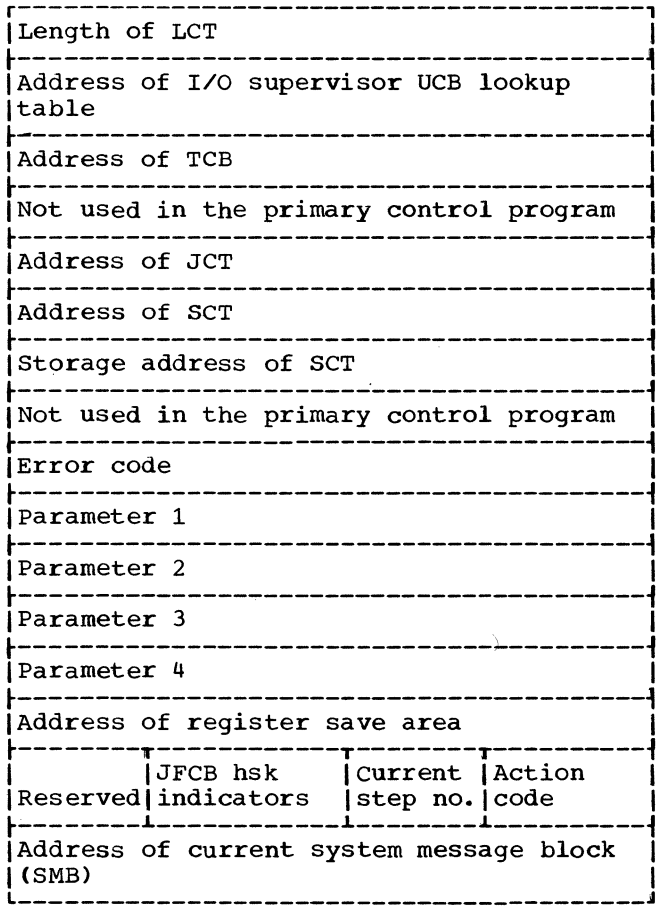


Figure 10. Linkage Control Table

SYSTEM CONTROL ROUTINE

The system control routine (Chart 15) is entered from the reader/interpreter when it completes the processing of a step that was followed by data in the input job stream, or when it reads the last step of a job. It is also entered from the step termination routine if additional steps remain to be initiated.

Upon entry, the system control routine updates the step number in the LCT. Then, if the step is the first step of the job, its job name is placed into the selected job queue. (See Figure 11.)

Jobname	Cancel ECB

Figure 11. Selected Job Queue

If the step being processed is the first step of the job, and if a DISPLAY JOBNAMES command has been issued, the WTO macro-instruction is used to write the message:

IEF401I jobname STARTED

on the console output device. Control is then passed to the execute statement condition code routine.

EXECUTE STATEMENT CONDITION CODE ROUTINE

The execute statement condition code routine (Chart 16) processes any step condition codes that were specified in the EXEC statement.

If, upon entry, it is found that the current step is the first step of the job or that no condition codes were specified in the EXEC statement, exit is made to the JFCB housekeeping routines. If neither of the above situations exists, each of the step conditions that were specified in the EXEC statement is compared with the corresponding return code.

Should the results of the comparison agree with the condition operator specified in the EXEC statement, the job step is canceled. (The step status indicator in the SCT is set to cancel.) The message subroutine is used to write a message to the programmer. Exit from this routine is made to the step termination routine if the step was canceled, or to the JFCB housekeeping routine if the step was not canceled.

JFCB HOUSEKEEPING ROUTINES

The JFCB housekeeping routines (Chart 17) complete volume information within certain tables, in preparation for their use by allocation routines. This information is generally the type that requires reference to the catalog (use of the LOCATE and OBTAIN macro-instructions) or to passed data sets. Tables in which entries are made include:

- Job file control block.
- Step input/output table.
- Step control table.
- Volume table.

For passed data sets, a PDQ is constructed and entries are made for the first occurrence of each data set being passed to a subsequent step. The existing data set

queue entries are then updated when a data set is received from a previous step.

The JFCB housekeeping routines include the following:

- JFCB housekeeping control routine.
- Allocate processing routine.
- Fetch DCB processing routine.
- GDG single processing routine.
- GDG all processing routine.
- Patterning DSCB processing routine.
- Error message processing routine.

JFCB Housekeeping Control Routine

The JFCB housekeeping control routine (Chart 18) determines what processing (if any) is required, and directs control to the first appropriate processing routine. Upon return of control, it redirects control to the next required processing routine. This routine places each SIOT for a job step into a main storage work area, examines it, and, depending on the type of information required, passes control to the processing routine which performs the actions necessary to retrieve the required information.

When all SIOTs for a job step have been examined, the JFCB housekeeping control routine passes control to the allocation and setup function of the initiator/terminator.

Allocate Processing Routine

The allocate processing routine (Chart 19) completes information about data sets which reference another data set by data set name (indicating a passed or cataloged data set) or by ddname or stepname.ddname (indicating a data set described in a previously processed DD statement).

When the data set reference is a data set name, the passed data set queue is examined and, if it contains an entry for the referenced data set, the SIOT and JFCB for that data set are placed into a main storage work area and are used to complete device and volume information for the subject data set.

If there is no entry for the referenced data set in the PDQ, a LOCATE macro-instruction is issued to find that data set in the catalog. Its volume control block or data set pointer entry is then used to complete the volume and device information for the subject data set.

When the data set reference is by ddname or stepname.ddname, a check is made to determine if the DD statement appeared in the step being processed. If so, the SIOT and JFCB associated with the referenced DD statement are placed into a main storage work area. These are used to complete the device and volume information of the subject data set.

If the DD statement appeared in a previous step of the job being processed, the SIOT and JFCBs constructed by the last step to reference the data set are placed into a main storage work area and are used to complete the volume and device information of the subject data set.

When a unit name is specified in the DD statement, the unit name is converted to unit type, through use of the device name table.

Fetch DCB Processing Routine

The fetch DCB processing routine (Chart 20) completes volume and device information when the data set referred to contains a program that was created in a previous step and is to be executed as the current step.

GDG Single Processing Routine

The GDG single processing routine(Chart 21) obtains the data set name of a generation data group (GDG) member and completes volume and device information entries for that member.

GDG All Processing Routine

The GDG all processing routine (Chart 22) builds an SIOT, JFCB, volume table entry, and PDQ entry for each GDG member when the entire generation data group is specified by the programmer.

Patterning DSCB Processing Routine

The patterning DSCB processing routine (Chart 23) completes control information in a JFCB when a new data set is to be patterned after a previously cataloged data set. The volume control block or data set pointer entry, which contains the volume serial number of the volume that contains the data set, is placed into a main storage work area. Fields in the JFCB are checked for zeros. If a field contains zeros, the corresponding field from the DSCB is moved into the JFCB.

Error Message Processing Routine

The error message processing routine (Chart 24) is entered and issues error messages whenever an error condition is encountered within a JFCB housekeeping routine.

ALLOCATION AND SETUP

The allocation and setup function of the initiator/terminator (Chart 25) allocates I/O devices, issues any necessary mounting instructions to the operator, and ensures that enough I/O requirements have been satisfied to begin execution of a job step. The routines in the allocation and setup function are:

- Allocation control routine, which performs housekeeping for the allocation and setup function by obtaining space for tables used during allocation.
- Demand allocation routine, which constructs the allocate tables and begins actual allocation by assigning devices to any data sets for which the programmer requested specific devices.
- Automatic volume recognition routine, (optional) which can determine that named volumes have been mounted on certain devices and which allocates those devices to satisfy requests for the volumes.
- Decision allocation routine, which performs allocation when a choice of devices is to be made.
- TIOT construction routine, which builds a task input/output table (TIOT) that will be used by data management routines during step execution.
- External action routine, which issues mounting instructions, verifies that volumes are mounted on the correct units, and unloads incorrectly mounted volumes.
- Space request routine, which obtains, from the direct-access device space management (DADSM) routines, space on direct-access devices, and which satisfies requests for data set space.
- Allocation error routines, which process error conditions encountered during allocation.

ALLOCATION CONTROL ROUTINE

The allocation control routine (Chart 26) performs housekeeping operations for the allocation and setup function of the initiator/terminator. It determines the size of certain tables to be constructed by subsequent allocation routines, obtains main storage space for the tables, and places the addresses of the portions of storage reserved for each table onto a directory of tables called the allocate control block.

Entry to the allocation control routine is made from the JFCB housekeeping control routine. Exit is to the demand allocation routine.

Upon entry, the storage requirements of the tables needed by allocation routines are calculated (see Figure 12). First, all requirements except those for the allocate volume table and TIOT are determined. The required amount of main storage space is requested and the addresses of the areas assigned to each table are calculated. (The first table is assigned the first available byte. Other addresses are determined by incrementing the last assigned address by length of the the respective table.) The relative position of each table except the device mask table is shown in Figure 13. The device mask table is included with the coding and is not positioned relative to the tables shown. As each address is determined, it is placed into the allocate control block.

DD number table*	4 |A|4
Buffer	176
Allocate control block	44
Channel load table	16
Allocate work table	(20 + 8 |D|32|) A
Potential user on device table	4 |D|4
Separation strikeout pattern	|D|32|
Each SIOT	68
Volume table	6S
TIOT	Determined by the TIOT construction routine
Allocate volume table	8B
Device mask table	4 + (8 + |D|32|) F

Legend
- * = Not used.
- \|\| = Next higher integer if a fraction.
- A = Number of DD statements.
- B = Number of volumes or devices (whichever is greater).
- D = Number of entries in the I/O supervisor UCB lookup table.
- F = Number of entries in device mask table.
- S = Number of volume serial numbers.

Figure 12. Formulas for Determining Allocation Table Sizes

When storage areas have been assigned for all but the allocate volume table (AVT) and task input/output table (TIOT), all step input/output tables (SIOTs) are placed into the area assigned to them. The size of the allocate volume table may then be determined. The number of volumes required by each data set (DD statement) is obtained from each SIOT and is used to calculate the number of AVT entries (one per volume) required. A second request for main storage space is issued and the address of the assigned area is placed into the allocate control block.

The storage requirements for the TIOT are calculated by the TIOT construction routine.

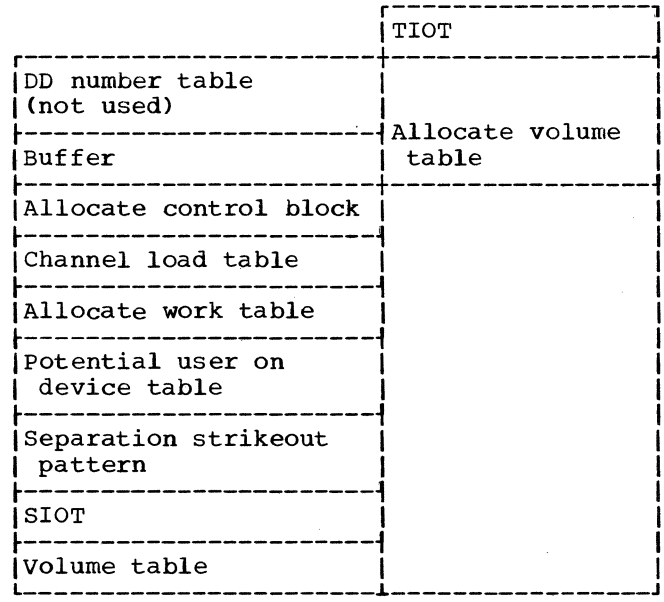


Figure 13. Relative Positions of Tables Used for Allocation

If, after a request for space, the required amount of main storage space is not available, the job is canceled.

Figure 14 shows the completed allocate control block. In addition to table addresses, the allocate control block contains other entries initialized by the allocate control routine.

All allocation tables are described in the descriptions of routines in which they are completed. When the allocate control block has been completed, control is passed to the demand allocation routine.

Channel load table address [4]	
Address of first empty slot in allocate volume table [4]	
Potential-user-on-device table address [4]	
Allocate work table (AWT) address [4]	
Allocate volume table address [4]	
Volume table address [4]	
Separation strikeout pattern address [4]	
Number of satisfied requests[1] [2]	Number of requests not satisfied[2] [2]
Number of bytes per AWT entry [2]	Length of allocate volume table [2]
Length of bit pattern[3] [2]	Number of DD statements in job step[4] [2]
Not Used [2]	Number of devices in configuration[5] [2]

Notes: (Entry length is shown in upper right corner of field.)

[1]Set to zero initially and incremented by one each time a request is satisfied.

[2]Initially set to the number of data sets to be allocated (the number of DD statements in the step). This number is decremented by one each time a request is satisfied.

[3]The length (in words) of the primary bit pattern.

[4]The number of DD statements to be processed.

[5]The number of UCB addresses in the I/O supervisor UCB lookup table.

Figure 14. Allocate Control Block

DEMAND ALLOCATION ROUTINE

The demand allocation routine (Chart 27) constructs the allocate work table and the allocate volume table. It also begins the allocation process by assigning devices to data sets that require specific devices. A specific device may be required because (1) the programmer specified it in a DD statement, or (2) all device requirements for a step could be met with only one combination of devices. The demand allocation routine performs the following eight functions:

- Allocate work table construction.
- Volume affinity resolution.
- Data set device requirement calculation.
- Channel load table construction.
- Allocation of resident devices.
- Device range reduction.
- System input device (SYSIN) allocation.
- Specific device allocation.

Allocate Work Table Construction

Two tables, the allocate volume table (see Figure 16) and the allocate work table (see Figure 15), are constructed by this function. The allocate work table contains information that describes a data set and certain other information that is used in allocating a device (or devices) to it. One entry, as shown in Figure 16, is built for each DD statement. The allocate volume table describes the volume on which the data set resides or will reside. One entry is made in the allocate volume table for each volume required by a data set.

DD number	Status E	UCB address
Pointer to volume serial number in volume table		Volume affinity link

Figure 15. Allocate Volume Table

Most entries made in the allocate work table are obtained directly from other tables. The source of each such entry is shown in Figure 17. The device type is obtained from the SIOT and placed into the device type field of the allocate work table. It is then used as a search argument and a search of the device mask table is made. When a matching device type is found, the bit pattern field of the device mask table is placed into the primary and secondary bit pattern fields of the allocate work table. These bit patterns indicate devices that are eligible for allocation to a data set.

The demand allocation routine moves the private and nonshareable flag bits from the step input/output table (SIOT) to the allocate work table (AWT). The demand allocation routine also sets the nonshareable bit in the allocate work table entry for a request if the request does not specify a direct-access device, and sets the private bit if the request is specifically for a nondirect-access device (unless request applies to passed data sets).

Data sets that have similar I/O device requirements are then linked together. Similar requirements are implied when the programmer specifies the following in a DD statement:

- POOL=poolname, which indicates that an output data set is to share a pool of tape units with other data sets.
- SPLIT=, which indicates that two or more data sets in the same job step are to share a cylinder of a direct-access device.
- SUBALLOC=stepname.ddname or ddname, which indicates that space for the data set will be suballocated from the space allocated to the data set described in the DD statement named ddname.

Pointers are placed into the POOL/SPLIT/SUBALLOC link field and unit affinity link field of the allocate work table to link all such groups together.

Number of devices available	POOL/SPLIT/ SUBALLOCATE link	Number of devices requested	Number of volumes
Status A	Status B	Status C	Status D
Number of devices allocated	Number of devices shared	Number of devices required	Unit affinity link
Address of first entry in volume table		Possible number of devices in secondary bit pattern	DD number
Device type			
Primary bit pattern (initially, a duplicate of secondary bit pattern)			
Secondary bit pattern			

Figure 16. Allocate Work Table Entry

Entry	Source
Number of devices available	Device mask table
POOL/SPLIT/SUBALLOC link	SIOT
Number of devices requested	SIOT
Number of volumes	SIOT
Status A	SIOT
Status B	SIOT
Status C	SIOT
Status D	SIOT
Number of devices allocated	Inserted as devices are allocated
Number of devices shared	Calculated
Number of devices required	Calculated
Unit affinity link	SIOT
Address of first entry in volume table	Calculated
Possible number of devices in secondary bit pattern	Device mask table
DD number	SIOT
Device type	SIOT
Primary bit pattern	Device mask table
Secondary bit pattern	Device mask table

Figure 17. Allocate Work Table Entry Sources

Volume Affinity Resolution

Volume affinity means that a certain volume is requested for more than one data set. Volume affinity may be requested explicitly by use of the REF parameter of the VOLUME field of the DD statement, or implicitly by specifying the same volume serials in one or more other DD statements. In either case, the subject volumes are linked with pointers placed into the volume link field of the allocate volume table by the demand allocation routine. All requests for the same volume that appear in the volume affinity chain subsequently will be satisfied with allocation of the device that bears the named volume.

Data Set Device Requirement Calculation

Information obtained from the allocate work table is used to determine the number of devices required by each data set. The following calculations are used:

1. For a data set marked parallel mount (the P subparameter of the UNIT keyword was specified in the DD statement):

   $D_1 = V_2$

2. For data sets not marked parallel mount:

   a. If $V_1 = V_2$ then $D_1 = V_2$

   b. If $V_1 < V_2$ and
      if $V_1 < D_2$ then $D_1 = D_2$ or
      if $V_1 \geq D_2$ then $D_1 = V_1 + 1$

where:

D_1 = Number of devices actually to be used for the data set.

D_2 = Number of devices requested for the data set.

V_1 = Number of volumes to be shared by two or more data sets.

V_2 = Number of volumes on which the data set exists.

The number of devices to be used (D_1) is placed into the number of devices required field of the allocate work table.

Channel Load Table Construction

Each unit control block (UCB) is examined to determine the number of data sets presently allocated on each channel. The totals are placed into the channel load table (see Figure 18). The channel load table is used to allocate devices so that channel usage is optimized.

Load Channel 0	Load Channel 1
Load Channel 2	Load Channel 3
Load Channel 4	Load Channel 5
Load Channel 6	

Figure 18. Channel Load Table

Allocation of Resident Devices

The resident device allocation routine allocates direct-access devices containing reserved and permanently resident volumes to satisfy requests by serial number for these volumes. The devices that contain these volumes are known as resident devices.

A volume is placed into the reserved status either when the operator issues a MOUNT command specifying the device on which the volume is mounted or when the volume is so listed in the PRESRES member of the procedure library data set (SYS1.PROCLIB). This type of volume cannot be dismounted unless its device is unloaded by means of an UNLOAD command.

A permanently resident volume has at least one of the following characteristics:

- The volume cannot be physically dismounted from its device.
- The volume is a system residence volume that contains the initial program loader (IPL) program.
- The volume contains the linkage library (SYS1.LINKLIB) data set, procedure library data set, or any part of the job queue (SYS1.SYSJOBQE) data set.
- The volume is listed as permanently resident in the PRESRES member of SYS1.PROCLIB.

For more information about the PRESRES data set member, refer to IBM System/360 Operating System: System Programmer's Guide, Form C28-6550. For more information about reserved and permanently resident volumes, refer to IBM System/360 Operating System: Job Control Language, Form C28-6539.

The resident device routine determines which direct-access devices are resident and then allocates them to satisfy any requests for the volumes they contain.

From the device mask table (DMT, Figure 32), the resident device routine first creates a special bit pattern that represents all direct-access devices in the system. It sets a bit in the pattern to one for each direct-access device. It then searches for unit control blocks representing direct-access devices, using this bit pattern to identify the unit control blocks.

The routine compares the volume serial number in each request with the serial number in each unit control block in which the permanently resident bit or the reserved bit is one. If the serial numbers match, the routine passes control to the device strikeout routine to allocate the device. (That is, it places the address of the unit control block into the allocate volume table entry, Figure 15, and increases, by one, the count of allocated devices in the allocate work table entry that represents the data set, Figure 16.)

Device Range Reduction

The device range reduction routine reduces the number of devices that can be allocated to satisfy certain requests. In addition, this routine allocates devices containing reserved tape volumes.

The device range reduction routine prevents allocation of devices that are ineligible to satisfy certain requests. Devices are ineligible under the following conditions:

- The device is the primary console.
- The device is off-line or is being changed to off-line status.
- The device has either been allocated or is resident, and the request is for an unspecified private volume. (Each such request requires an unused volume.)
- The device has either been allocated or is resident; the device contains a private volume; and the request is for temporary data set space on a volume that is neither specific nor private.
- The device is a resident, direct-access device, and the request is for a specific volume.
- The device is neither a direct-access device nor a tape device (unit record or graphic equipment, for example) and is allocated, unless one of the two following conditions exists:
  - The device is the system output device, and the request is for a SYSOUT data set.
  - The device is the system input device, and the request is for a SYSIN data set.
- The device does not contain a storage volume, and the request has all of the following characteristics:
  - The request is not for temporary data set space.
  - The request is not for a specific volume.
  - The request is not for a private volume.

A storage volume is a permanently resident or reserved volume that may be used to keep any data set specified in a DD statement in which KEEP has been specified.

To prevent allocation of these ineligible devices, the device range reduction routine alters primary bit patterns representing devices that are available for allocation. In each bit pattern, ones represent devices that can be allocated, and zeros represent those that can not. A primary bit pattern forms part of each allocate work table (AWT) entry. (Each entry stands for one request.) The device range reduction routine eliminates each device that is ineligible to satisfy a particular request by changing the bit corresponding to the device from a one to a zero in the bit pattern corresponding to the request. The final bit pattern thus represents only devices that can satisfy the request.

As each ineligible device is disqualified, a count of eligible devices in each affected allocate work table entry is reduced by one. If this count becomes less than the number of devices needed to satisfy the request represented by the entry, the device range reduction routine passes control to the allocation error recovery routine. If recovery is possible, this routine provides a list of devices that can satisfy the request. The operator may either reply with a three-character device name or cancel the job. (If allocation error recovery is necessary, the entire allocation procedure is repeated.)

If, during this processing, the device range reduction routine finds a unit control block representing a tape unit with a reserved volume mounted on it, it allocates the device if the volume was requested.

SYSIN Allocation

If the device range reduction routine encounters a request for the device designated as the system input device, it allocates that device.

Specific Device Allocation

Allocation is next made to requests for specific devices or requests which, because of range reduction or previous allocation, can be satisfied only by a specific device.

Exits From Demand Allocation

When all processing is completed in the demand allocation routine, all requests within the step may have been satisfied. If so, exit is made to the TIOT construction routine. If, however, some requests remain outstanding, control is passed to the automatic volume recognition routine if it was specified during system generation. If additional requests remain, control is passed to the decision allocation routine. When allocation is complete, the "number of unallocated entries" field in the allocate control block (ACB) reaches zero. If the number of devices required exceeds the number of devices available, control is passed to an allocation error routine.

AUTOMATIC VOLUME RECOGNITION

The automatic volume recognition (AVR) routine decreases the time required for job step initiation by enabling the operator to mount volumes needed for subsequent job steps as soon as devices become available. During subsequent job step initiation, the AVR routine recognizes that volumes needed for the current job step are mounted, thus saving the time that the system otherwise would spend waiting for the operator to find and mount them.

Before the next job step after a volume has been mounted, the AVR routine reads the volume label and associates the volume with the device containing it, using information from the label. When the volume is needed for a subsequent job step, the AVR routine can then identify and allocate the device on which it is mounted.

The AVR routine, which may be specified during system generation, allocates devices to satisfy requests that specify 2311 and 2314 direct-access volumes, 7-track tape volumes having a tape density specified during system generation, and 9-track tape volumes. These volumes must be specified by either a serial number or a data set name that implies a serial number. The order in which the AVR routine allocates these devices depends upon when the volumes are mounted.

The AVR routine (Chart 28) first allocates devices containing volumes mounted before the start of the last job step. It can do this immediately, since these volumes already have been associated by serial number with the devices containing them.

The AVR routine then allocates devices containing volumes mounted after the start of the preceding job step. It reads the labels of these volumes to obtain their serial numbers and allocates any of the devices containing volumes requested for the current job step.

Finally, the AVR routine attempts to satisfy any remaining requests for 2311 and 2314 direct-access volumes and 9-track tape volumes. The AVR routine attempts to

obtain devices to satisfy all requests either by selecting devices that are not being used or, if necessary, by unloading volumes that are not needed for the job step. If the AVR routine can obtain enough devices, it prints a list of the requested volume serial numbers and allocates the devices as the operator mounts the volumes. If enough devices are not available or if all of the needed volumes cannot be mounted, however, the operator must cancel the job.

Processing Requests for Previously Mounted Volumes

The AVR routine first satisfies requests for volumes that were mounted before the start of the last job step (Chart 29). The AVR routine can identify these volumes immediately, because the system already has associated the volumes with the devices containing them.

The AVR routine first determines which devices contain volumes that were mounted before the start of the last job step. The AVR routine searches for such volumes by examining all unit control blocks that represent on-line, ready 2311 and 2314 direct-access devices, 9-track tape devices, and 7-track tape devices. The AVR routine can identify previously mounted volumes, because each unit control block containing a volume serial number represents a device containing such a volume.

If the AVR routine finds such a volume, it next determines whether the volume is needed for the current job step. To make this determination, it searches in the volume table (VOLT) for the serial number of the mounted volume. (Each entry in this table represents a volume that has been specifically requested.) If the AVR routine locates the serial number, the volume is needed for the job step. The AVR routine then uses the device strikeout routine to allocate the device to satisfy all requests for the volume. If the serial number is not in the volume table entries for this job step, however, the volume is not presently needed. The AVR routine subsequently ignores the device and looks for another previously mounted volume.

Processing Requests for Newly Mounted Volumes

The AVR routine next allocates devices to satisfy requests for volumes mounted since the last job step was initiated (see Chart 30). The AVR routine determines the serial number of each of these newly mounted volumes and then allocates its device to fulfill any unsatisfied requests for the volume. If any request for a newly mounted volume is already satisfied or if any device containing such a volume is already allocated, the AVR routine determines whether the volume was improperly mounted during demand allocation. If so, it takes corrective action.

IDENTIFYING A NEWLY MOUNTED VOLUME: Because the control program could not have previously read the labels of newly mounted volumes, the AVR routine must identify each of these volumes to the system and associate the volume with its device.

The AVR routine identifies these volumes by searching for a serial number in each unit control block that represents an on-line, ready device. If the serial number is zero, the volume was mounted after the start of the last job step.

To associate the device with the volume serial number, the AVR routine reads the volume label into main storage, extracts the serial number from the label, and records it in the unit control block representing the device. (To extract the serial number from a nonstandard label, the AVR routine uses a volume serial number routine, IEFXVNSL, which must be supplied by the user. A routine with the same name is supplied by IBM to indicate an error if the user has provided a nonstandard label but has not substituted his own routine to read it.)

The AVR routine uses the external action routine to unload the device, if no serial number can be found, because the volume cannot be identified.

If the device already has been allocated but the volume was mounted on the wrong device during the demand allocation procedure, the AVR routine notifies the operator and takes corrective action.

ALLOCATING A DEVICE: The AVR routine determines whether it should allocate the device by searching for a request for the mounted volume (Chart 30). To determine whether the volume was requested, the AVR routine searches for the serial number in the volume table entries for the job step, as before. If the serial number can be found, the device containing the volume can be allocated. The request must not have been previously satisfied, however. If the serial number is not found in the volume table, the volume is not needed for the job step, and the AVR routine searches for another newly mounted volume.

To determine whether the request is unsatisfied, so that it can allocate a device, the AVR routine searches for the address of a unit control block in the allocate volume table entry for the request (see Figures 15 and 45). If the AVR

routine finds no unit control block address, the request was not previously satisfied. The AVR routine then uses the device strikeout routine to allocate the device and to satisfy any other requests for the same volume.

If the volume request was already satisfied, but the volume was mounted on the wrong device during the demand allocation procedure, the AVR routine takes action to correct the error.

Processing Requests for Unmounted Volumes

The AVR routine finally attempts to satisfy all remaining specific volume requests. For these requests to be satisfied, enough devices for all of the requests either must be available or must be made available. If enough devices become available, the AVR routine provides the operator with a list of volumes to mount and allocates the devices as he mounts the volumes on them. If sufficient devices for the job step cannot be made available or if all of the required volumes cannot be mounted, the operator must cancel the job.

OBTAINING DEVICES: Before the AVR routine requests that the operator mount any unmounted volumes, it determines whether enough devices to contain them are available (see Chart 31). If there are not enough devices without mounted volumes to begin with, the AVR routine determines whether it can unload enough devices. The devices it considers for unloading contain mounted volumes not needed for the job step. If it can, it unloads these devices so that the operator can replace the mounted volumes with volumes needed for the job step. Otherwise, the AVR routine attempts to have enough off-line devices placed into on-line status to satisfy the remaining specific requests.

To determine whether there are enough devices, the AVR routine compares, by device type, a count of available devices with a count of needed devices. Because the need for each device type is filled separately, a shortage of any one type means that not enough devices are available for the job step.

The available devices comprise all on-line 9-track tape units, 2311 disk units, and 2314 disk units that have not been allocated. Separate counts are made of devices not in the ready status (which normally do not contain mounted volumes) and devices that are ready (all of which have mounted volumes).

To eliminate any unnecessary unloading of devices, the AVR routine compares, first, the number of devices needed with the number of on-line devices not having mounted volumes (that is, those that are not in the ready status). If there are enough such devices, none need be unloaded, and the AVR routine can immediately print a list of volumes to be mounted.

If ready devices must be unloaded, the AVR routine determines the number of ready devices still needed and whether enough can be unloaded.

If the AVR routine has determined that enough ready devices can be unloaded, it stores the identities of a sufficient number of devices and then unloads them. To fill the quota, it first tries to obtain enough ready devices not containing retained volumes or volumes with data sets. If the AVR routine cannot find enough devices, it obtains the remainder needed from among devices containing these kinds of volumes. The AVR routine unloads the devices with the external action routine, which also prints a list of unit addresses so that the operator will know which devices have volumes to be dismounted. The AVR routine then provides the operator with a list of the serial numbers of volumes to mount.

In an attempt to make more devices available, if it is apparent that enough ready devices cannot be unloaded, the AVR routine uses the allocation error recovery routine (IEFXJIMP) to print a list of off-line devices that can be made available. The operator either may reply with a three-character device name to place each device into on-line status or cancel the job. (If allocation error recovery is necessary, the entire allocation procedure is repeated.)

ALLOCATING DEVICES ON WHICH VOLUMES HAVE BEEN MOUNTED: When the AVR routine has determined that the required number of devices is available for allocation, it provides the operator with a list of serial numbers of the needed volumes (Chart 28). As the operator mounts these volumes, the AVR routine allocates the corresponding devices to satisfy requests for these volumes.

After printing the list, the AVR routine waits for the operator to mount a volume. A device-end I/O interruption releases the AVR routine from its waiting status when the operator mounts the first volume and presses the START button on the device. The AVR routine extracts the new serial number from the volume label (Chart 30), removes the serial number from the list of required volumes, and allocates the device. Then the AVR routine waits for the operator either to mount the next volume or to

cancel the job. It repeats the procedure until either all specific volume requests have been satisfied or the job is canceled.

When the devices have been allocated, the AVR routine passes control to the TIOT construction routine, unless there are more volume requests. If there are, the AVR routine passes control to the decision allocation routine, which satisfies the remaining requests.

DECISION ALLOCATION ROUTINE

The decision allocation routine (Chart 32) allocates devices to most data sets for which devices have not yet been allocated by either the demand allocation or the automatic volume recognition routine. This includes all remaining requests except requests for space on unspecified public or unspecified storage volumes. The latter requests are fulfilled by the space request routine.

Upon entry to the decision allocation routine, an attempt is made to reduce the number of devices that are candidates for allocation. A request for unit or channel separation from devices allocated by either the demand allocation or automatic volume recognition routines eliminates the units or additional devices on the selected channels from further consideration. If this is the case, the separation strikeout subroutine is entered. This subroutine, by changing corresponding bits in the primary bit pattern, eliminates these devices from consideration for allocation.

The number of data sets directed to each channel is then determined and added to the totals in the channel load table (see Figure 18). This table is later used to "spread the load" across the channels, thereby:

- Obtaining maximum overlap of I/O activity.
- Reducing the possibility of making a channel ineligible because all of its devices had been allocated too early. (Some channel separation requests would then be impossible to satisfy.)

The maximum number of data sets that could use each device is next determined and placed into the potential user on device table (see Figure 19). This table is later used to determine the order in which devices will be selected for data sets. (Devices first selected are those with the fewest potential users.)

No. of data sets for first device		No. of data sets for nth device

Figure 19. Potential User on Device Table

The remainder of the decision allocation routine allocates devices. First, devices are allocated to data sets for which only one device is eligible. Then all other requests (except those for unspecified public or unspecified storage volumes) are processed in the following manner. A data set is selected and then a device for the data set is selected and allocated to it. Another data set is then processed.

Data Set Selection

Data sets are selected by considering the number of devices eligible for allocation to them. That is, the first data set selected is the one for which the smallest number of devices is eligible.

The decision allocation routine selects two kinds of requests, both of which must be satisfied with the allocation of devices containing nonshareable volumes:

- Requests for nonshareable volumes. (Each such request has a nonshareable flag in its allocate work table entry, shown in Figure 16.)
- Requests that may be satisfied with the allocation of either a direct- or sequential-access device, if sequential-access devices are available for them. (As each of these requests is satisfied, a nonshareable flag is placed into its allocate work table entry to mark the allocation of a device containing a nonshareable volume.)

Selection is performed by scanning the allocate work table. If two or more data sets have the same number of eligible devices, they are selected in the following order:

1. Data sets with separation requests.
2. Data sets with affinity requests.
3. Passed data sets.
4. All others.

Device Selection

When a data set has been selected, a device is selected and allocated for it. Devices are considered in the following order:

1. If the possible devices for a data set exist on more than one channel, the channel with the greatest number of

free devices of the type requested is chosen.

2. If two channels have the same number of free devices of the requested type, the channel with the lightest load is chosen; the device which has the fewest possible users is chosen.

3. To satisfy requests for public nonspecific (scratch) tape volumes, devices with mounted tape volumes are given preference. To satisfy requests for direct-access volumes and specific tape volumes (including private volumes and volumes which are used for multi-volume public data sets), devices without mounted volumes are given preference.

4. If two devices have the same number of possible users, the first one in the I/O supervisor UCB lookup table is chosen.

Device Allocation

As indicated previously, the decision allocation routine selects a data set and an eligible device, allocates the device, and then selects another data set. To allocate a device, the decision allocation routine places the address of the unit control block representing the device into the allocate volume table entry (Figure 15) representing the required volume and adds one to the "number of devices allocated" field of the allocate work table entry for the data set (Figure 16).

While a request is being satisfied, the same device is also allocated to satisfy any other requests that specify the same volume. Multiple allocations may be performed in this case, because all requests for the same volume appear in a volume affinity chain, which is a series of linked allocate volume table entries (Figure 15). The decision allocation routine satisfies, in the same way, requests that specify unit affinity or that have a split or suballocate relationship (Figure 16).

When a device is allocated, the decision allocation routine alters bit patterns in the allocate work table entries for certain other requests. Each bit pattern specifies the devices that are eligible to contain the data set represented by the allocate work table entry.

If a private volume request was satisfied, the decision allocation routine changes the bit representing the allocated device to zero in all primary and secondary bit patterns so that the device cannot be selected to satisfy another request. Such devices are exempted from further allocation because each private volume may not contain other data sets and must be removed after use.

If the request was satisfied with a device containing a nonshareable volume, the decision allocation routine changes the bit representing the device to zero in the primary and secondary bit patterns of the allocate work table entries that represent all other data sets that require nonshareable volumes. A device allocated to satisfy a request for a nonshareable volume thus cannot satisfy additional requests of this kind.

If all eligible devices are allocated before all data sets for a step have been selected for allocation, the decision allocation routine passes control to an allocation error routine.

Upon successful completion of processing by the decision allocation routine, exit is made to the TIOT construction routine.

TIOT CONSTRUCTION ROUTINE

The task input/output table (TIOT) construction routine (Chart 29) obtains space for and builds the processing program's task input/output table. The primary function of the TIOT is to provide the data management open, close, and end-of-volume (EOV) routines with pointers to JFCBs and allocated devices.

Entry to the TIOT construction routine is made when all requests for I/O devices have been satisfied except requests for unspecified public or unspecified storage volumes. Therefore, entry may be from the demand allocation routine, the automatic volume recognition routine, or the decision allocation routine. Exit is to the external action routine.

Upon entry, main storage space required to build the TIOT is calculated using the first formula shown in Figure 20, and space is requested. The standard TIOT is shown in Figure 21. TIOT entries are constructed for each data set in a step. Entries are also constructed when use of the job library is requested or when a program, created in a previous step, is to be executed as the current step. Figure 22 shows the sources of entries in the TIOT.

The TIOT construction routine determines, for each request for an unspecified storage or unspecified public volume, which devices are eligible to be allocated by the space request routine. It obtains this information from the allocate work table

entry (Figure 16) for the request, which contains a primary bit pattern representing the devices that are eligible to satisfy the request.

Space required to build TIOT = $28 + 16N_1 + 4N_2 + 12N_3 + 4N + 4(N \times N)$
Space occupied by completed TIOT = $28 + 16N_1 + 4N_2 + 12N_3 + 4N$
where:
N_1 = Number of DD statements.
N_2 = Number of devices allocated to the step.
N_3 = Number of pools of devices.
N = Number of slots for all pools entries in the step.
N = Number of requests for public volumes.
N = Number of devices available for public volumes.

Figure 20. Formulas for Determining Task Input/Output Table Space Requirements

The TIOT construction routine places pointers to all unit control blocks representing eligible devices into the TIOT entry for each such request. If more than one device can satisfy a request, it selects, first, the channel with the lightest load, and, on this channel, the device that has been allocated to satisfy the smallest number of requests. When the first device has been selected, it places other devices in order, using the following criteria:

1. Devices on the same channel as the first device selected, but which do not contain passed data sets.
2. Devices that do not contain passed data sets and do not violate requests for separation.
3. Devices that contain passed data sets and do not violate separation requests.
4. Devices that do not contain passed data sets and violate separation requests.
5. All other devices eligible to receive public volumes.

Should more than one device have similar attributes, their pointers are arranged in the order in which the devices are represented in the primary bit pattern.

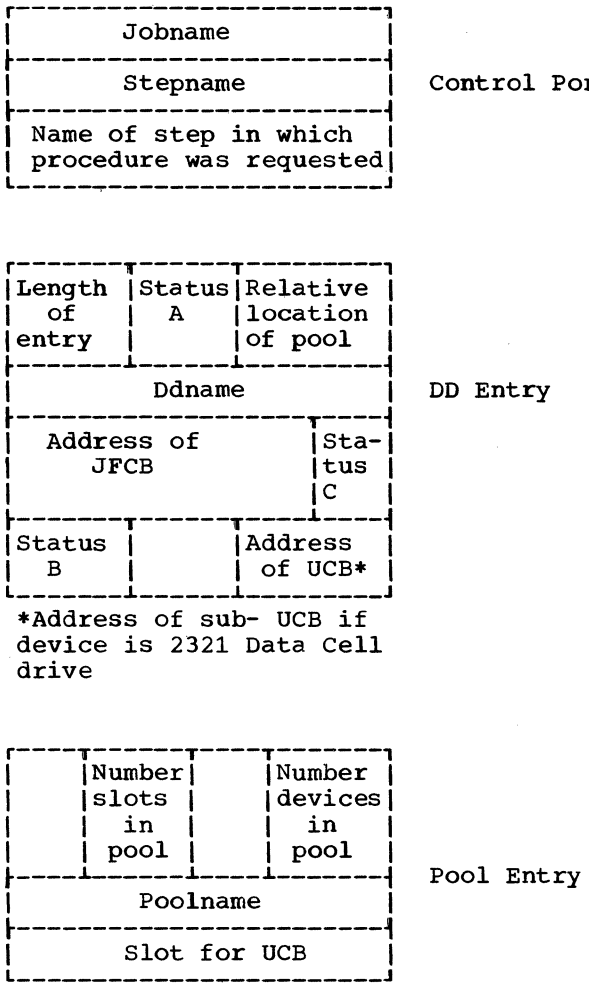


Figure 21. Task Input/Output Table

Entry	Source
Jobname	JCT
Stepname	SCT
Stepname of step in which procedure was requested	SCT
Length of entry	Calculated
Status A	Calculated
Relative location of pool	Calculated
Ddname	SIOT
Address of JFCB	SIOT
Status C	Calculated
Status B	Calculated
Address of UCB	I/O supervisor UCB Lookup Table
No. of slots in pool	Calculated
No. of devices in pool	SIOT
Poolname	SIOT
Slot for UCB	I/O supervisor UCB Lookup Table

Figure 22. Task Input/Output Table Entry Sources

EXTERNAL ACTION ROUTINE

The external action routine (Chart 34) issues mounting instructions, verifies that the correct volumes have been mounted, and unloads incorrectly mounted volumes.

Entry to the external action routine is made from the TIOT construction routine. Exit is made to the space request routine.

Upon entry, devices allocated to each data set are checked and any required dismounting is requested. (The operator is notified of volume dispositions.) Messages instructing the operator to mount the required volumes are then issued, and checks are made to ensure that volumes were mounted on the correct units.

SPACE REQUEST ROUTINE

The space request routine (Chart 35) processes requests for space on direct-access volumes. It determines whether a volume has enough space for the data set specified in a particular request, and, if so, it obtains space on the volume for the data set. If space is not available initially, the space request routine attempts to locate another volume with sufficient space.

The space request routine, which receives control from the external action routine, searches among the task input/output table (TIOT) entries for requests for direct-access volume space. It processes these requests in two different ways, depending on whether or not a device was previously allocated to satisfy the request.

Obtaining Space If a Device Was Allocated

If a device has been allocated to satisfy the request (because a specific device or volume was named), the space request routine attempts to obtain space on the volume that is mounted on the device. It passes control to the direct-access device space management (DADSM) routines, which record the limits of an extent on the volume into a data set control block (DSCB) if space is available. If the mounted volume does not have space for the data set, and is not being used to contain another data set for the job step, the space request routine passes control to the external action routine, which directs the operator to mount another volume on the allocated device.

Obtaining Space If a Device Was Not Allocated

If a device has not been allocated to satisfy the request, the space request routine attempts to obtain space on an unspecified public or unspecified storage volume, depending on the type of request. (Either unspecified public or unspecified storage volumes can contain temporary data sets, but only storage volumes are eligible to contain data sets that are to be kept.) If the space request routine determines that a volume has space for a data set, it allocates the device containing the volume.

The space request routine first attempts to obtain space for the data set on a volume that is mounted on an eligible device. (The devices that are eligible to satisfy a particular request are indicated in the task input/output table entry for the request. Each entry contains pointers to the unit control blocks representing eligible devices.) To determine whether

space is available, the space request routine passes control to the direct-access device space management (DADSM) routines, which attempt to specify an extent on the volume, as indicated previously. If space is not available on the first volume checked, the space request routine determines whether the volume is unused and is removable (that is, not reserved or permanently resident). If so, it gives control to the external action routine, which directs the operator to mount another volume on the same device. If the volume is being used for other data sets for the job step or if it is not removable, the space request routine attempts to obtain space on another mounted volume.

If no volumes can be dismounted, the space request routine passes control to the external action routine. The external action routine requests the operator to mount a volume on an eligible device that does not contain a volume. If the operator mounts a volume, the space request routine allocates the device.

If no eligible devices are free to have volumes mounted on them, however, the space request routine determines whether any unused, reserved volumes are mounted on eligible devices. If any are, the space request routine prints out a list of unit addresses. The operator may either cancel the job if a reserved volume cannot be dismounted or reply with one of the listed unit addresses. In the latter case, the external action routine issues a dismounting message that indicates the reserved volume and a mounting message for a new volume to replace it. (A volume thus mounted assumes both the volume and use attributes of the dismounted volume, or, in other words, the reserved attribute and either the public or storage attribute. These attributes are described in Job Control Language, Form C28-6539.) If all reserved volumes on eligible devices are being used, the job is cancelled, because space for the data set cannot be obtained.

There are three exits from this routine:

- Exit 1 is taken when all requests for space were satisfied. Exit is to the step initiation routine of the initiator/terminator.
- Exit 2 is taken when space was not available on a volume that was requested. Exit is to the external action routine, if the request could be satisfied by having another volume mounted.
- Exit 3 is taken when space could not be obtained on any direct-access devices. Exit is to an allocation error routine, which passes control to the termination function of the initiator. The step is canceled, and subsequent steps of the job are interpreted, but are not initiated.

ALLOCATION ERROR ROUTINES

Allocation error routines are entered when error conditions are encountered by allocation and setup routines. There are two error routines: the recovery routine and the nonrecovery routine.

The recovery routine is entered if an error condition is detected before a TIOT is built for the step. It may be entered from the demand allocation, automatic volume recognition, decision allocation, or TIOT construction routine. If allocation requirements can be satisfied by changing the status of a device from off-line to on-line (determined by checking the secondary bit pattern), the recovery routine issues a message to the operator requesting him to place additional devices on-line. If he does, allocation for the step is begun anew by entry to the allocation control routine. If the operator does not or cannot add devices to the configuration, the recovery routine cancels the job.

The nonrecovery routine is entered when an error condition is detected after the TIOT has been built for the step. It passes control to the step termination portion of the initiator/terminator.

STEP INITIATION

The step initiation routine of the initiator/terminator (Chart 36) makes preparations for passing control to the processing program. If a JOBLIB DD statement is included in the job, the job library data set is opened. If the program to be executed exists on a data set created in a previous step, a DCB is created for that data set and is opened. Also, several tables are stored, releasing to the processing program the space they occupied. Step initiation passes control to the processing program.

The step initiation routine is entered from the space request routine. Upon entry, control is passed to the pseudo-sysout subroutine, which writes the contents of system message blocks (SMBs) onto the system output data set.

When control returns from the pseudo-sysout subroutine, the step initiation routine inserts the address of the UCB for the device containing the system output data set into each TIOT entry that indicates a SYSOUT disposition. The LCT and

JCT are then stored and the space that they occupied is released.

Main storage space to be used by the processing program is then obtained. A portion of this area is reserved for the following:

- Job library DCB (if any).
- Fetch DCB (if any).
- Macro-parameter list.
- TIOT.
- Processing program register save area.

First, the TIOT is moved from the initiator/terminator work area to the area of processing program storage assigned to it. The TIOT is also stored, and the space it occupied is released. The macro parameter list (see Figure 23) is then built and the programname entry and initializing parameter values entry (PARM information) are inserted. The SCT is then stored, and the space it occupied is released. If a job library has been requested for the job, the job library data set is opened, and the address of its DCB is placed into the TCB. If a fetch DCB is required (PGM=*.stepname.ddname was specified in the EXEC statement) a DCB is created and opened, and its address is placed into the macro parameter list.

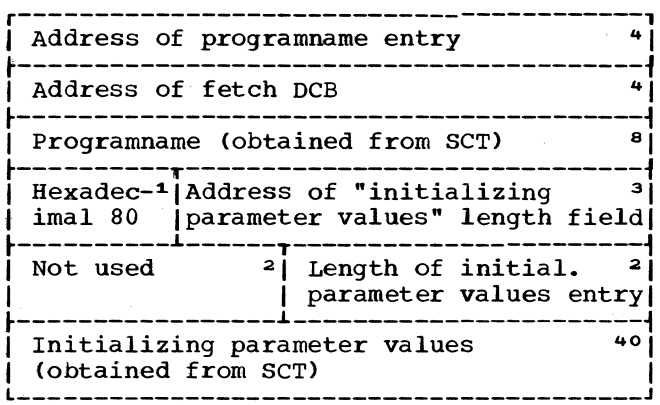


Figure 23. Macro Parameter List

The cancel ECB in the selected job queue[1] is then set up for the processing program: i.e., the low-order byte is changed to the number 255. If a CANCEL command was issued, the step initiation routine issues the ABEND macro-instruction.

---

[1]Just prior to passing control to the job step, the low-order byte of the cancel ECB in the selected job queue is changed to all ones. This causes issuance of an ABEND or ABTERM rather than a POST by the master scheduler if the operator issues a CANCEL command for the job.

If a CANCEL command was not issued, an XCTL macro-instruction is used to pass control to the processing program.

TERMINATION

The termination function of the initiator/terminator (Chart 37) performs post-step and post-job housekeeping. It is normally given control following step execution, but is also given control when a job management routine encounters an irrecoverable error while processing a job step. Termination routines:

- Release space occupied by tables.
- Free I/O devices.
- Dispose of data sets referred to or created during execution.

Major components of termination are:

- The step termination routine, which performs post-step housekeeping functions.
- The job termination routine, which performs post-job housekeeping functions.

The disposition and unallocation subroutine is used by both the step and job termination routines. Basically, this subroutine handles disposition of data sets and frees devices allocated to a step. The disposition and unallocation subroutine is described in Appendix A.

STEP TERMINATION ROUTINE

The step termination routine (Chart 38) performs its functions when a step has been terminated either normally due to successful completion of execution or abnormally due to an error condition. It uses five major routines:

- Step termination control routine.
- Step termination data set driver routine.
- Job statement condition code routine.
- Disposition and unallocation subroutine.
- User's accounting routine (if included in the configuration).

Upon successful execution of a step or abnormal termination of execution, control is passed from the supervisor to the step termination control routine. In addition, when a job management routine encounters an

irrecoverable error, it immediately passes control to the step termination control routine.

First, the initiator/terminator TIOT and the LCT are placed into a main storage work area. Next, the cancel ECB in the selected job queue is set to zero. The JCT and the SCT are then placed into a main storage work area (if they are not in this area at the time), and a step status code is inserted into the SCT.

The step data set driver routine is then entered. It places the SIOT for each data set into a main storage work area and branches to the disposition and unallocation subroutine. The loop through the data set driver routine and the disposition and unallocation subroutine is then repeated for each SIOT.

When all data sets have been processed by the disposition and unallocation subroutine, the updated SCT is stored. Control is then passed to the job statement condition code routine, unless it is known that there are no further steps for the job (the reader/interpreter had encountered a JOB or null statement). In the latter case the job statement condition code routine is bypassed.

The job statement condition code routine (Chart 39) processes condition codes specified in the JOB statement.

If, upon entry, it is found that there were no condition codes specified in the JOB statement, control is returned to the step termination routine. Each condition code in the JCT for the job is in turn compared with the step completion code of the previous step, which appears in its SCT. Up to eight conditions are checked by this routine for each step. Any additional condition codes are ignored. If any of the condition operators are satisfied by the codes, the job-failed indicator in the JCT is updated to indicate that the job failed, the message subroutine is used to issue a message to the programmer, and control is returned to the step termination routine.

Upon return from the job statement condition code routine, or if it had been bypassed, exit is made to the user's accounting routine, if one is present. On return from the accounting routine, or if there was none, control is passed to:

- The job termination routine, if the current step is known to be the last step of the job.
- The initiator/terminator system control routine, if additional steps have been interpreted and are ready to be initiated.
- The reader/interpreter control routine, which resumes processing the input job stream.

JOB TERMINATION ROUTINE

The job termination routine (Chart 40) performs its functions when an entire job has been executed and step termination for its last step has been completed. It consists of four major routines:

- Job termination control routine.
- Release job queue routine.
- Disposition and unallocation subroutine.
- User's accounting routine (if included in the configuration).

Control is passed to the job termination control routine from the step termination routine.

The job termination control routine determines if a passed data set queue exists and, if so, places each block into main storage work area and tests for unreceived data sets. (An unreceived data set is a passed data set to which no reference is made after PASS is specified.) When an unreceived data set is found, entry is made to the disposition and unallocation subroutine. When all unreceived data sets have been processed, or if no passed data set queue exists, the job termination control routine passes control to the accounting routine, if there is one.

When the accounting routine returns, or if there is none, the completed job's control tables are removed from the system by the release job queue routine. This routine releases the auxiliary storage space (or, if the resident job queue option was selected during system generation, the main storage space) occupied by all control tables for the job. If the job notification switch is on, the message

IEF402I jobname ENDED

is written on the console device. Control is then passed to the reader/interpreter control routine.

TABLE STORE SUBROUTINE

The table store subroutine stores records into and retrieves records from the SYS1.SYSJOBQE data set. This data set may be either completely on a resident direct-access device, or partly in main storage and partly on such a device, depending on whether the resident job queue (RESJQ) option was specified during system generation. The table store subroutine provides the following services on request:

- Supplies the requester with an auxiliary storage address or addresses into which records may later be written.
- Writes a record (or records) onto SYS1.SYSJOBQE locations specified by the requester.
- Reads a record (or records) from SYS1.SYSJOBQE locations specified by the requester .

The table store subroutine is used by job management routines to temporarily store tables and work areas that need to be communicated from one routine to another.

As part of the preparation for system generation (initializing system data sets), a specified number of tracks is assigned to data set SYS1.SYSJOBQE. During IPL, this extent is formatted for 176-byte records. (All records handled by the table store subroutine are 176-byte records.)

If the resident job queue option was selected during system generation, a specified number of records, starting at the beginning of the data set, will occupy a main storage area, thus saving time when tables are to be stored or retrieved. If there is room within this area of main storage, the I/O supervisor causes the records to be moved in response to the table store subroutine's WRITE macro-instruction; if desired records are stored in this main storage area, the I/O supervisor causes them to be moved in response to a READ macro-instruction.

The calling routine may request one of five functions. These are:

- Assign and start. The requested number of track addresses are assigned, beginning with the first assignable address in the extent.
- Assign. The requested number of track addresses are assigned, beginning with the first available address in the extent.
- Write and assign. The requested number of records are written, and the requested number of addresses are assigned.
- Write. The requested number of records are written.
- Read. The requested number of records are read.

Before passing control to the table store subroutine, calling routines must construct a parameter area (see Figure 24) and place its address into general register 1. Calling routines must also provide a QMPCA-QMPEX list (see Figure 25). Figure 26 shows the parameters required when a function is requested. The parameters are:

- QMPOP. A function code that indicates the function to be performed.
- QMPCM. The number of records (maximum of 15) for which addresses are to be assigned.
- QMPNC. The number of records (maximum of 15) to be stored into or retrieved from SYS1.SYSJOBQE.
- QMPCL. The beginning address of the QMPCA-QMPEX list.
- QMPCA. The main storage address from which the record is to be read or into which the record is to be written.
- QMPEX. The record address (in SYSJOBQE) into which the record is to be written or from which the record is to be read.

An entry in the QMPCA-QMPEX list is required for each record when a read or write function is requested. For assign functions, the table store subroutine returns the assigned track addresses in these parameters. The first assigned record address is placed into QMPCA1, the second into QMPEX1, and the remaining record addresses into ...QMPCAn, QMPEXn.

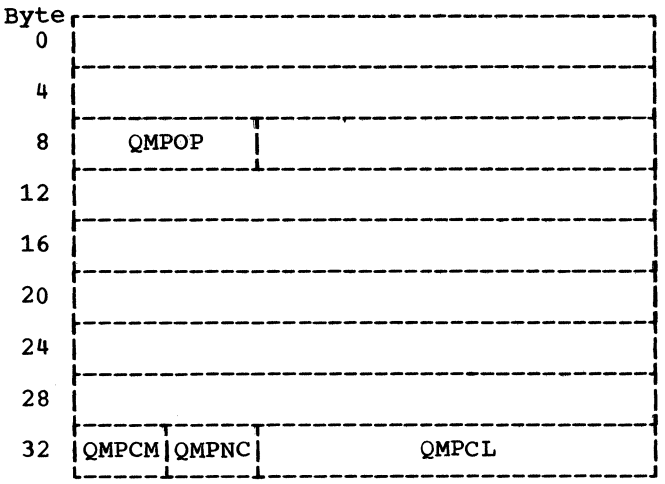


Figure 24. Table Store Subroutine Parameter Area

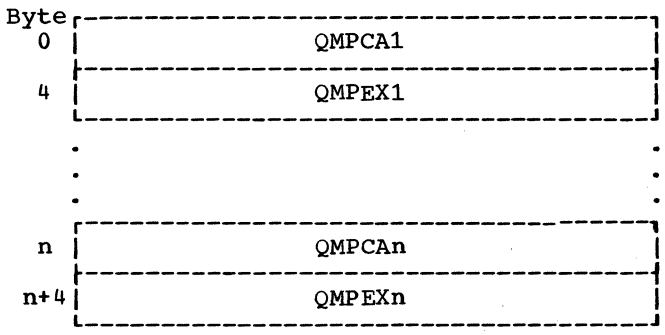


Figure 25. QMPCA-QMPEX List

	Input Parameters					
	QMPOP	QMPCM	QMPNC	QMPCL	QMPCA	QMPEX
Assign and start	00	X		X	X	X
Assign	01	X		X	X	X
Write and assign	02	X	X	X	X	X
Write	03		X	X	X	X
Read	04		X	X	X	X

Figure 26. Table Store Subroutine Parameter Requirements

DISPOSITION AND UNALLOCATION SUBROUTINE

The disposition and unallocation subroutine is divided into two sections: disposition processing, which performs data set dispositions specified in the DISP field of DD statements, and device availability processing, which makes the associated devices available for allocation to the next job step. Control enters the disposition and unallocation subroutine from the step termination routine and the job termination routine. In all cases, disposition processing is performed, followed by device availability processing. A message containing the data set name, its disposition, and the serial numbers of the volume (or volumes) in which it is contained, is always issued to the programmer.

ENTRY FROM THE STEP TERMINATION ROUTINE

When the step termination routine passes control to the disposition and unallocation subroutine (Chart 37), it provides pointers to the TIOT and SIOT of a data set. The disposition field of the SIOT indicates the disposition to be performed.

Disposition Processing

Dispositions that may have been specified in the DD statement are DELETE, KEEP, PASS, CATLG, and UNCATLG.

If the disposition is DELETE and the data set is cataloged, and if the JFCB housekeeping routine obtained volume information from the catalog, the UNCATALOG macro-instruction is issued. If the devices containing the data set are not direct-access devices, no SCRATCH macro-instruction is issued. If the devices are direct-access devices, a check is made to determine if the SCRATCH macro-instruction can be issued. It can be issued if one of the following conditions exists:

- All volumes containing the data set are mounted.
- All volumes containing the data set are not mounted, but at least one dismountable volume is mounted.

If neither of these requirements is met, an error message is issued. 1-ity

If the disposition specified in the DD statement is KEEP, the disposition subroutine issues a message to the operator and passes control directly to device availability processing.

If the disposition is PASS, no message is issued to the operator. Control is passed to device availability processing.

If the disposition is CATLG, the disposition subroutine determines if the data set is already cataloged. If not, the CATALOG macro-instruction is issued. If it

is cataloged, a further check is made to determine whether its volume list was altered during execution of the job step. (The data management OPEN, CLOSE, or EOV routines may have altered the volume list.) If the volume list was altered, a RECATALOG macro-instruction is issued. If the volume list was not altered, control passes directly to device availability processing.

An UNCATLG disposition causes an UNCATALOG macro-instruction to be issued.

If a disposition is not specified in the DD statement, but if the SYSOUT keyword is specified, control returns directly to the step termination routine.

When neither a DISP nor a SYSOUT keyword is specified in the DD statement a check is made to determine if an entry for the data set exists in the passed data set queue (PDQ), and if so, the status indicator in that entry is checked. If the status is old (the data set was created by a previous step or job), a KEEP disposition is assumed. If the status is new, a DELETE disposition is assumed. If there is no entry for the data set in the PDQ, the status indicator in the step input/output table is examined, and as in the conditions for a PDQ entry, either a KEEP or DELETE disposition is assumed.

Device Availability Processing

After the disposition of a data set is determined and processed, the device availability portion of the disposition and unallocation subroutine is entered. First, a check is made to determine if the operator has issued a VARY or UNLOAD command. If so, the status of the device is changed, and a message indicating that the command was processed is issued to the operator.

When there are no pending VARY or UNLOAD commands or when these commands have been processed, tests are made to determine if any of the volumes containing the data set can be dismounted. Dismount messages are issued for any that can be dismounted. The following volumes are not dismountable:

- Public volumes.
- Volumes on system residence or RESERVED devices.
- Volumes on permanently resident devices.
- Volumes whose status is RETAINED.
- Volumes on system input or system output devices.
- Volumes containing data sets with PASS dispositions.

The addresses of appropriate UCBs are obtained from the TIOT, and the status of the devices used is changed to ALLOCATABLE. When device availability processing of a data set is completed, the disposition and unallocation subroutine returns control to the step termination routine.

ENTRY FROM THE JOB TERMINATION ROUTINE

When the job termination routine passes control to the disposition and unallocation subroutine (Chart 38), only two types of data sets remain to be processed:

- Data sets that were passed but were not received.
- Data sets contained on volumes that were retained but to which reference was never made.

Each time that the job termination routine passes control to the disposition and unallocation subroutine, it passes a pointer to an entry in the PDQ describing a data set that was passed but not received. Only two dispositions may exist when entry is made from the job termination routine -- DELETE and KEEP. If the data set existed before the job, a KEEP disposition is assigned, otherwise a DELETE disposition is assigned. These dispositions are processed in the same manner as when entry is from the step termination routine.

When the job termination routine has scanned all PDQ entries for a job, it enters the disposition and unallocation subroutine, but provides no pointer to a PDQ entry. The disposition and unallocation subroutine scans all UCBs and issues dismount messages for any dismountable volumes on devices whose UCB contains the current job identification. Control is then returned to the job termination routine.

APPENDIX B: TABLES AND WORK AREAS

This appendix contains descriptions and formats of major tables and work areas that are used by job management routines and that are not described in the body of this publication. Most table entries are self-explanatory. Those entries that require further explanation are described with each table. Tables are shown here four or eight bytes wide for convenience, but are not necessarily drawn to scale.

The length of each field of the tables is given in bytes in the upper right corner of the field, and each table is limited to a 176-byte length by convention. The tables are presented in the following alphabetical order:

Account control table
DD list table
DD major field table
DD name table
DD parameter list table
Device mask table
Dsname table
EXEC key field table
Generation data group (GDG) bias count table
Job control table
JOB keyword table
New reader or writer table
Passed data set queue
Reader/Interpreter TTR table
Step control table
Step input/output table
System message block
Volume table

Auxiliary storage addresses appearing in the tables are relative track addresses (TTRs), in relation to the beginning of the SYS1.SYSJOBQE data set, whether the table is stored into main storage or into auxiliary storage by the table store subroutine and the I/O supervisor. All TTRs are three bytes long and begin on a full-word boundary. The format of all storage addresses appearing in the following tables is:

Relative track address (2)	Relative record address (1)	Not used (1)

ACCOUNT CONTROL TABLE

The account control table (ACT), shown in Figure 27, contains accounting information obtained from JOB and EXEC statements. This information is made available to user accounting routines. One or more ACTs are created for each job. The job routine of the reader/interpreter creates one ACT for each JOB statement, and the execute routine creates an ACT whenever the accounting (ACCT) parameter with its subsequent information is specified on an EXEC statement. The "number of accounting fields" entry contains the number of elements of accounting information specified in the ACCT parameter of the EXEC statement, or in the first positional parameter of the JOB statement (see IBM System/360 Operating System: Job Control Language). ACTs are stored by the table store subroutine.

Storage address of ACT (3)	Table ID=01 (1)	Not used (4)	
Programmer's name if JOB ACT; blanks if step (EXEC) ACT (20)			
Not used (3)	No. of accounting fields (1)	Length of first accounting field (1)	First accounting field (Variable)
Other accounting fields (if any)	Length of Nth accounting field (1)	Nth accounting field (Variable)	

Figure 27. Account Control Table

DD LIST TABLE

The DD list table (DDLT), shown in Figure 28, contains ddnames specified in the DDNAME parameters of DD statements for one job step, and pointers to associated dummy JFCBs and dummy SIOTs. When a DDNAME parameter is encountered in the operand field of a DD statement, a dummy JFCB and dummy SIOT are created by the DD routine of the reader/interpreter, and the ddname and pointers to the dummy tables are entered in the DDLT. The DDLT may contain five unresolved ddnames at any one time. After a ddname is used in the name field of a DD statement, its information is deleted from the DDLT and the space is made available for a new table entry.

When a DD statement is encountered, the DD routine scans the DDLT to determine whether the ddname in the name field of that statement had been specified previously in a DDNAME parameter. If so, the routine processes the new DD statement parameters and places them into the corresponding JFCB and SIOT, which are no longer dummy tables. If the ddname was not specified in a DDNAME parameter of a previous statement, the parameters of the new statement are processed and a new JFCB and SIOT are formed by the reader/interpreter.

The DDLT is initialized for each job step, and the maximum size of the table is 82 bytes.

DD MAJOR FIELD TABLE

The DD major field table (DDMFT), an entry of which is shown in Figure 29, contains one entry for each keyword that defines a major field (e.g., DCB, DSNAME, UNIT) in a data definition (DD) statement. The DD routine of the reader/interpreter refers to the DDMFT to check the validity of keywords, and also to obtain pointers to an appropriate entry in the DD parameter list table (DDPLT), which contains entries that relate to the parameters that may follow the keyword.

DDNAME TABLE

The ddname table (DDNT), shown in Figure 30, is used by the DD routine of the reader/interpreter to resolve ddname references during creation of the JFCB and SIOT. Ddnames, references to previous DD statements, and dsname references are placed in the ddname table by the DD scan routine of the reader/interpreter. The pointers to ddname or dsname entries in the

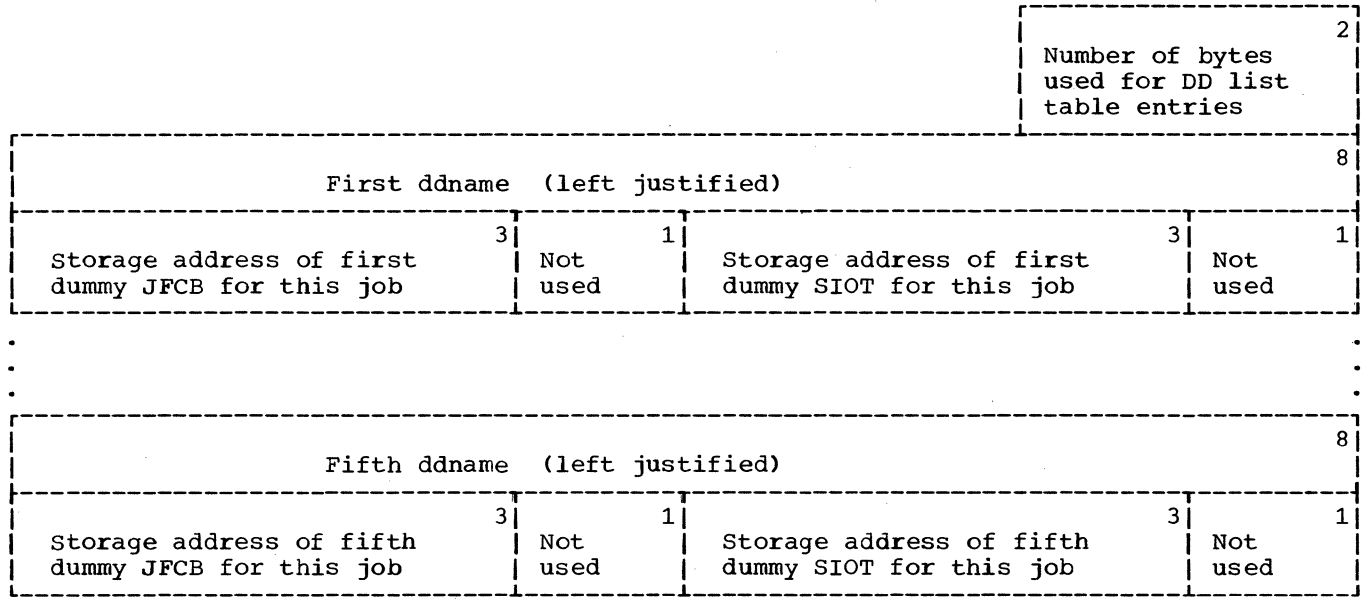


Figure 28. DD List Table

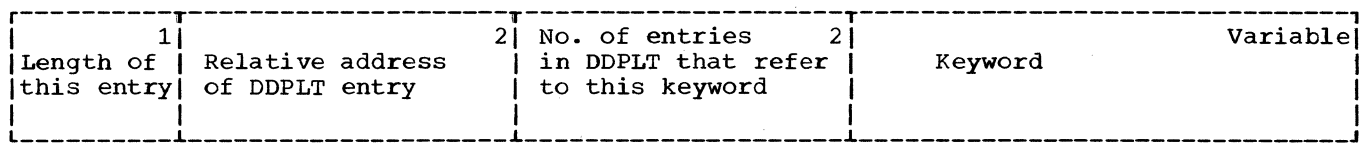


Figure 29. DD Major Field Table Entry

table are relative to byte 15, which is the "not used" portion following the 14-byte table header, which, except for the "number of bytes" count, consists entirely of pointers to the table entries. After the entire DD statement has been scanned, references entered in the ddname table are resolved to complete the JFCB and SIOT.

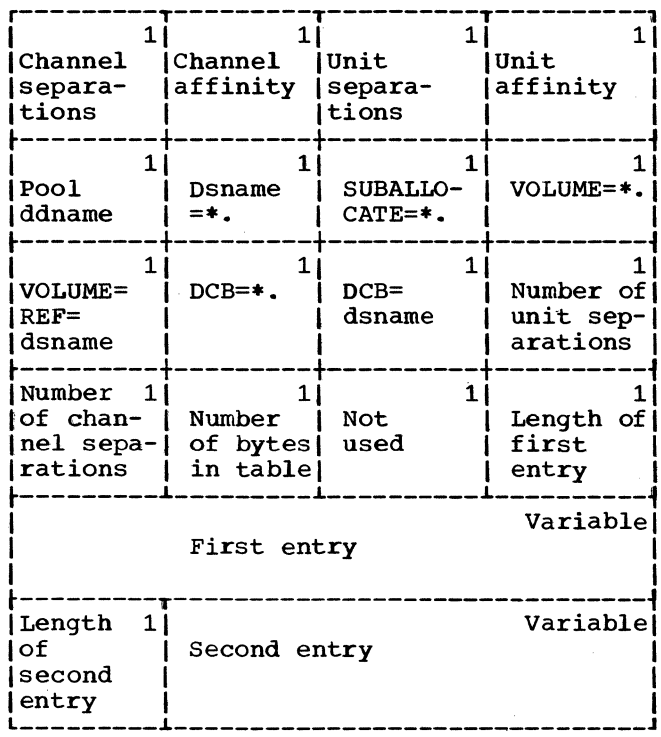


Figure 30. Ddname Table

DD PARAMETER LIST TABLE

The DD parameter list table (DDPLT), shown in Figure 31, contains one entry for each parameter that may appear in each DD statement field. The DD routine of the reader/interpreter refers to the DDMFT, which points to a block of DDPLT entries (corresponding to one DD major keyword), to determine which entries to make in the SIOT or the JFCB for DD statement parameters. Each DDPLT entry consists of a foundation and an extension, both of which are variable-length fields, and which contain attributes of the entry to be made in the dummy SIOT or dummy JFCB (e.g., length of the entry and its location in the SIOT or JFCB table). Because the dummy JFCB follows the dummy SIOT in storage, all locations in the dummy SIOT or JFCB are defined relative to the start of the dummy SIOT. The dummy tables are used to form the SIOT and JFCB by the DD output routine, after the DD scan routine.

The syntax type (key), contained in byte 3 of each entry may be one of 11 keys, corresponding to 11 different formats for the entry extensions. Each format depends upon the form of the information that appears in each DD statement field.

Within a DDPLT entry, the start of an extension is found by incrementing the foundation start location by the foundation length contained in the first byte. Information from the DD statement either is inserted directly into the table, or is converted into binary (CVB), or the bit configuration representing the value is ORed into the dummy JFCB or dummy SIOT.

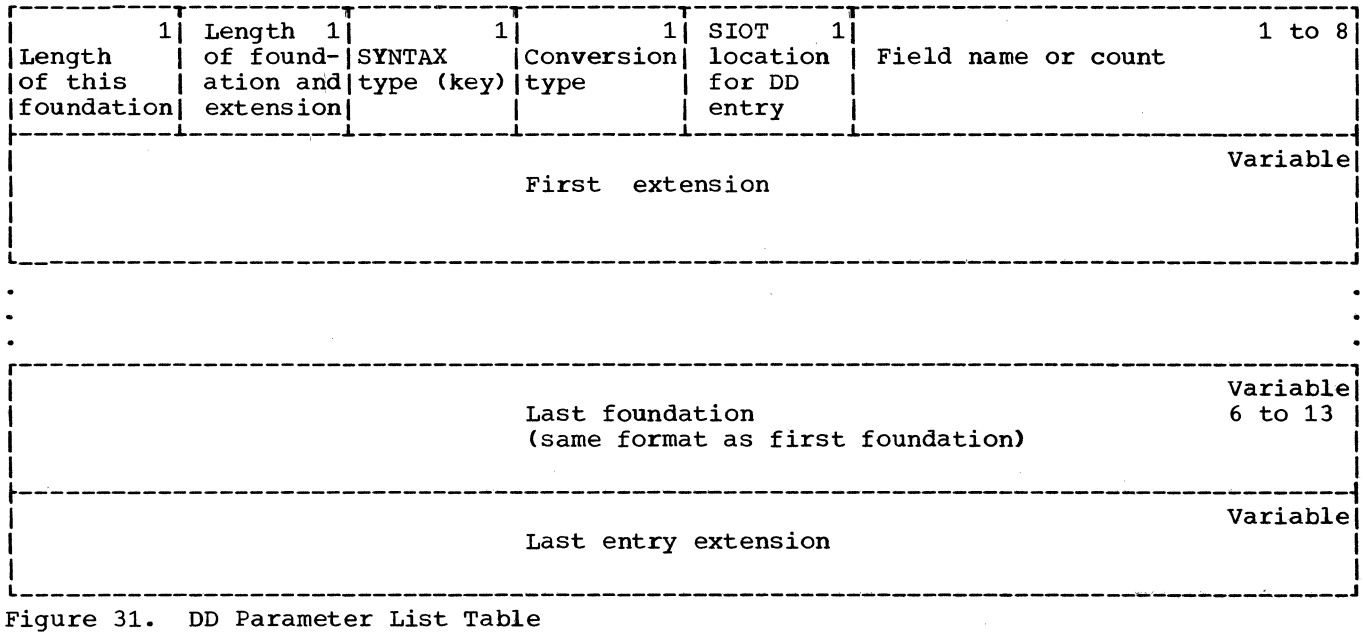


Figure 31. DD Parameter List Table

DEVICE MASK TABLE

The device mask table (DMT), shown in Figure 32, is built at SYSGEN time, and permits system access to the unique group of I/O devices represented by one unit name. This group may consist of any combination of device types or device numbers, and will be unique for any user's system. The user may determine specific device assignment bit patterns for his system from a symbolic listing taken after system generation. There is one table entry for each device. Within each entry, the bit pattern signifies the devices associated with a particular device name. The bit pattern within any entry is extended in full-word increments when the number of devices exceeds 32 or a multiple of 32. The entry status byte, bit 0, if 1, signifies that the group of devices is a homogeneous group.

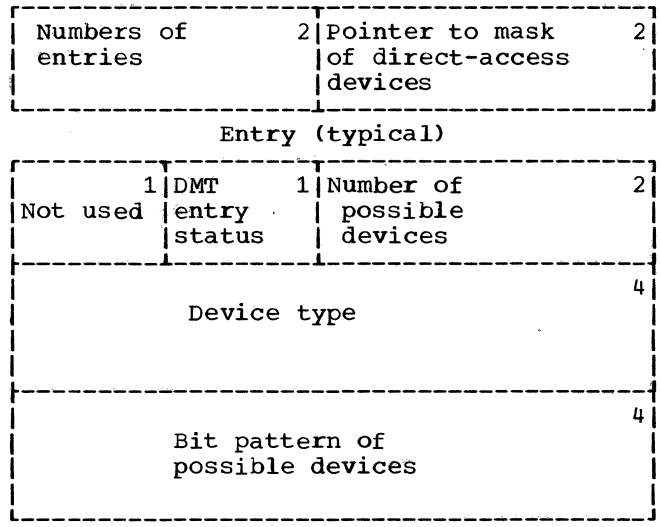


Figure 32. Device Mask Table

At SYSGEN time, device type codes are obtained from tables internal to the SYSGEN program, or are generated, and placed in the device mask table. The DMT is used as a source of device-type codes for the device name table (DNT) (see IBM System/360 Operating System: System Control Blocks). During device allocation, these codes are used as search keys to gain access to the DMT for device groups or single devices.

DSNAME TABLE

The dsname table, (see Figure 33), contains the volume reference data set names for one step as found in the DD statement. The table is created by the DD routine of the reader/interpreter for each job step. One entry is made in the dsname table for each DD statement containing the VOLUME=REF=dsname parameter.

The step control table (SCT) points to the dsname table, and also contains a count of the total bytes occupied in the dsname table by dsnames for the current step. The SIOT for each data set also contains a pointer to the dsname table entry for this SIOT before volume resolution and a pointer to the volume table (VOLT) after volume information has been resolved.

The dsname table is used by the JFCB housekeeping routine of the initiator/terminator to retrieve volume information concerning data sets referred to by data set name in the DD statement VOLUME=REF parameter. The dsname table is fragmented into 176-byte blocks before being stored, prior to job step execution.

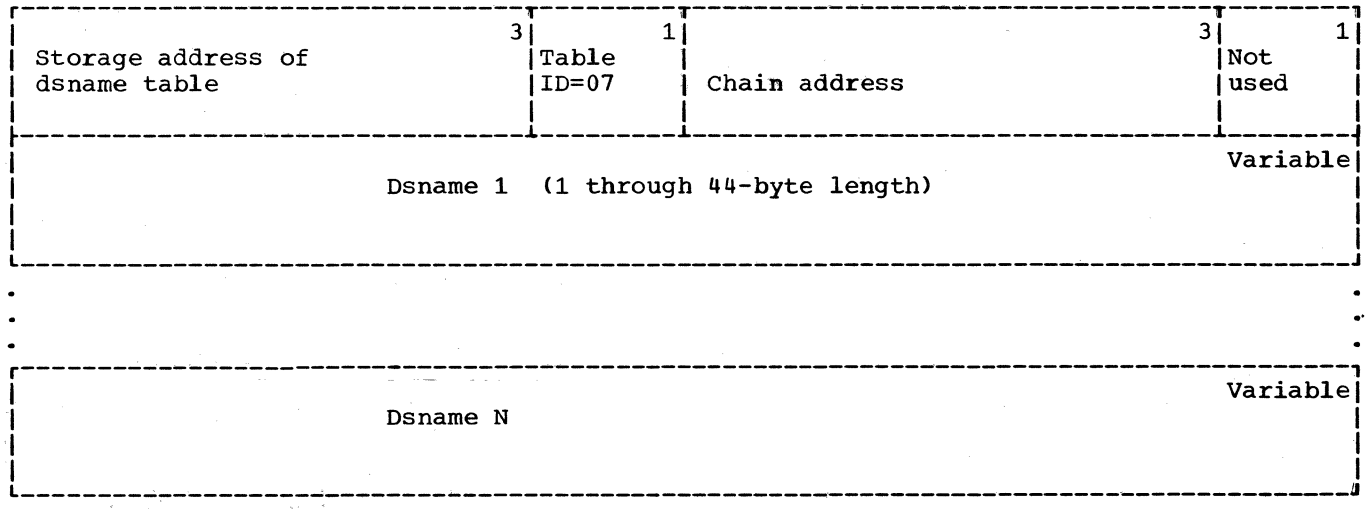


Figure 33. Dsname Table

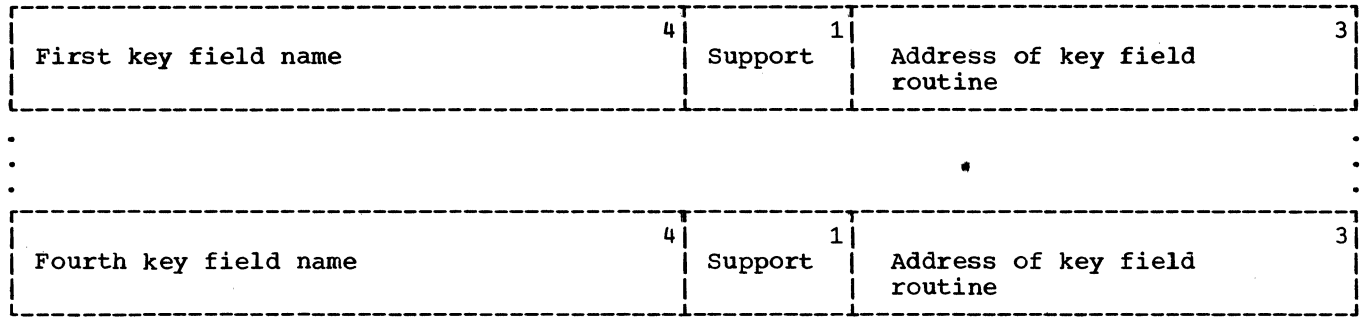


Figure 34. EXEC Key Field Table

EXEC KEY FIELD TABLE

Each entry in the EXEC Key Field Table, (see Figure 34), contains one of four EXEC statement key field names, an indication of whether the key field is supported (that is, processed and inserted in the appropriate table by means of a key field processing routine), and the address of the routine appropriate to the particular key field.

GENERATION DATA GROUP BIAS COUNT TABLE

The generation data group (GDG) bias count table, shown in Figure 35, makes GDG information available to the data management portion of the system, and allows the user to refer to a particular GDG member by the same number in different steps of the same job. The programmer refers to GDG members serially from the start of a job, but data management refers to GDG members serially from the last-cataloged member. The last member cataloged in a previous job, if any, is referenced as member number zero. The programmer will refer to the first new data set in the present job as number +1. This table is used to convert a reference that is relative to the start of the present job, as specified by the programmer, to a reference that is relative to the last-cataloged member, as required by data management.

created by the GDG single processing routine of JFCB housekeeping when a single GDG is requested by the user. When a step is completed by JFCB housekeeping, the JFCB housekeeping control routine transfers the GDG work bias byte to the GDG bias byte location if the value of the work byte is greater than that of the bias byte. In subsequent steps of the same job, any reference by the programmer to a GDG member will be decremented by the value of the bias count, which is contained in the GDG bias byte, to obtain a corrected member number for data management reference.

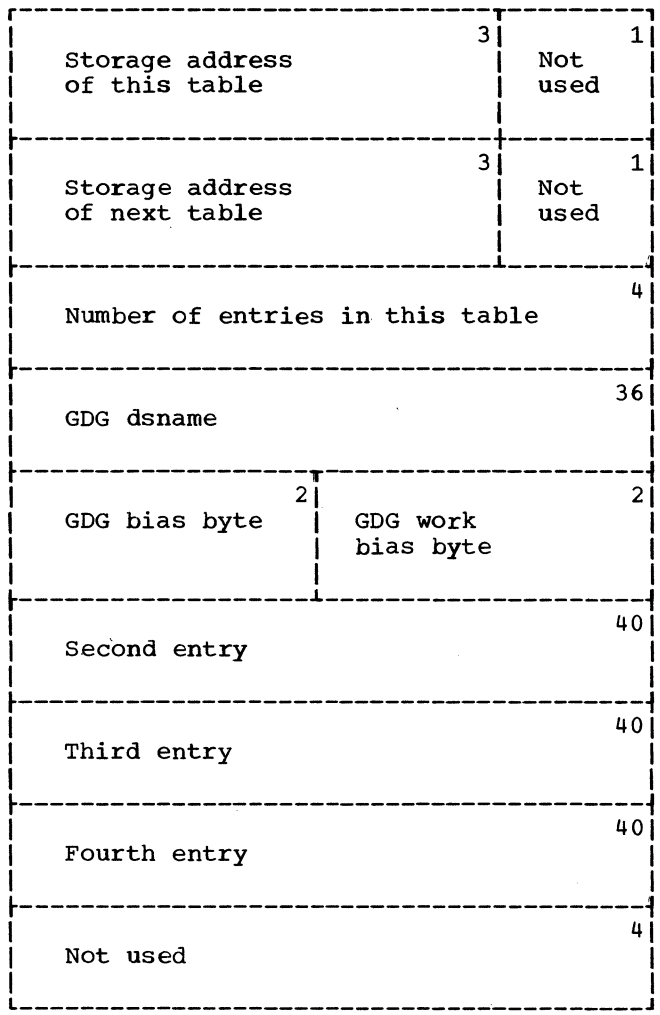


Figure 35. GDG Bias Count Table

JOB CONTROL TABLE

The job control table (JCT), shown in Figure 36, is created by the job routine of the reader/interpreter upon receipt of a job statement. It contains information taken from the job statement, and also storage addresses of major tables. After all steps within a job have been

interpreted, the JCT is stored by the reader/interpreter. The JCT is used by the initiator/terminator in preparing a job step for execution, and is stored by the step initiation routine of the initiator/terminator, before control is passed to the job step.

The JCT includes the following entries:

Job Serial Number: Always contains 1 in the primary control program.

Job Status Indicators:

Bit 0: The job library indicator contains a 1 if a JOBLIB DD statement is included with the job.

Bit 5: The job-failed indicator contains a 1 if an error condition caused the job to be terminated.

Message Level: Bits 0-3 contain zeros for message level 0; bit 3 contains a 1 for message level 1. Bits 4-7 are not used.

3	1	1	1	1	1
Storage address of job control table	Table ID=00	Job serial number	Job status indicators	Message class	Message level
8					
Jobname (padded with blanks)					
8					
Not used in the primary control program					
3	1	3	1		
Storage address of PDQ	Not used	Storage address of GDG bias count table	Not used		
3	1	3	1		
Storage address of first step control table	Not used	Storage address of first system message block	Not used		
3	1	3	1		
Storage address of job account control table	Not used	Storage address of first data set SYSOUT block	Not used		
3	1	2	2		
Storage address of last data set SYSOUT block	Not used	Not used	First job condition code		
1	1	2	1	1	
First job condition operator	Not used	. . . . .	Eighth job condition code	Eighth job condition operator	Not used

Figure 36. Job Control Table

JOB KEYWORD TABLE

All JOB statement keywords are listed in the JOB Keyword Table, (see Figure 37). Bytes 14 through 16 of each entry contain the address of the routine which processes the keyword parameters.

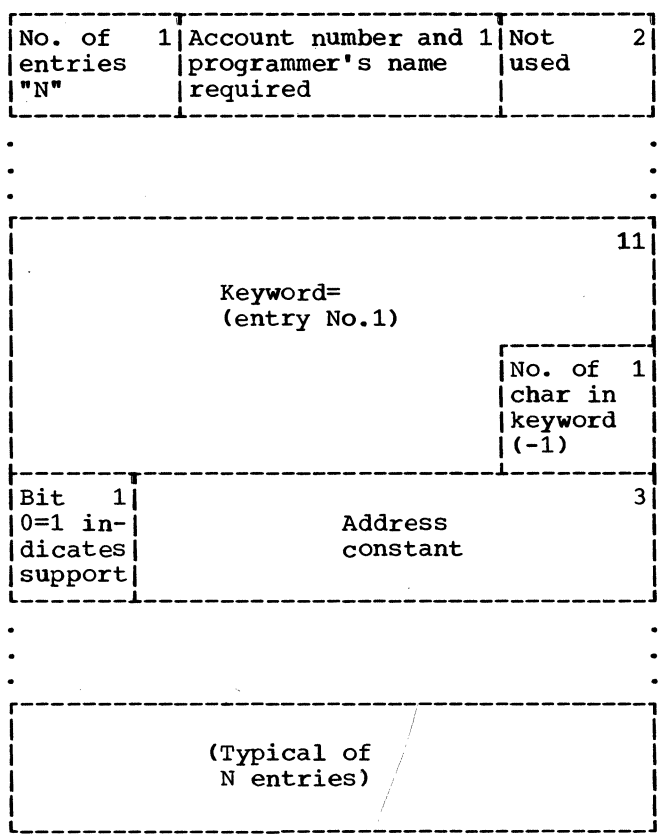


Figure 37. JOB Keyword Table

NEW READER OR WRITER TABLE

The new reader or writer table (NRWT), shown in Figure 38, is a control block that contains OPEN requirements for reader and writer routines. At initial program load time, the table is written onto auxiliary storage. The table is read into main storage from auxiliary storage and is used by the reader/interpreter and SYSOUT routines. Each entry (except jobname) consists of an active section and an inactive section. Whether the lower or higher order part of the entry is active is indicated by a 1 in bit 0 of the flag 1 byte in the active section. When a NRWT entry is active, the data set has been opened, and the device indicated by the applicable UCB pointer is active. The currently inactive section of the entry receives information from new START commands. The table is always available in the SYS1.SYSJOBQE data set.

The bits in location Flags 1 have the following meanings:

Bit 0 -- 1= active
0= inactive

Bit 1 -- 1= START RDR at jobname
0= START RDR at first job

The bits in location Flags 2 are not used in the sequential scheduling system.

PASSED DATA SET QUEUE

The passed data set queue (PDQ), shown in Figure 39, contains information regarding previously processed data sets which have been passed from executed steps of the job, that may be referenced by subsequent steps of the same job. Each PDQ contains a set of tables, consisting of three types of blocks: the PDQ directory block, the PDQ block, and the PDQ overflow block (if required). The PDQ directory block and the PDQ block are created by the initiator/terminator JFCB housekeeping routine. The directory blocks are chained together with pointers, and each PDQ directory block also points to its respective PDQ block. If more than ten additional UCB pointers are needed for any one PDQ entry, one or more PDQ overflow blocks are added in a chain to each such PDQ block entry by allocation routines.

Initiator/terminator routines use the PDQ to obtain pointers to UCBs when allocating devices to passed data sets. Step termination routines use the PDQ to obtain UCB allocation pointers and disposition information.

When control passes to the initiator/terminator, the JFCB housekeeping routine inspects the disposition field of the SIOT for the disposition "PASS" to determine whether a new entry may be required in the PDQ.

If a PASS disposition is found and the dsname is not in the PDQ directory because it was not placed into the directory by a prior PASS, an entry is made in the PDQ for this dsname. If the last PDQ directory block and PDQ block already contain the maximum number of three entries, auxiliary storage space is assigned for a new PDQ directory block and a new PDQ block, thereby providing space for three more dsname entries.

When a passed data set is to be referenced by a subsequent step in the same job, the dsname is specified in the DD statement. The JFCB housekeeping routine checks for the dsname in the PDQ directory

to see if the data set was received (passed from a previous step).

If the dsname is found in the PDQ directory, the existing PDQ entry for this dsname is updated to identify the reference as the latest reference to this dsname and the data set is marked as being received in the PDQ entry. If no entry is found, the data set must have been cataloged, so the JFCB routine searches the catalog for this dsname, assuming that this is an initial reference for this job to a cataloged data set.

Bits of the terminate work area byte of the PDQ block have the following status significance:

Bit	Significance	Status
0	Initial status	1 = old
1	Current status	1 = old
2	Pass satisfied	1 = passed 0 = received
3	SYSIN specified	1 = SYSIN
4	SYSOUT specified	1 = SYSOUT

Start Reader Entry

Track address (3)	Flags 1 (1)	Flags 2 (not used) (2)	UCB pointer (2)
Track address (3)	Flags 1 (1)	Flags 2 (not used) (2)	UCB pointer (2)

Jobname Entry

Jobname from START command of operator (8)

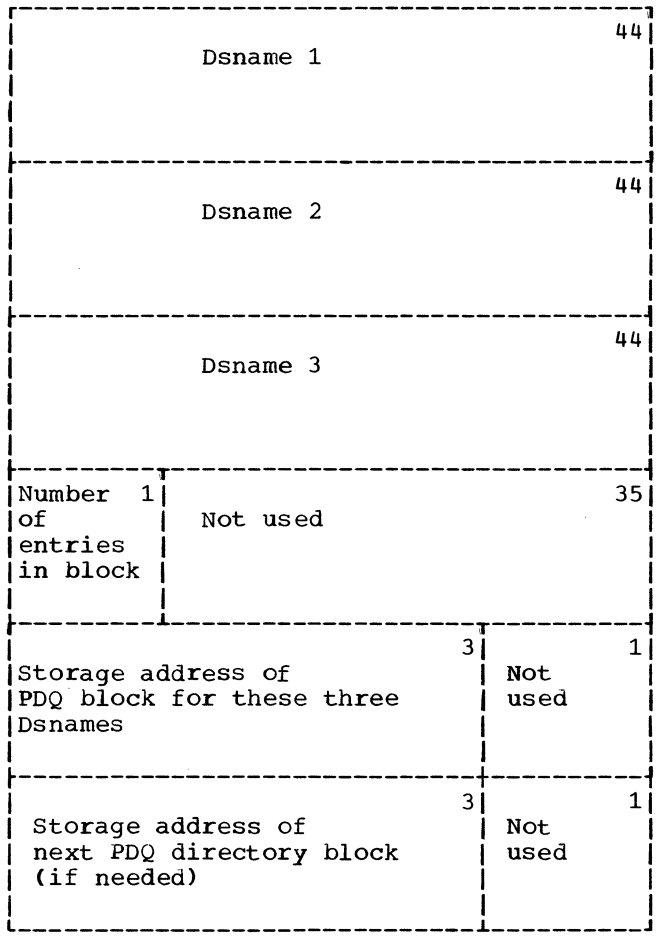
Start Writer Entry

Track address (3)	Flags 1 (1)	Flags 2 (not used) (2)	UCB pointer (2)
Track address (3)	Flags 1 (1)	Flags 2 (not used) (2)	UCB pointer (2)

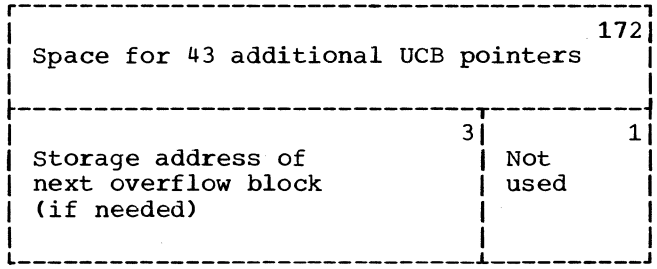
Cataloged Procedures Entry

Track address (3)	Flags 1 (1)	Flags 2 (not used) (2)	UCB pointer (2)
Track address (3)	Flags 1 (1)	Flags 2 (not used) (2)	UCB pointer (2)

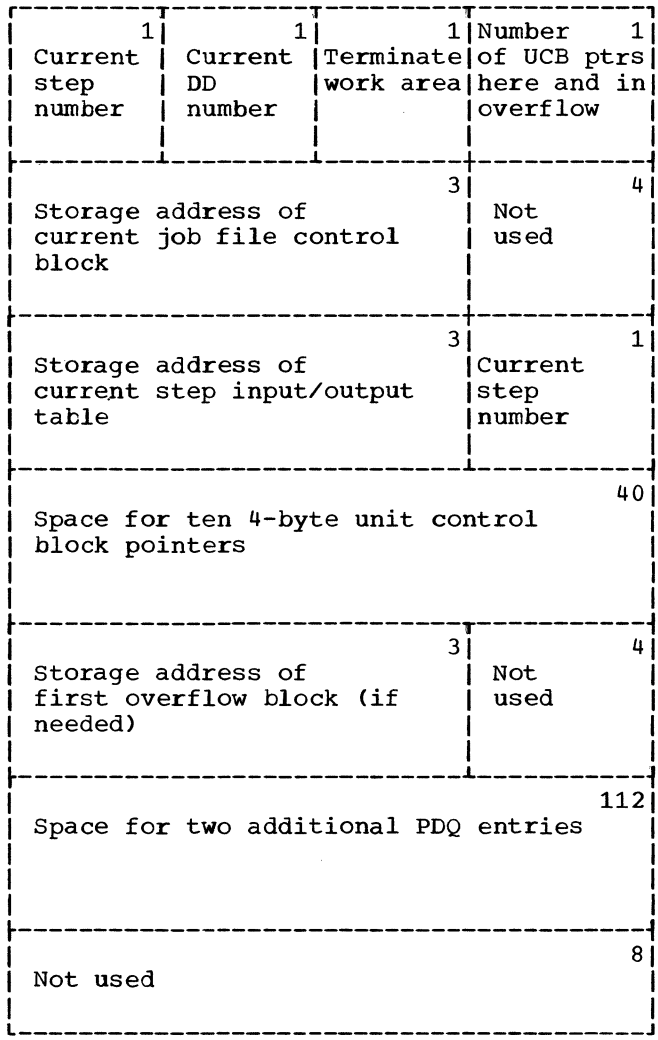
Figure 38. New Reader or Writer Table



PDQ Directory Block



PDQ Overflow Block



PDQ Block

Figure 39. Passed Data Set Queue Tables

READER/INTERPRETER TTR TABLE

The reader/interpreter TTR table (see Figure 40) provides a means of chaining certain tables by allowing routines to insert a chain address into a table before that table is stored, prior to job step execution. The 16-word TTR table is built and updated by the reader/interpreter. First, the table-store subroutine assigns 15 3-byte addresses, and the reader/interpreter places them into the TTR table left-justified on a full-word boundary. When a table is to be stored, the reader/interpreter DD, job, or execute routine places the secondary address into the chaining field of the table to be stored. The routine then uses the table store subroutine write-and-assign function to place the table into the storage location

specified by the primary address. The present secondary address, which specifies the next available storage location, is then moved into the primary address field of the TTR table, and a new secondary address, assigned by the table store subroutine, is placed into the TTR table.

Primary Address	Secondary Address
Job control table	None
Account control table	
Step control table	
Step I/O table	
Job file control block	
System message block	
Volume table	
Dsname table	

Figure 40. Reader/Interpreter TTR Table

STEP CONTROL TABLE

The step control table (SCT), shown in Figure 41, is used to pass control information to the DD routine of the reader/interpreter and to the initiator/terminator routines, which also contribute information to the table. This table is created and initialized by the execute routine of the reader/interpreter when an EXEC statement is read. One SCT is created for each step of a job, and is stored by the reader/interpreter control routine and the initiator/terminator step initiation routine.

If the step is part of a previously cataloged procedure, the name of the step that called the procedure, if any, is entered. The following variable-content and indicator fields are included in the table:

1. Internal Step Status Indicators:

   Bit 7 contains a one if an error condition caused the step to be terminated.

2. PARM Count or Step Status Code:

   a. Reader/Interpreter: The number of characters specified in the PARM parameter of the EXEC statement is placed in this entry.

   b. Initiator/Terminator: This table entry contains the condition code returned by the processing program.

3. Step Type Indicators:

   Bit 0 contains a one if the following parameter definition appears in the EXEC statement:

   PGM=*.stepname.ddname

   Bit 1 indicates SYSIN is specified (DD *).

   Bit 2 indicates SYSOUT is specified.

   Bit 3 contains a 1 if JFCB housekeeping is complete.

   Bits 4, 5, and 6 are unused.

Storage address of step control table (3)	Table ID=02 (1)	Inter-nal step status indicators (1)	Maximum step running time (3)
PARM count or step status code (2)	Length of allocate work area or number of SIOTs (2)	Storage address of first SIOT entry (3)	Not used (1)
Storage address of allocate work area (3)	Not used (1)	Storage address of next SCT (3)	Not used (1)
Storage address of first SMB for next step (3)	Not used (1)	Storage address of last SMB for this step (3)	Not used (1)
Storage address of first ACT entry for this step (3)	Not used (1)	Storage address of volume table (3)	Not used (1)
Storage address of dsname table for this step (3)	Not used (1)	Name of step that called procedure (if any) (8)	
Stepname (8)			
Not used (2)	Length of volume table (2)	No. of SIOTs in this step (1)	No. of setup messages' (1)	No. of JFCBs to allocate (1)	Step type indicators (1)
Initializing parameter values (40)					
Programname (8)					
Length of dsname table in bytes (2)	First step condition code (2)	First step condition operator (1)	Storage address of first condition SCT (3)		
Second through seventh step condition entries. . . . . (36)					
Eighth step condition code (2)	Eighth step condition operator (1)	Storage address of eighth condition SCT (3)	Not used (2)		

Figure 41. Step Control Table

STEP INPUT/OUTPUT TABLE

The Step Input/Output Table (SIOT), shown in Figure 42, makes DD statement information available to the initiator/terminator for use as a source of information for the TIOT and for providing DD information to allocation and disposition routines. When a DD statement is read, the reader/interpreter creates a new SIOT and places the DD information into it. The individual bits of indicator bytes 56 through 60 in the SIOT are set to one to indicate the following conditions:

BYTE 56: Disposition Status Byte (SCTSDISP)

- Bit 0 Nonshareable volume
- Bit 1 Retain volume
- Bit 2 Private volume
- Bit 3 Pass data set
- Bit 4 Keep data set
- Bit 5 Delete data set
- Bit 6 Catalog data set
- Bit 7 Uncatalog data set

BYTE 57: Status Byte 1 (SCTSBYT1)

- Bit 0 Dummy data set
- Bit 1 SYSIN data set
- Bit 2 Split (primary)
- Bit 3 Split (secondary)
- Bit 4 Suballocate
- Bit 5 Parallel mount
- Bit 6 Unit affinity
- Bit 7 Unit separation

BYTE 58: Status Byte 2 (SCTSBYT2)

- Bit 0 Channel affinity
- Bit 1 Channel separation
- Bit 2 Volume affinity
- Bit 3 Not used
- Bit 4 Unlabeled
- Bit 5 Pool DD statement
- Bit 6 Defer mounting
- Bit 7 Received data set

BYTE 59: Status Byte 3 (SCTSBYT3)

- Bit 0 Volume reference is dsname
- Bit 1 SYSIN expected (procedures only)
- Bit 2 No associated volume serial in volume table
- Bit 3 Intra-step suballocate
- Bit 4 SYSOUT was specified
- Bit 5 New data set
- Bit 6 Modified data set
- Bit 7 Old data set

BYTE 60: Status Byte 4 (SCTSBYT4)

- Bit 0 Set by reader/interpreter to indicate GDG single
- Bit 2 Volume serial was found in passed data set queue (PDQ)
- Bit 4 Step processed
- Bit 5 Intra-step volume affinity
- Bit 6 Data set is in PDQ
- Bit 7 1 = old or modified data set
  0 = new data set

SYSTEM MESSAGE BLOCK

The system message block (SMB), shown in Figure 43, temporarily stores all control statements and diagnostic error messages before they are printed via the system output writer routine. The reader/interpreter control routine creates and initializes one or more SMBs for each job step. Initiator/terminator routines also may add messages to the SMB. The chain address of the next SMB is given in bytes 4 through 8 of each table but the last, resulting in a chain of SMBs for each job. The status byte of each entry concerns the following entry, and contains the message length, zero if there are no more messages, or all ones if a data set entry follows, the format of which is shown in Figure 44.

VOLUME TABLE

The volume table (VOLT), shown in Figure 45, consists of a series of chained blocks, and contains the list of volume serial numbers to be used in a given step. Use of the list reduces the number of times that the SYS1.SYSJOBQE data set must be referenced during allocation. The table is built by the DD routine for each step, and is modified by the JFCB housekeeping routine. The maximum extent of each block of the table is 176 bytes, and the maximum number of volumes listed per block is 28.

						3 Auxiliary storage address of SIOT	1 Table ID=03
8 Ddname							
8 Channel separation and affinity							
8 Unit separation and affinity							
3 Storage address of next SIOT in chain	1 Not used			3 Storage address of JFCB (left adjusted)			1 Not used
3 Storage address of SIOT for VOLREF/SUBALLOCATE	1 Not used			3 Storage address of SIOT system output / dependency block			1 Not used
4 Not used				1 Internal No. of pool DD	1 No. of volumes	2 Relative pointer to volume table entry	
1 Internal DD number	1 No. of units	1 Not used	5 Status bytes (see text)				
8 Unit type							
61 Not used							
44 DCB reference dsname						2 Not used	

Figure 42. Step Input/Output Table

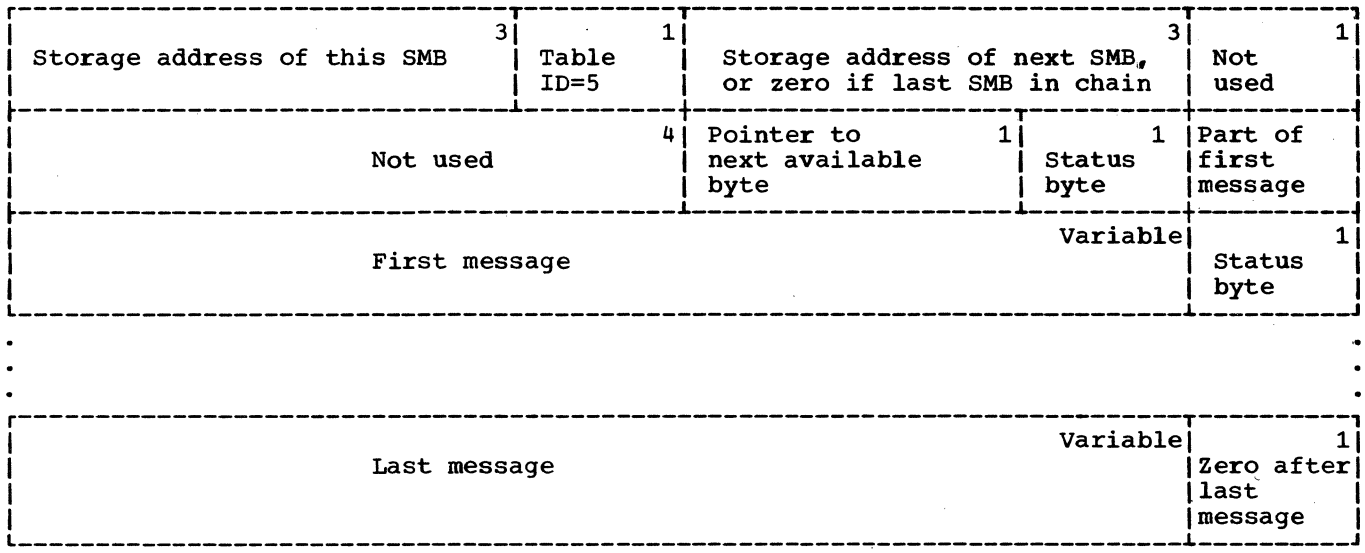


Figure 43. System Message Block

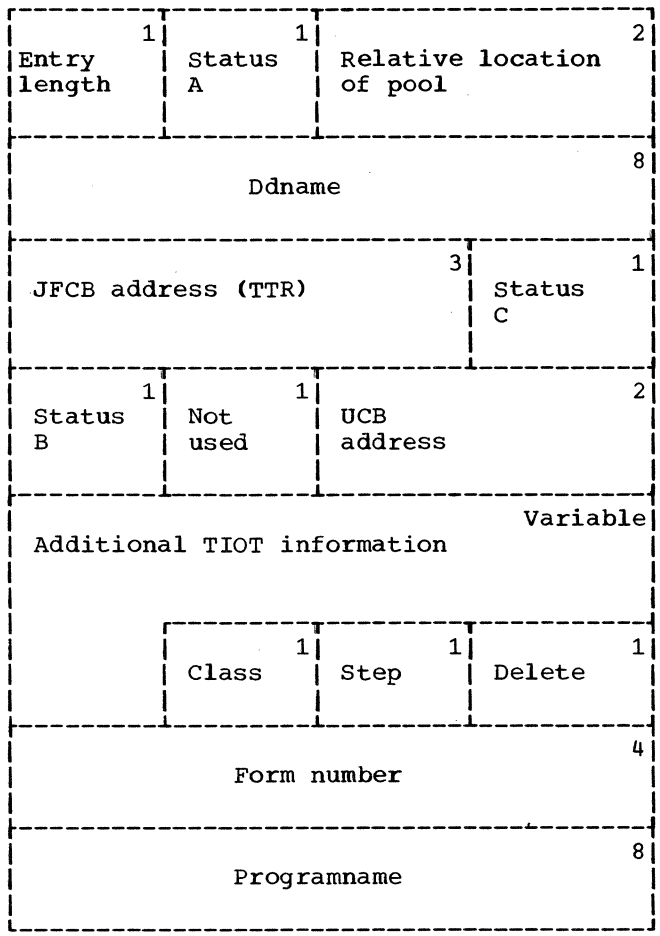


Figure 44. SMB Data Set Message Format

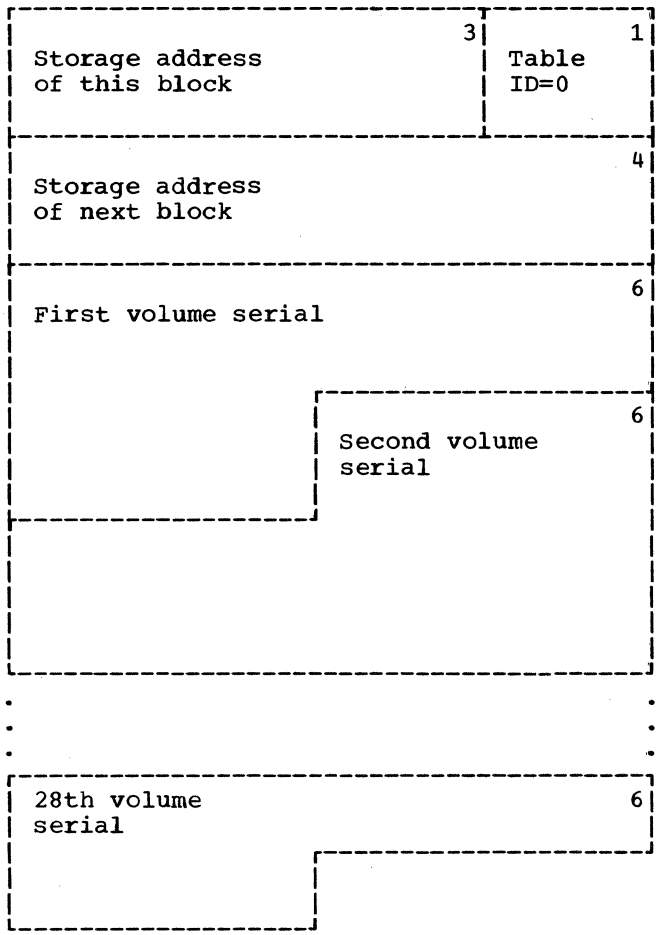


Figure 45. Volume Table

This appendix lists job management load modules and indicates the assembly modules that are processed by the linkage editor into each load module during system generation. Included is a separate list that shows the load modules in which each assembly module is contained.

Job management routines for sequential scheduling systems are packaged in three configurations: 18K, 44K, and 100K (where K is 1024 bytes of main storage). The numbers represent the maximum amount of main storage occupied by job management routines and work areas at any time. All three job management configurations function identically but differ in both the number of their load modules and the number of assembly modules within each load module. Job management routines occupy the dynamic portion of main storage alternately with processing programs, and therefore these size designations bear a direct relationship to the main storage required for each configuration.

LOAD MODULES

In each configuration, all load modules are contained in three data sets: SYS1.NUCLEUS, SYS1.SVCLIB, and SYS1.LINKLIB. These data sets also contain other parts of the control program. The load modules in the first two data sets remain the same for all three job management configurations, but the SYS1.LINKLIB data set contains a different set of load modules for each configuration, depending on which one was selected at system generation time. In the 18K configuration, LINKLIB contains 36 load modules; in the 44K configuration, it contains 25 load modules; and in the 100K configuration, 18 load modules.

Charts 43 through 45 show the control flow among load modules. The decision to transfer control (XCTL) to a particular succeeding load module is made in the previous load module. Each subsequent module loaded in response to an XCTL macro-instruction is read into main storage directly over the previous load module. Such load modules are read into the low-numbered end of the dynamic, or problem-program, area of main storage.

Modules that are brought into storage with LINK macro-instructions and LOAD macro-instructions occupy separate storage areas within the problem program area; such modules are shown on the control-flow charts. Because storage is used in this manner, the load module lists may be used with Charts 43, 44 or 45 to determine the approximate layout of main storage at different times during the execution of job management routines. Other items present in the problem program area at the same time as the load modules are not shown on the control flow charts because, although these items are necessary, control is not passed among them. They are, generally, the tables and control blocks, work areas, access methods, buffers, and register save areas.

In the following load module lists, entry points are shown if a load module contains more than one assembly module. If only one assembly module is named, the entry point is the same as the assembly module's control section (CSECT) name given in the Assembly Modules and Control Sections table in this appendix.

LOAD MODULES CONTAINED IN THE SYS1.NUCLEUS DATA SET

The load modules and assembly modules in the following list are contained in the SYS1.NUCLEUS data set, and are always present in the nucleus, or fixed area of main storage, regardless of the job management configuration.

Load Module Name: SYS1.NUCLEUS

Assembly Modules:

IEEBC1PE	External interrupt routine.
IEECIR01	Console interrupt routine.
IEERSC01	Master scheduler buffers, switches, input/output block (IOB), event control block (ECB), channel control word (CCW), and data extent block (DEB). This load module forms master scheduler resident main storage in the nucleus area when the primary or alternate console (1052) is used.
IEERSR01	Master scheduler buffers, switches, IOB, ECB, CCW, and DEB. This load module forms master scheduler resident main storage in the nucleus area when the composite console is used.
IEFDPOST	Unsolicited interrupt routine.
IEFKRESA	Table store subroutine work area.

LOAD MODULES CONTAINED IN THE SYS1.SVCLIB DATA SET

The load modules and assembly modules in the following list are contained in the SYS1.SVCLIB data set, and are called in response to SVC instructions.

Load Module Name: IGC0003D
Assembly Modules:
IEEMXC01 Master command EXCP routine (Part 1) -- primary/alternate console.
IEEMXR01 Master command EXCP routine (Part 1) -- composite console.

Load Module Name: IGC0003E
Assembly Modules:
IEEWTC01 Write-to-operator (WTO) routine -- primary/alternate console.
IEEWTR01 Write-to-operator (WTO) routine -- composite console.

Load Module Name: IGC0103D
Assembly Modules:
IGC0103D Command processing routine for 'MOUNT, VARY ONLINE/OFFLINE, and UNLOAD. This routine issues an XCTL to IGC0203D if command is other than listed.'
IGC0203D Command processing routine for 'DISPLAY JOBNAMES, STOP JOBNAMES, CANCEL' (SHIFT command not used primary control program.)

Load Module Name: IGC0003F
Assembly Module:
IEEBH1PE Not used in sequential scheduling system.

MODULES CONTAINED IN THE SYS1.LINKLIB DATA SET

The load modules and assembly modules in the following lists are contained in the SYS1.LINKLIB data set. A list is provided for each of the three packaging configurations in which job management routines are available.

18K CONFIGURATION

Load Module Name: IEFSTERM
Alias: IEFYN
Alias: GO
Entry Point: IEFSD011
Assembly Modules:
IEFSD011 Entry to job management from supervisor.
IEFW42SD Passes control to IEFIDUMP (in IEFIDUMP Load Module) if necessary, or to IEFYNIMP (in this module).
IEFYNIMP Step termination routine.
IEFYPJB3 Step data set driver routine.
IEFVJIMP Job statement condition code routine.
IEFZGST1 Disposition and unallocation subroutine.
IEFACTLK Linkage to user's accounting routine.
IEFACTRT Dummy, to be replaced by user's accounting routine.
(The preceding two modules may be replaced by IEFACTFK assembly module if no accounting routine is specified as a system generation option.)
IEFSD017 Places logical track address (TTR) of first system message block (SMB) into job control table (JCT).
IEFW22SD Passes control to IEFYNIMP (in this load module), then to IEFSD002 (in this load module) or to IEFZAJB3 (in IEFJTERM load module).
IEFSD002 Exit to IEF08FAK or IEF09FAK (both in this load module).
IEFSD006 Converts record number to logical track address (TTR).
IEFSD007 Call to table store subroutine.
IEFYSSMB Message enqueuing routine, enqueues SMBs.
IEFQMSSS Table store subroutine.
IEFVJMSG Contains initiator/terminator messages.
IEFYNMSG Contains initiator/terminator messages.
IEFYPMSG Contains initiator/terminator messages.
IEFZGMSG Contains initiator/terminator messages.
IEFZHMSG Contains initiator/terminator messages.
IEFIDFAK Linkage to IEFIDUMP (in IEFIDUMP load module).
IEFZAFAK Linkage to IEFZAJB3 (in IEFJTERM load module).
IEF08FAK Linkage to IEFSD008 (in IEFINTFC load module).
IEF09FAK Linkage to IEFSD009 (in IEFSELCT load module).

Load Module Name: IEFSELCT
Alias: IEFSD009
Entry Point: IEFSD009
Assembly Modules:
IEFSD006 Converts record number to logical track address (TTR).
IEFSD009 Initializes initiator/terminator.

Load Modules
(18K Configuration, Continued)

IEFW21SD	System control routine.
IEFVKIMP	Execute statement condition code routine.
IEFVMLS1	JFCB housekeeping (H/K) control routine.
IEFVM2LS	JFCB H/K fetch DCB routine.
IEFVM3LS	JFCB H/K generation data group (GDG) single routine.
IEFVM4LS	JFCB H/K generation data group (GDG) all routine.
IEFVM5LS	JFCB H/K patterning data set control block (DSCB) routine.
IEFVM76	Processes passed, non-labeled tape data sets.
IEFWSTRT	Job started message routine.
IEFYSSMB	Message enqueuing routine, enqueues SMBs.
IEFQMSSS	Table store subroutine.
IEFWMAS1	Device name table.
IEFVKMSG	Contains initiator terminator messages.
IEFVMLK5	Linkage to IEFVMLS6 (in IEFERROR load module).
IEFXAFAK	Linkage to IEFXCSSS (in IEFALOC1 load module).
IEFYNFAK	Linkage to IEFYNIMP (in IEFSTERM load module).

Load Module Name: IEFALOC1
Alias: IEFXJ000
Alias: IEFXA
Entry Point: IEFXA
Assembly Modules:

IEFXCSS	Allocation control routine.
IEFXJIMP	Allocation error recovery routine.
IEFYSSMB	Message enqueuing routine.
IEFQMSSS	Table store subroutine.
IEFXAMSG	Contains initiator/terminator messages.
IEFXJMSG	Contains initiator/terminator messages.
IEFWAFAK	Linkage to IEFWA000 (in IEFALOC2 load module).
IEFWCFAK	Linkage to IEFWCIMP (in IEFALOC3 load module).
IEFYNFAK	Linkage to IEFYNIMP (in IEFSTERM load module).

Load Module Name: IEFALOC2
Alias: IEFWA000
Alias: IEFX5000
Entry Point: IEFWA000
Assembly Modules:

IEFWA000	Demand allocation routine.
IEFWSWIN	Passes control to decision allocation or automatic volume recognition (AVR) routine.
IEFX5000	Decision allocation routine.
IEFX300A	Device strikeout routine.
IEFXH000	Separation strikeout routine.
IEFWMSKA	Device mask table.
IEFXVFAK	Linkage to IEFXV001 (in load module IEFALOC4).

Load Modules
(18K Configuration, Continued)

IEFWCFAK	Linkage to IEFWCIMP (in IEFALOC3 load module).
IEFXJFAK	Linkage to IEFXCSSS (in IEFALOC1 load module).

Load Module Name: IEFALOC3
Alias: IEFWC000
Entry Point: IEFWC000
Assembly Modules:

IEFWCIMP	Task Input/Output Table construction routine.
IEFXH000	Separation strikeout routine.
IEFWDFAK	Linkage to IEFWD000 (in IEFALOC4 module).
IEFXJFAK	Linkage to IEFXCSSS (in IEFALOC1 module).

Load Module Name: IEFALOC4
Alias: IEFWD000
Alias: IEFXV001
Entry Point: IEFWD000
Assembly Modules:

IEFWD000	External action routine.
IEFWD001	Message directory for external action routine.
IEFXKIMP	Allocation error non-recovery routine.
IEFYSSMB	Message enqueuing routine, enqueues SMB's.
IEFQMSSS	Table store subroutine.
IEFXKMSG	Contains initiator/terminator messages.
IEFYNFAK	Linkage to IEFYNIMP (in IEFSTERM load module).
IEFSD006	Converts record number to logical track address (TTR).
IEFXTFAK	Linkage to IEFXT000 (in load module IEFALOC5).
IEFXV001	Automatic volume recognition.
IEFXV002	Processes new volumes (AVR).
IEFXV003	Processes specific requests for unmounted volumes.
IEFXV004	AVR unloading routine.
IEFXVNSL	AVR volume serial routine.
IEFXVMSG	AVR message routine.
IEFX1FAK	Linkage to IEFXJIMP (in load module IEFALOC1).
IEFX2FAK	Linkage to IEFX5000 (in load module IEFALOC2).
IEFX3FAK	Linkage to IEFWCIMP (in load module IEFALOC3).
IEFX300A	Device strikeout routine.

Load Module Name: IEFALOC5
Alias: IEFXT000
Entry Point: IEFXT000
Assembly Modules:

IEFXKIMP	Allocation error non-recovery routine.
IEFXTDMY	Queue overflow routine.
IEFXT00D	Space request routine.
IEFYSSMB	Message enqueuing routine, enqueues SMBs.

IEFQMSSS Table store subroutine.
IEFXKMSG Contains initiator/terminator messages.
IEFXTMSG Contains initiator/terminator messages.
IEFW41SD Exit to IEF04FAK (in this load module).
IEFSD006 Converts record number to logical track address (TTR).
IEF04FAK Linkage to IEFSD004 (in IEFATACH load module).
IEFYNFAK Linkage to IEFYNIMP (in IEFSTERM load module).
IEFWDFAK Linkage to IEFWD000 (in IEFALOC4 load module).

Load Module Name: IEFATACH
Alias: IEFSD004
Entry Point: IEFSD004
Assembly Modules:
IEFSD004 Step initiation routine, with exit to processing program.
IEFSD006 Converts record number to logical track address (TTR).
IEFSD007 Call to table store subroutine.
IEFSD010 Dequeues and writes out system message blocks (SMBs).
IEFQMSSS Table store subroutine.

Load Module Name: IEFCNTRL
Alias: IEF5DDHD
Alias: IEFMF
Alias: IEFMC
Alias: IEFKA
Entry Point: INDMRTN
Assembly Modules:
IEFSPIE Program check handling routine.
IEF7KAXX Reader/interpreter control routine.
IEF6DDHD DD routine.
IEF6BOCM Breakout routine.
IEF6MFXX Verb identification routine.
IEF6MCXX Scans job control language (JCL) statements.
IEF6STNM Scan stepname routine.
IEF6NAME Qualified name routine.
IEF6FRRS Resolves DD forward references.
IEF6DCB0 DCB refer-back routine.
IEF6MKXX Continuation routine.
IEF7MMCM Reader/interpreter message routine.
IEFSD006 Converts record number to logical track address (TTR).
IEF6MIXX Reader/interpreter call to IEFQMSSS.
IEFQMSSS Table store subroutine.
IEFK4DUM Linkage to IEFK4ENT (in IEFK4 load module).
IEF6DHX1 Linkage to IEF6SCAN (in IEFDD load module).
IEFKLDUM Linkage to IEF6KLXX (in IEF1STMT load module).

Load Modules
(18K Configuration, Continued)

IEFJMDUM Linkage to IEF6NCJB (in IEFJOB load module).
IEFEMDUM Linkage to IEF6NJEX (in IEFEXEC load module).
IEF6OUT2 DD output routine, with exit to IEF7KAXX (in this load module).
IEFKGDUM Linkage to IEF7KGXX (in IEFINTFC load module).
IEFKPDUM Linkage to IEF7KPXX (in IEFCOMND load module).
IEFK3DUM Linkage to IEF7K3XX (in IEFEOF load module).

Load Module Name: IEFDD
Alias: IEF5SCAN
Entry Point: INDMON
Assembly Modules:
IEFSPIE Program check handling routine.
IEF6SCAN DD scan routine.
IEF6BOCM Breakout routine.
IEFSD012 DD* statement routine.
IEF6DDNM DD name routine.
IEF6DSNM DS name routine.
IEF6RFWD Processes DD forward references.
IEF6RTPR Right parenthesis routine.
IEF6LFPR Left parenthesis routine.
IEF6EQUL Equal sign routine.
IEF6LIST Subparameter list routine.
IEF6NLST Routine for no subparameter list.
IEF6NDDP DD parameter list table.
IEF6NDDX Alternative DD parameter list table (DDPLT).
IEF6DCDP Data control block (DCB) DD parameter list table.
IEFSD013 Assigns unit to system output (SYSOUT).
IEF6ORDR Order subroutine.
IEF6INST Insert routine.
IEF6VALU Value subroutine.
IEF6CLNP Clean up after DD routine.
IEFSD006 Converts record number to logical track address (TTR).
IEF6ERR1 DD error-handling routine.
IEF7MMCM Reader/interpreter message routine.
IEF6MIXX Reader/interpreter call to IEFQMSSS
IEFQMSSS Table store subroutine.
IEF6CN17 Linkage to IEF6DDHD (in IEFCNTRL load module).

Load Module Name: IEFINTFC
Alias: IEFSD008
Alias: IEFSD001
Alias: IEFKG
Entry Point: IEFSD008
Assembly Modules:
IEFSPIE Program check handling routine.
IEFSD008 Initiator/terminator to reader/interpreter interface.
IEF7KGXX Output tables for step.
IEFSD006 Converts record number to logical track address (TTR).

Load Modules
(18K Configuration, Continued)

IEFSD007	Call to table store subroutine.
IEEMCR01	Master command routine.
IEF7MMCM	Reader/interpreter message routine.
IEF6MIXX	Reader/interpreter call to IEFQMSSS.
IEFQMSSS	Table store subroutine.
IEFSD001	Reader/interpreter entry to IEF09FAK or to IEFW23SD (both in this load module).
IEF09FAK	Linkage to IEFSD009 (in IEFSELCT load module).
IEF23FAK	Linkage to IEFW23SD (in IEFJTERM load module).
IEFMFDUM	Linkage to IEF6MFXX (in IEFCNTRL load module).
IEFKADUM	Linkage to IEF7KAXX (in IEFCNTRL load module).
IEFK3DUM	Linkage to IEF7K3XX (in IEFEOF load module).

Load Module Name: IEFEXEC
Alias: IEFEM
Entry Point: IEFEM
Assembly Modules:

IEFSPIE	Program check handling routine.
IEF6NJEX	Execute (EXEC) statement routine.
IEF6BOCM	Breakout routine.
IEF6STNM	Scan stepname routine.
IEF6NAME	Qualified name routine.
IEF6RFBK	Refer-back routine.
IEF6PROC	Procedure name routine.
IEF6TIME	TIME keyword routine.
IEF6COND	Condition (COND) keyword routine.
IEF6PARM	Parameter (PARM) keyword routine.
IEF6NFCM	Accounting information routine.
IEF7MMCM	Reader/interpreter message routine.
IEF6MIXX	Reader/interpreter call to IEFQMSSS.
IEFQMSSS	Table store subroutine.
IEF8LINK	Linkage to IEF6COND (in this module).
IEFMFDUM	Linkage to IEF6MFXX (in IEFCNTRL load module).

Load Module Name: IEFJOB
Alias: IEFJM
Entry Point: IEFJM
Assembly Modules:

IEFSPIE	Program check handling routine.
IEF6NCJB	Job (JOB) statement routine.
IEF6BOCM	Breakout routine.
IEF6STNM	Scan stepname routine.
IEF6NAME	Qualified name routine.
IEF6NFCM	Accounting information routine.
IEF6NIJB	TYPRUN keyword routine.
IEF6NYJB	Priority (PRTY) keyword routine.
IEF6COND	Condition (COND) keyword routine.

Load Modules
(18K Configuration, Continued)

IEF6NXJB	Message level (MSGLEVEL) keyword routine.
IEF6NZJB	Message class (MSGCLASS) keyword routine.
IEF6NIJB	Parenthesis routine.
IEF7MMCM	Reader/interpreter message routine.
IEF6MIXX	Reader/interpreter call to IEFQMSSS.
IEFQMSSS	Table store subroutine.
IEFMFDUM	Linkage to IEF6MFXX (in IEFCNTRL load module).

Load Module Name: IEFJTERM
Alias: IEFW23SD
Alias: IEFZA
Entry Point: IEFZA
Assembly Modules:

IEFW23SD	Initializes for job termination, exits to IEFZAJB3 (in this load module).
IEFZAJB3	Job termination routine.
IEFWTERM	Job ended message routine.
IEFZGJB1	Disposition and unallocation subroutine.
IEFACTLK	Linkage to user's accounting routine.
IEFACTRT	Dummy module to be replaced by user's accounting routine.

(The preceding two modules may be replaced by IEFACTFK assembly module if no accounting routine is specified as a system generation option.)

IEFSD006	Converts record number to logical track address (TTR).
IEFSD007	Call to table store subroutine.
IEFYSSMB	Message enqueuing routine, enqueues SMBs.
IEFQMSSS	Table store subroutine.
IEFZHFAK	Call to ZPOQMGR1 subroutine, in IEFZGJB1 of this load module.
IEFZGMSG	Contains initiator terminator messages.
IEFZHMSG	Contains initiator terminator messages.
IEFW31SD	Exit to IEFSD003 (in this load module).
IEFSD003	Passes control to IEFSD010, then to IEF08FAK, (both in this load module).
IEFSD010	Dequeues and writes out system message blocks (SMBs).

Load Module Name: IEFCOMND
Alias: IEFKP
Entry Point: IEFKP
Assembly Modules:

IEF7KPXX	Processes commands in input stream.
IEEMCR01	Master command routine.
IEEILCDM	Prevents unresolved IEEICCAN symbol after initialization.
IEFSD006	Converts record number to logical track address (TTR).

IEF7MMCM	Reader/interpreter message routine.
IEF6MIXX	Reader/interpreter call to IEFQMSSS.
IEFQMSSS	Table store subroutine.
IEFKADUM	Linkage to IEF7KAXX (in IEFCNTRL load module).

Load Module Name: IEF1STMT
Alias: IEFKL
Entry Point: IEFKL
Assembly Modules:

IEFSPIE	Program check handling routine.
IEF6KLXX	First statement routine.
IEF7MMCM	Reader/interpreter message routine.
IEF6MIXX	Reader/interpreter call to IEFQMSSS.
IEFQMSSS	Table store subroutine.
IEFMCDUM	Linkage to IEF6MCXX (in IEFCNTRL load module).
IEFKADUM	Linkage to IEF7KAXX (in IEFCNTRL load module).

Load Module Name: IEFEOF
Alias: IEFK3
Entry Point: IEFK3
Assembly Modules:

IEFSPIE	Program check handling routine.
IEF7MMCM	Reader/interpreter message routine.
IEF7K3XX	Input stream end-of-file (EOF) routine.
IEF7K4XX	Close devices routine.
IEF7K2XX	Open devices routine.
IEFSD006	Converts record number to logical track address (TTR).
IEFSD007	Call to table store subroutine.
IEF6MIXX	Reader/interpreter call to IEFQMSSS.
IEFQMSSS	Table store subroutine.
IEEMCR01	Master command routine.
IEFKADUM	Linkage to IEF7KAXX (in IEFCNTRL load module).
IEEILCDM	Prevents unresolved IEEICCAN symbol after initialization (IPL).

Load Module Name: IEFK4
Entry Point: IEFK4DUM
Assembly Modules:

IEFSPIE	Program check handling routine.
IEF7MMCM	Reader/interpreter message routine.
IEFK4ENT	Switch input readers routine.
IEF7K4XX	Close devices routine.
IEF7K2XX	Open devices routine.
IEFSD006	Converts record number to logical track address (TTR).
IEFSD007	Call to table store subroutine.
IEF6MIXX	Reader/interpreter call to IEFQMSSS.
IEFQMSSS	Table store subroutine.

Load Modules
(18K Configuration, Continued)

Load Module Name: IEFERROR
Alias: IEFVM6LS
Entry Point: IEFVMSGR
Assembly Modules:

IEFVMLS6	JFCB housekeeping error message processing routine.
IEFYSSMB	Message enqueuing routine, enqueues SMBs.
IEFQMSSS	Table store subroutine.
IEFVMLS7	Contains initiator terminator messages.
IEFYNFAK	Linkage to IEFYNIMP (in IEFSTERM load module).

Load Module Name: IEFIDUMP
Entry Point: IEFIDUMP
Assembly Modules:

IEFIDUMP	Indicative dump routine.
IEFYSSMB	Message enqueuing routine, enqueues SMBs.
IEFQMSSS	Table store subroutine.
IEFIDMPM	Contains initiator terminator messages.
IEFYNFAK	Linkage to IEFYNIMP (in IEFSTERM load module).

Load Module Name: IEFDCB
Alias: IEF5DCDP
Assembly Module:

IEF6DCDP	Data control block (DCB) DD parameter list table.

Load Module Name: IEFMSG01
Assembly Module:

IEF3MSG1	Contains reader/interpreter messages.

Load Module Name: IEFMSG02
Assembly Module:

IEF3MSG2	Contains reader/interpreter messages.

Load Module Name: IEFMSG03
Assembly Module:

IEF3MSG3	Contains reader/interpreter messages.

Load Module Name: IEFMSG04
Assembly Module:

IEF3MSG4	Contains reader/interpreter messages.

Load Module Name: IEFMSG05
Assembly Module:

IEF3MSG5	Contains reader/interpreter messages.

Load Module Name: IEFMSG06
Assembly Module:

IEF3MSG6	Contains reader/interpreter messages.

Load Modules
(18K Configuration, Continued)

Load Module Name: IEFMSG07
Assembly Module:
IEF3MSG7 Contains reader/interpreter messages.

Load Module Name: IEFINITL
Alias: IEFK1
Assembly Modules:
IEFSPIE Program check handling routine.
IEF7MMCM Reader/interpreter message routine.
IEF7K1XX Entry to job management from nucleus initialization program (NIP).
IEFK1MSG Reader/interpreter message routine.
IEEMCR01 Master command routine.
IEEILC01 Automatic command routine.
IEF7K2XX Open devices routine.
IEFWSDIP Linkage control table (LCT) initialization routine.
IEFSD006 Converts record number to logical track address (TTR).
IEFSD007 Call to table store subroutine.
IEF6MIXX Reader/interpreter call to IEFQMSSS.
IEFQMSSS Table store subroutine.
IEFPRFAK Linkage to IEFPRES load module.

Load Module Name: IEFPRES
Assembly Modules:
IEFPRES Volume attribute initialization routine.
IEFKIMSG IEFPRES messages.
IEFKADUM Linkage to IEFKA load module.

Load Module Name: IEESET
Alias: IEEGESTO
Assembly Module:
IEEGES01 Master scheduler SET command routine.

Load Module Name: IEFJOBQE
Alias: IEFINTQS

Load Modules
(18K Configuration, Continued)

Assembly Module:
IEFINTQA Initializes SYS1.SYSJOBQE data set.

Load Module Name: IEETIME
Alias: IEEQOT00
Assembly Module:
IEEQOT00 Sets time and date.

Load Module Name: IEEFAULT
Alias: IEEGK1GM
Assembly Module:
IEEGK1GM Fault routine -- issues master scheduler messages.

Load Module Name: IEESTART
Alias: IEEIC1PE
Entry Point: IEEIC1PE
Assembly Modules:
IEESTART START command routine.
IEEREADR Start reader routine.
IEEWRITR Start writer routine.

Load Module Name: IEEJFCB
Alias: IEEIC3JF
Assembly Module:
IEEIC3JF Contains preformatted JFCB for one START command.

Load Module Name: IEESJFCB
Alias: IEEIC2NQ

Entry Point: IEEIC2NQ
Assembly Modules:
IEEIC2NQ Saves START command JFCBs.
IEFQMSSS Table store subroutine.

Load Module Name: IEFPRINT
Alias: SPRINTER
Alias: IEFPRT
Assembly Module:
IEFPRTXX Tape SYSOUT to printer.

44K CONFIGURATION

Load Module Name: IEFSTERM
Alias: IEFYN
Alias: IEFSD009
Alias: GO
Entry Point: IEFSD011
Assembly Modules:
IEFSD011 Entry to job management from supervisor.
IEFW42SD Passes control to IEFIDUMP (in IEFIDUMP load module) if indicative dump is needed, or to IEFYNIMP (in this load module).
IEFYNIMP Step termination routine.
IEFYPJB3 Step data set driver routine.
IEFVJIMP JOB statement condition code routine.
IEFZGST1 Disposition and unallocation subroutine.
IEFACTLK Linkage to user's accounting routine.
IEFACTRT Dummy user's accounting routine.
(The preceding two modules may be replaced by IEFACTFK assembly module if no accounting routine is specified as a system generation option.)

IEFSD017 Places logical track address (TTR) of first system message

	block (SMB) in job control table (JCT).
IEFW22SD	Passes control to IEFYNIMP (in this load module), and then to IEFSD002 (in this load module) or to IEFZAJB3 (in IEFJTERM load module).
IEFSD002	Exit to IEF08FAK or IEFSD009 (both in this load module).
IEFSD009	Initiator/terminator initialization of output unit, passes control to IEFW21SD (in this load module).
IEFW21SD	System control routine.
IEFVKIMP	EXEC statement condition code routine.
IEFVMLS1	JFCB housekeeping control routine.
IEFVM2LS	Fetch DCB routine.
IEFVM3LS	GDG single routine.
IEFVM4LS	GDG all routine.
IEFVM5LS	Patterning DSCB routine.
IEFVM76	Processes passed nonlabeled tape data sets.
IEFWSTRT	Job started message routine.
IEFWMAS1	Device name table.
IEFSD006	Converts record number to logical track address (TTR).
IEFSD007	Call to table store subroutine.
IEFYSSMB	Message enqueuing routine, enqueues SMBs.
IEFQMSSS	Table store subroutine.
IEFVJMSG	Contains initiator/terminator messages.
IEFVKMSG	Contains initiator/terminator messages.
IEFYNMSG	Contains initiator/terminator messages.
IEFYPMSG	Contains initiator/terminator messages.
IEFZGMSG	Contains initiator/terminator messages.
IEFZHMSG	Contains initiator/terminator messages.
IEFIDFAK	Linkage to IEFIDUMP (in IEFIDUMP load module).
IEFVMLK5	Linkage to IEFVMLS6 (in IEFERROR load module).
IEFXAFAK	Linkage to IEFXCSSS (in IEFALLOC load module).
IEFZAFAK	Linkage to IEFZAJB3 (in IEFJTERM load module).
IEF08FAK	Linkage to IEFSD008 (in IEFCNTRL load module).

Load Module Name: IEFALLOC
Alias: IEFXA
Entry Point: IEFXA
Assembly Modules:

IEFXCSSS	Allocation control routine.
IEFXJIMP	Allocation error recovery routine.
IEFWA000	Demand allocation routine.
IEFWSWIN	Passes control to decision allocation or AVR routine.

Load Modules
(44K Configuration, Continued)

IEFXV001	Automatic volume recognition.
IEFXV002	Processes new volumes (AVR).
IEFXV003	Processes specific requests for unmounted volumes.
IEFXV004	AVR unloading routine.
IEFXVNSL	AVR volume serial routine.
IEFXVMSG	AVR message routine.
IEFX5000	Decision allocation routine.
IEFX300A	Device strikeout routine.
IEFXH000	Separation strikeout routine.
IEFWMSKA	Device mask table.
IEFWCIMP	Task input/output table (TIOT) construction routine.
IEFWD000	External action routine.
IEFWD001	Message directory for external action routine.
IEFXT00D	Space request routine.
IEFXKIMP	Allocation error nonrecovery routine.
IEFW41SD	Exit to step initiation routine.
IEFSD004	Step initiation routine, with exit to processing program.
IEFSD006	Convert record number to logical track address (TTR).
IEFSD007	Call to table store subroutine.
IEFSD010	Dequeue and write out system message blocks (SMBs).
IEFXTDMY	Queue overflow routine.
IEFYSSMB	Message enqueuing routine.
IEFQMSSS	Table store subroutine.
IEFXAMSG	Contains initiator/terminator messages.
IEFXJMSG	Contains initiator/terminator messages.
IEFXKMSG	Contains initiator/terminator messages.
IEFXTMSG	Contains initiator/terminator messages.
IEFYNFAK	Linkage to IEFYNIMP (in IEFSTERM load module).

Load Module Name: IEFCNTRL
Alias: IEF5DDHD
Alias: IEFZA
Alias: IEFSD008
Alias: IEFMC
Entry Point: IEFKA
Assembly Modules:

IEFSPIE	Program check handling routine.
IEF7KAXX	Reader/interpreter control routine.
IEF6MCXX	Scans job control language (JCL) statements.
IEF6BOCM	Breakout routine.
IEF6NAME	Qualified name routine.
IEF6STNM	Scan stepname routine.
IEF6MFXX	Verb identification routine.
IEF6MKXX	Continuation routine.
IEF6NCJB	JOB statement routine.
IEF6NFCM	Accounting information routine.
IEF6N1JB	TYPRUN keyword routine.
IEF6NYJB	Priority (PRTY) keyword routine.
IEF6COND	Condition (COND) keyword routine.
IEF6NXJB	MSGLEVEL keyword routine.

Load Modules
(44K Configuration, Continued)

IEF6NZJB	MSGCLASS keyword routine.
IEF6NIJB	Parenthesis routine.
IEF6NJEX	EXEC statement routine.
IEF6RFBK	Refer-back routine.
IEF6PROC	Procedure name routine.
IEF6TIME	TIME keyword routine.
IEF6PARM	Parameter (PARM) keyword routine.
IEF6DDHD	DD statement routine.
IEF6SCAN	DD scan routine.
IEFSD012	DD* statement routine.
IEF6DDNM	DD name routine.
IEF6FRRS	Resolves DD forward references.
IEF6DSNM	DS name routine.
IEF6RFWD	Processes DD forward references.
IEF6LFPR	Left parenthesis routine.
IEF6RTPR	Right parenthesis routine.
IEF6EQUL	Equal sign routine.
IEF6LIST	Subparameter list routine.
IEF6NLST	Routine for no subparameter list.
IEF6NDDP	DD parameter list table.
IEF6NDDX	Alternative DD parameter list table (DDPLT).
IEF6DCB0	DCB refer-back routine.
IEF6DCDP	DCB DD parameter list table.
IEF6ORDR	Order subroutine.
IEF6INST	Insert routine.
IEF6VALU	Value subroutine.
IEF6CLNP	Clean up after DD routine.
IEF6ERR1	DD error-handling routine.
IEF7KPXX	Processes command in input stream.
IEEMCR01	Master command routine.
IEF7MMCM	Reader/interpreter message routine.
IEF6MIXX	Reader/interpreter call to table store subroutine.
IEFQMSSS	Table store subroutine.
IEF7KGXX	Output tables for step.
IEFSD013	Assigns unit for system output (SYSOUT).
IEFSD008	Initiator/terminator to reader/interpreter interface routine.
IEFSD001	Reader/interpreter entry to IEF09FAK or to IEFW23SD (both in this load module).
IEFW23SD	Performs initialization for job termination routine and exits to IEFZAJB3 (in this load module).
IEFSD006	Converts record number to logical track address (TTR).
IEFSD007	Call to table store subroutine.
IEEILCDM	Prevents unresolved IEEICCAN symbol after IPL time.
IEFK4DUM	Linkage to IEFK4ENT (in IEFK4 load module).
IEF09FAK	Linkage to IEFSD009 (in IEFSTERM load module).
IEF8LINK	Linkage to IEF6COND (in this load module).
IEFKLDUM	Linkage to IEF6KLXX (in IEF1STMT load module).

Load Modules
(44K Configuration, Continued)

IEF6OUT2	Linkage to IEF6SCAN (in this load module).
IEFK3DUM	Linkage to IEF7K3XX (in IEFEOF load module).
IEFZAFAK	Linkage to IEFZAJB3 (in load module IEFJTERM)

Load Module Name: IEFJTERM
Alias: IEFZA
Entry Point: IEFZA
Assembly Modules:

IEFZAJB3	Job termination routine.
IEFWTERM	Job ended message routine.
IEFZGJB1	Disposition and unallocation subroutine.
IEFACTLK	Linkage to user accounting routine.
IEFACTRT	Dummy routine to be replaced by user's accounting routine.

(The preceding two modules may be replaced by IEFACTFK assembly module if no accounting routine is specified as a system generation option.)

IEFSD006	Converts record number to logical track address (TTR).
IEFSD007	Call to table store subroutine.
IEFYSSMB	Message enqueuing routine enqueues SMB's.
IEFQMSSS	Table store subroutine.
IEFZGMSG	Contains initiator/terminator messages.
IEFZHMSG	Contains initiator/terminator messages.
IEFW31SD	Job termination exit to IEFSD003 (in this load module).
IEFSD003	Passes control to IEFSD010, in this load module.
IEFSD010	Dequeue and write out system message blocks (SMBs).
IEF08FAK	Linkage to IEFSD008 (in IEFCNTRL load module).
IEFZHFAK	Linkage to subroutine ZPOQMGR1, in IEFZGJB1 of this load module.

Load Module Name: IEF1STMT
Alias: IEFKL
Entry Point: IEFKL
Assembly Modules:

IEF6KLXX	First statement routine.
IEF7MMCM	Reader/interpreter message routine.
IEF6MIXX	Reader/interpreter call to table store subroutine.
IEFQMSSS	Table store subroutine.
IEFMCDUM	Linkage to IEF6MCXX (in IEFCNTRL load module).
IEFKADUM	Linkage to IEF7KAXX (in IEFCNTRL load module).

Load Module Name: IEFEOF
Alias: IEFK3
Entry Point: IEFK3
Assembly Modules:

IEFSPIE	Program check handling routine.

IEF7MMCM Reader/interpreter message routine.
IEF7K3XX Input stream end-of-file (EOF) routine.
IEF7K4XX Close devices routine.
IEF7K2XX Open devices routine.
IEFSD006 Converts record number to logical track address (TTR).
IEFSD007 Call to table store subroutine.
IEF6MIXX Reader/interpreter call to table store subroutine.
IEFQMSSS Table store subroutine.
IEEMCR01 Master command routine.
IEEILCDM Prevent unresolved IEEICCAN symbol after IPL.
IEFKADUM Linkage to IEF7KAXX (in IEFCNTRL load module).

Load Module Name: IEFK4
Entry Point: IEFK4DUM
IEFSPIE Program check handling routine.
IEF7MMCM Reader/interpreter message routine.
IEFK4ENT Switch input readers routine.
IEF7K4XX Close devices routine.
IEF7K2XX Open devices routine.
IEFSD006 Convert record number to logical track address (TTR).
IEFSD007 Call to table store subroutine.
IEF6MIXX Reader/interpreter call to table store subroutine.
IEFQMSSS Table store subroutine.

Load Module Name: IEFERROR
Alias: IEFVM6LS
Entry Point: IEFVMSGR
Assembly Modules:
IEFVMLS6 JFCB housekeeping error message processing routine.
IEFYSSMB Message enqueuing routine, enqueues SMBs.
IEFQMSSS Table store subroutine.
IEFVMLS7 Contains initiator terminator messages
IEFYNFAK Linkage to IEFYNIMP (in IEFSTERM load module).

Load Module Name: IEFIDUMP
Entry Point: IEFIDUMP
Assembly Modules:
IEFIDUMP Indicative dump routine.
IEFYSSMB Message enqueuing routine, enqueues SMBs.
IEFQMSSS Table store subroutine.
IEFIDMPM Contains initiator terminator messages.
IEFYNFAK Linkage to IEFYNIMP (in IEFSTERM load module).

Load Module Name: IEFDCB
Alias: IEF5DCDP
Assembly Module:
IEF6DCDP DCB DD parameter list table.

Load Modules
(44K Configuration, Continued)

Load Module Name: IEFMSG01
Assembly Module:
IEF3MSG1 Contains reader/interpreter messages.

Load Module Name: IEFMSG02
Assembly Module:
IEF3MSG2 Contains reader/interpreter messages.

Load Module Name: IEFMSG03
Assembly Module:
IEF3MSG3 Contains reader/interpreter messages.

Load Module Name: IEFMSG04
Assembly Module:
IEF3MSG4 Contains reader/interpreter messages.

Load Module Name: IEFMSG05
Assembly Module:
IEF3MSG5 Contains reader/interpreter messages.

Load Module Name: IEFMSG06
Assembly Module:
IEF3MSG6 Contains reader/interpreter messages.

Load Module Name: IEFMSG07
Assembly Module:
IEF3MSG7 Contains reader/interpreter messages.

Load Module Name: IEFINITL
Alias: IEFK1
Entry Point: IEFK1

Assembly Modules:
IEFSPIE Program check handling routine.
IEF7MMCM Reader/interpreter message routine.
IEF7K1XX Entry to job management from nucleus initialization program (NIP).
IEFPRES Volume attribute initialization routine.
IEFKIMSG IEFPRES messages.
IEFK1MSG Reader/interpreter message routine.
IEEMCR01 Master command routine.
IEEILC01 Automatic command routine.
IEF7K2XX Open devices routine.
IEFWSDIP Linkage control table (LCT) initialization.
IEFSD006 Converts record number to logical track address (TTR).
IEFSD007 Call to table store subroutine.
IEF6MIXX Reader/interpreter call to table store subroutine.
IEFQMSSS Table store subroutine.
IEFKADUM Linkage to IEF7KAXX (in IEFCNTRL load module).

Load Modules
(44K Configuration, Continued)

Load Module Name: IEESET
Alias: IEEGESTO
Assembly Module:
IEEGES01 Master scheduler SET command routine.

Load Module Name: IEFJOBQE
Alias: IEFINTQS
Assembly Module:
IEFINTQA Initializes SYS1.SYSJOBQE data set.

Load Module Name: IEETIME
Alias: IEEQOT00
Assembly Module:
IEEQOT00 Sets time and date.

Load Module Name: IEEFAULT
Alias: IEEGK1GM
Assembly Module:
IEEGK1GM Fault routine, issues master scheduler messages.

Load Module Name: IEESTART
Alias: IEEIC1PE

Load Modules
(44K Configuration, Continued)

Entry Point: IEEIC1PE
Assembly Modules:
IEESTART START command routine.
IEEREADR Start reader routine.
IEEWRITR Start writer routine.

Load Module Name: IEEJFCB
Alias: IEEIC3JF
Assembly Module:
IEEIC3JF Contains preformatted JFCB for one START command.

Load Module Name: IEESJFCB
Alias: IEEIC2NQ
Entry Point: IEEIC2NQ
Assembly Modules:
IEEIC2NQ Save JFCBs for START commands.
IEFQMSSS Table store subroutine.

Load Module Name: IEFPRINT
Alias: SPRINTER
Alias: IEFPRT
Assembly Module:
IEFPRTXX Tape SYSOUT to printer.

100K CONFIGURATION

Load Module Name: IEFCNTRL
Alias: IEFKA
Alias: GO
Alias: IEFYN
Entry Point: IEFSD011
Assembly Modules:
IEFSPIE Program check handling routine.
IEFSD011 Entry to job management from supervisor.
IEFW42SD Passes control to IEFIDUMP if needed, or to IEFYNIMP (both in this load module).
IEFIDUMP Indicative dump routine.
IEFYNIMP Step termination routine.
IEFYPJB3 Step data set driver routine.
IEFVJIMP JOB statement condition code routine.
IEFZGST1 Disposition and unallocation subroutine.
IEFSD017 Places logical track address of first system message block (SMB) into job control table (JCT).
IEFW22SD Passes control to IEFYNIMP assembly module, and then to IEFSD002 or IEFZAJB3 (all in this load module).
IEFSD002 Exit to IEFSD008 or to IEFSD009 (both in this load module).
IEFSD008 Initiator/terminator to reader/interpreter interface.
IEF7KAXX Reader/interpreter control routine.
IEF6MCXX Job control language (JCL) statement scan routine.
IEF6BOCM Breakout routine.
IEF6NAME Qualified name routine.
IEF6STNM Scan stepname routine.
IEF6MFXX Verb identification routine.
IEF6MKXX Continuation routine.
IEF6NCJB JOB statement routine.
IEF6NFCM Accounting information routine.
IEF6N1JB TYPRUN keyword routine.
IEF6NYJB Priority (PRTY) keyword routine.
IEF6COND Condition (COND) keyword routine.
IEF6NXJB MSGLEVEL keyword routine.
IEF6NZJB MSGCLASS keyword routine.
IEF6NIJB Parenthesis routine.
IEF6NJEX EXEC statement routine.
IEF6RFBK Refer-back routine.
IEF6PROC Procedure name routine.
IEF6TIME TIME keyword routine.
IEF6PARM Parameter (PARM) keyword routine.
IEF6DDHD DD statement routine.
IEF6SCAN DD scan routine.
IEFSD012 DD* statement routine.
IEF6DDNM DD name routine.
IEF6FRRS Resolves DD forward references.
IEF6DSNM DS name routine.
IEF6RFWD Processes DD forward references.
IEF6LFPR Left parenthesis routine.
IEF6RTPR Right parenthesis routine.
IEF6EQUL Equal sign routine.
IEF6LIST Subparameter list routine.
IEF6NLST No subparameter list routine.

Load Modules
(100K Configuration, Continued)

IEF6NDDP	DD parameter list table.
IEF6NDDX	Alternative DD parameter list table (DDPLT).
IEF6DCB0	DCB refer-back routine.
IEF6DCDP	DCB DD parameter list table.
IEF6ORDR	Order subroutine.
IEF6INST	Insert routine.
IEF6VALU	Value subroutine.
IEF6CLNP	Clean up after DD routine.
IEF6ERR1	DD error-handling routine.
IEF7KPXX	Command in input stream.
IEF7K3XX	Input stream end-of-file (EOF) routine.
IEF7K4XX	Close devices routine.
IEFK4ENT	Switch input readers routine.
IEF7K2XX	Open devices routine.
IEF6KLXX	First statement routine.
IEEMCR01	Master command routine.
IEF7MMCM	Reader/interpreter message routine.
IEF6OUT2	Linkage to IEF6SCAN (in this load module).
IEF7KGXX	Output tables for step.
IEFSD013	Assign unit for system output (SYSOUT).
IEFSD001	Reader/interpreter entry to IEFSD009 or to IEFW23SD (both in this load module).
IEFSD009	Initiator/terminator initialization of output unit.
IEFW21SD	System control routine.
IEFVKIMP	EXEC statement condition code routine.
IEFVMSL1	JFCB housekeeping control routine.
IEFVM2LS	JFCB H/K fetch data control block (DCB) routine.
IEFVM3LS	JFCB H/K generation data group (GDG) single routine.
IEFVM4LS	JFCB H/K generation data group (GDG) all routine.
IEFVM5LS	JFCB H/K patterning data set control block (DSCB) routine.
IEFVMLS6	JFCB H/K error message processing routine.
IEFVM76	Processes passed, non-labeled tape data sets.
IEFWSTRT	Job started message routine.
IEFWMAS1	Device name table.
IEFXCSSS	Allocation control routine.
IEFXJIMP	Allocation error recovery routine.
IEFWA000	Demand allocation routine.
IEFWSWIN	Passes control to decision allocation or AVR routine.
IEFXV001	Automatic volume recognition.
IEFXV002	Processes new volumes (AVR).
IEFXV003	Processes specific requests for unmounted volumes.
IEFXV004	AVR unloading routine.
IEFXVNSL	AVR volume serial routine.
IEFXVMSG	AVR message routine.
IEFX5000	Decision allocation routine.
IEFX300A	Device strikeout routine.

Load Modules
(100K Configuration, Continued)

IEFXH000	Separation strikeout routine.
IEFWMSKA	Device mask table.
IEFWCIMP	Task input/output table (TIOT) construction routine.
IEFWD000	External action routine.
IEFWD001	Message directory for external action routine.
IEFXT00D	Space request routine.
IEFXTDMY	Queue overflow routine.
IEFXKIMP	Allocation error nonrecovery routine.
IEFW41SD	Exit to step initiation routine.
IEFSD004	Step initiation routine, with exit to processing program.
IEFW23SD	Initializes for job termination and exits to IEFZAJB3 (in this load module).
IEFZAJB3	Job termination routine.
IEFWTERM	Job ended message routine.
IEFZGJB1	Disposition and unallocation subroutine.
IEFACTLK	Linkage to user's accounting routine.
IEFACTRT	Dummy routine to be replaced by user's accounting routine.

(The preceding two modules may be replaced by IEFACTFK assembly module if no accounting routine is specified as a system generation option.)

IEFW31SD	Job termination exit to IEFSD003.
IEFSD003	Passes control to IEFSD010 and then goes to IEFSD008 (both in this load module).
IEFSD010	Dequeues and writes out system message blocks (SMBs).
IEFYSSMB	Message enqueuing routine, enqueues SMBs.
IEFSD006	Converts record number to logical track address (TTR).
IEFSD007	Call to table store subroutine.
IEF6MIXX	Reader/interpreter call to IEFQMSSS.
IEFQMSSS	Table store subroutine.
IEEILCDM	Prevents unresolved IEEICCAN symbol after initialization (Job management IPL).
IEF8LINK	Linkage to IEF6COND (in this module).
IEFIDMPM	Contains initiator/terminator messages.
IEFVJMSG	Contains initiator/terminator messages.
IEFVKMSG	Contains initiator/terminator messages.
IEFVMLS7	Contains initiator/terminator messages.
IEFXAMSG	Contains initiator/terminator messages.
IEFXJMSG	Contains initiator/terminator messages.
IEFXKMSG	Contains initiator/terminator messages.
IEFXTMSG	Contains initiator/terminator messages.

Load Modules
(100K Configuration, Continued)

IEFYNMSG Contains initiator/terminator messages.
IEFYPMSG Contains initiator/terminator messages.
IEFZGMSG Contains initiator/terminator messages.
IEFZHMSG Contains initiator/terminator messages.

Load Module Name: IEFDCB
Alias: IEF5DCDP
Assembly Module:
IEF6DCDP Data control block (DCB) DD parameter list table.

Load Module Name: IEFMSG01
Assembly Module:
IEF3MSG1 Contains reader/interpreter messages.

Load Module Name: IEFMSG02
Assembly Module:
IEF3MSG2 Contains reader/interpreter messages.

Load Module Name: IEFMSG03
Assembly Module:
IEF3MSG3 Contains reader/interpreter messages.

Load Module Name: IEFMSG04
Assembly Module:
IEF3MSG4 Contains reader/interpreter messages.

Load Module Name: IEFMSG05
Assembly Module:
IEF3MSG5 Contains reader/interpreter messages.

Load Module Name: IEFMSG06
Assembly Module:
IEF3MSG6 Contains reader/interpreter messages.

Load Module Name: IEFMSG07
Assembly Module:
IEF3MSG7 Contains reader/interpreter messages.

Load Module Name: IEFINITL
Alias: IEFK1
Entry Point: IEFK1
Assembly Modules:
IEFSPIE Program check handling routine.
IEF7MMCM Reader/interpreter message routine.
IEF7K1XX Initial entry to job management from nucleus initialization program (NIP).
IEFPRES Volume attribute initialization routine.
IEFKIMSG IEFPRES messages.
IEFK1MSG Reader/interpreter message routine.

Load Modules
(100K Configuration, Continued)

IEEMCR01 Master command routine.
IEEILC011 Automatic command routine
IEF7K2XX Open devices routine.
IEFWSDIP Linkage control table (LCT) initialization.
IEFSD006 Converts record number to logical track address (TTR).
IEFSD007 Call to table store subroutine.
IEF6MIXX Reader/interpreter call to table store subroutine.
IEFQMSSS Table store subroutine.
IEFKADUM Linkage to IEF7KAXX (in IEFCNTRL load module).

Load Module Name: IEESET
Alias: IEEGESTO
Assembly Modules:
IEEGES01 Master scheduler SET command routine.

Load Module Name: IEFJOBQE
Alias: IEFINTQS
Assembly Module:
IEFINTQA Initializes SYS1.SYSJOBQE data set.

Load Module Name: IEETIME
Alias: IEEQOT00
Assembly Module:
IEEQOT00 Sets time and date in response to SET command.

Load Module Name: IEEFAULT
Alias: IEEGK1GM
Assembly Modules:
IEEGK1GM Fault routine, issues master scheduler messages.

Load Module Name: IEESTART
Alias: IEEIC1PE
Entry Point: IEEIC1PE
Assembly Modules:
IEESTART START command routine.
IEEREADR Start reader routine.
IEEWRITR Start writer routine.

Load Module Name: IEEJFCB
Alias: IEEIC3JF
Assembly Module:
IEEIC3JF Contains preformatted JFCB for one START command.

Load Module Name: IEESJFCB
Alias: IEEIC2NQ
Entry Point: IEEIC2NQ
Assembly Modules:
IEEIC2NQ Save START command JFCB.
IEFQMSSS Table store subroutine.

Load Module Name: IEFPRINT
Alias: SPRINTER
Alias: IEFPRT
Assembly Module:
IEFPRTXX Transfers tape system output (SYSOUT) to printer.

ASSEMBLY MODULES AND CONTROL SECTIONS

The following table shows in which load modules each assembly module is used in the three configurations of job management. The first column lists the assembly module names in alphameric order. Except as indicated, all assembly modules are contained in load modules in the SYS1.LINKLIB data set. The second column lists the control section names that correspond to the assembly module names in the first column. The next three columns of the table indicate which load modules of each configuration contain each assembly module. The two right-hand columns refer to the CHARTS section. If a control section is shown as a subroutine block, the flowchart number is listed in the "Appears As Subr. Block" column; if the flow within a control section is given in a chart, the flowchart number is listed in the "Flow is Defined" column.

Assembly Modules and Control Sections (Part 1 of 6)

Assembly Module Name	Notes	Control Section Name	Load Modules in Which Assembly Modules are Used			Chart Number	
			18K	44K	100K	Appears As Subr. Block	Flow is Defined
IEEBC1PE	*	IEEBC1PE				02	07
IEEBH1PE	Not Used	IEEBH1	IGC0003F	IGC0003F	IGC0003F		
IEECIR01	*	IEEBA1				02	03
IEEGES01		IEEGESTO	IEESET	IEESET	IEESET	05	
IEEGK1GM		IEEGK1GM	IEEFAULT	IEEFAULT	IEEFAULT	05,15	
IEEIC2NQ		IEEIC2NQ	IEESJFCB	IEESJFCB	IEESJFCB		
IEEIC3JF	**	IEEIC3JF	IEEJFCB	IEEJFCB	IEEJFCB		
IEEILCDM		IEEICCAN	IEFINTFC	IEFCNTRL	IEFCNTRL	05	
			IEFCOMND	IEFEOF			
			IEFEOF				
IEEILC01	**	IEEICCAN	IEFINITL	IEFINITL	IEFINITL		
IEEMCR01		IEEBB1	IEFINTFC	IEFCNTRL	IEFCNTRL	02,09	05
			IEFCOMND	IEFINITL	IEFINITL		
			IEFINITL	IEFEOF			
			IEFEOF				
IEEMXC01	***	IGC03D	IGC0003D	IGC0003D	IGC0003D	02	04
IEEMXR01	***	IGC03D	IGC0003D	IGC0003D	IGC0003D	02	04
IEEQOT00		IEEQOT00	IEETIME	IEETIME	IEETIME		
IEEREADR		IEEICRDR	IEESTART	IEESTART	IEESTART		
IEERSC01	*	IEEMSLT					
IEERSR01	*	IEEMSLT					
IEESTART		IEEIC1PE	IEESTART	IEESTART	IEESTART		
IEEWRITR		IEECWTR	IEESTART	IEESTART	IEESTART		
IEEWTC01	***	IGC03E	IGC0003E	IGC0003E	IGC0003E		06
IEEWTR01	***	IGC03E	IGC0003E	IGC0003E	IGC0003E		06
IEFACTFK	****	IEFACTFK	IEFSTERM	IEFSTERM	IEFCNTRL		
			IEFJTERM	IEFCNTRL			
IEFACTLK	****	IEFACTLK	IEFSTERM	IEFSTERM	IEFCNTRL		
			IEFJTERM	IEFJTERM			
IEFACTRT	****	IEFACTRT	IEFSTERM	IEFSTERM	IEFCNTRL	34	
			IEFJTERM	IEFJTERM			
IEFDPOST	*	IEFDPOST					
IEFEMDUM		IEFEM	IEFCNTRL				
IEFIDFAK		IEFIDUMP	IEFSTERM	IEFSTERM			
IEFIDMPM		IEFIDMPM	IEFIDUMP	IEFIDUMP	IEFCNTRL		
IEFIDUMP		IEFIDUMP	IEFIDUMP	IEFIDUMP	IEFCNTRL		
IEFINTQA		IEFINTQS	IEFJOBQE	IEFJOBQE	IEFJOBQE		
IEFJMDUM		IEFJM	IEFCNTRL				
IEFKADUM		IEFKA	IEFPRES	IEFINITL	IEFINITL		
			IEFEOF	IEFEOF			
			IEFINTFC	IEF1STMT			
			IEFCOMND				
IEFKGDUM		IEFKG	IEFCNTRL				

(Part 1 of 6)

Assembly Modules and Control Sections (Part 2 of 6)

Assembly Module Name	Notes	Control Section Name	Load Modules in Which Assembly Modules are Used			Chart Number	
			18K	44K	100K	Appears As Subr. Block	Flow is Defined
IEFKLDUM		IEFKL	IEFCNTRL	IEFCNTRL			
IEFKPDUM		IEFKP	IEFCNTRL				
IEFKRESA	*	IEFJOB					
IEFKIMSG		IEFKIMSG	IEFPRES	IEFINITL	IEFINITL		
IEFK1MSG		IEFK1MSG	IEFINITL	IEFINITL	IEFINITL		
IEFK3DUM		IEFK3	IEFCNTRL	IEFCNTRL			
			IEFINTFC				
IEFK4DUM		IEFK4	IEFCNTRL	IEFCNTRL			
IEFK4ENT		IEFK4DUM	IEFK4	IEFK4	IEFCNTRL		
IEFMCDUM		IEFMC	IEF1STMT	IEF1STMT			
IEFMFDUM		IEFMF	IEFINTFC				
			IEFEXEC				
			IEFJOB				
IEFPRES		IEFPRES	IEFPRES	IEFINITL	IEFINITL		
IEFPRFAK		IEFPRES	IEFINITL				
IEFPRTXX		SPRINTER	IEFPRINT	IEFPRINT	IEFPRINT		
IEFQMSSS		IEFQMSSS	IEFSTERM	IEFSTERM	IEFCNTRL	12,16	
			IEFSELCT	IEFALLOC	IEFINITL		
			IEFALOC1	IEFCNTRL			
			IEFALOC4	IEF1STMT			
			IEFALOC5				
			IEFATACH	IEFEOF			
			IEFCNTRL	IEFK4			
			IEFDD	IEFERROR			
			IEFINTFC	IEFIDUMP			
			IEFEXEC	IEFINITL			
			IEFJOB	IEESJFCB			
			IEFJTERM	IEFJTERM			
			IEFCOMND				
			IEF1STMT				
			IEFEOF				
			IEFK4				
			IEFERROR				
			IEFIDUMP				
			IEFINITL				
			IEESJFCB				
IEFSD001		IEFSD001	IEFINTFC	IEFCNTRL	IEFCNTRL	09	
IEFSD002		IEFSD002	IEFSTERM	IEFSTERM	IEFCNTRL		
IEFSD003		IEFSD003	IEFJTERM	IEFJTERM	IEFCNTRL		
IEFSD004		IEFSD004	IEFATACH	IEFALLOC	IEFCNTRL		36
IEFSD006		IEFSD006	IEFSTERM	IEFSTERM	IEFCNTRL		
			IEFALOC2	IEFALLOC	IEFINITL		
			IEFALOC4	IEFCNTRL			
			IEFALOC5				
			IEFATACH	IEFEOF			
			IEFCNTRL	IEFK4			
			IEFDD	IEFINITL			
			IEFINTFC	IEFJTERM			
			IEFJTERM				
			IEFCOMND				
			IEFEOF				
			IEFK4				
			IEFINITL				
IEFSD007		IEFSD007	IEFSTERM	IEFSTERM	IEFCNTRL		
			IEFATACH	IEFALLOC	IEFINITL		
			IEFINTFC	IEFCNTRL			
			IEFJTERM	IEFEOF			
			IEFEOF	IEFK4			
			IEFK4	IEFINITL			

(Part 2 of 6)

Assembly Modules and Control Sections (Part 3 of 6)

Assembly Module Name	Notes	Control Section Name	Load Modules in Which Assembly Modules are Used			Chart Number	
			18K	44K	100K	Appears As Subr. Block	Flow is Defined
			IEFINITL	IEFJTERM			
IEFSD008		IEFSD008	IEFINTFC	IEFCNTRL	IEFCNTRL		09
IEFSD009		IEFSD009	IEFSELCT	IEFSTERM	IEFCNTRL		
IEFSD010		IEFSD010	IEFATACH	IEFALLOC	IEFCNTRL		
			IEFJTERM	IEFJTERM			
IEFSD011		IEFSD011	IEFSTERM	IEFSTERM	IEFCNTRL	37	40
IEFSD012		IEFSD012	IEFDD	IEFCNTRL	IEFCNTRL		
IEFSD013		IEFSD013	IEFDD	IEFCNTRL	IEFCNTRL		
IEFSD017		IEFSD017	IEFSTERM	IEFSTERM	IEFCNTRL		
IEFSPIE	x	IEFSPIE	IEFINITL	IEFINITL	IEFINITL		
			IEFCNTRL	IEFCNTRL	IEFCNTRL		
			IEFEOF	IEFEOF			
			IEF1STMT	IF1STMT			
			IEFINTFC	IEFNEWRD			
			IEFJOB				
			IEFEXEC				
			IEFDD				
			IEFK4				
IEFVJIMP		IEFVJ	IEFSTERM	IEFSTERM	IEFCNTRL	38	39
IEFVJMSG		IEFVJMSG	IEFSTERM	IEFSTERM	IEFCNTRL		
IEFVKIMP		IEFVK	IEFSELCT	IEFSTERM	IEFCNTRL	14	16
IEFVKMSG		IEFVKMSG	IEFSELCT	IEFSTERM	IEFCNTRL		
IEFVMLK5		IEFVM6	IEFSELCT	IEFSTERM			
IEFVMLS1		IEFVM1	IEFSELCT	IEFSTERM	IEFCNTRL	17	18
IEFVMLS6		IEFVM6	IEFERROR	IEFERROR	IEFCNTRL	17,18	24
IEFVMLS7		IEFVM7	IEFERROR	IEFERROR	IEFCNTRL		
IEFVM2LS		IEFVM2	IEFSELCT	IEFSTERM	IEFCNTRL	17,18	20
IEFVM3LS		IEFVM3	IEFSELCT	IEFSTERM	IEFCNTRL	17,18	21
IEFVM4LS		IEFVM4	IEFSELCT	IEFSTERM	IEFCNTRL	17,18	22
IEFVM5LS		IEFVM5	IEFSELCT	IEFSTERM	IEFCNTRL	17,18	23
IEFVM76		IEFVM76	IEFSELCT	IEFSTERM	IEFCNTRL		
IEFWAFAK		IEFWA000	IEFALOC1				
IEFWA000		IEFWA7	IEFALOC2	IEFALLOC	IEFCNTRL	25	27
IEFWCFAK		IEFWC000	IEFALOC1				
			IEFALOC2				
IEFWCIMP		IEFWC000	IEFALOC3	IEFALLOC	IEFCNTRL	25	33
		IEFWC002	IEFALOC3	IEFALLOC	IEFCNTRL		
IEFWDFAK		IEFWD000	IEFALOC3				
			IEFALOC5				
IEFWD000		IEFWD000	IEFALOC4	IEFALLOC	IEFCNTRL	25,28	34
IEFWD001		IEFWD001	IEFALOC4	IEFALLOC	IEFCNTRL		
IEFWMAS1	**	DEVNAMET	IEFSELCT	IEFSTERM	IEFCNTRL		
IEFWMSKA	**	DEVMASKT	IEFALOC2	IEFALLOC	IEFCNTRL		
IEFWSDIP		IEFWSDIP	IEFINITL	IEFINITL	IEFINITL		
IEFWSTRT		IEFWSTRT	IEFSELCT	IEFSTERM	IEFCNTRL		
IEFSWIN		IEFSWIT	IEFALOC2	IEFALLOC	IEFCNTRL		
IEFWTERM		IEFWTERM	IEFJTERM	IEFJTERM	IEFCNTRL		
IEFW21SD		IEFW21SD	IEFSELCT	IEFSTERM	IEFCNTRL	14	15
IEFW22SD		IEFW22SD	IEFSTERM	IEFSTERM	IEFCNTRL	38	
IEFW23SD		IEFW23SD	IEFJTERM	IEFCNTRL	IEFCNTRL		
IEFW31SD		IEFW31SD	IEFJTERM	IEFJTERM	IEFCNTRL		
IEFW41SD		IEFW41SD	IEFALOC5	IEFALLOC	IEFCNTRL		
IEFW42SD		IEFW42SD	IEFSTERM	IEFSTERM	IEFCNTRL		
IEFXAFAK		IEFXA	IEFSELCT	IEFSTERM			
IEFXAMSG		IEFXAMSG	IEFALOC1	IEFALLOC	IEFCNTRL		
IEFXCSSS		IEFXA	IEFALOC1	IEFALLOC	IEFCNTRL	25	26
IEFXH000		IEFXH000	IEFALOC2	IEFALLOC	IEFCNTRL		
			IEFALOC3				

(Part 3 of 6)

Assembly Modules and Control Sections (Part 4 of 6)

Assembly Module Name	Notes	Control Section Name	Load Modules in Which Assembly Modules are Used			Chart Number	
			18K	44K	100K	Appears As Subr. Block	Flow is Defined
IEFXJFAK		IEFXJ000	IEFALOC2				
			IEFALOC3				
IEFXJIMP		IEFXJ000	IEFALOC1	IEFALLOC	IEFCNTRL	28	
IEFXJMSG		IEFXJMSG	IEFALOC1	IEFALLOC	IEFCNTRL		
IEFXKIMP		IEFXK000	IEFALOC4	IEFALLOC	IEFCNTRL		
			IEFALOC5				
IEFXKMSG		IEFXKMSG	IEFALOC4	IEFALLOC	IEFCNTRL		
			IEFALOC5				
IEFXTFAK		IEFXT000	IEFALOC4				
IEFXTDMY		IEFXTDMY	IEFALOC5	IEFALLOC	IEFCNTRL		
IEFXTMSG		IEFXTMSG	IEFALOC5	IEFALLOC	IEFCNTRL		
IEFXT00D		IEFXT000	IEFALOC5	IEFALLOC	IEFCNTRL	25	35
IEFXVMSG		IEFXVMSG	IEFALOC4	IEFALLOC	IEFCNTRL		
IEFXVNSL		IEFXVNSL	IEFALOC4	IEFALLOC	IEFCNTRL		
IEFXV001		IEFXV001	IEFALOC4	IEFALLOC	IEFCNTRL		28
IEFXV002		IEFXV002	IEFALOC4	IEFALLOC	IEFCNTRL		
IEFXV003		IEFXV003	IEFALOC4	IEFALLOC	IEFCNTRL		
IEFXV004		IEFXV004	IEFALOC4	IEFALLOC	IEFCNTRL		
IEFXVFAK		IEFXV001	IEFALOC2				
IEFX1FAK		IEFXJ000	IEFALOC4				
IEFX2FAK		IEFX5000	IEFALOC4				
IEFX3FAK		IEFWC000	IEFALOC4				
IEFX300A		IEFX3000	IEFALOC2	IEFALLOC	IEFCNTRL		
			IEFALOC4				
IEFX5000		IEFX5000	IEFALOC2	IEFALLOC	IEFCNTRL	25	32
IEFYNFAK		IEFYN	IEFSELCT	IEFALLOC			
			IEFALOC1	IEFERROR			
			IEFALOC4	IEFIDUMP			
			IEFALOC5				
			IEFERROR				
			IEFIDUMP				
IEFYNIMP		IEFYN	IEFSTERM	IEFSTERM	IEFCNTRL		
IEFYNMSG		IEFYNMSG	IEFSTERM	IEFSTERM	IEFCNTRL		
IEFYPJB3		IEFYP	IEFSTERM	IEFSTERM	IEFCNTRL	38	41
IEFYPMSG		IEFYPMSG	IEFSTERM	IEFSTERM	IEFCNTRL		
IEFYSSMB		IEFYS	IEFSTERM	IEFSTERM	IEFCNTRL		
			IEFSELCT	IEFALLOC			
			IEFALOC1				
			IEFALOC4	IEFERROR			
			IEFALOC5				
			IEFJTERM	IEFIDUMP			
			IEFERROR	IEFTERM			
			IEFIDUMP				
IEFZAFAK		IEFZA	IEFSTERM	IEFSTERM			
IEFZAJB3		IEFZA	IEFJTERM	IEFTERM	IEFCNTRL	37	40
IEFZGJB1		IEFZGJ	IEFJTERM	IEFTERM	IEFCNTRL	40	42
IEFZGMSG		IEFZGMSG	IEFSTERM	IEFSTERM	IEFCNTRL		
			IEFJTERM	IEFTERM			
IEFZGST1		IEFZG	IEFSTERM	IEFSTERM	IEFCNTRL		
IEFZHFAK		IEFZPOQM	IEFJTERM	IEFTERM			
IEFZHMSG		IEFZH	IEFSTERM	IEFSTERM	IEFCNTRL	41,42	
			IEFJTERM	IEFTERM			
IEF04FAK		IEFSD004	IEFALOC5				
IEF08FAK		IEFSD008	IEFSTERM	IEFSTERM			
			IEFINTFC	IEFTERM			
			IEFJTERM				
IEF09FAK		IEFSD009	IEFSTERM	IEFCNTRL			
			IEFINTFC				

(Part 4 of 6)

Assembly Modules and Control Sections (Part 5 of 6)

Assembly Module Name	Notes	Control Section Name	Load Modules in Which Assembly Modules are Used			Chart Number	
			18K	44K	100K	Appears As Subr. Block	Flow is Defined
IEF23FAK		IEFW23SD	IEFINTFC				
IEF3MSG1		IEFMSG1	IEFMSG01	IEFMSG01	IEFMSG01		
IEF3MSG2		IEFMSG2	IEFMSG02	IEFMSG02	IEFMSG02		
IEF3MSG3		IEFMSG3	IEFMSG03	IEFMSG03	IEFMSG03		
IEF3MSG4		IEFMSG4	IEFMSG04	IEFMSG04	IEFMSG04		
IEF3MSG5		IEFMSG5	IEFMSG05	IEFMSG05	IEFMSG05		
IEF3MSG6		IEFMSG6	IEFMSG06	IEFMSG06	IEFMSG06		
IEF3MSG7		IEFMSG7	IEFMSG07	IEFMSG07	IEFMSG07		
IEF6BOCM		IEFBM	IEFCNTRL	IEFCNTRL	IEFCNTRL		
			IEFDD				
			IEFJOB				
			IEFEXEC				
IEF6CLNP		IEFMO2	IEFDD	IEFCNTRL	IEFCNTRL		
IEF6CN17		INDMRTN	IEFDD				
IEF6COND		IEF6COND	IEFEXEC	IEFCNTRL	IEFCNTRL		
			IEFJOB				
IEF6DCB0		IEF6DCB0	IEFCNTRL	IEFCNTRL	IEFCNTRL		
IEF6DCDP		INDMB	IEFDD	IEFCNTRL	IEFCNTRL		
			IEFDCB	IEFDCB	IEFDCB		
IEF6DDHD		INDMRTN	IEFCNTRL	IEFCNTRL	IEFCNTRL	12	
IEF6DDNM		INDMO1	IEFDD	IEFCNTRL	IEFCNTRL		
IEF6DHX1		INDMON	IEFCNTRL				
IEF6DSNM		INDMO3	IEFDD	IEFCNTRL	IEFCNTRL		
IEF6EQUL		INDMOP	IEFDD	IEFCNTRL	IEFCNTRL		
IEF6ERR1		IEF6ERR1	IEFDD	IEFCNTRL	IEFCNTRL		
IEF6FRRS		IEF6FRRS	IEFCNTRL	IEFCNTRL	IEFCNTRL		
IEF6INST		INDMOZ	IEFDD	IEFCNTRL	IEFCNTRL		
IEF6KLXX		IEFKL	IEF1STMT	IEF1STMT	IEFCNTRL		
IEF6LFPR		INDMOS1	IEFDD	IEFCNTRL	IEFCNTRL		
IEF6LIST		INDMOY	IEFDD	IEFCNTRL	IEFCNTRL		
IEF6MCXX		IEFMC	IEFCNTRL	IEFCNTRL	IEFCNTRL	09	
IEF6MFXX		IEFMF	IEFCNTRL	IEFCNTRL	IEFCNTRL	09	
IEF6MIXX		IEFMI	IEFCNTRL	IEF1STMT	IEFCNTRL		
			IEFDD	IEFK4	IEFINITL		
			IEFINTFC	IEFEOF			
IEF6MIXX			IEFEXEC	IEFINITL			
(Cont.)			IEFJOB	IEFCNTRL			
			IEFCOMND				
			IEF1STMT				
			IEFEOF				
			IEFK4				
			IEFINITL				
IEF6MKXX		IEFMK	IEFCNTRL	IEFCNTRL	IEFCNTRL		
IEF6NAME		INNAME	IEFCNTRL	IEFCNTRL	IEFCNTRL		
			IEFEXEC				
			IEFJOB				
IEF6NCJB	**	IEFJM	IEFJOB	IEFCNTRL	IEFCNTRL	08	10
IEF6NDDP		INDMA	IEFDD	IEFCNTRL	IEFCNTRL		
IEF6NFCM		IEFAM	IEFEXEC	IEFCNTRL	IEFCNTRL		
			IEFJOB				
IEF6NIJB		IEFNI	IEFJOB	IEFCNTRL	IEFCNTRL		
IEF6NJEX		IEFEM	IEFEXEC	IEFCNTRL	IEFCNTRL	08	11
IEF6NLST		INDMOX	IEFDD	IEFCNTRL	IEFCNTRL		
IEF6NXJB		IEFNX	IEFJOB	IEFCNTRL	IEFCNTRL		
IEF6NYJB		IEFNY	IEFJOB	IEFCNTRL	IEFCNTRL		
IEF6NZJB		IEFNZ	IEFJOB	IEFCNTRL	IEFCNTRL		
IEF6N1JB		IEFN1	IEFJOB	IEFCNTRL	IEFCNTRL		
IEF6ORDR		INDMOV	IEFDD	IEFCNTRL	IEFCNTRL		
IEF6OUT2		INDMOH	IEFCNTRL	IEFCNTRL	IEFCNTRL	12	

Assembly Modules and Control Sections (Part 6 of 6)

Assembly Module Name	Notes	Control Section Name	Load Modules in Which Assembly Modules are Used			Chart Number	
			18K	44K	100K	Appears As Subr. Block	Flow is Defined
IEF6PARM		IEFPARM	IEFEXEC	IEFCNTRL	IEFCNTRL		
IEF6PROC		IEFPROC	IEFEXEC	IEFCNTRL	IEFCNTRL		
IEF6RFBK		IEFRFBK	IEFEXEC	IEFCNTRL	IEFCNTRL		
IEF6RFWD		IEF6RFWD	IEFDD	IEFCNTRL	IEFCNTRL		
IEF6RTPR		INDMOR	IEFDD	IEFCNTRL	IEFCNTRL		
IEF6SCAN		INDMON	IEFDD	IEFCNTRL	IEFCNTRL	08	12
IEF6STNM		IEFSTNM	IEFCNTRL	IEFCNTRL	IEFCNTRL		
			IEFEXEC				
			IEFJOB				
IEF6TIME		IEFTIME	IEFEXEC	IEFCNTRL	IEFCNTRL		
IEF6VALU		INDMOT	IEFDD	IEFCNTRL	IEFCNTRL		
IEF7KAXX		IEFKA	IEFCNTRL	IEFCNTRL	IEFCNTRL	09	
IEF7KGXX		IEFKG	IEFINTFC	IEFCNTRL	IEFCNTRL	09	
IEF7KPXX		IEFKP	IEFCOMND	IEFCNTRL	IEFCNTRL	09	
IEF7K1XX		IEFK1	IEFINITL	IEFINITL	IEFINITL	08	09
			IEFK4				
IEF7K2XX		IEFK2	IEFINITL	IEFINITL	IEFINITL		
			IEFK4	IEFK4	IEFCNTRL		
			IEFEOF	IEFEOF			
IEF7K3XX		IEFK3	IEFEOF	IEFEOF	IEFCNTRL		
IEF7K4XX		IEFK4	IEFK4	IEFK4	IEFCNTRL		
			IEFEOF	IEFEOF			
IEF7MMCM		IEFWMSG	IEFCNTRL	IEFCNTRL	IEFCNTRL		
			IEF1STMT	IEF1STMT			
			IEFDD				
			IEFINTFC				
			IEFEXEC				
			IEFJOB				
			IEFCOMND				
IEF8LINK		IEFLINK	IEFEXEC	IEFCNTRL	IEFCNTRL		
IGC0103D	***	IGC0103D	IGC0103D	IGC0103D	IGC0103D	02,04	
IGC0203D	***	IGC0203D	IGC0203D	IGC0203D	IGC0203D		

Notes: *Assembly modules in SYS1.NUCLEUS data set.
**Modules are assembled during system generation.
***Assembly modules in SYS1.SVCLIB data set.
****IEFACTFK may replace both IEFACTLK and IEFACTRT during system generation.
x This assembly module must be first in any load module in which it is included.

CONTROL SECTIONS AND ASSEMBLY MODULES

The following list provides a cross-reference between job management control section (CSECT) names, which appear in alphameric order, and the corresponding assembly module names. Control section names are also listed in the preceding assembly module to load module cross reference table.

CSECT NAME	ASSEMBLY MODULE NAME
DEVMASKT	IEFWMSKA
DEVNAMET	IEFWMAS1
IEEBA1	IEECIR01
IEEBB1	IEEMCR01
IEEBC1PE	IEEBC1PE
IEEBH1	IEEBH1PE
IEEGESTO	IEEGES01
IEEGK1GM	IEEGK1GM
IEEICCAN	IEEILCDM
IEEICCAN	IEEILC01
IEEICRDR	IEEREADR
IEEICWTR	IEEWRITR
IEEIC1PE	IEESTART
IEEIC2NQ	IEEIC2NQ
IEEIC3JF	IEEIC3JF
IEEMSLT	IEERSC01
IEEMSLT	IEERSR01
IEEQOT00	IEEQOT00
IEFACTLK	IEFACTLK
IEFACTRT	IEFACTRT
IEFAM	IEF6NFCM
IEFBM	IEF6BOCM
IEFCOND	IEF6COND
IEFDPOST	IEFDPOST
IEFEM	IEFEMDUM
IEFEM	IEF6NJEX
IEFIDMPM	IEFIDMPM
IEFIDUMP	IEFIDFAK
IEFIDUMP	IEFIDUMP
IEFINTQS	IEFINTQA
IEFJM	IEFJMDUM
IEFJM	IEF6NCJB
IEFJOB	IEFKRESA
IEFKA	IEFKADUM
IEFKA	IEF7KAXX
IEFKG	IEFKGDUM
IEFKG	IEF7KGXX
IEFKL	IEFKLDUM
IEFKL	IEF6KLXX
IEFKP	IEFKPDUM
IEFKP	IEF7KPXX
IEFK1	IEF7K1XX
IEFKIMSG	IEFKIMSG
IEFK2	IEF7K2XX
IEFK3	IEFK3DUM
IEFK3	IEF7K3XX
IEFK4	IEFK4DUM
IEFK4	IEF7K4XX
IEFK4DUM	IEFK4ENT
IEFLINK	IEF8LINK
IEFMC	IEFMCDUM
IEFMC	IEF6MCXX
IEFMF	IEFMFDUM
IEFMF	IEF6MFXX
IEFMI	IEF6MIXX
IEFMK	IEF6MKXX
IEFMO2	IEF6CLNP
IEFMSG1	IEF3MSG1
IEFMSG2	IEF3MSG2
IEFMSG3	IEF3MSG3
IEFMSG4	IEF3MSG4
IEFMSG5	IEF3MSG5
IEFMSG6	IEF3MSG6
IEFMSG7	IEF3MSG7
IEFNI	IEF6NIJB
IEFNX	IEF6NXJB
IEFNY	IEF6NYJB
IEFNZ	IEF6NZJB
IEFN1	IEF6N1JB
IEFPARM	IEF6PARM
IEFPRES	IEFPRES
IEFPRES	IEFPRFAK
IEFPROC	IEF6PROC
IEFQMSSS	IEFQMSSS
IEFRFBK	IEF6RFBK
IEFSD001	IEFSD001
IEFSD002	IEFSD002
IEFSD003	IEFSD003
IEFSD004	IEFSD004
IEFSD004	IEF04FAK
IEFSD006	IEFSD006
IEFSD007	IEFSD007
IEFSD008	IEFSD008
IEFSD008	IEF08FAK
IEFSD009	IEFSD009
IEFSD009	IEF09FAK
IEFSD010	IEFSD010
IEFSD011	IEFSD011
IEFSD012	IEFSD012
IEFSD013	IEFSD013
IEFSD017	IEFSD017
IEFSPIE	IEFSPIE
IEFSTNM	IEF6STNM
IEFTIME	IEF6TIME
IEFVJMSG	IEFVJMSG
IEFVJ	IEFVJIMP
IEFVKMSG	IEFVKMSG
IEFVK	IEFVKIMP
IEFVM1	IEFVMLS1
IEFVM2	IEFVM2LS
IEFVM3	IEFVM3LS
IEFVM4	IEFVM4LS
IEFVM5	IEFVM5LS
IEFVM6	IEFVMLK5
IEFVM6	IEFVMLS6
IEFVM76	IEFVM76
IEFVM7	IEFVMLS7
IEFWA000	IEFWAFAK
IEFWA7	IEFWA000
IEFWC000	IEFWCFAK
IEFWC000	IEFWCIMP
IEFWC002	IEFWCIMP
IEFWD000	IEFWDFAK
IEFWD000	IEFWD000

CSECT NAME	ASSEMBLY MODULE NAME
IEFWD001	IEFWD001
IEFWMSG	IEF7MMCM
IEFWSDIP	IEFWSDIP
IEFWSTRT	IEFWSTRT
IEFWSWIT	IEFWSWIN
IEFWTERM	IEFWTERM
IEFW21SD	IEFW21SD
IEFW22SD	IEFW22SD
IEFW23SD	IEFW23SD
IEFW23SD	IEF23FAK
IEFW31SD	IEFW31SD
IEFW41SD	IEFW41SD
IEFW42SD	IEFW42SD
IEFXAMSG	IEFXAMSG
IEFXA	IEFXAFAK
IEFXA	IEFXCSSS
IEFXH000	IEFXH000
IEFXJMSG	IEFXJMSG
IEFXJ000	IEFXJFAK
IEFXJ000	IEFXJIMP
IEFXKMSG	IEFXKMSG
IEFXK000	IEFXKIMP
IEFXTDMY	IEFXTDMY
IEFXTMSG	IEFXTMSG
IEFXT000	IEFXT00D
IEFXT000	IEFXTAK
IEFXVMSG	IEFXVMSG
IEFXVNSL	IEFXVNSL
IEFXV001	IEFXV001
IEFXV002	IEFXV002
IEFXV003	IEFXV003
IEFXV004	IEFXV004
IEFX3000	IEFX300A
IEFX5000	IEFX5000
IEFYNIMP	IEFYNIMP
IEFYNMSG	IEFYNMSG
IEFYN	IEFYNFAK

CSECT NAME	ASSEMBLY MODULE NAME
IEFYPMSG	IEFYPMSG
IEFYP	IEFYPJB3
IEFYS	IEFYSSMB
IEFZA	IEFZAFAK
IEFZA	IEFZAJB3
IEFZGMSG	IEFZGMSG
IEFZG	IEFZGJB1
IEFZG	IEFZGST1
IEFZH	IEFZHMSG
IEFZPOQM	IEFZHFAK
IEF6DCB0	IEF6DCB0
IEF6ERR1	IEF6ERR1
IEF6FRRS	IEF6FRRS
IEF6RFWD	IEF6RFWD
IGC0103D	IGC0103D
IGC0203D	IGC0203D
IGC03D	IEEMXC01
IGC03E	IEEWTC01
INDMA	IEF6NDDP
INDMB	IEF6DCDP
INDMOH	IEF6OUT2
INDMON	IEF6DHX1
INDMON	IEF6SCAN
INDMOP	IEF6EQUL
INDMOR	IEF6RTPR
INDMOS1	IEF6LFPR
INDMOT	IEF6VALU
INDMOV	IEF6ORDR
INDMOX	IEF6NLST
INDMOY	IEF6LIST
INDMOZ	IEF6INST
INDMO1	IEF6DDNM
INDMO3	IEF6DSNM
INDMRTN	IEF6CN17
INNAME	IEF6NAME
SPRINTER	IEFPRTXX

Chart 01. Job Management

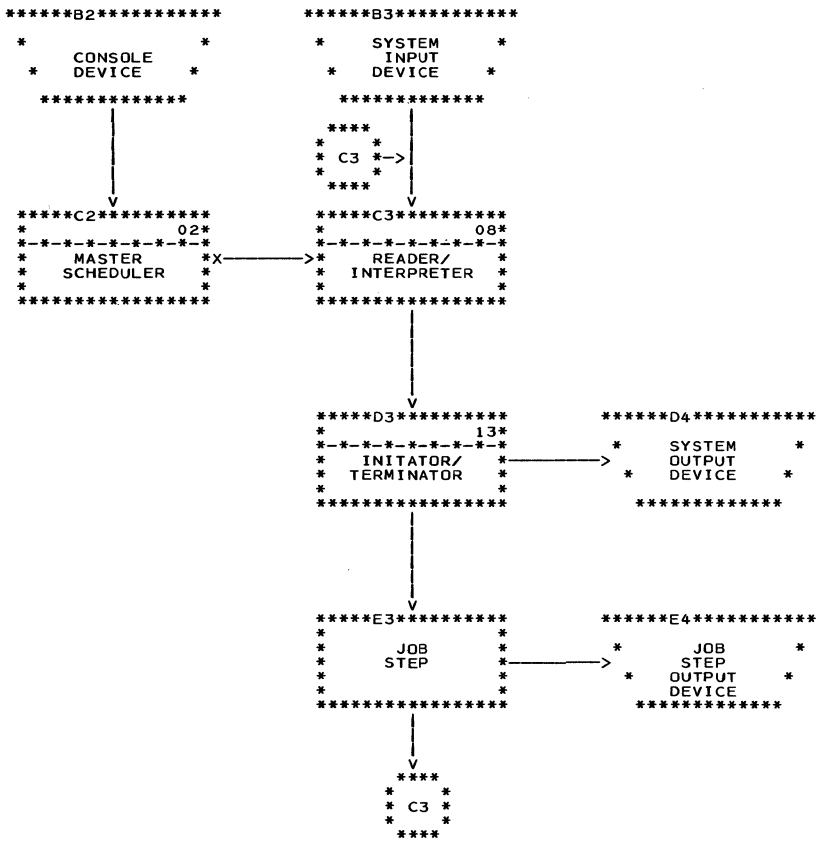
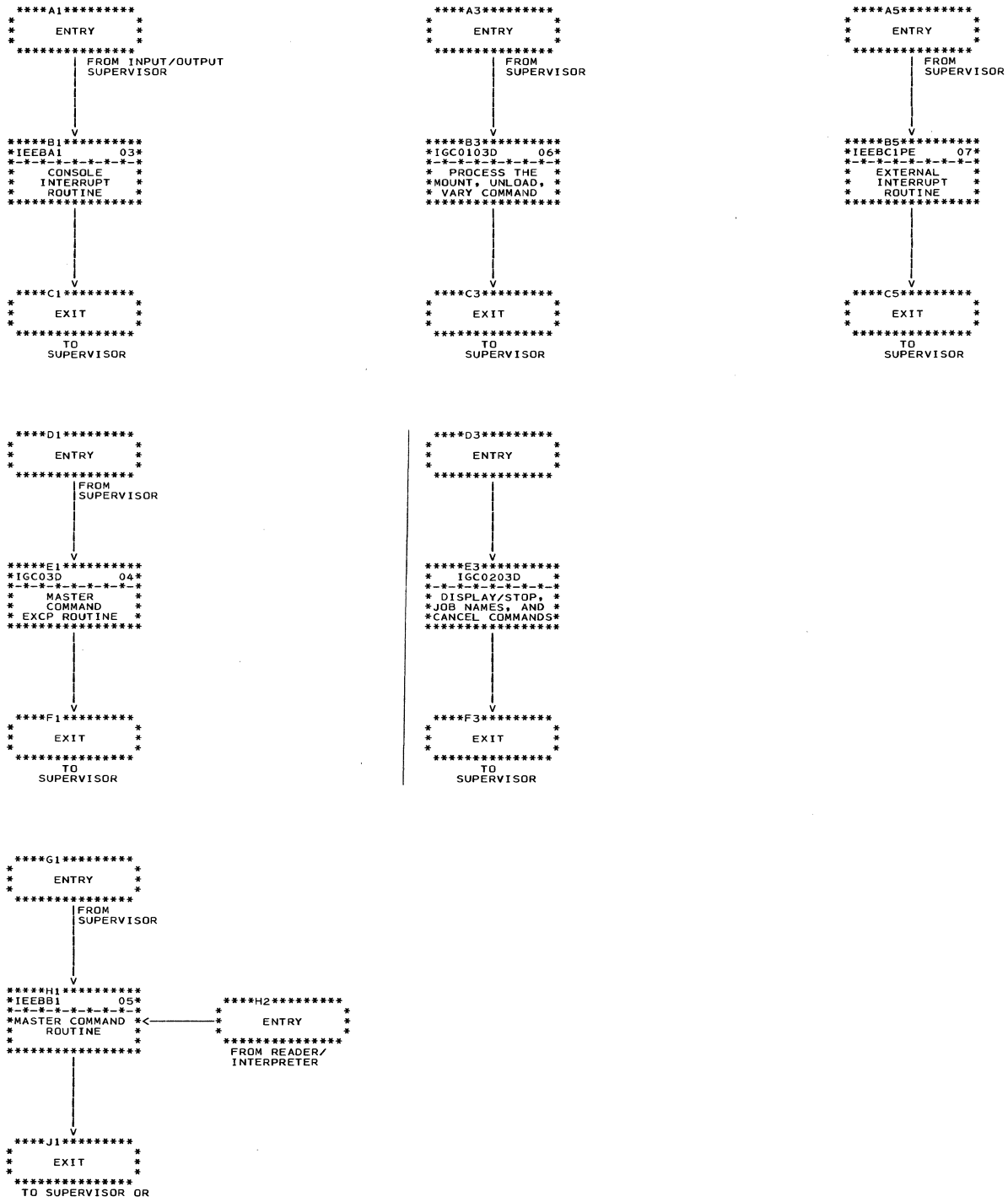
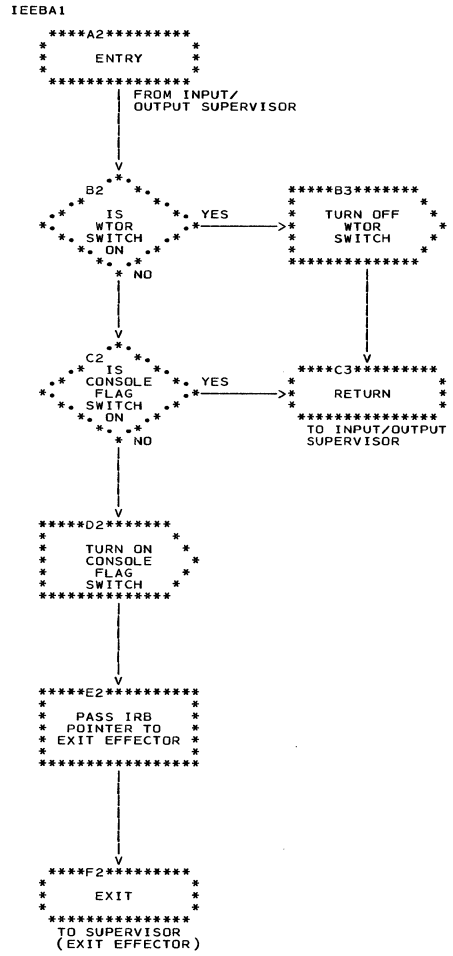


Chart 02. Master Scheduler



• Chart 03. Console Interrupt Routine



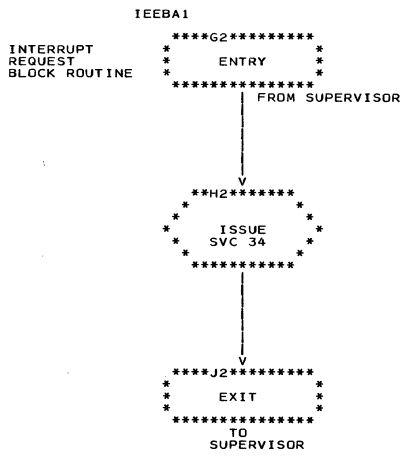
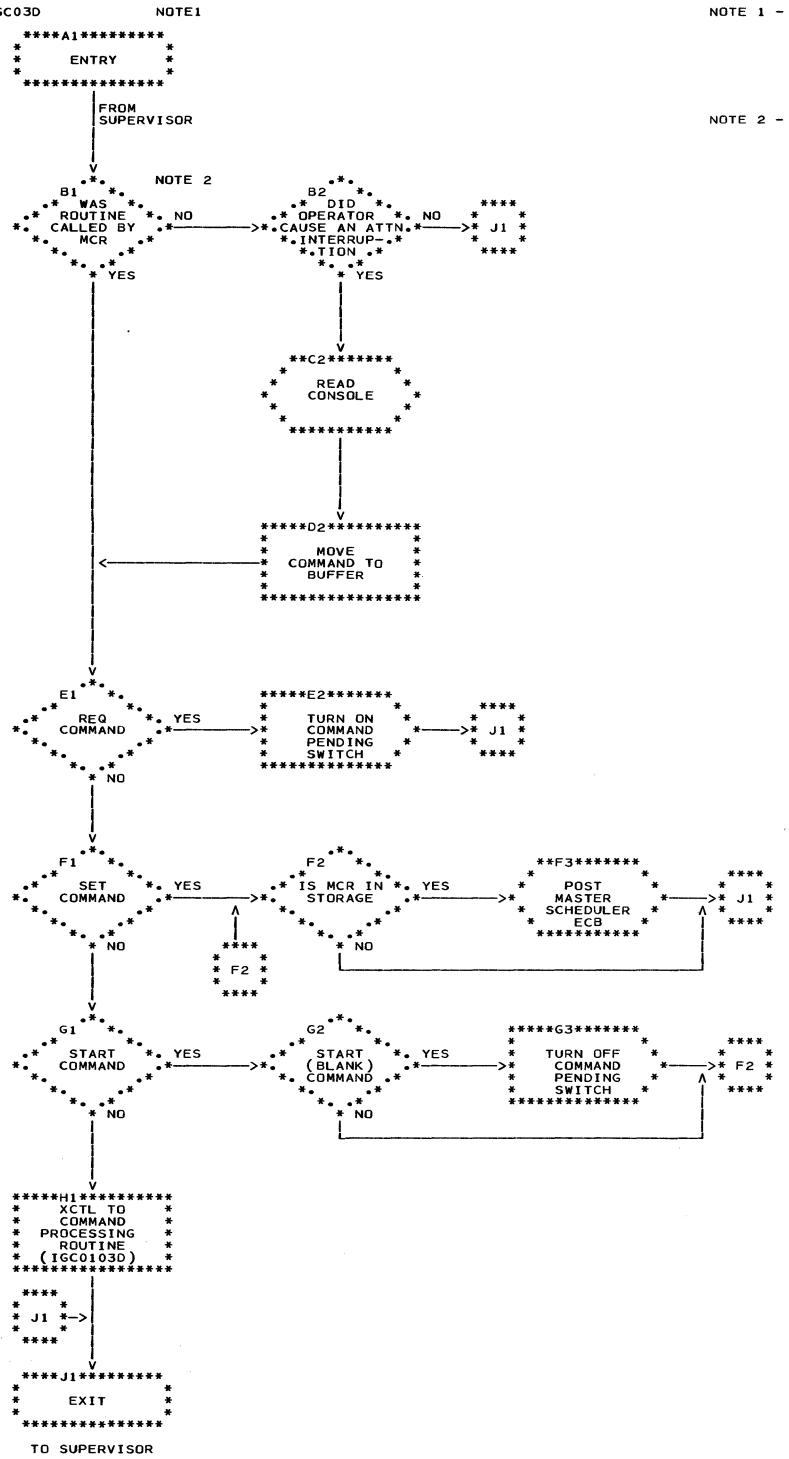


Chart 04. Master Command EXCP Routine



NOTE 1 - IF PRIMARY OR ALTERNATE CONSOLE IS IN USE, ENTRY IS TO IEEMXC01.

IF COMPOSITE CONSOLE IS IN USE, ENTRY IS TO IEEMXR01.

NOTE 2 - MCR = MASTER COMMAND ROUTINE.

Chart 05. Master Command Routine

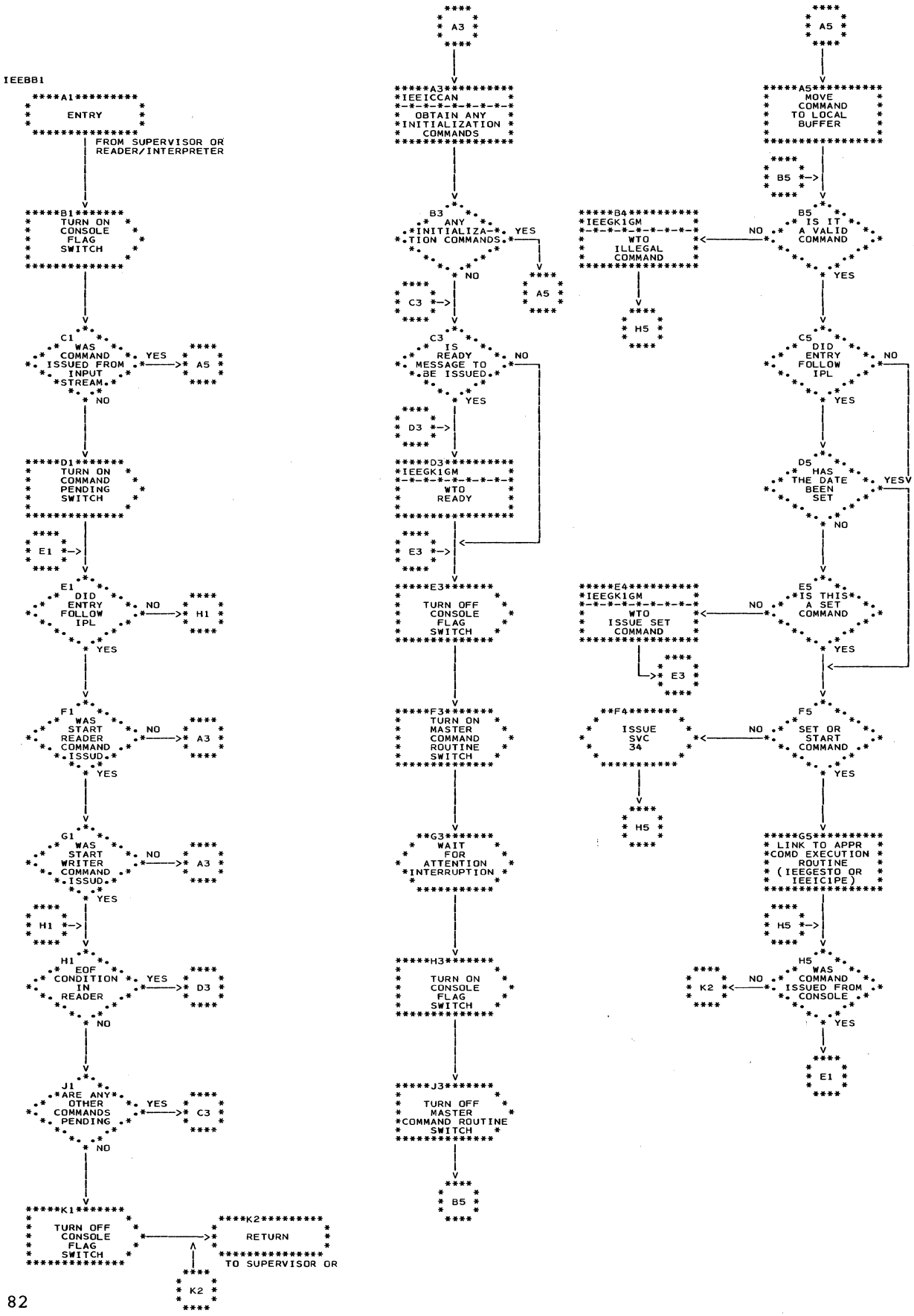
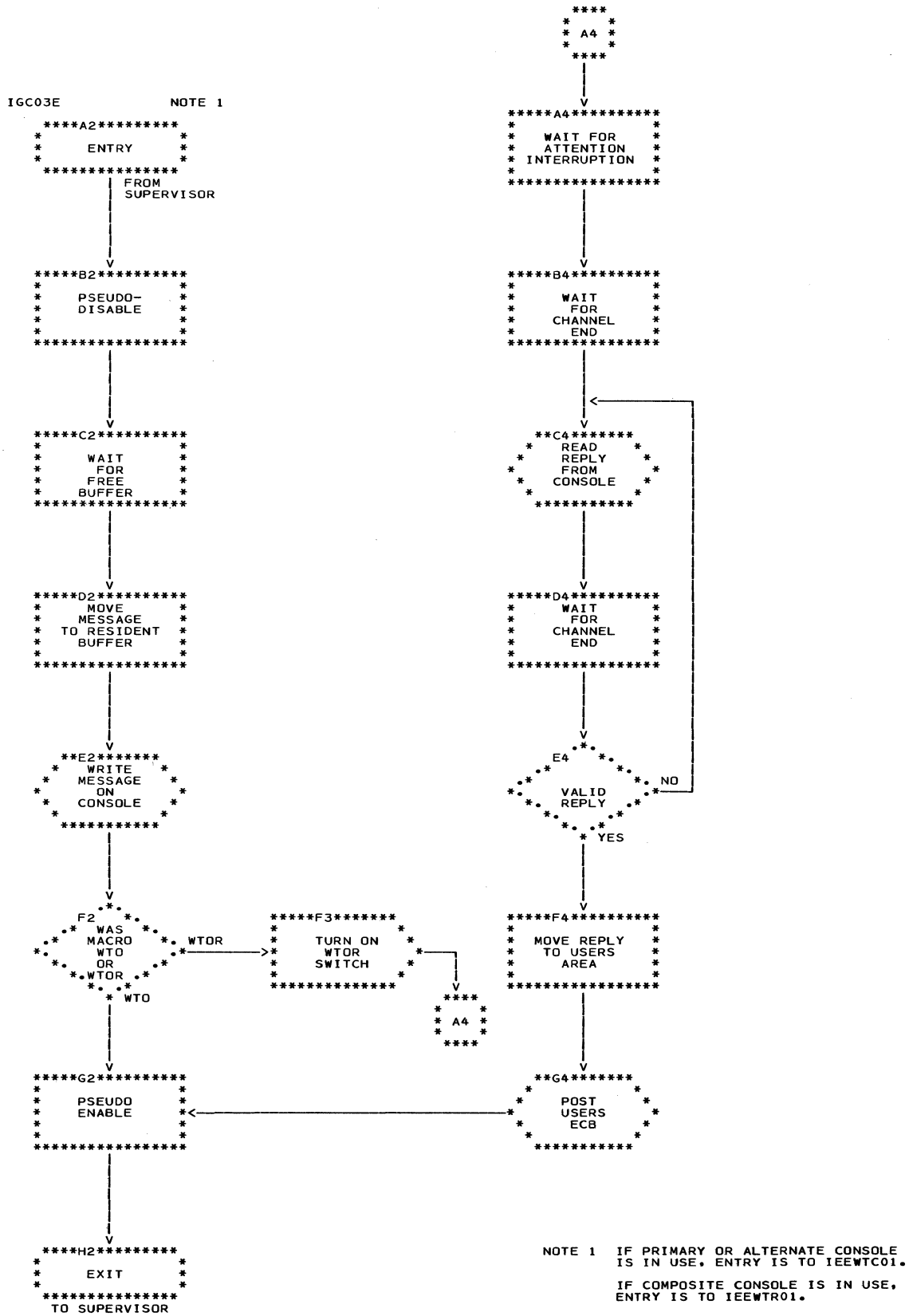


Chart 06. Write-to-Operator Routine



NOTE 1 IF PRIMARY OR ALTERNATE CONSOLE IS IN USE, ENTRY IS TO IEEWTC01.

IF COMPOSITE CONSOLE IS IN USE, ENTRY IS TO IEEWTR01.

Chart 07. External Interrupt Routine

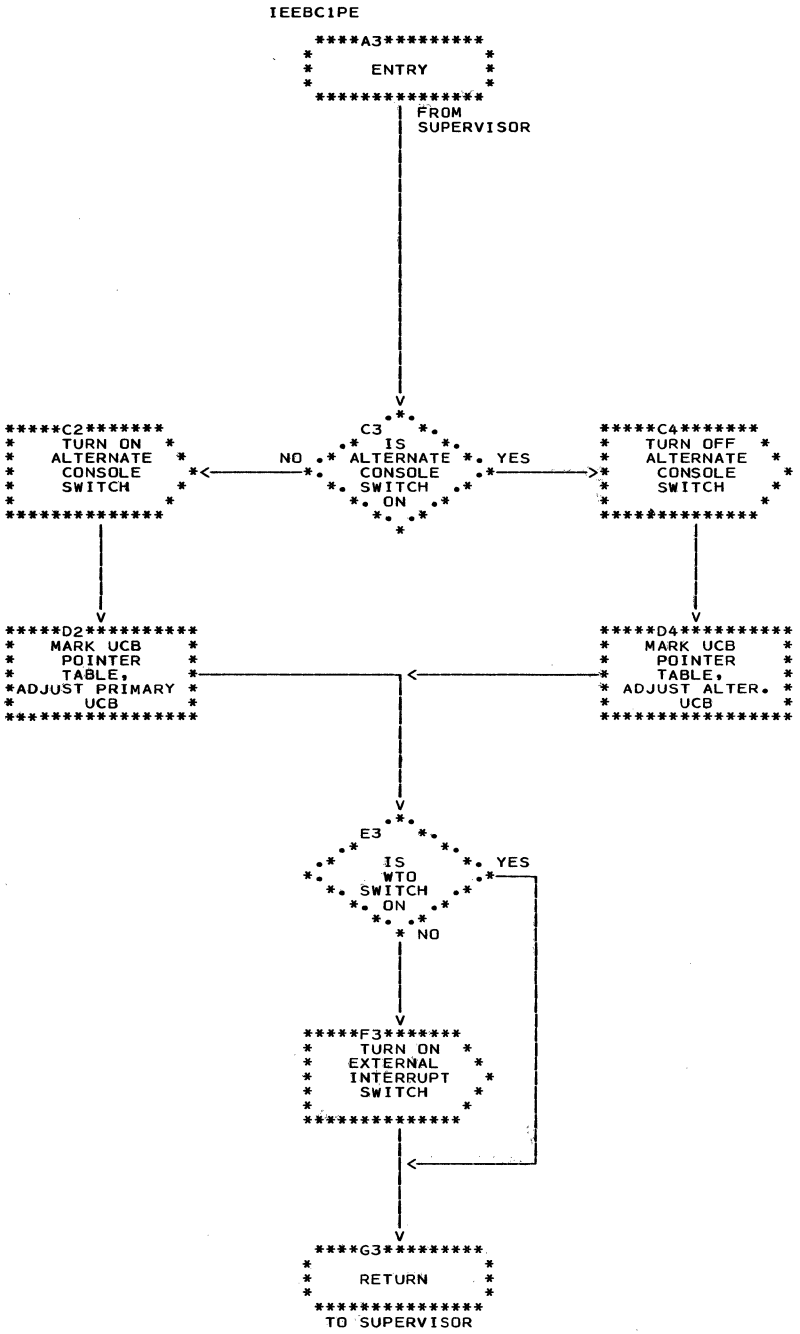


Chart 08. Reader/Interpreter

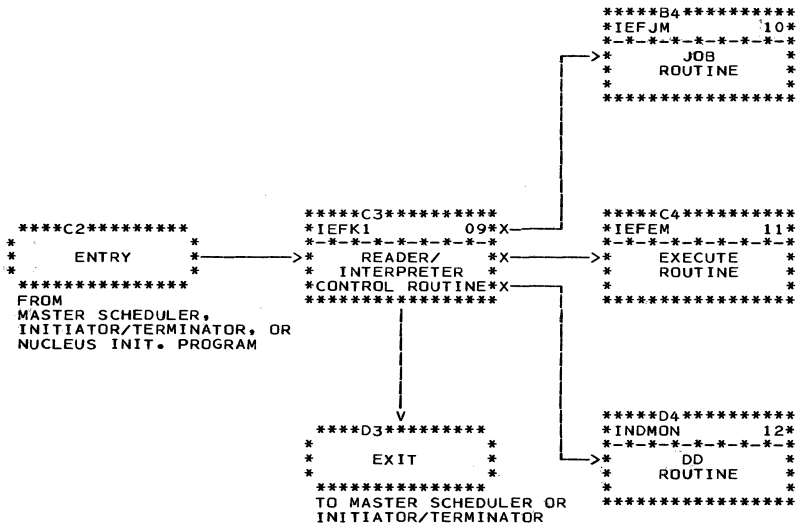


Chart 09. Reader/Interpreter Control Routine

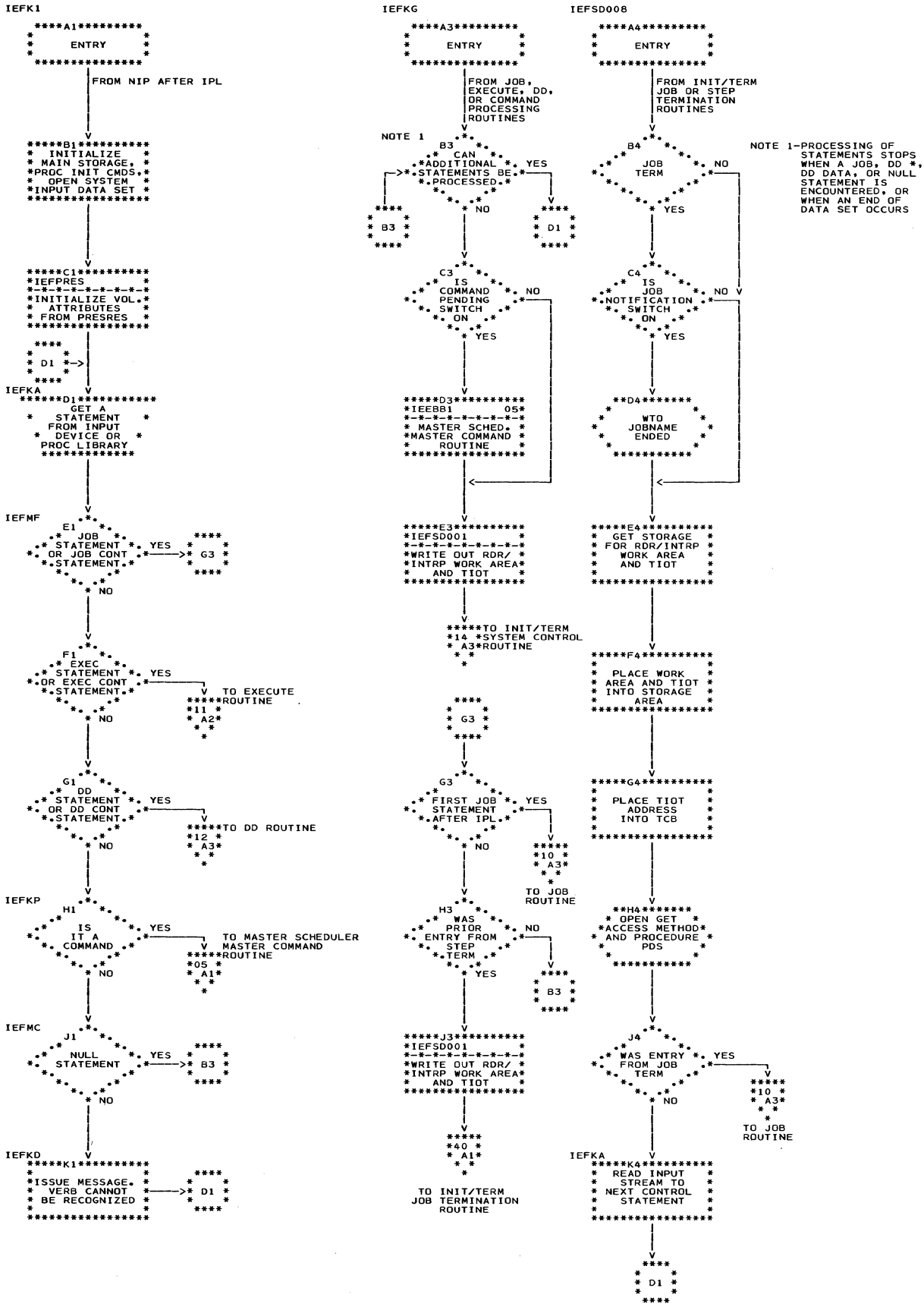


Chart 10. Job Routine

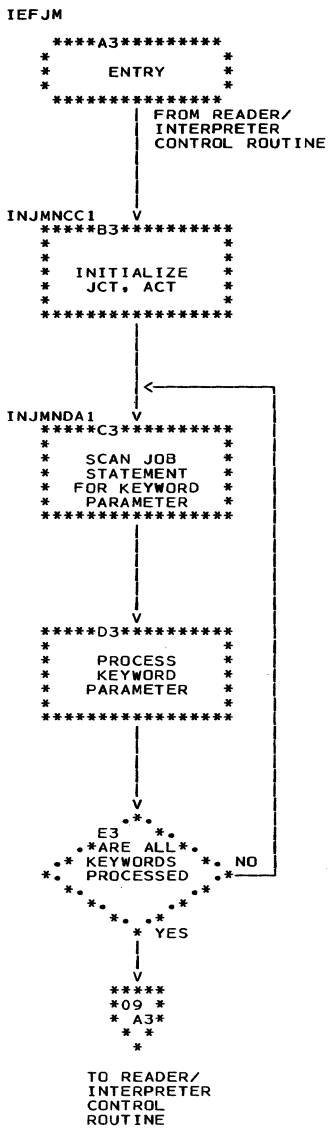


Chart 11. Execute Routine

IEFEM

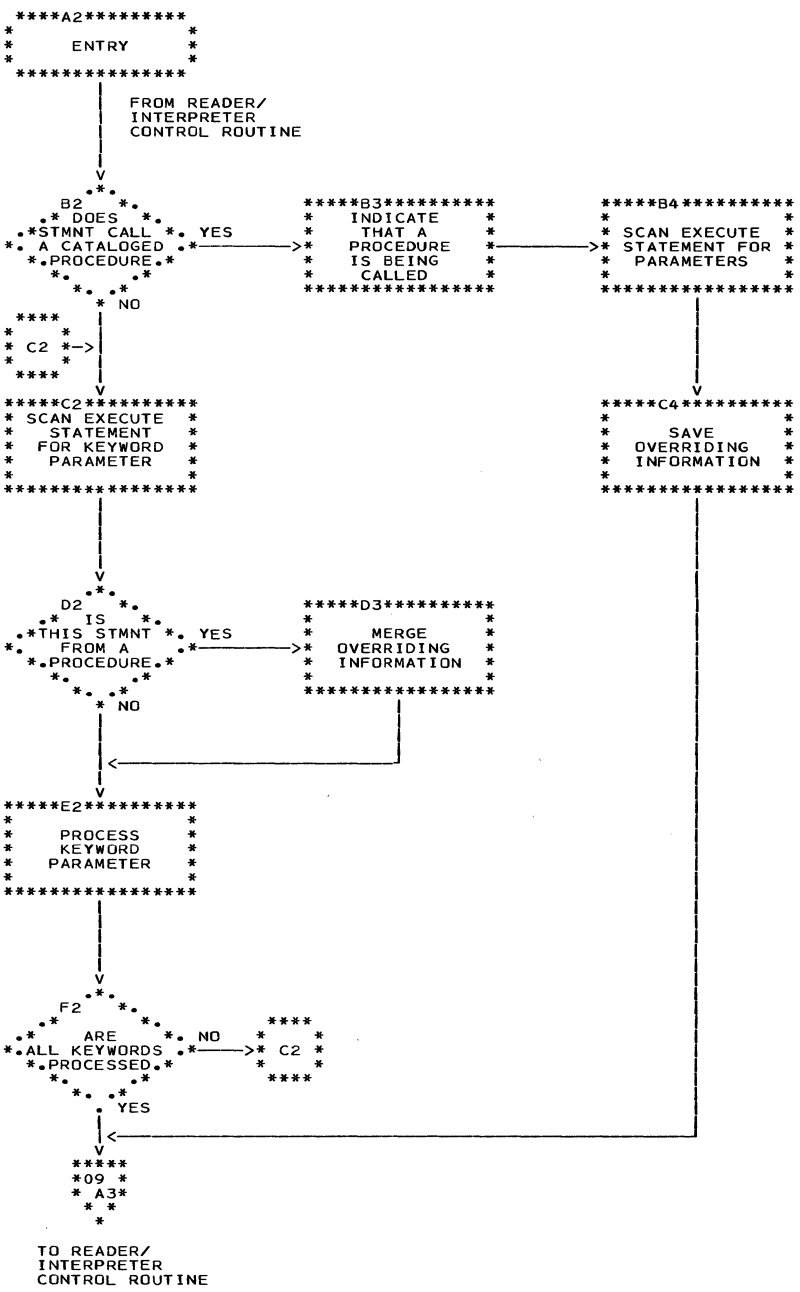


Chart 12. DD Routine

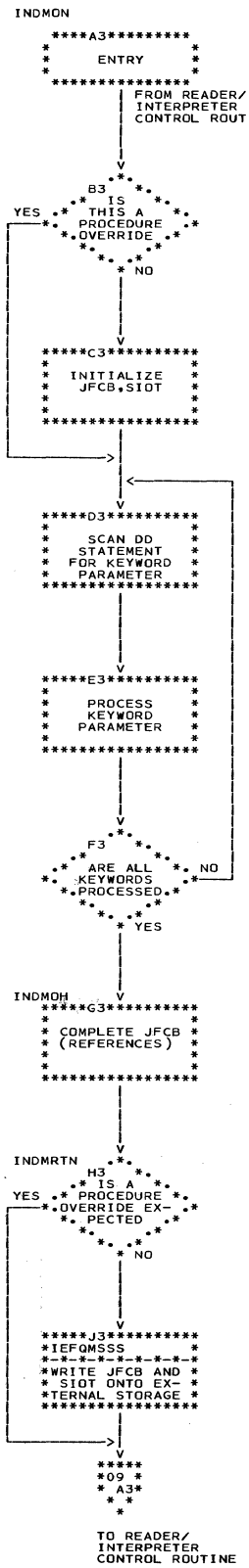
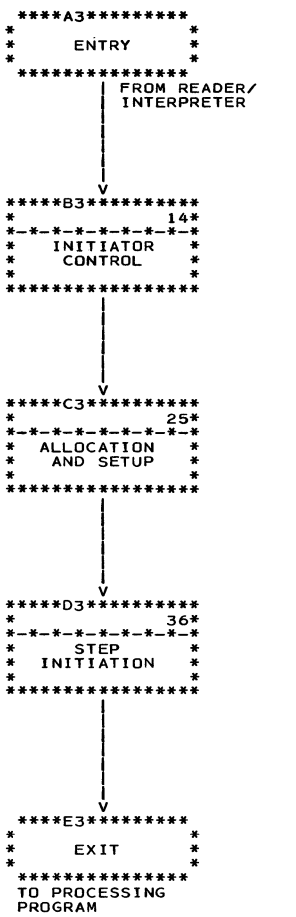


Chart 13. Initiator/Terminator



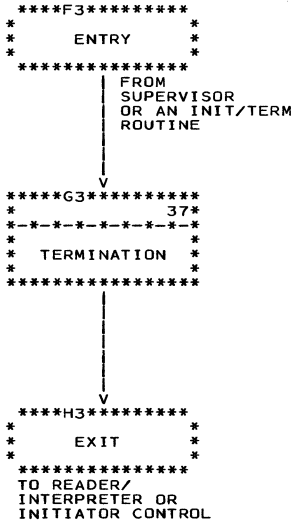


Chart 14. Initiator Control

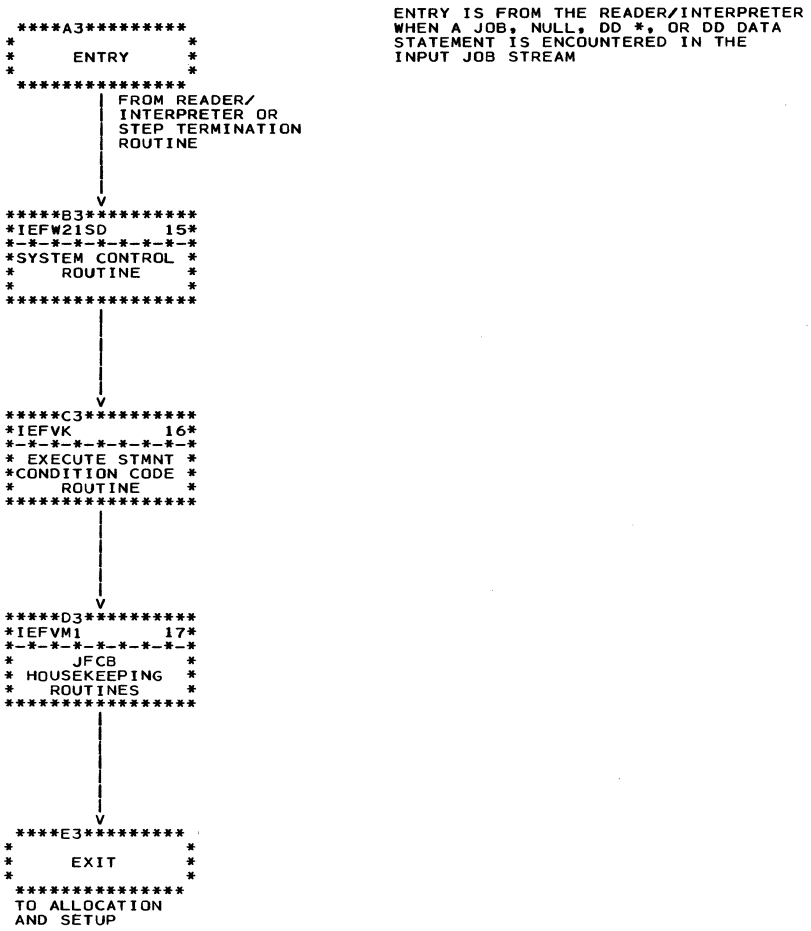


Chart 15. System Control Routine

IEFW21SD

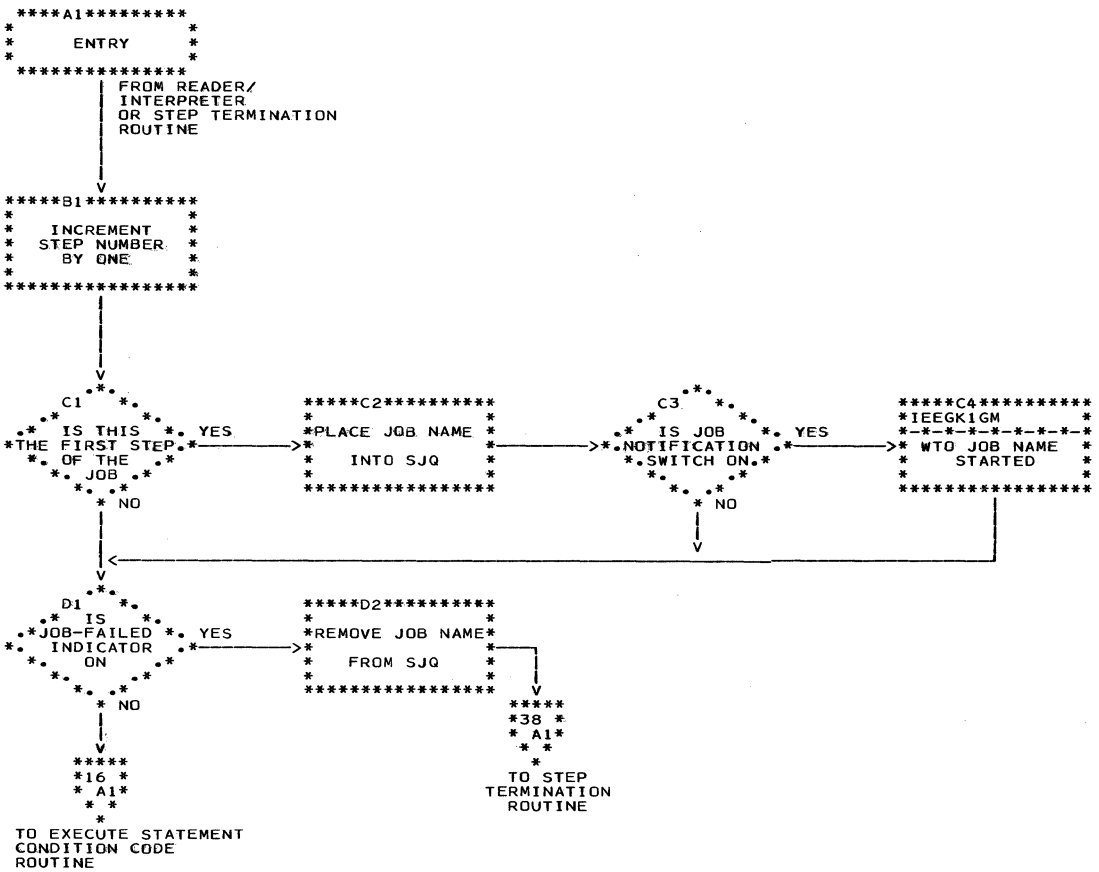


Chart 16. Execute Statement Condition Code Routine

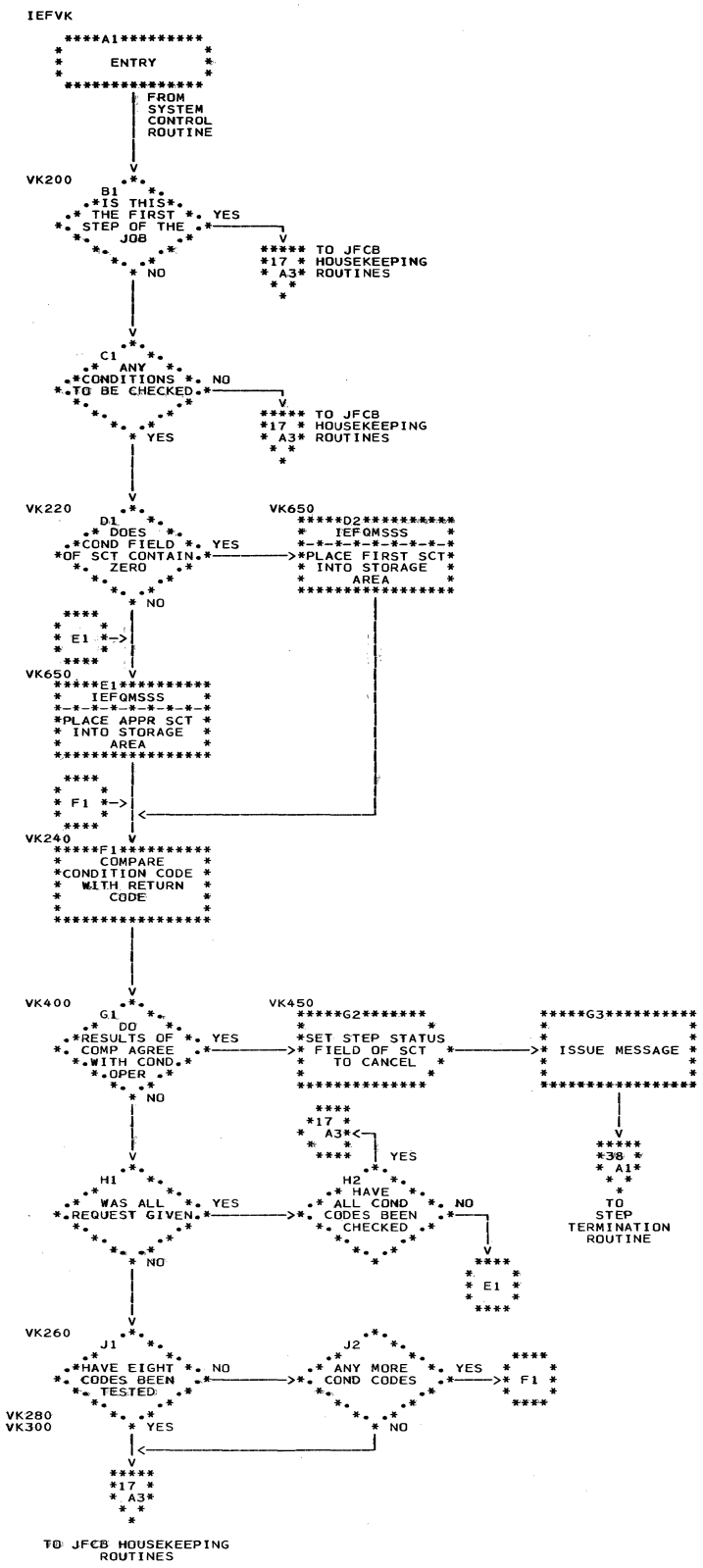


Chart 17. JFCB Housekeeping Routines

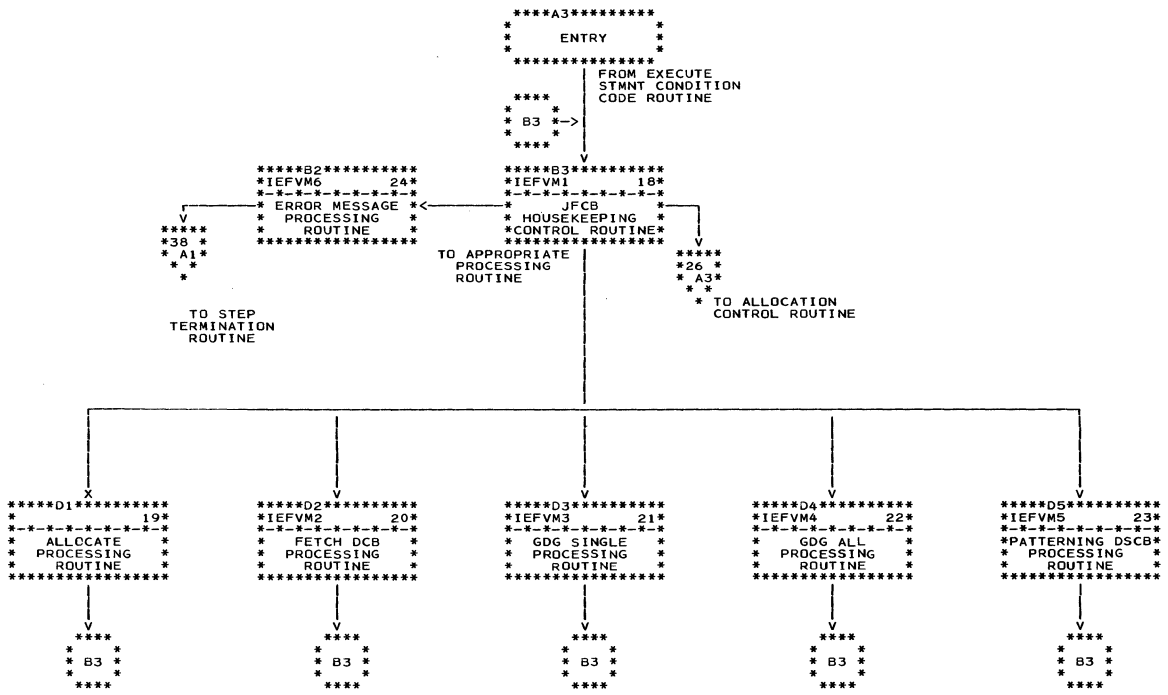


Chart 18. JFCB Housekeeping Control Routine

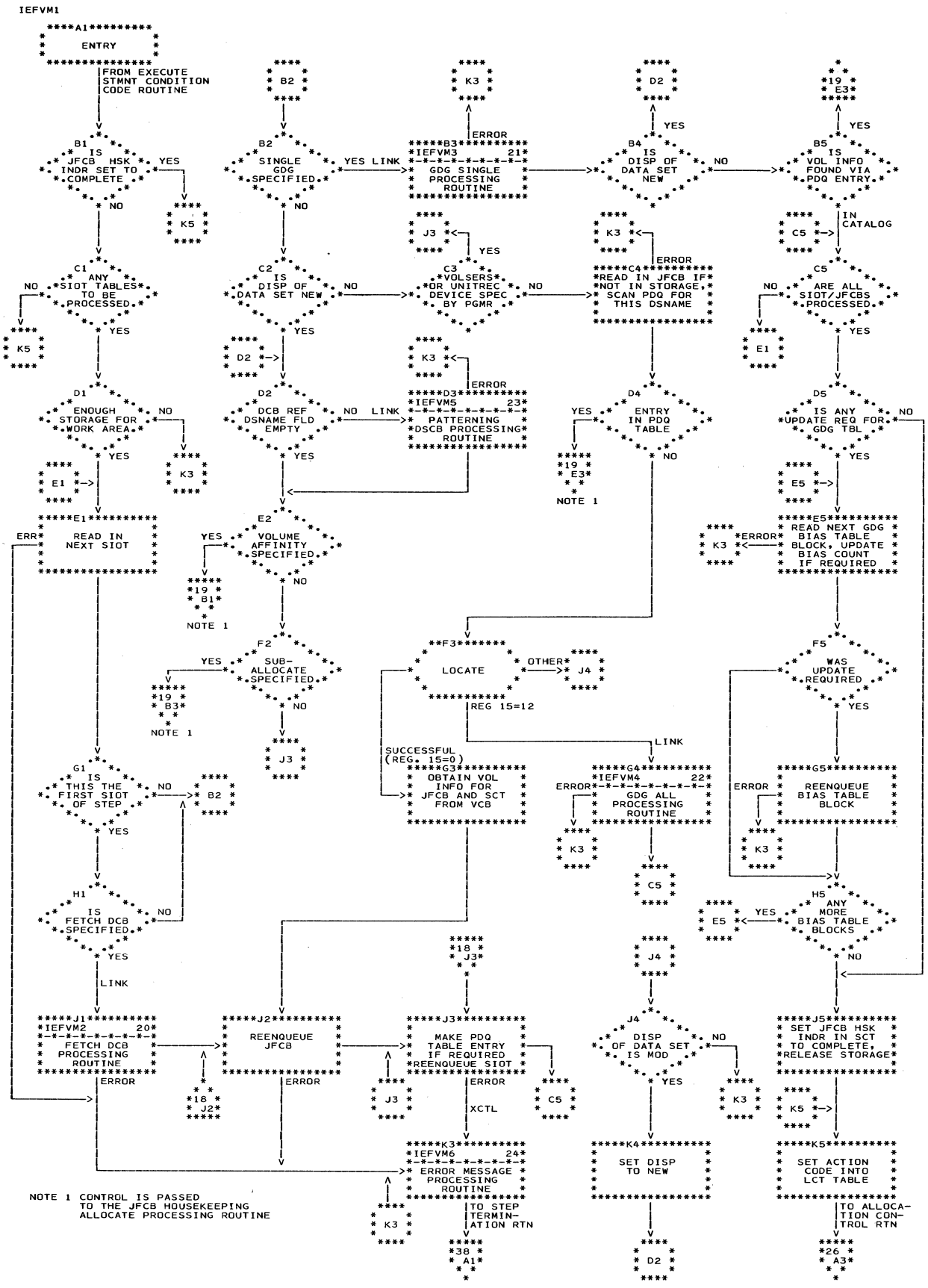
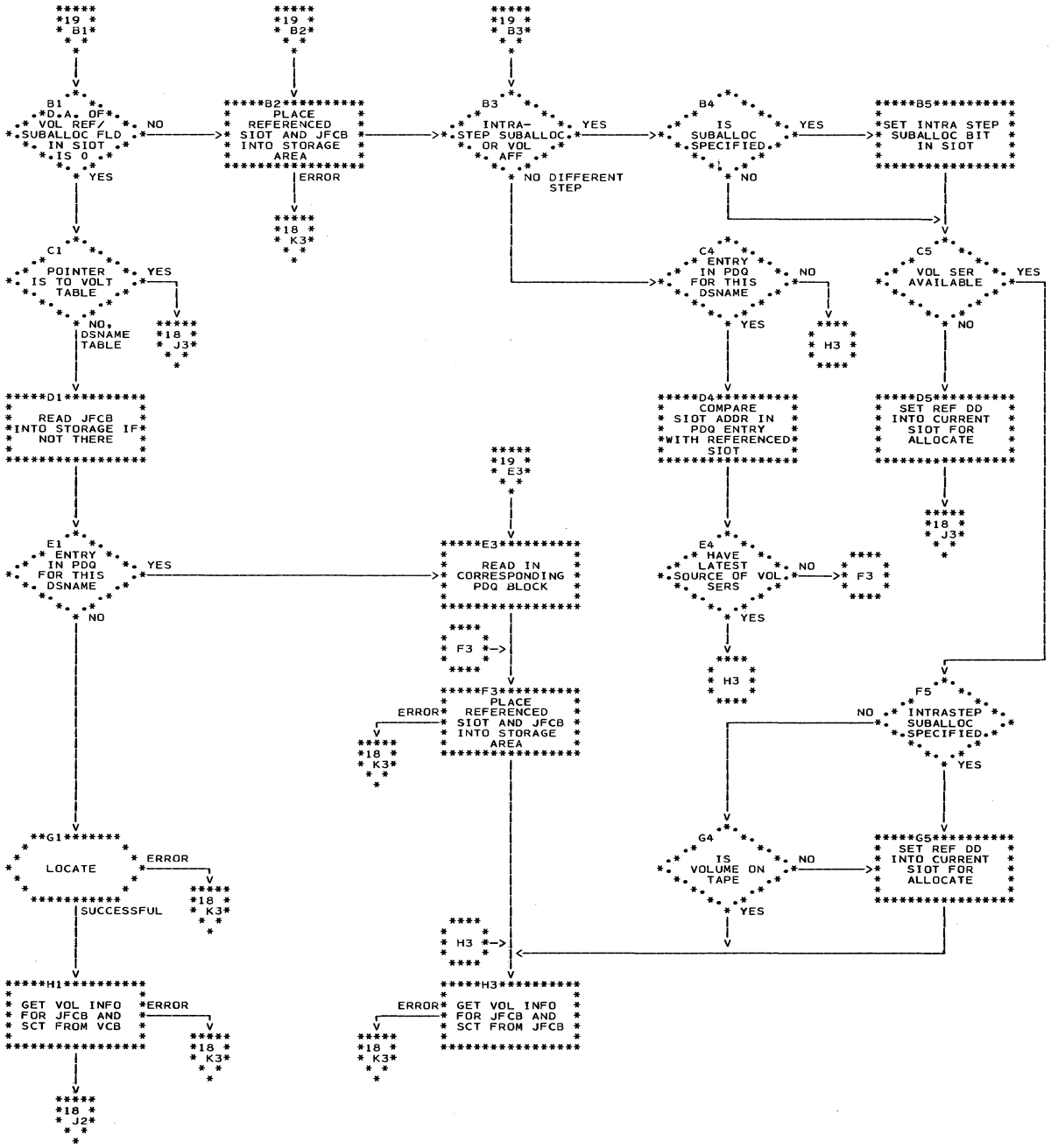


Chart 19. Allocate Processing Routine



ALL ENTRIES/EXITS ARE FROM/TO THE JFCB HOUSEKEEPING CONTROL ROUTINE

Chart 20. Fetch DCB Processing Routine

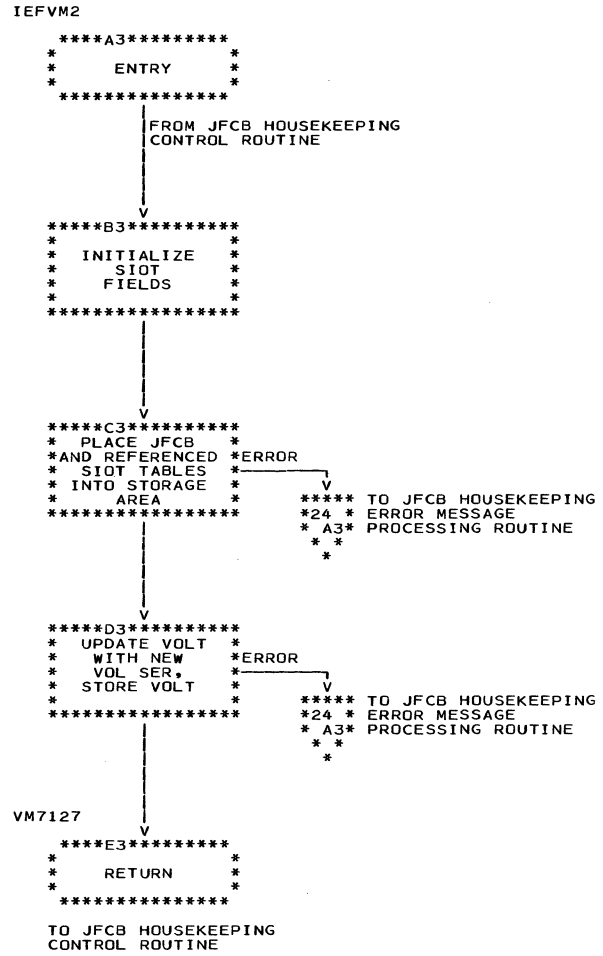


Chart 21. GDG Single Processing Routine

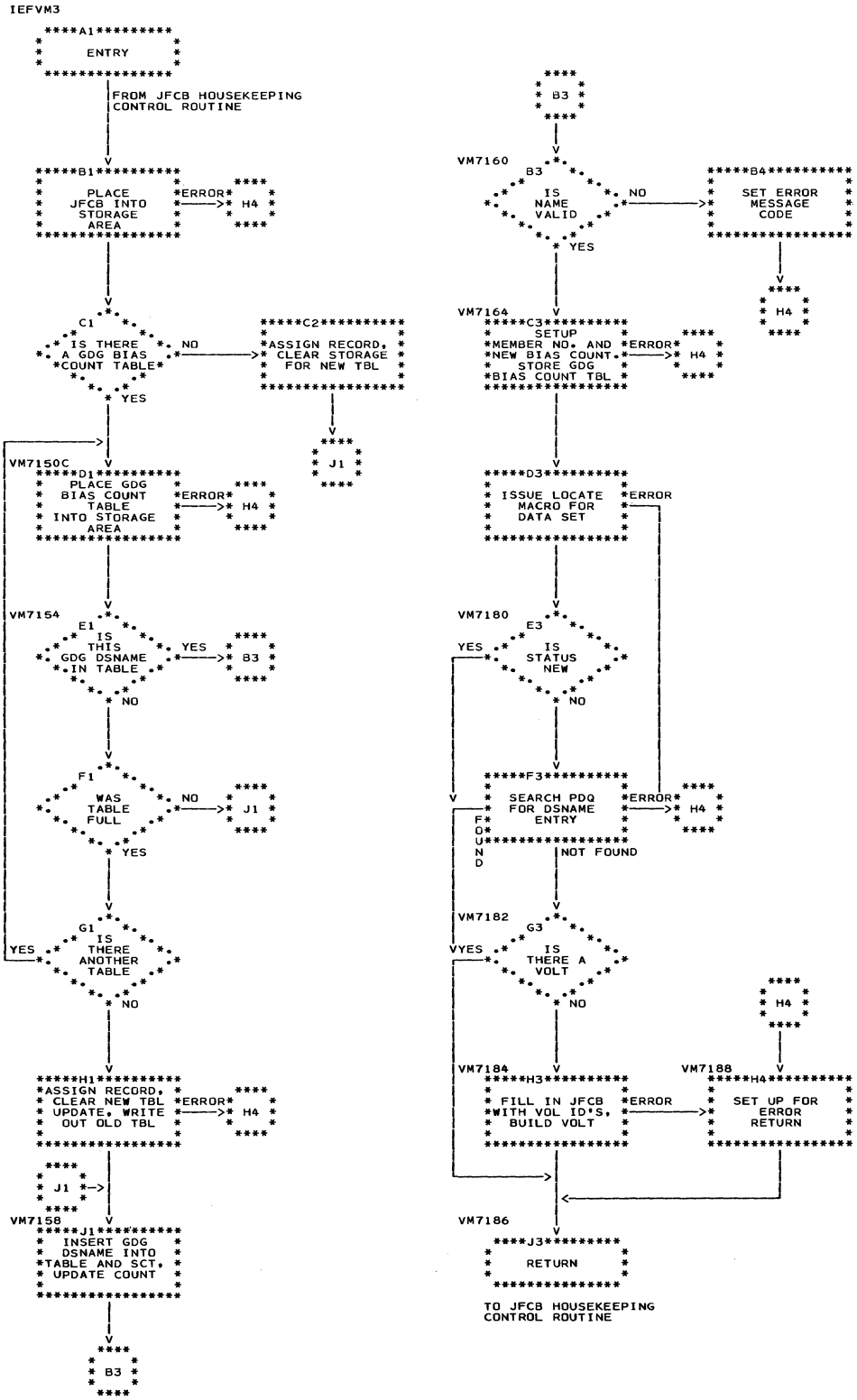


Chart 22. GDG All Processing Routine

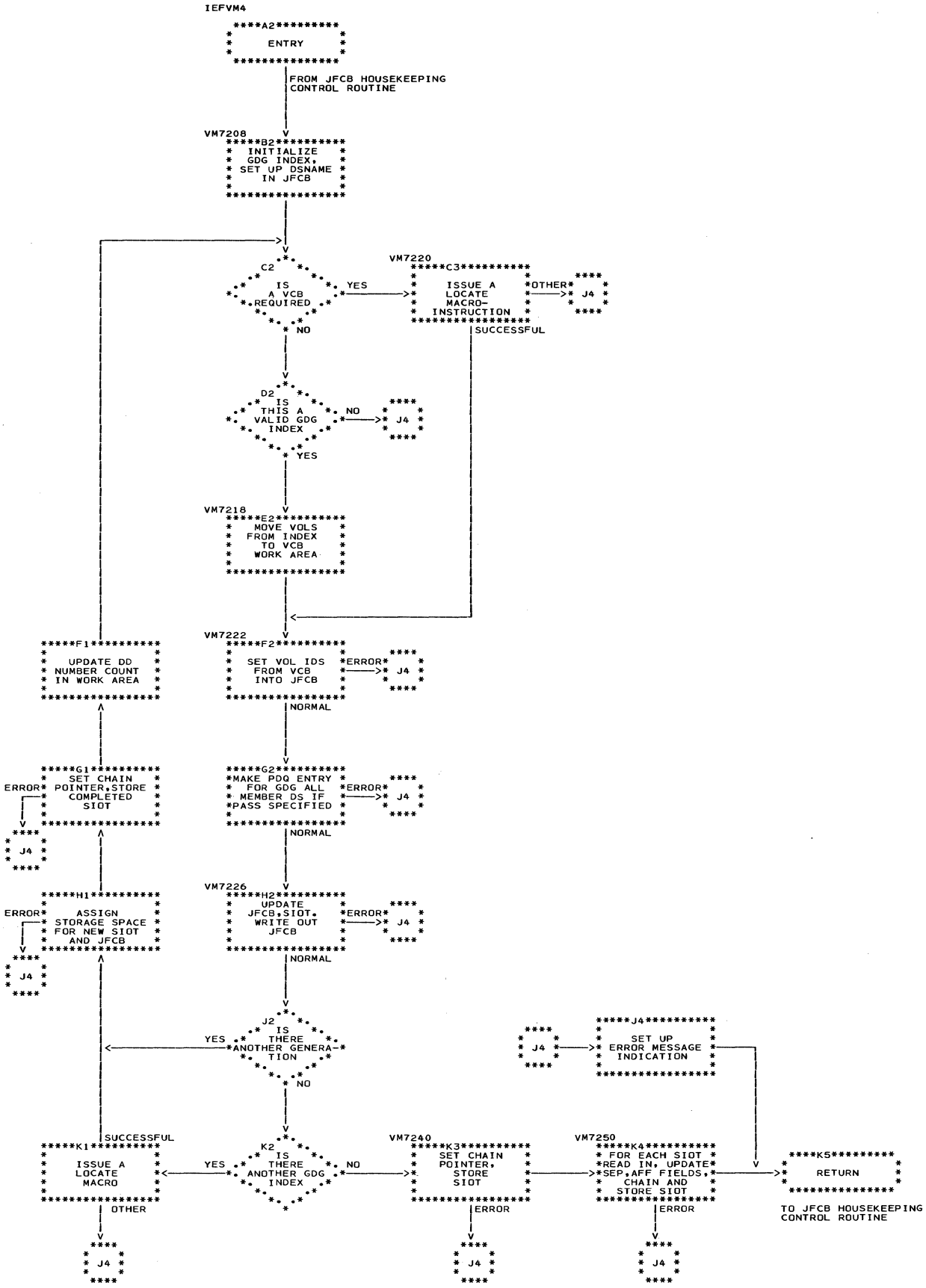


Chart 23. Patterning DSCB Processing Routine

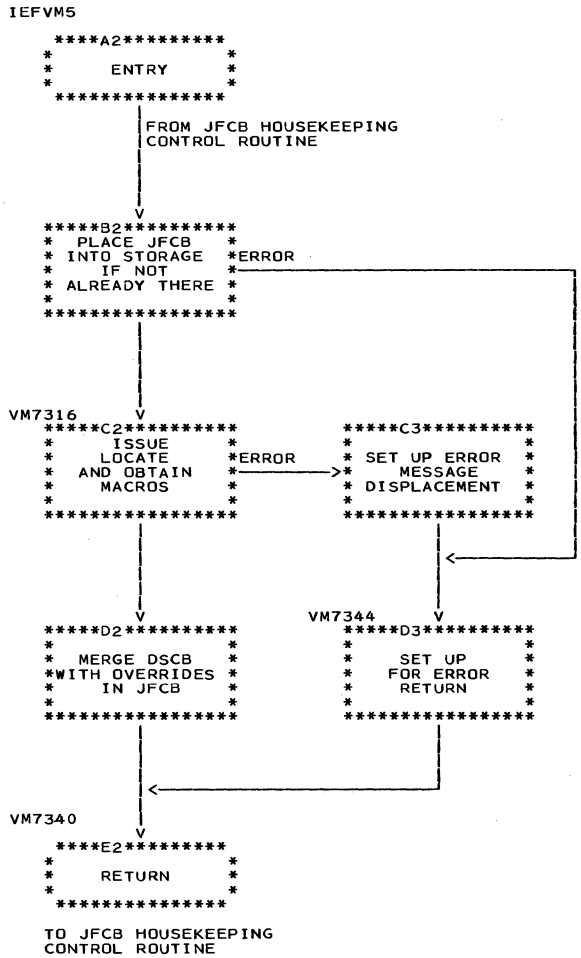


Chart 24. Error Message Processing Routine

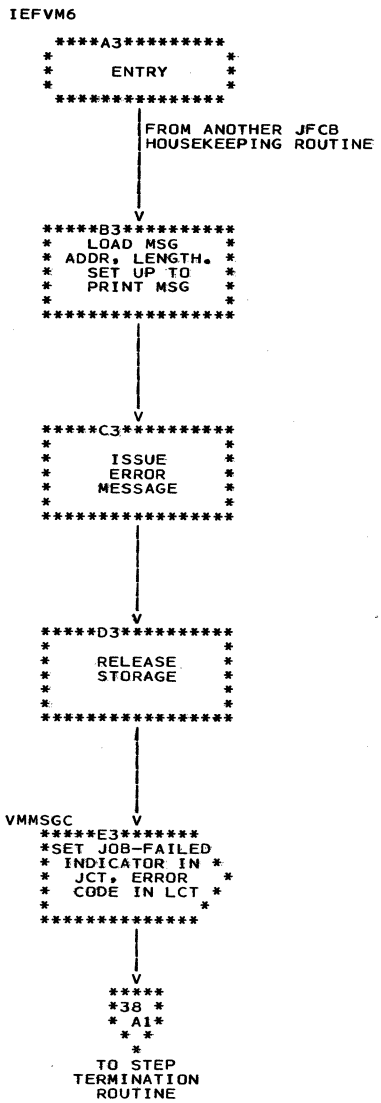


Chart 25. Allocation and Setup

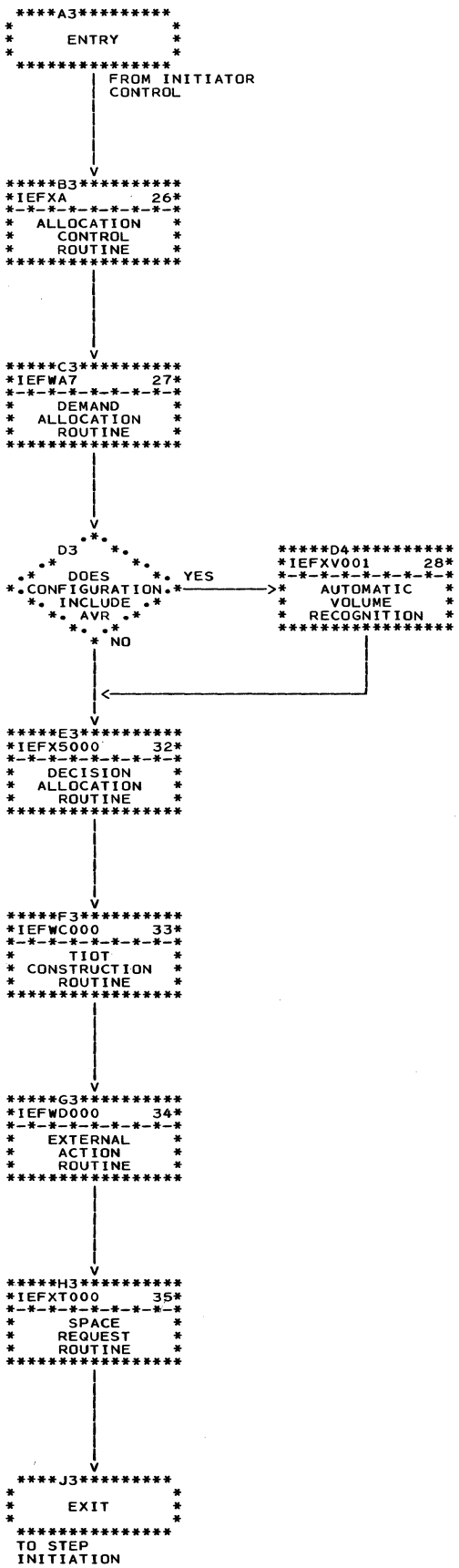


Chart 26. Allocation Control Routine

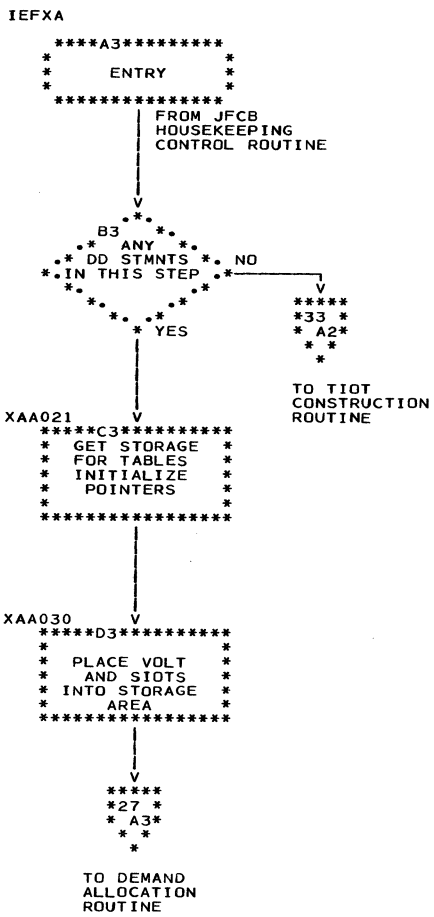
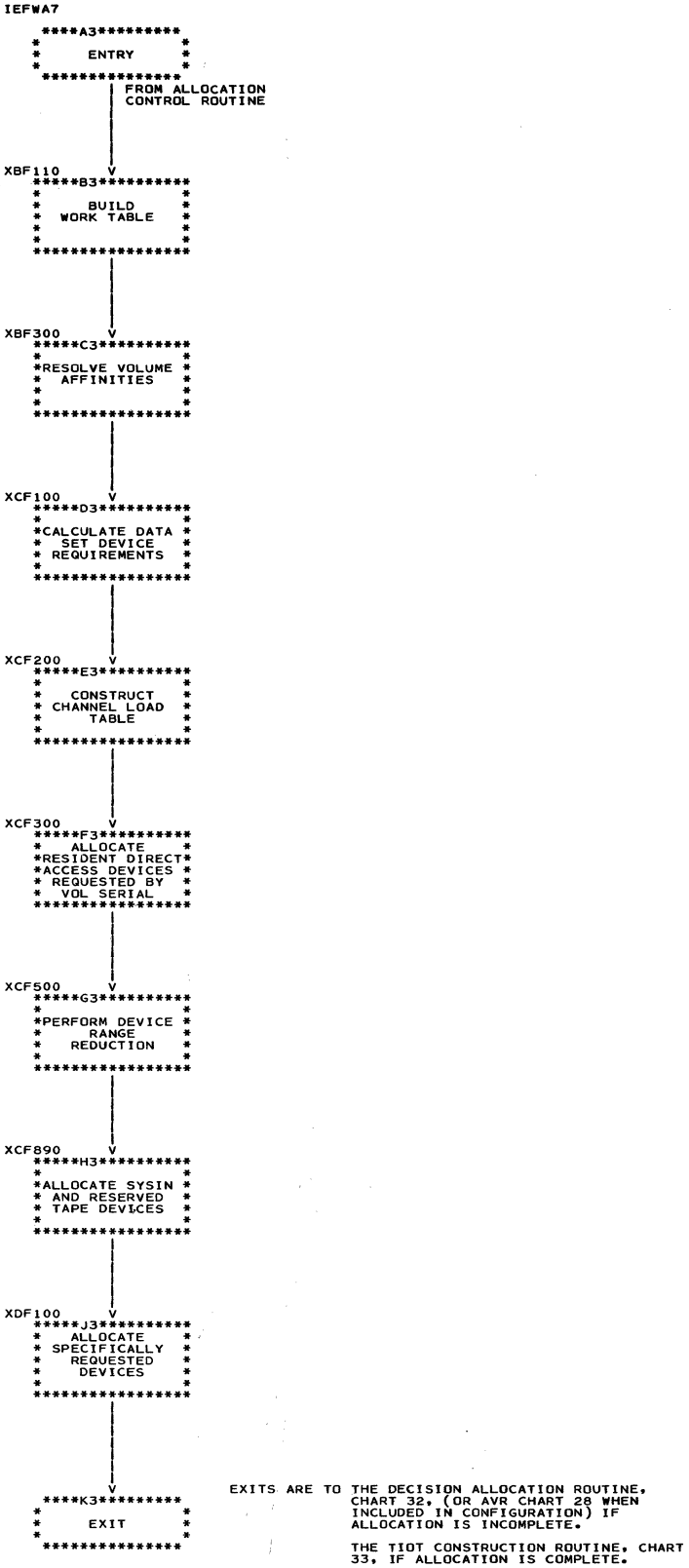


Chart 27. Demand Allocation Routine



• Chart 28. Automatic Volume Recognition

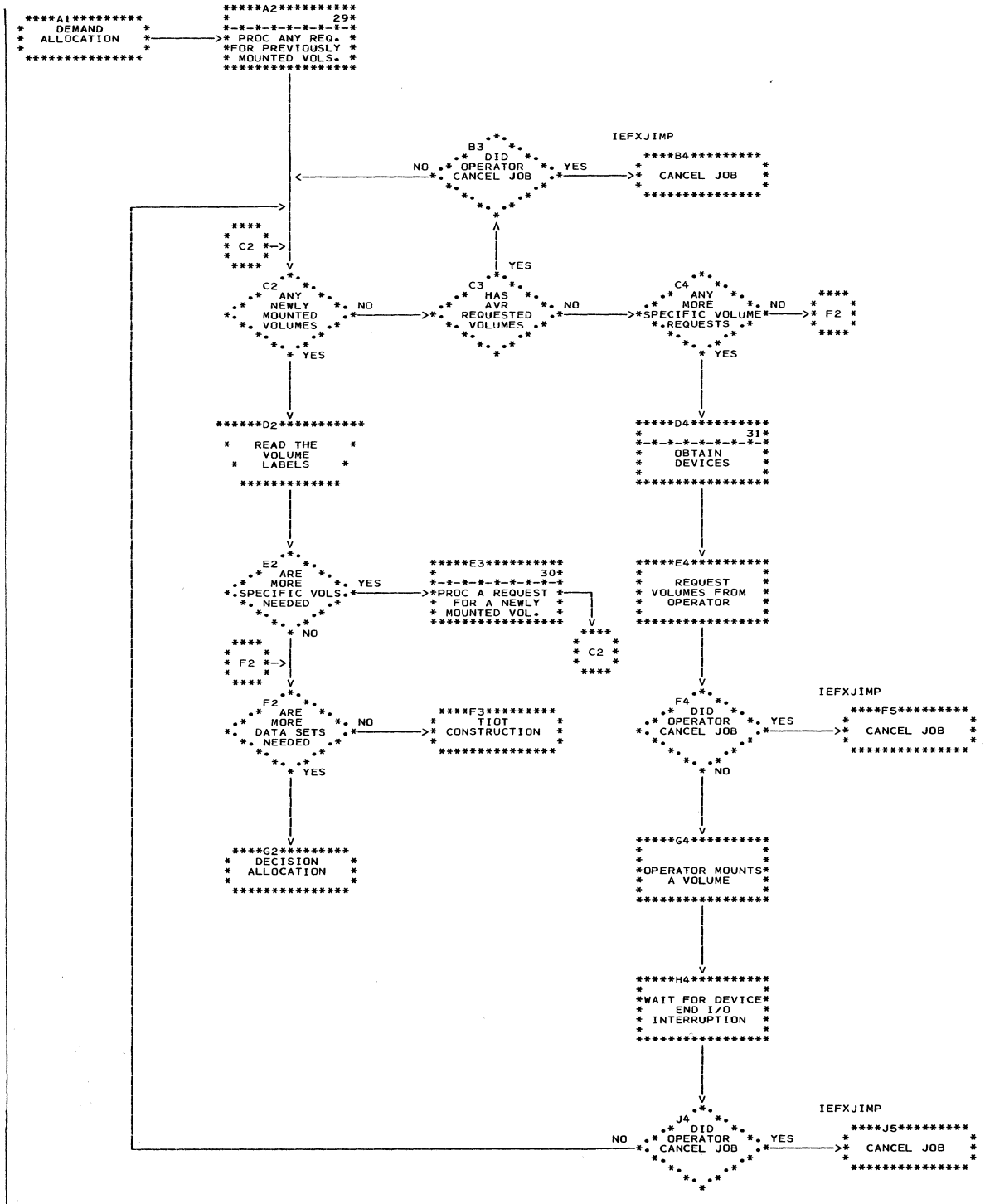
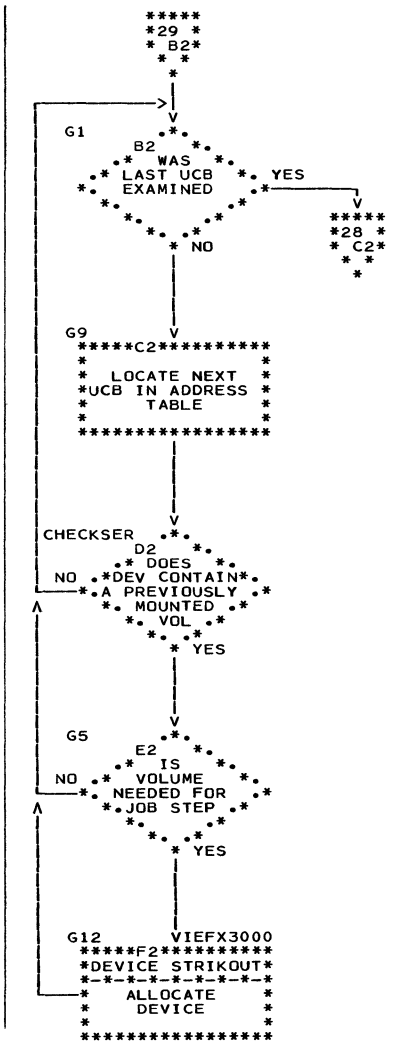
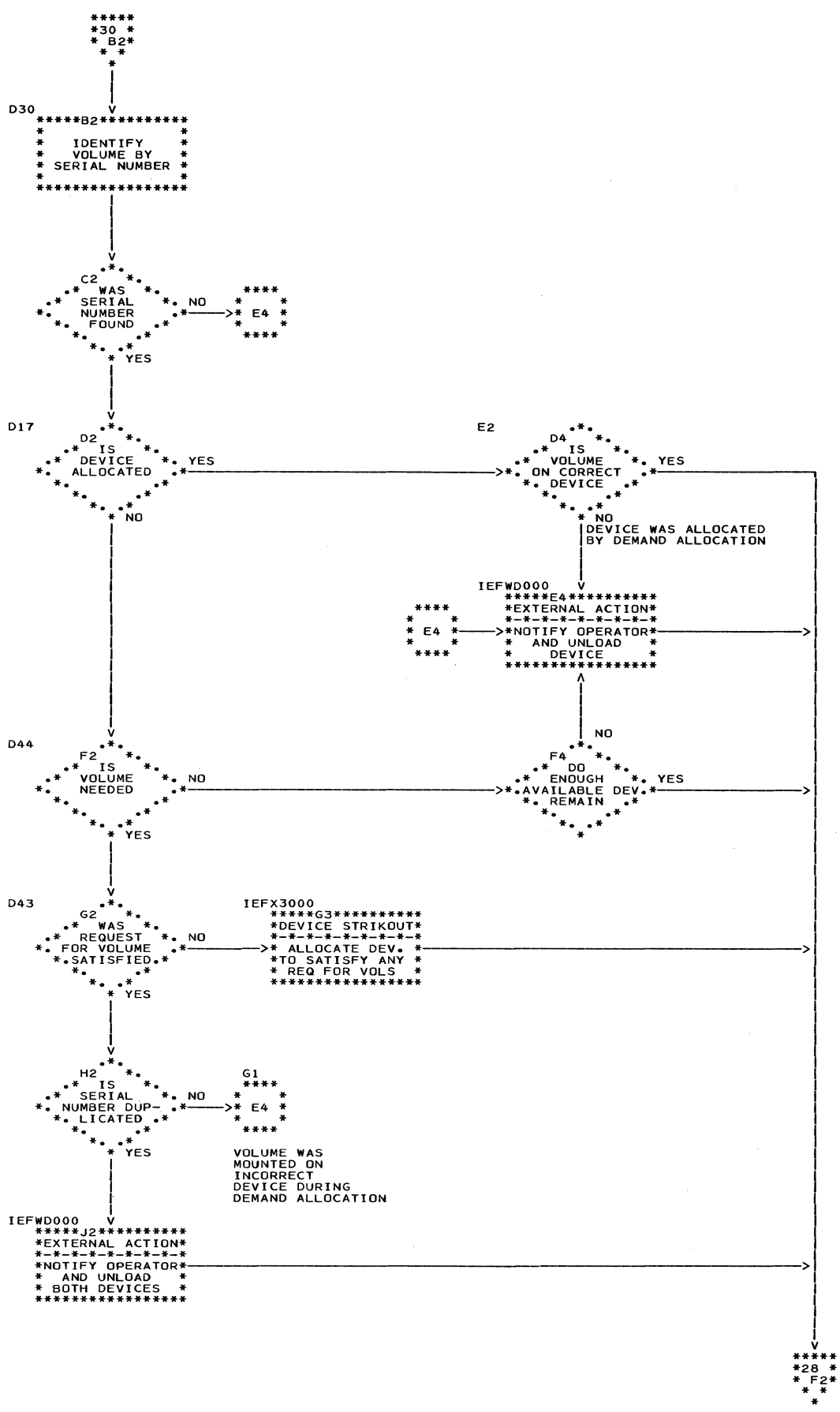


Chart 29. Process Any Requests for Previously Mounted Volumes



- Chart 30. Process a Request for a Newly Mounted Volume



- Chart 31. Obtain Devices

*D30,D31,D80
IEFXVNSL

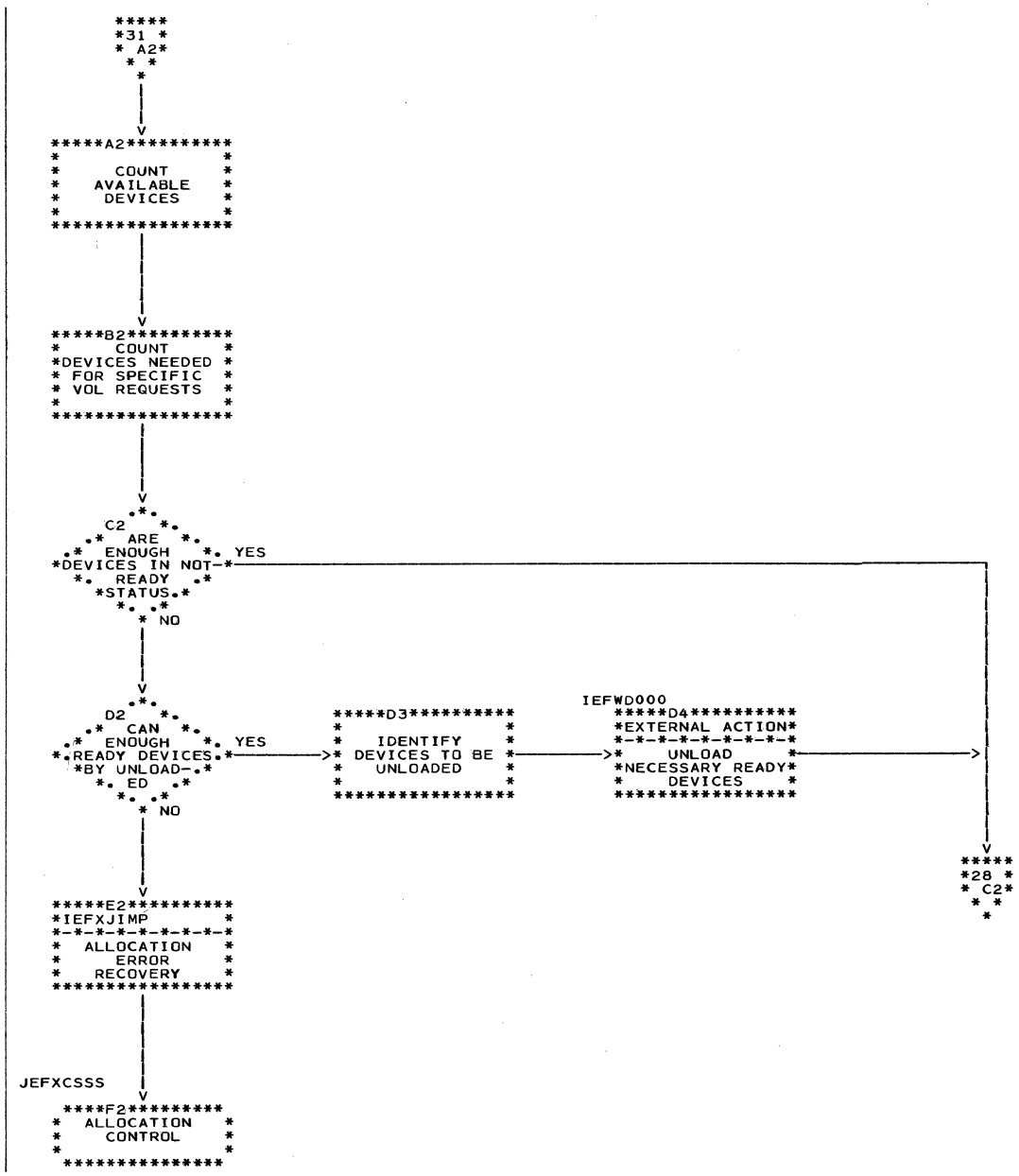


Chart 32. Decision Allocation Routine

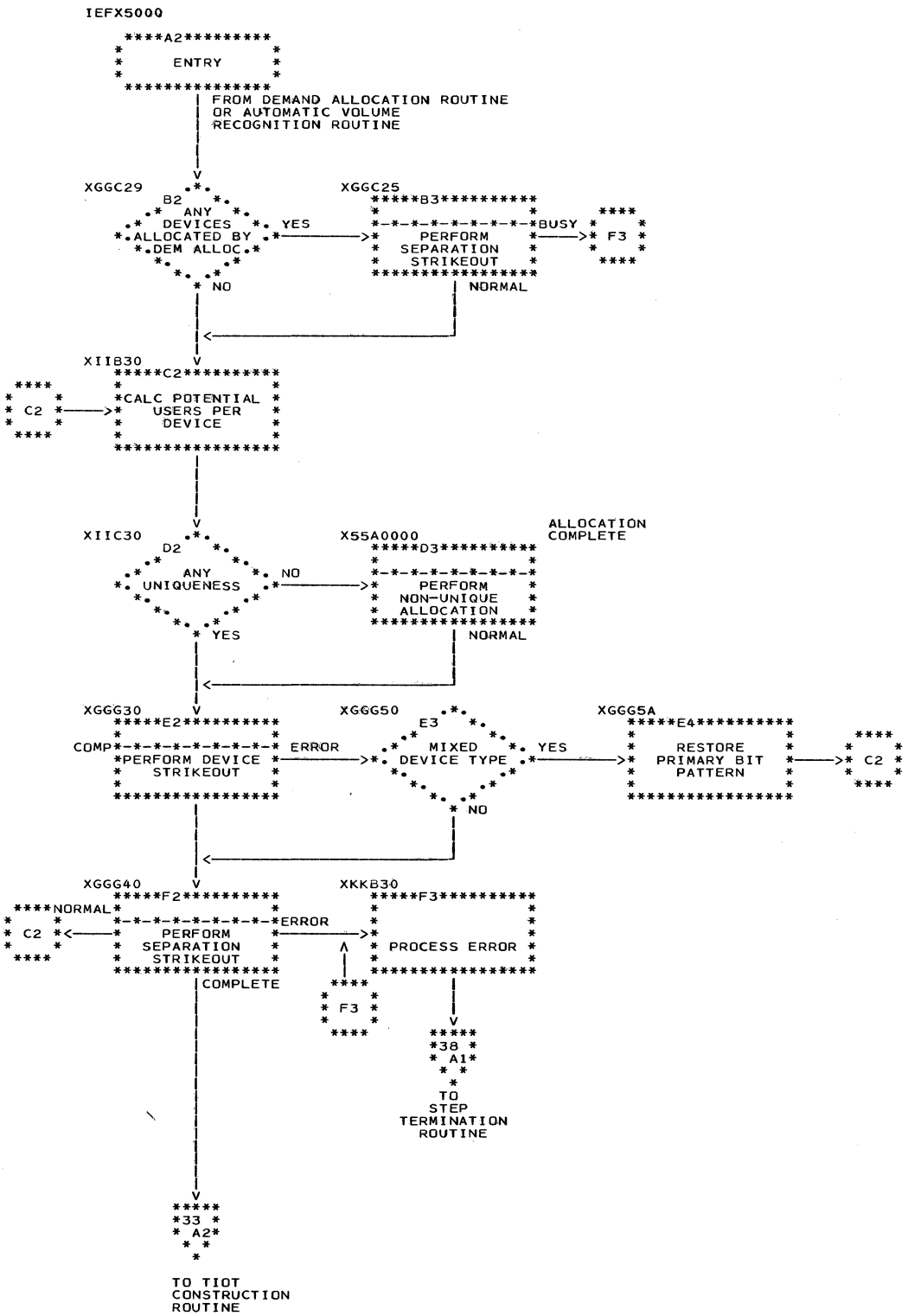


Chart 33. TIOT Construction Routine

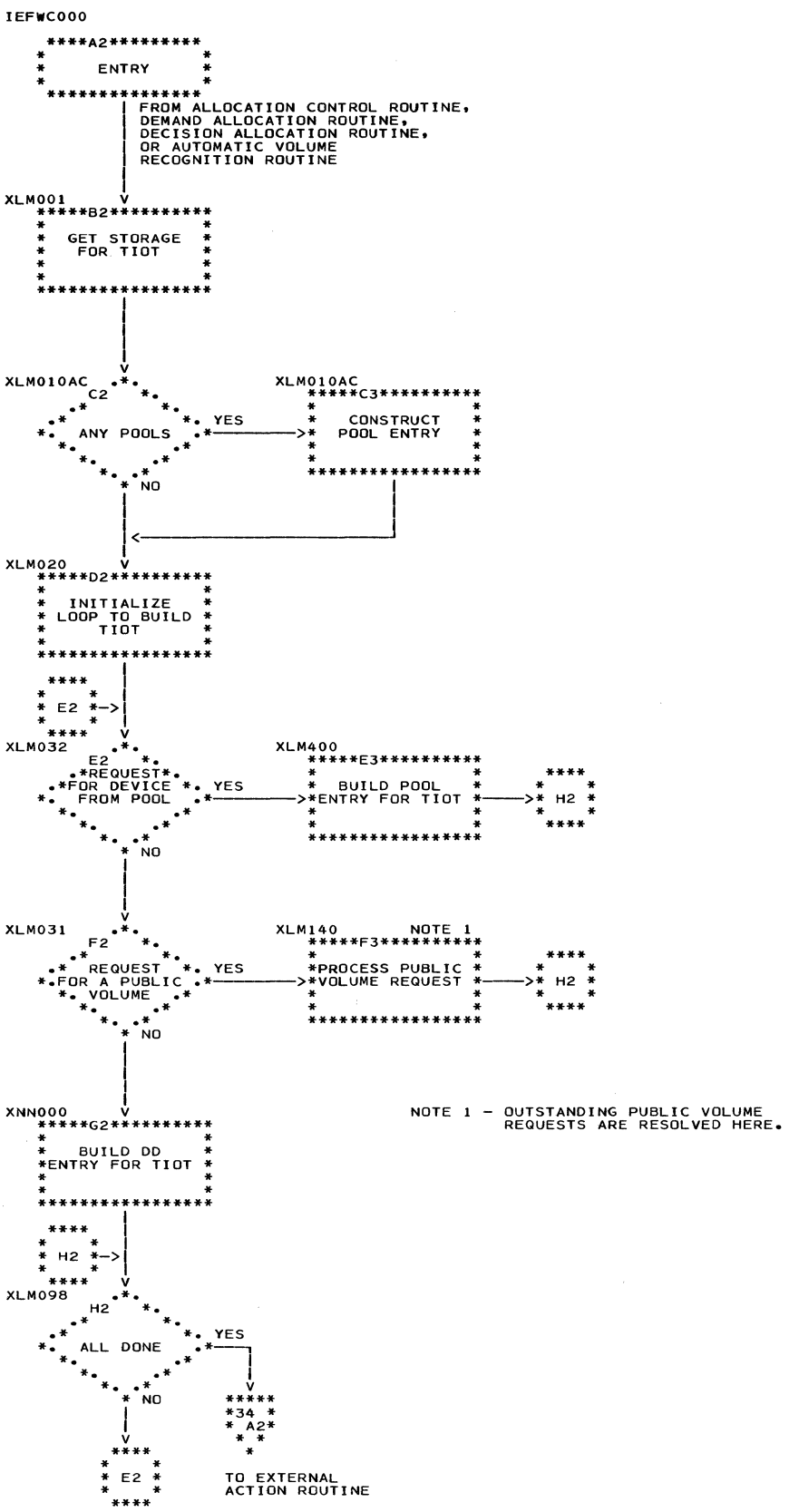
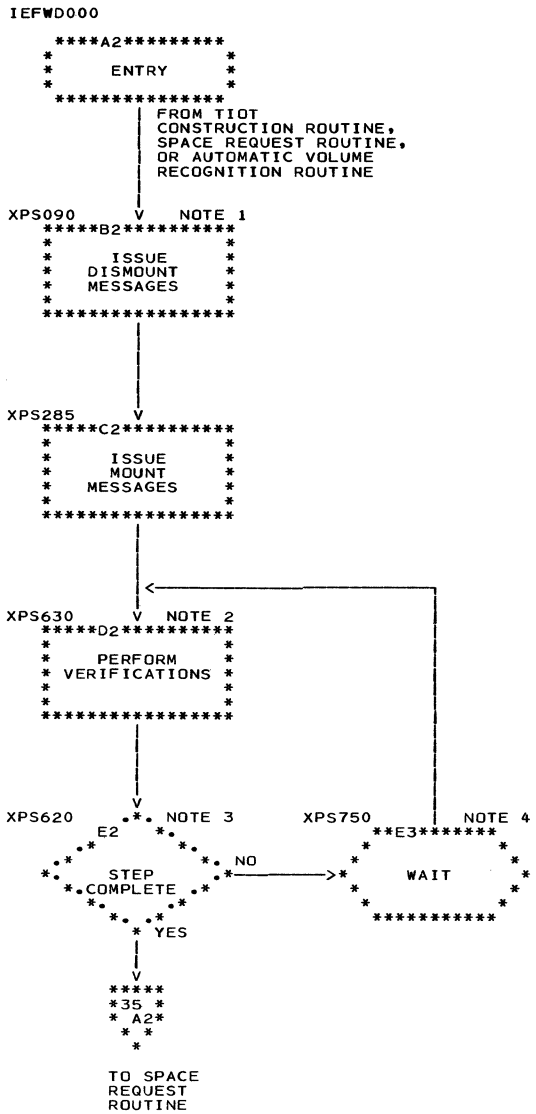


Chart 34. External Action Routine



NOTE 1 INCLUDES A SCAN TO DETERMINE IF THE REQUIRED VOLUMES ARE MOUNTED ON UNALLOCATED DEVICES.

NOTE 2 UNLOAD COMMANDS FROM MASTER SCHEDULER ARE HONORED. UCB'S ARE UPDATED IF REQUIRED.

NOTE 3 THE STEP IS COMPLETE WHEN ALL SETUP MESSAGES HAVE BEEN ISSUED AND VERIFICATION HAS BEEN PERFORMED. COUNTS IN THE SCT CONTROL THIS MECHANISM.

NOTE 4 EITHER OF TWO EVENTS IS WAITED UPON. ISSUANCE OF A CANCEL COMMAND OR A DEVICE BEING MADE READY.

Chart 35. Space Request Routine

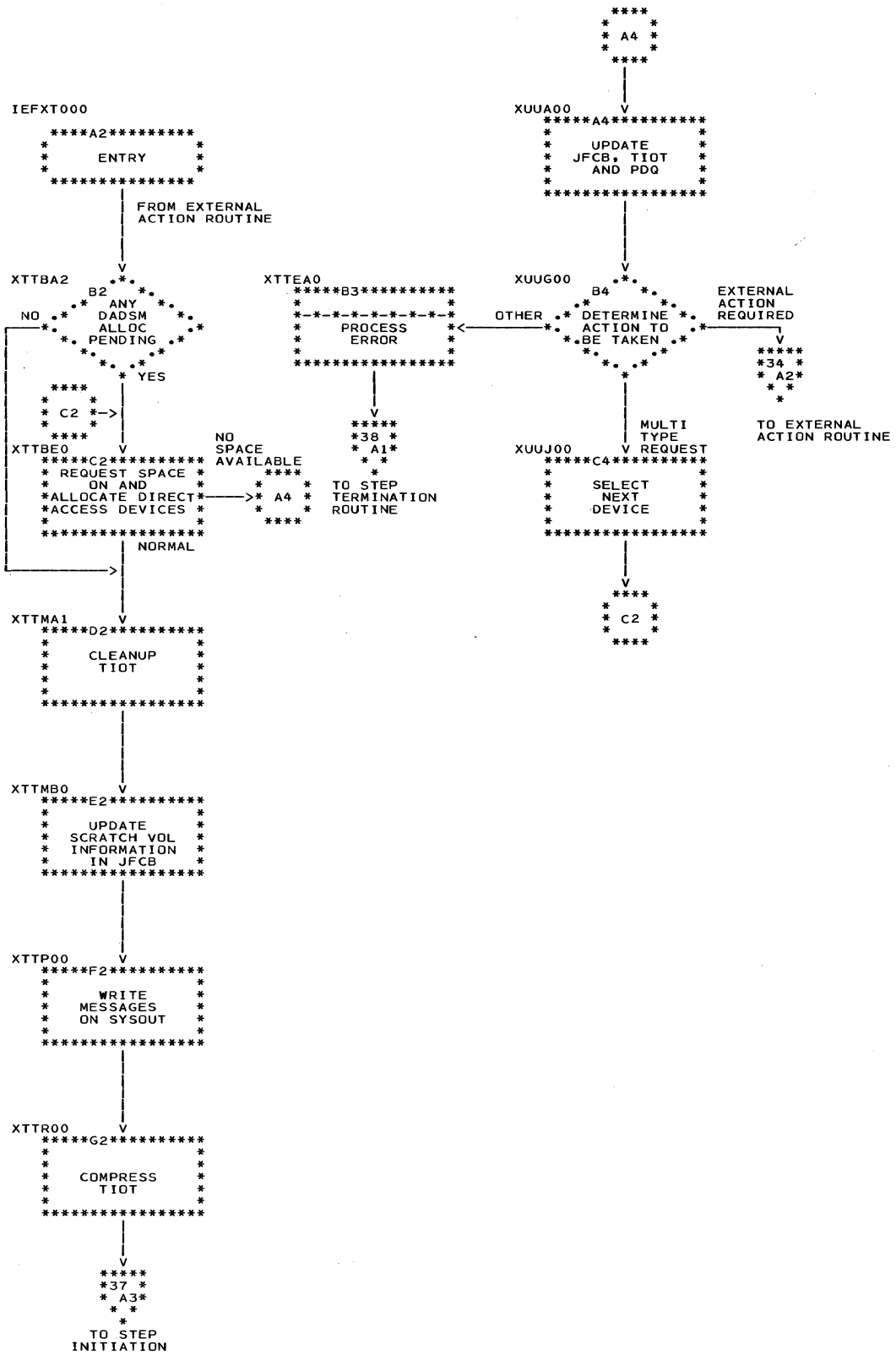
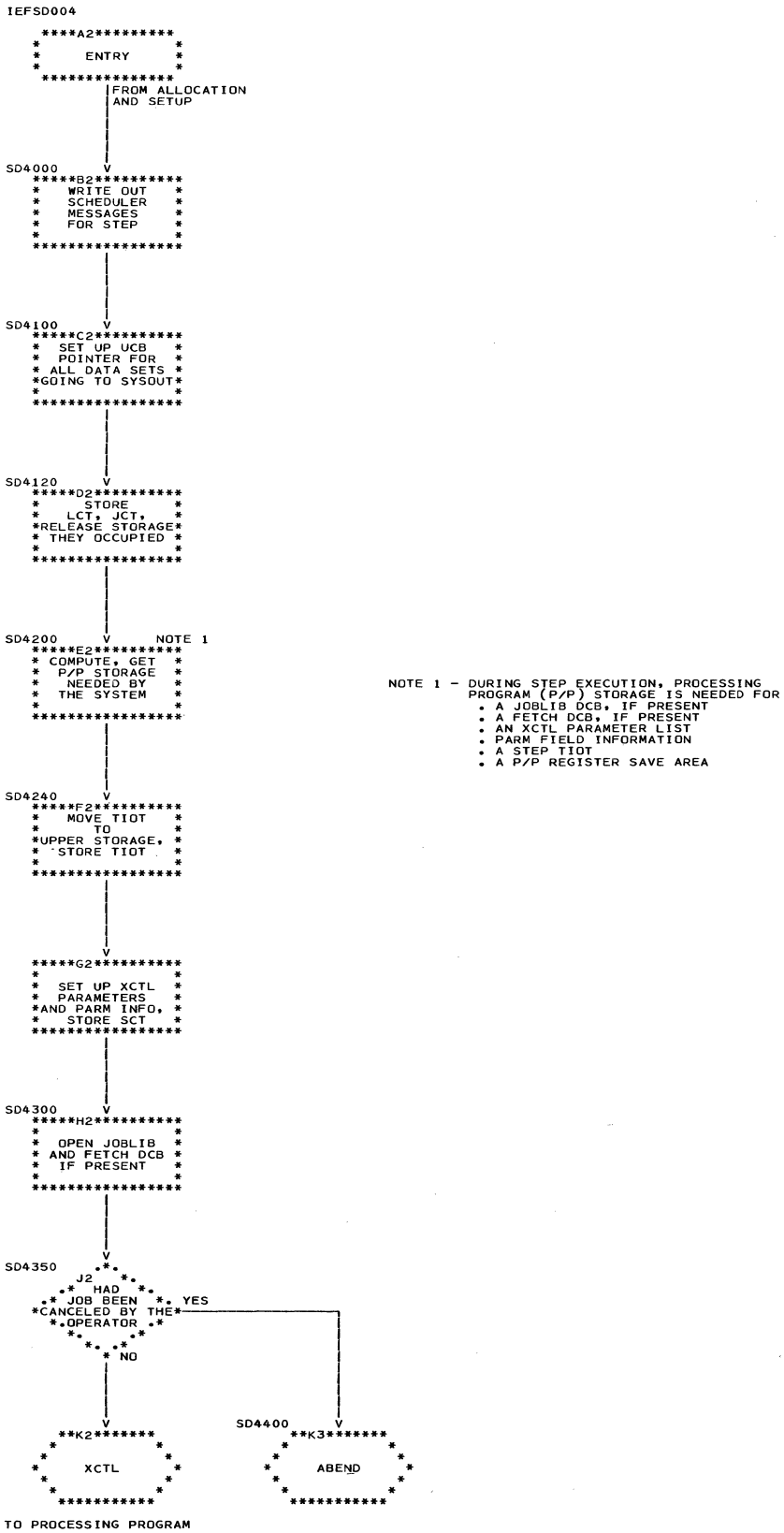


Chart 36. Step Initiation



NOTE 1 - DURING STEP EXECUTION, PROCESSING PROGRAM (P/P) STORAGE IS NEEDED FOR
- A JOBLIB DCB, IF PRESENT
- A FETCH DCB, IF PRESENT
- AN XCTL PARAMETER LIST
- PARM FIELD INFORMATION
- A STEP TIOT
- A P/P REGISTER SAVE AREA

Chart 37. Termination

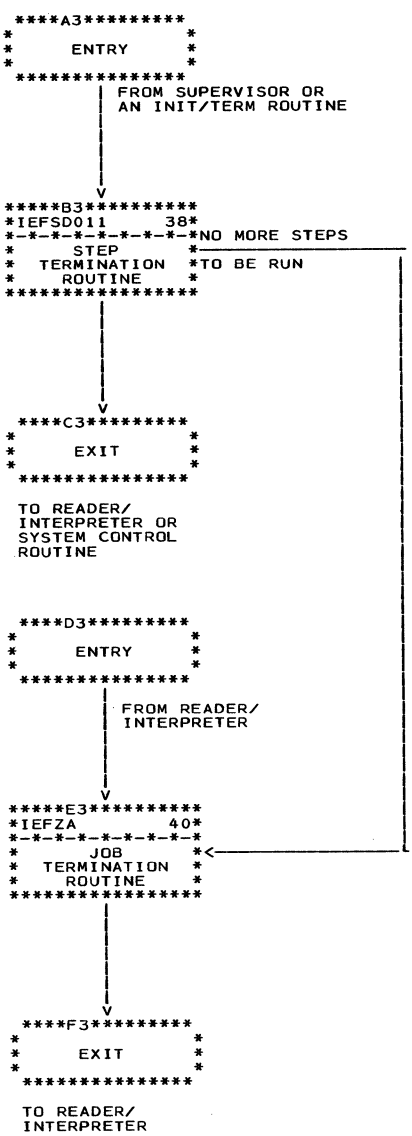


Chart 38. Step Termination Routine

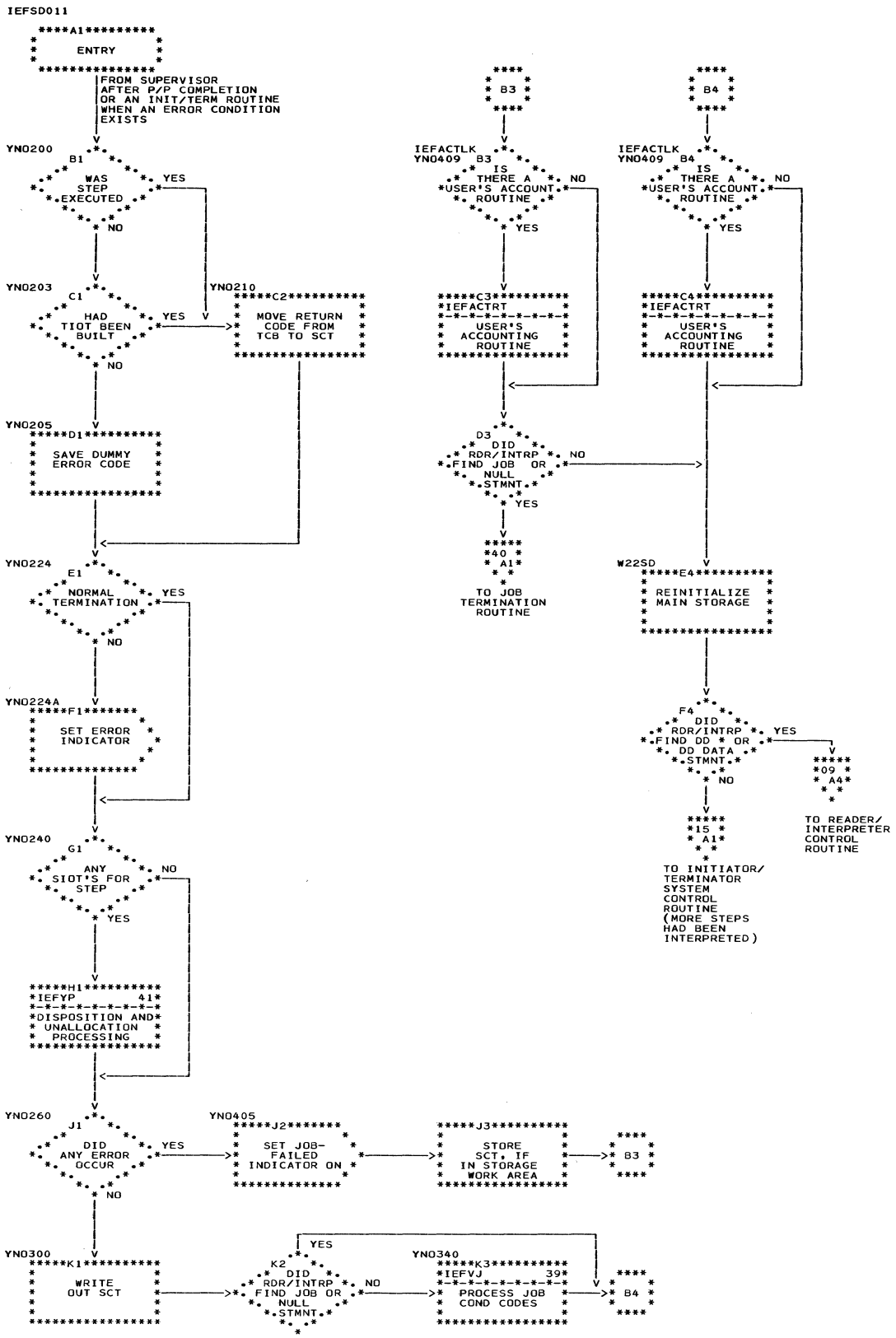


Chart 39. Job Statement Condition Code Routine

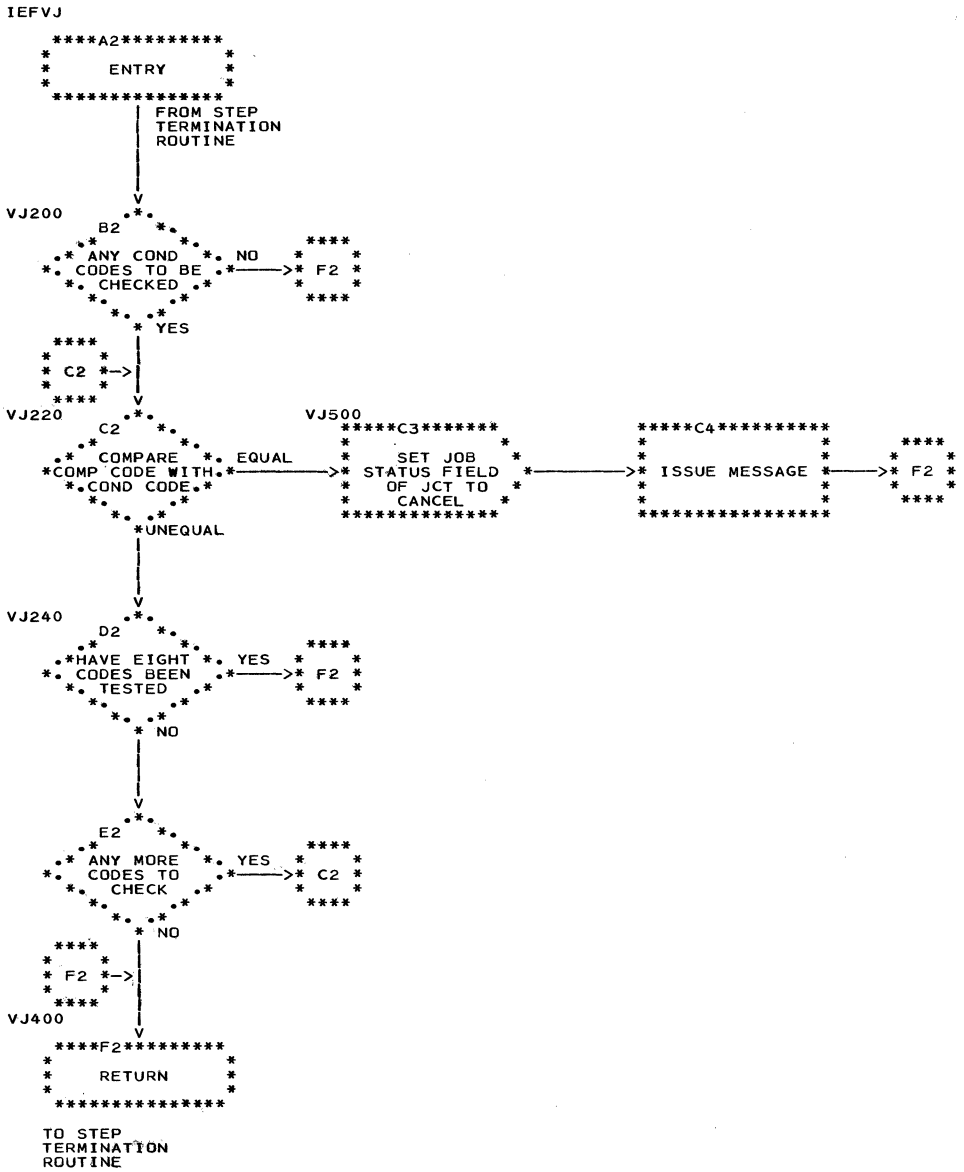


Chart 40. Job Termination Routine

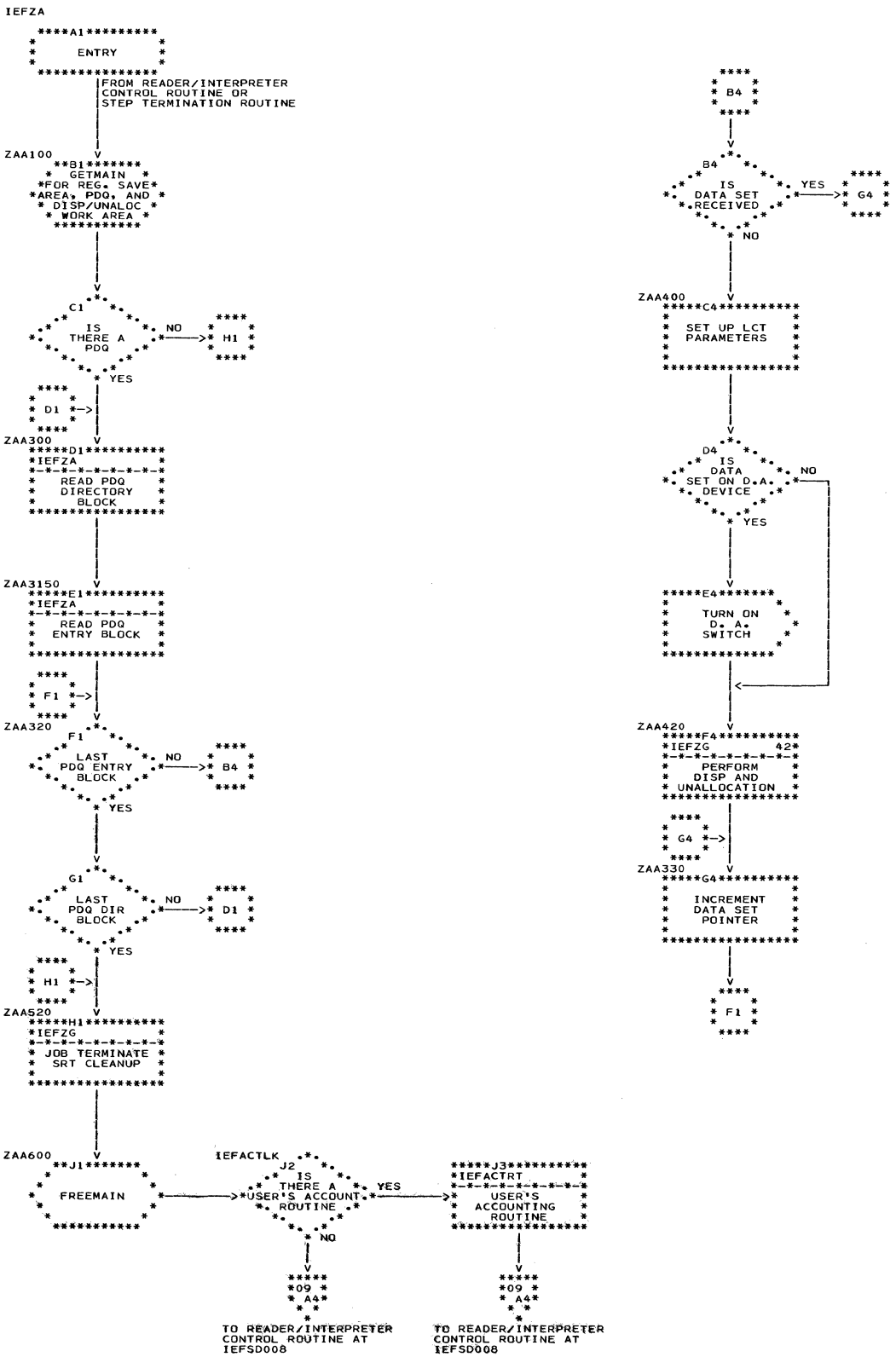


Chart 41. Disposition and Unallocation Subroutine - Entry From Step Termination Routine

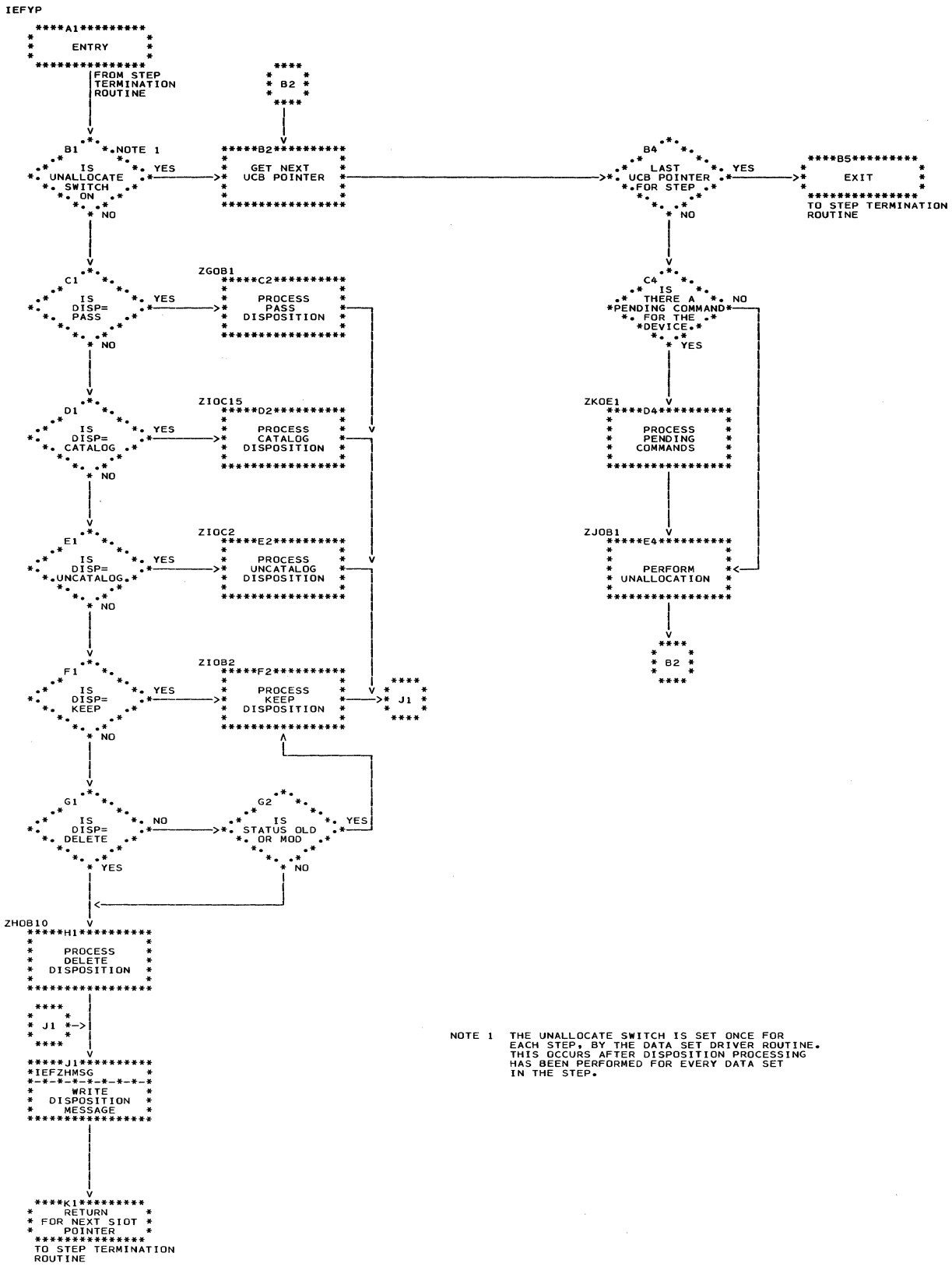


Chart 42. Disposition and Unallocation Subroutine - Entry From Job Termination Routine

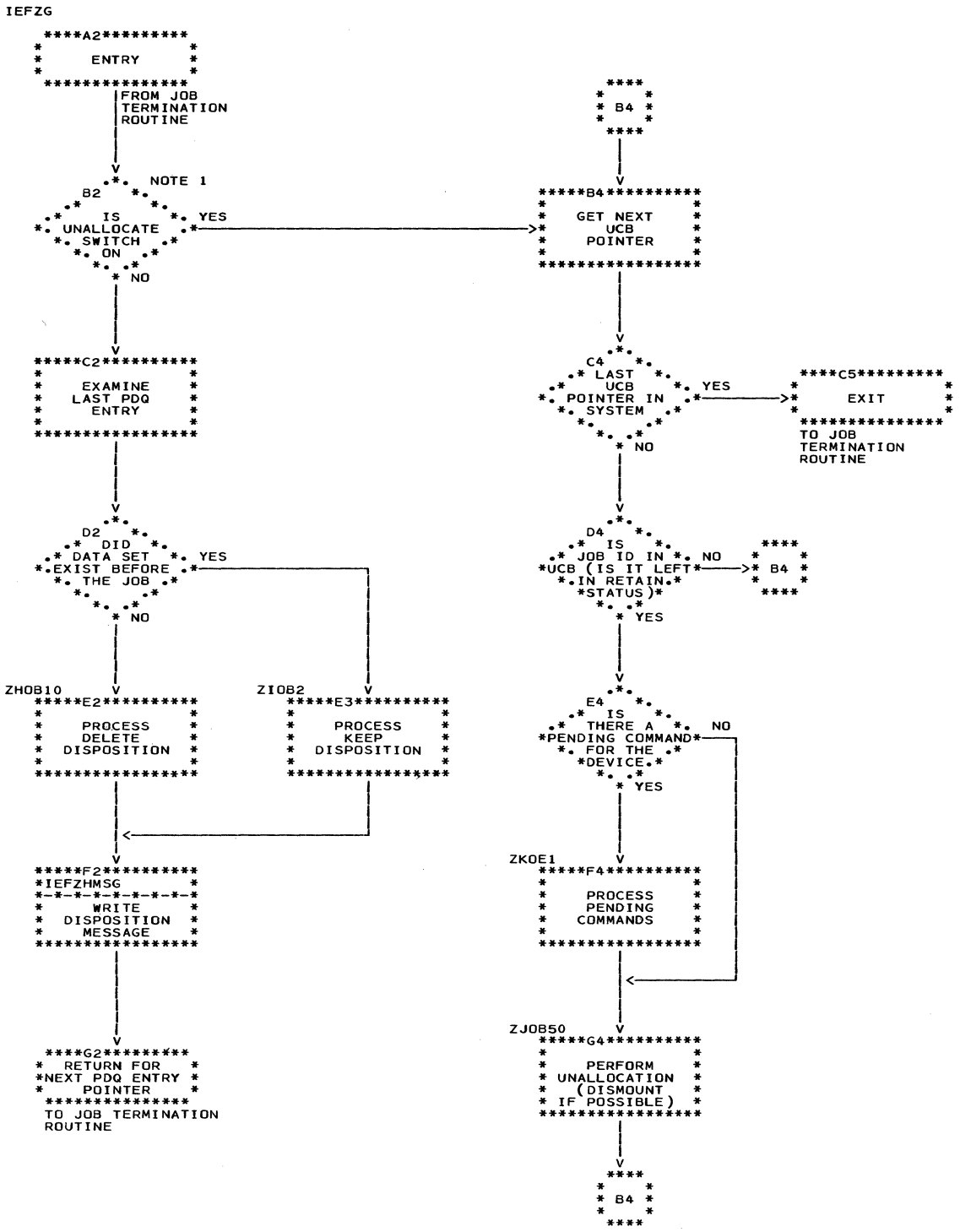


Chart 43. 18K Configuration Load Module Control Flow

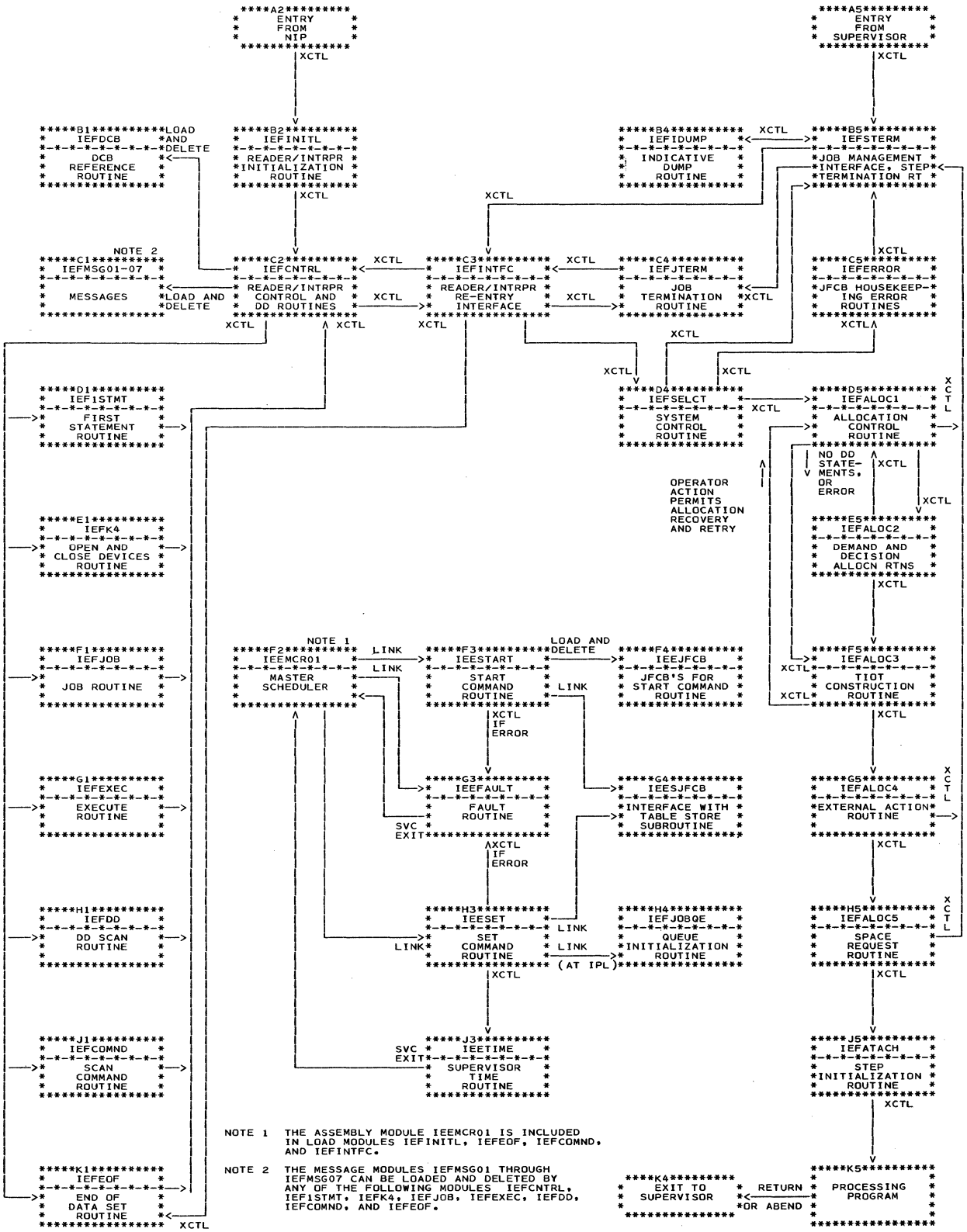
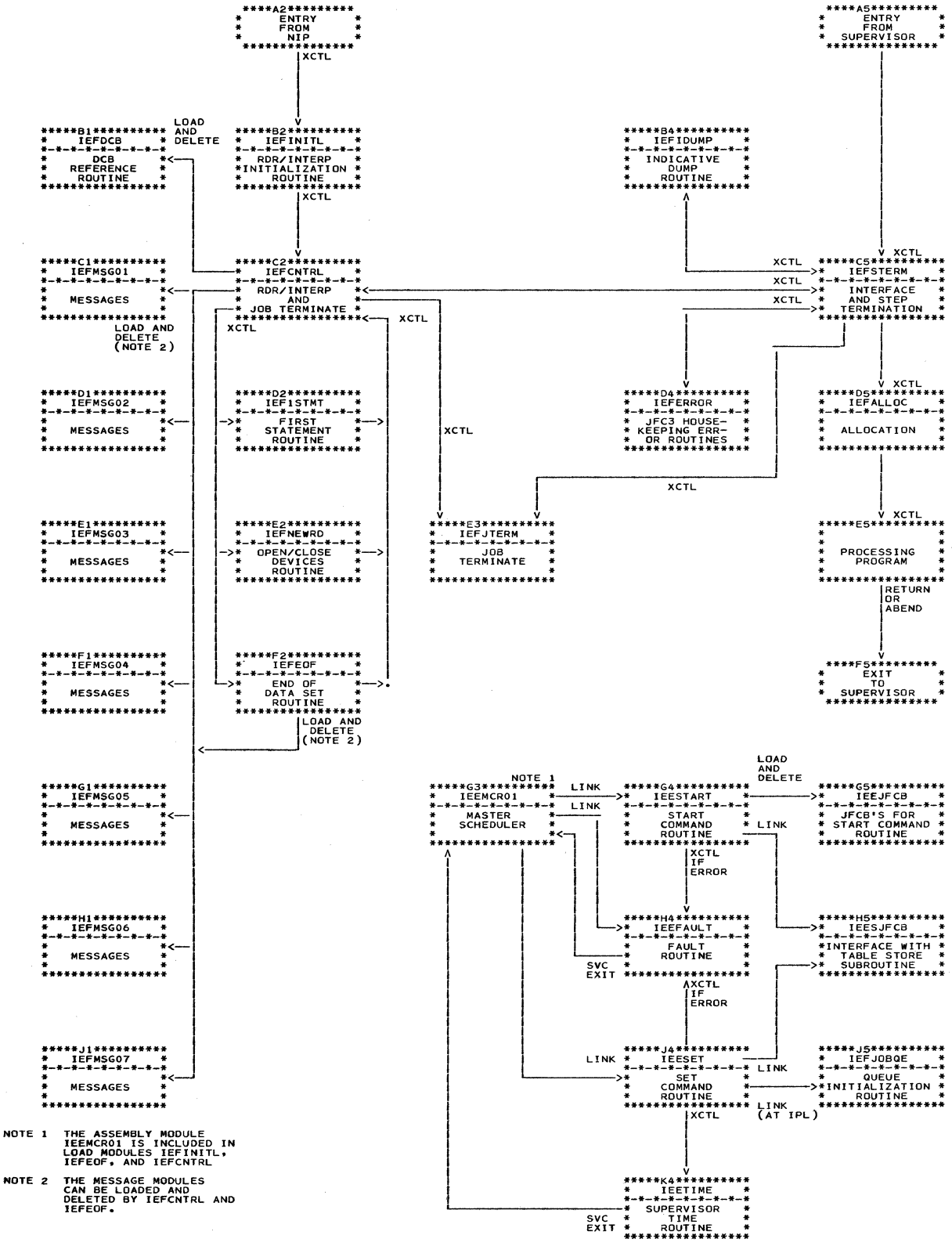


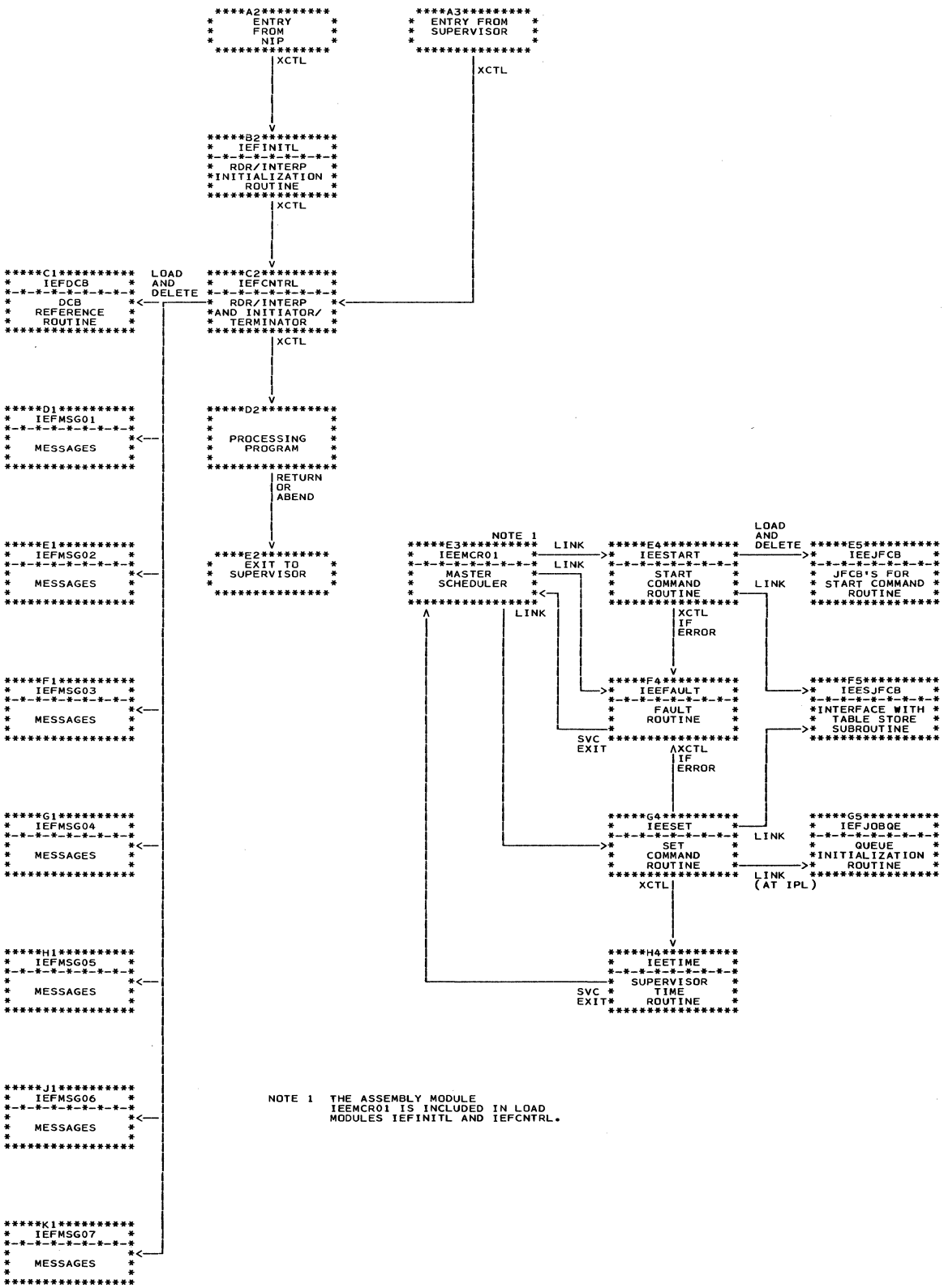
Chart 44. 44K Configuration Load Module Control Flow



NOTE 1 THE ASSEMBLY MODULE IEEMCR01 IS INCLUDED IN LOAD MODULES IEFINITL, IEFEOF, AND IEFCNTRL

NOTE 2 THE MESSAGE MODULES CAN BE LOADED AND DELETED BY IEFCNTRL AND IEFEOF.

Chart 45. 100K Configuration Load Module Control Flow



International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

READER'S COMMENTS

Title: IBM System/360 Operating System
Job Management
Program Logic Manual

Form: Y28-6613-1

Is the material:	Yes	No
Easy to Read?	___	___
Well organized?	___	___
Complete?	___	___
Well illustrated?	___	___
Accurate?	___	___
Suitable for its intended audience?	___	___

How did you use this publication?
___ As an introduction to the subject ___ For additional knowledge
Other _____

fold

Please check the items that describe your position:

___ Customer personnel	___ Operator	___ Sales Representative
___ IBM personnel	___ Programmer	___ Systems Engineer
___ Manager	___ Customer Engineer	___ Trainee
___ Systems Analyst	___ Instructor	Other_____

Please check specific criticism(s), give page number(s),and explain below:
___ Clarification on page(s)
___ Addition on page(s)
___ Deletion on page(s)
___ Error on page(s)

Explanation:

fold

FOLD ON TWO LINES,STAPLE AND MAIL
No Postage Necessary if Mailed in U.S.A.

staple sta

fold f

FIRST CLASS
PERMIT NO. 81

POUGHKEEPSIE, N.Y.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

IBM CORPORATION
P.O. BOX 390
POUGHKEEPSIE, N. Y. 12602

ATTN: PROGRAMMING SYSTEMS PUBLICATIONS
DEPARTMENT D58

fold f

Printed in U.S.A. Y28-6613-1

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

sta