LY28-1745-1
File No. S370-36

Program Product

MVS/Extended Architecture System Logic Library: Scheduler Restart

MVS/System Product:

JES3 Version 2	**5665-291**
JES2 Version 2	**5740-XC6**

Second Edition (September, 1989)

This is a major revision of, and obsoletes, LY28-1745-0. See the Summary of Amendments following the Contents for a summary of the changes made to this manual. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

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

References in this publication to IBM products or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product in this publication is not intended to state or imply that only IBM's product may be used. Any functionally equivalent product may be used instead. This statement does not expressly or implicitly waive any intellectual property right IBM may hold in any product mentioned herein.

Publications are not stocked at the address given below. Requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Information Development, Department D58, Building 921-2, PO Box 950, Poughkeepsie, N.Y. 12602. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

PREFACE

The MVS/Extended Architecture System Logic Library is intended for people who debug or modify the MVS control program. It describes the logic of most MVS control program functions that are performed after master scheduler initialization completes. For detailed information about the MVS control program prior to this point, refer to MVS/Extended Architecture System Initialization Logic. For general information about the MVS control program and the relationships among the components that make up the MVS control program, refer to the MVS/Extended Architecture Overview. To obtain the names of publications that describe some of the components not in the System Logic Library, refer to the section Corequisite Reading in the Master Preface in MVS/Extended Architecture System Logic Library: Master Index.

TRADEMARKS

The following are trademarks of International Business Machines Corporation.

- MVS/DFP(TM)
- MVS/XA(TM)

HOW THE LIBRARY IS ORGANIZED

SET OF BOOKS

The System Logic Library consists of a set of books. Two of the books provide information that is relevant to the entire set of books:

1. The MVS/Extended Architecture System Logic Library: Master Index contains the master preface master index for the other books in the set.
2. The MVS/Extended Architecture System Logic Library: Module Descriptions contains module descriptions for all of the modules in the components documented in the System Logic Library.

Each of the other books (referred to as component books) in the set contains its own table of contents and index, and describes the logic of one of the components in the MVS control program.

ORGANIZATION OF THE COMPONENTS

Most component books contain information about one component in the MVS control program. However, some component books (such as System Logic Library: Initiator/Terminator) contain more than one component if the components are closely related, frequently referenced at the same time, and not so large that they require a book of their own.

A three or four character mnemonic is associated with each component book and is used in all diagram and page numbers in that book. For example, the mnemonic ASM is associated with the book MVS/Extended Architecture System Logic Library: Auxiliary Storage Management. All diagrams in this book are identified as Diagram ASM-n, and all pages as ASM-n, where n represents the specific diagram or page number. Whenever possible, the existing component acronym is used as the mnemonic for the component book. The Table of Book Titles in the Master Preface in MVS/Extended Architecture System Logic Library: Master Index lists the book titles, the components included in each book (if a book contains more than one component), the mnemonics for the books, and the order number for each book.

HOW TO USE THE LIBRARY

To help you use this library efficiently, the following topics cover

- How to find information using book titles and the master index
- What types of information are provided for each component
- How to obtain further information about other books in the System Logic Library

FINDING INFORMATION USING THE BOOK TITLES

As you become familiar with the book titles, MVS component names and mnemonics, and the book contents, you will be able to use the System Logic Library as you would an encyclopedia and go directly to the book that you need. We recommend that you group the books in alphabetical order for easy reference, or, if you are familiar with MVS, that you group the books by related functions.

The Table of Book Titles in the Master Preface in MVS/Extended Architecture System Logic Library: Master Index contains a list of book titles and mnemonics. It provides a quick reference to all the books, and their corresponding components, in the System Logic Library.

FINDING INFORMATION USING THE MASTER INDEX

If you are not sure which book contains the information you are looking for, you can locate the book and the page on which the information appears by using the master index in System Logic Library: Master Index. For the component books, the page number in an index entry consists of the mnemonic for the component and the page number.

For example:

ASM-12 refers to MVS/Extended Architecture System Logic Library: Auxiliary Storage Management, page ASM-12.

INFORMATION PROVIDED FOR MOST COMPONENTS

The following information is provided for most of the components described in the System Logic Library.

1. An introduction that summarizes the component's function
2. Control block overview figures that show significant fields and the chaining structure of the component's control blocks
3. Process flow figures that show control flow between the component's object modules
4. Module information that describes the functional organization of a program. This information can be in the form of:
   - Method-of-Operation diagrams and extended descriptions.
   - Automatically-generated prose. The automated module information is generated from the module prologue and the code itself. It consists of three parts: module description, module operation summary, and diagnostic aids.
5. Module descriptions that describe the operation of the modules (the module descriptions are contained in System Logic Library: Module Descriptions)

Some component books also include diagnostic techniques information following the Introduction.

FURTHER INFORMATION

For more information about the System Logic Library, including the order numbers of the books in the System Logic Library, see the Master Preface in MVS/Extended Architecture System Logic Library: Master Index.

CONTENTS

FIGURES

Summary of Amendments for LY28-1745-1 for MVS/System Product Version 2 Release 2.3

This major revision contains changes to support MVS/System Product Version 2 Release 2.

Changes to this publication include:

- MVS/Extended Architecture Data Facility Product (MVS/XA DFP) Version 3 Release 1.0, which introduces the storage management subsystem (SMS). SMS provides new function for data and storage management.
- Update module IEFXB609.
- The addition of several new checkpoint restrictions.
- The addition of the new modules to the process flow.
- The addition of six new modules (in logic tool format).
  - IEFXBSJX
  - IEFXBDYS
  - IEFXBGDG
  - IEFXBUSJ
  - IEFXBRDC
  - IEFXBSWA
- The Preface has been updated to include the new title for the MVS/XA System Logic Library: Master Index and the deletion of the index from MVS/XA System Logic Library: Module Descriptions.
- Minor editorial changes.

Summary of Amendments for LY28-1745-0 for MVS/System Product Version 2 Release 2.0

Scheduler Restart is now the name used for this component. It had been called Checkpoint/Restart.

This publication is new for MVS System Product Version 2 Release 2.0. It contains information that was reorganized from the Checkpoint/Restart section in MVS/XA System Logic Library Volume 3, LY28-1214, which applies to MVS/System Product Version 2 Release 1.7.

This publication contains changes to support MVS/System Product Version 2 Release 2.0. The changes include:

- Updates to modules IEFXB602 and IEFRPREP.
- Addition of information for new module IEFRCSTP.
- Minor technical and editorial changes throughout the publication.

INTRODUCTION

The job scheduler restart facility consists of routines that collect job-related information and process this information in the event of a job or system failure. This information is recorded either at programmer-designated checkpoints or whenever SWA-contained control blocks that are critical to a job's processing are updated. This permits termination of active jobs in the event of a system failure. It also allows the restarting of a job step from the beginning, its most recent checkpoint if automatic restart is requested, or from a programmer-designated checkpoint if deferred restart is requested.

RESTRICTIONS

You can use checkpoints with the following restrictions:

- A routine that is restricted from issuing SVCs (for example, a routine executing in SRB, disabled, or cross memory mode) is also restricted from establishing checkpoints because programmer-designated checkpoints require the use of the checkpoint SVC.
- An exit routine other than the end-of-volume exit routine cannot request a checkpoint.
- A routine invoked by a program call (PC) cannot request checkpoints because the system environment might be different at the time of the restart from what it was at the time of the checkpoint. This could lead to unpredictable results on the return linkage (PT).
- A routine with a PCLINK STACK request outstanding cannot establish a checkpoint.
- Routines that use both PC/AUTH and scheduler restart facilities must reestablish their PC/AUTH environment at restart. In addition, they must not use any PC/AUTH data (for example, a PC number) that was obtained prior to the restart.
- Subsystems that use the TCB subsystem affinity service cannot issue checkpoints. This is because the subsystem affinity table (SSAT) index values might change from one system initialization to another.
- Permanent dynamically concatenated JCL defined DDs will not be properly resolved on a restart. The result is a missing DD statement.
- The use of permanently concatenated dynamically allocated datasets with automatic checkpoint restart may produce incorrect results.
- Dynamically concatenated DDs may invalidate unit affinity requests to the DDs being concatenated.

For additional information concerning the restrictions on the use of checkpoints, see MVS/Extended Architecture Checkpoint/Restart.

DSDR PROCESSING

Data set descriptor record (DSDR) processing routines use data from a checkpoint data set to update the control blocks in the scheduler work area (SWA). A CHKPT macro instruction results in the issuing of a checkpoint SVC to save information in the checkpoint data set. Scheduler restart routines use this information when a job restarts at the designated checkpoint. The information required by the scheduler is saved in the DSDR.

THE JOB JOURNAL

The job journal, a logical sequential data set residing on JES2's or JES3's direct access spool volume, provides backup direct access storage for the scheduler work area. It contains copies of SWA control blocks that are critical to the restart processing of a job. Each job has its own job journal, which is a temporary data set that exists for the life of the job. For each job, the initial entry to the job journal is a job header record (JHR). A step header record (SHR) is written to the job journal just prior to the job step allocation processing for each step. At the completion of allocation processing, the critical SWA control blocks for the job step are written to the job journal.

As SWA is updated during the processing of each job step, a copy of each critical control block for the step is written to the job journal. Thus, an audit trail of the necessary control blocks for each job is maintained to provide for the reconstruction of the SWA for the following forms of restart:

- Automatic checkpoint restart
- Automatic step restart
- System restart
- Continue restart

To support multiple subsequent restarts, all critical blocks for all steps up to the failing steps are rewritten to the job journal.

Job-processing routines write the following critical SWA control blocks to the job journal:

- Job control table (JCT)
- Job control table extension (JCTX)
- Step control table (SCT)
- Step input/output table (SIOT)
- Job file control block (JFCB)
- Job file control block extension for 3800 printer (JFCBE)
- Job file control block extension (JFCBX)
- Passed data set information block (PDIB)
- Generation data group name table (GDGNT)
- Account control table (ACT)
- Volume unload table (VUT)
- Virtual data set control block (VDSCB)
- Data set page control table header (DSPCT)

JOURNAL ROUTINES

The journal write routine is responsible for maintaining the job journal. The journal write routine determines which control blocks are necessary for restart and writes those blocks to the job journal. As critical blocks are altered during the processing of a job step (for example, due to an open, scratch, close, checkpoint, or dynamic allocation procedure) the journal write routine writes the updated blocks to the journal.

The SWA resides in virtual (pageable) storage and contains the control blocks used by the scheduler during job processing. When initiator routines perform termination processing at job step failure time, they free the SWA. If a failing job step is authorized for automatic restart, the SWA must be reconstructed using the information preserved in the job journal. The SWA contents are unrecoverable in the event of a system failure. For this situation, restart routines use the information preserved in the job journal to reconstruct the SWA in order to perform termination processing for all active jobs that are eligible for restart.

For system and automatic restart processing, the journal merge routine reconstructs the SWA so it appears as follows:

- When used for automatic checkpoint restart, the SWA contains the control blocks in effect as of the last time they were journalled. Dynamically allocated SIOTS that were present at the checkpoint are not present if they were unallocated after the checkpoint.
- When a system failure occurs, the SWA contains the control blocks in effect at the point of failure. If a system failure occurs during job step termination processing, the job is re-enqueued for step-continue processing if there are any additional steps in the job.
- When used for step-continue processing, the SWA contains the control blocks necessary to permit a restart at the next job step.
- For automatic step restarts, the SWA contains the control blocks necessary to start at the beginning of the failing step.

Figure 1 on page SCR-8 shows an overview of the following control blocks for scheduler restart.

Acronym	Control Block Name
ACT	Accounting control table
CVT	Communications vector table
EPA	External parameter area
GDG	Generation data group
JCT	Job control table
JCTX	Job control table extension
JFCB	Job file control block
JFCBE	Job file control block extension for 3800
JFCBX	Job file control block extension
JSCB	Job step control block
LCT	Linkage control table
MEL	Merge entrance list
PDI	Passed data set information
QMPA	Queue manager parameter area
SCT	Step control table
SIOT	Step input/output table
SWA	Scheduler work area
SWB	Scheduler work block
TCB	Task control block
VAT	Virtual address table
VATX	Virtual address table extension
VUT	Volume unload table

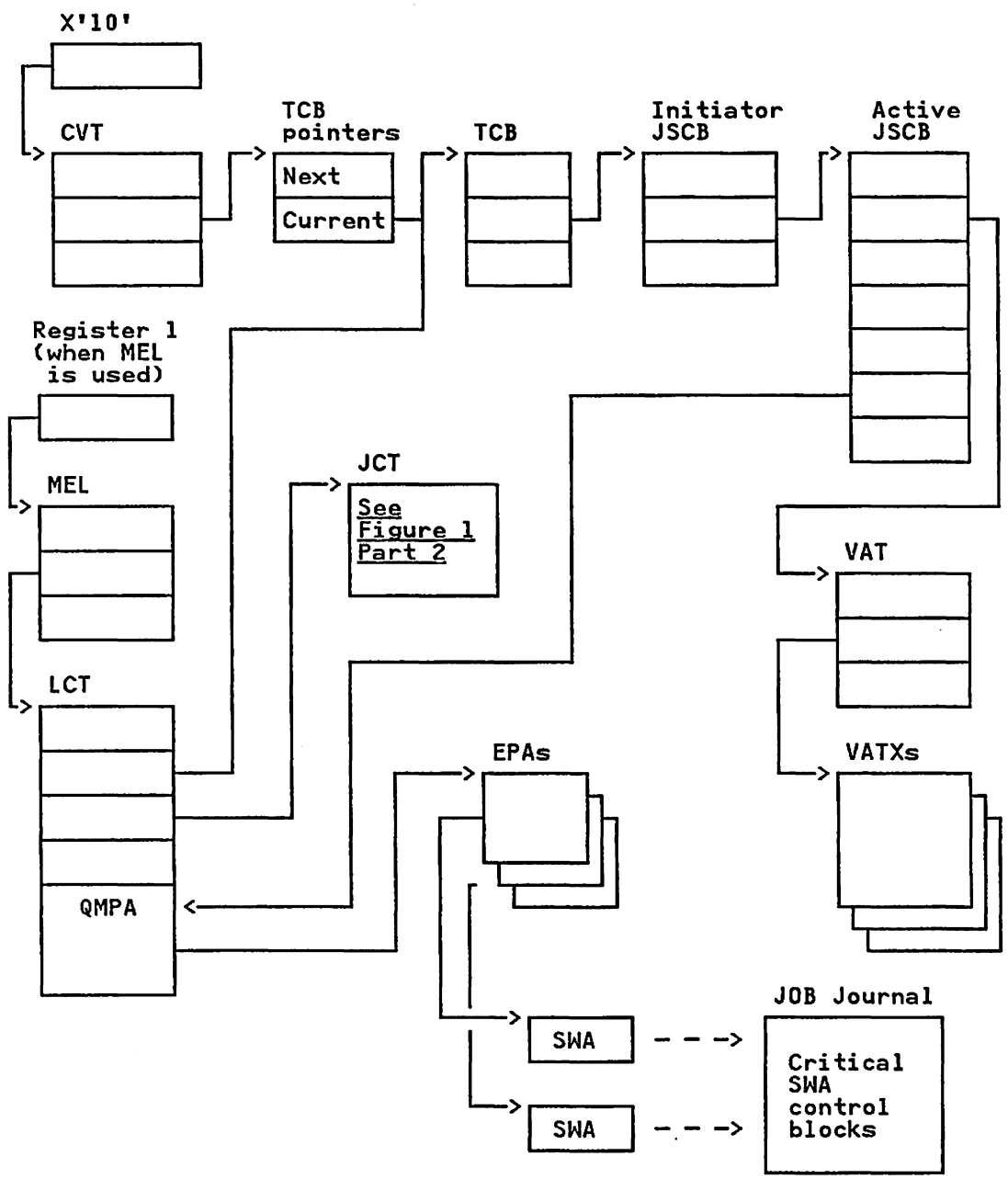


Figure 1 (Part 1 of 2). Scheduler Restart Control Block Overview

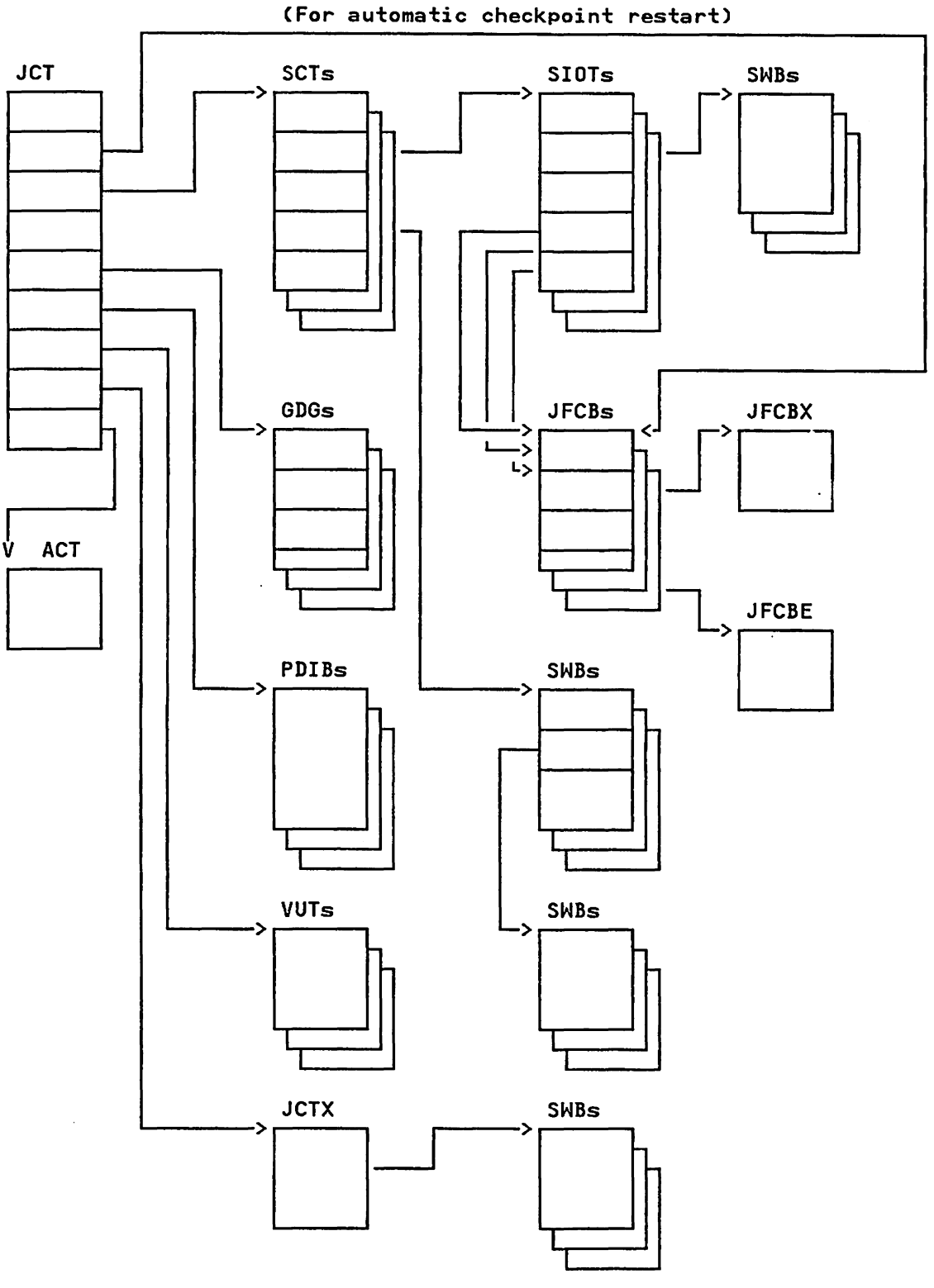


Figure 1 (Part 2 of 2). Scheduler Restart Control Block Overview

PROCESS FLOW

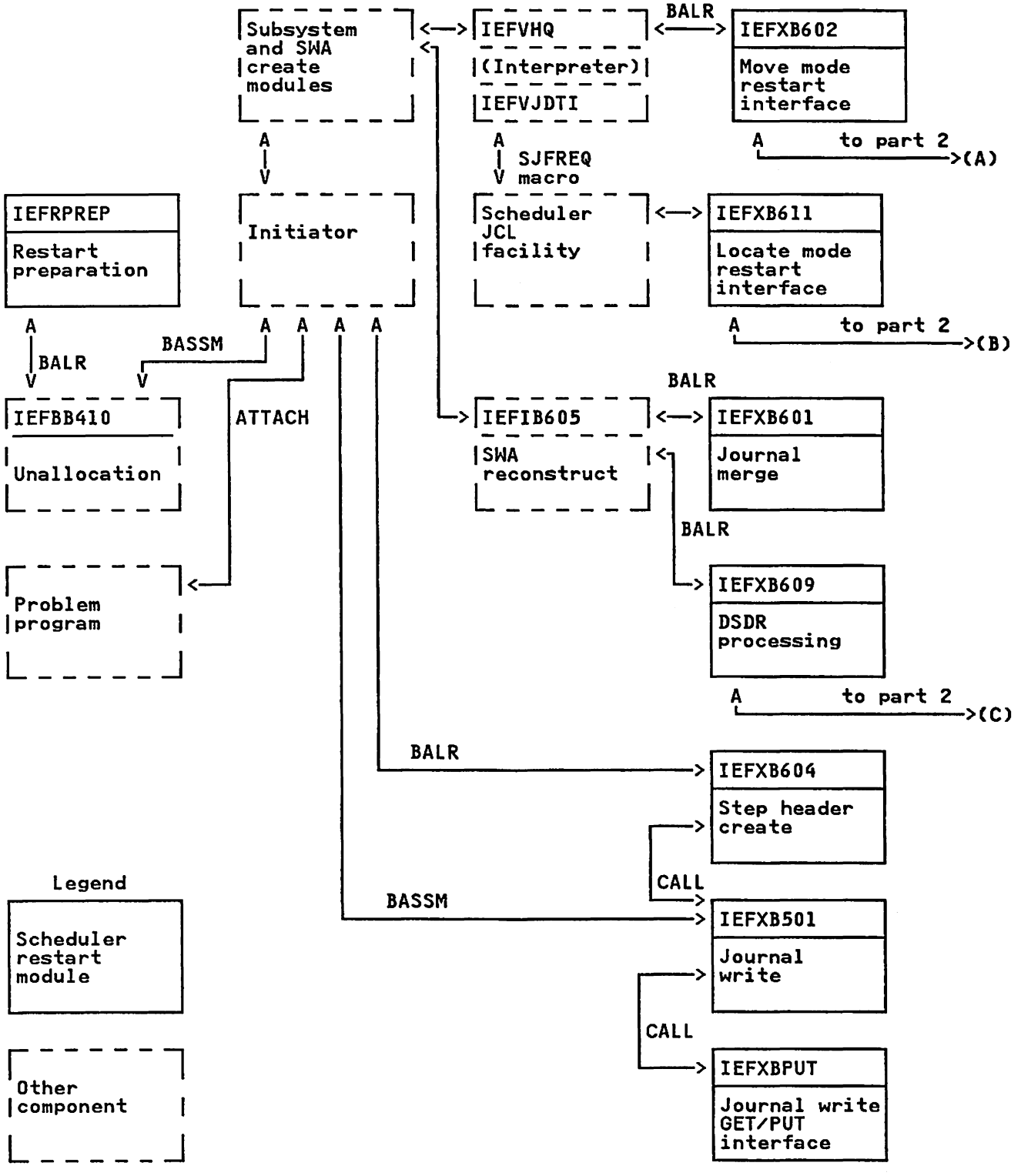


Figure 2 (Part 1 of 2). Scheduler Restart Process Flow

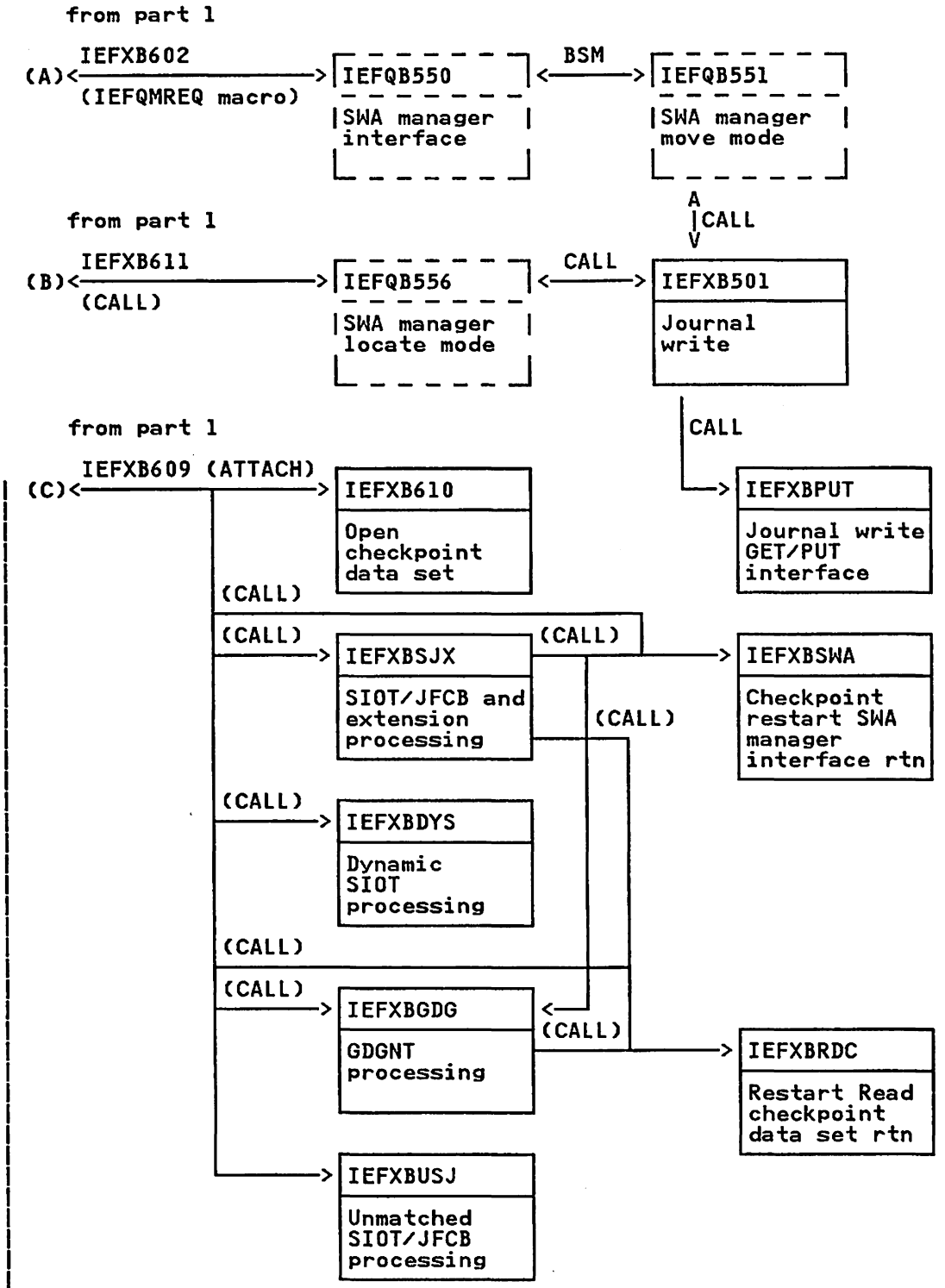


Figure 2 (Part 2 of 2). Scheduler Restart Process Flow

METHOD OF OPERATION

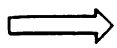
This section uses diagrams and text to describe the functions performed during scheduler restart processing. There are two types of diagram formats: HIPO and prologue.

HIPO diagrams are arranged in an input-processing-output format. The left side of the diagram contains data that serves as input to the processing steps in the center of the diagram, and the right side contains the data that is output from the processing steps. Each processing step is numbered; the number corresponds to an amplified explanation of the step in the "Extended Description" box. The object module name and labels in the extended description point to the code that performs the function.

The following figure shows the symbols used in HIPO diagrams. The relative size and the order of fields in control block illustrations do not always represent the actual size and format of the control block.

Primary processing -- indicates major functional flow.

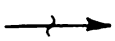
Secondary processing -- indicates functional flow within a diagram.



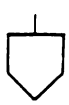
Data movement, modification, or use.

Data reference -- indicates the testing or reading of a data area to determine the course of subsequent processing.

Pointer -- indicates that a data area contains the address of another data area.



Indirect pointer -- indicates intermediate pointers have been omitted.



Connector -- indicates that a diagram is continued on the next page.

Figure 3. Key to HIPO Diagrams

Note: Brief scheduler restart module descriptions appear in MVS/Extended Architecture System Logic Library: Module Descriptions, which contains module descriptions for all the MVS/Extended Architecture components described in the System Logic Library.

The prologue format diagrams contain detailed information that is broken down into four different headings. The four headings and the topics they document are:

Module Description, which includes:

- Descriptive name
- Function (of the entire module)
- Entry point names, which includes:
  - Purpose (of the entry point)
  - Linkage
  - Callers
  - Input
  - Output
  - Exit normal
  - Exit error, if any
- External references, which includes:
  - Routines
  - Data areas, if any
  - Control blocks
- Tables
- Serialization

Module Operation, which includes:

- Operation, which explains how the module performs its function.

- Recovery operation, which explains how the module performs any recovery.

Diagnostic aids, which provide information useful for debugging program problems; this includes:

- Entry point names
- Messages
- Abend codes
- Wait state codes
- Return codes for each entry point. Within each entry point, return codes might be further categorized by exit-normal and exit-error.
- Entry register contents for each entry point
- Exit register contents for each entry point

Logic Diagram, which illustrates the processing of the module, the input it uses, the output it produces, and the flow of control.

Note: All modules in the prolog format are grouped in alphabetic order following the HIPOs. The following figure illustrates the graphic symbols and format used in the prolog format diagrams.

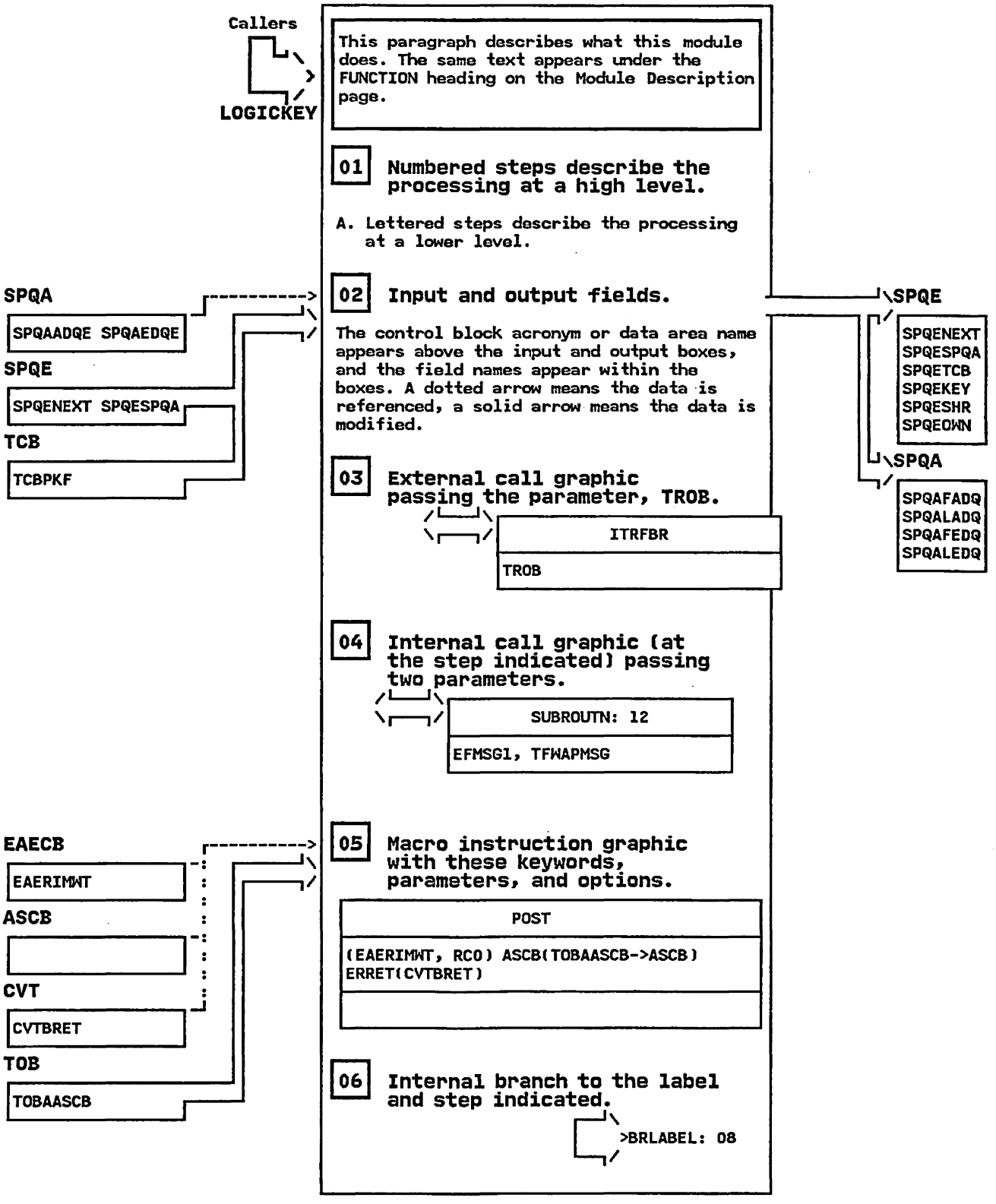


Figure 4. Key to the Logic Diagrams (Part 1 of 2)

LOGICKEY - Key to the Logic Diagrams **STEP 07**

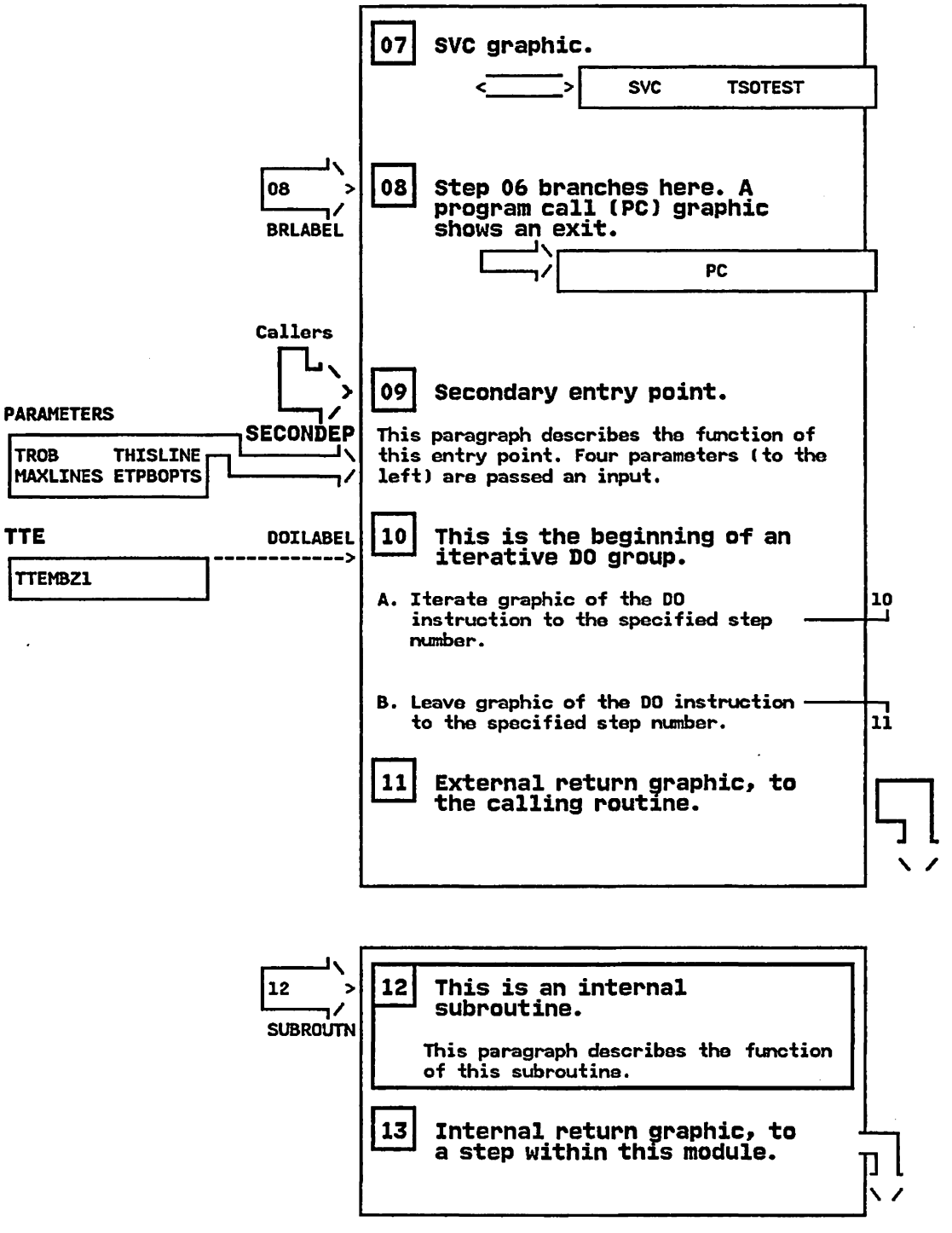
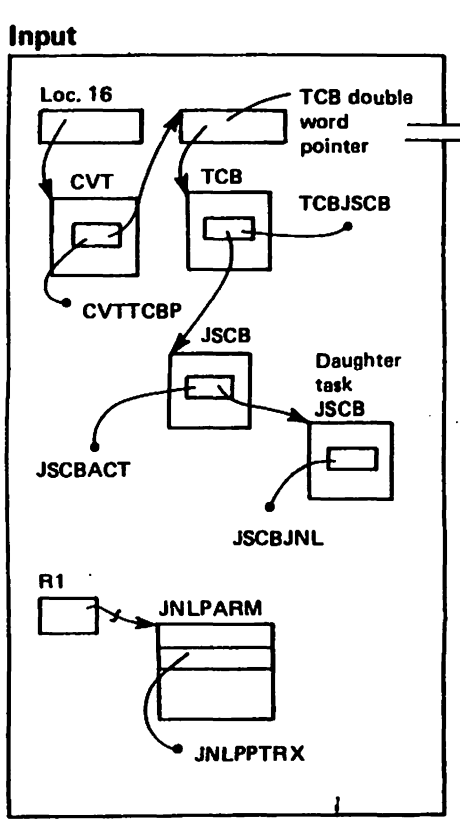


Figure 4. Key to the Logic Diagrams (Part 2 of 2)

This page intentionally left blank.

Writing Blocks to the Job Journal (IEFXB501) (Part 1 of 4)

- Initiator (IEFSD162),
- SWA manager (IEFQB551 or IEFQB556),
- Step header create (IEFXB604),
- Direct write (IDDWIJRN) (via IEFXB500 interface routine),
- Scheduler JCL facility



Process

1 Determine if the job journal can be written (updated). Unable to write. → Caller

Writing possible.

2 Determine which routine called for the journal write routine.

A consideration of the calling routine determines which step will be executed. The following list associates a symbol with each calling routine.

The symbols, in turn, are associated with the processing steps that are required to respond to the indicated calling routine.

Calling Routine	Symbol
SWA Write Routine (IEFQB551)	A
SWA Write/Locate Routine (IEFQB556)	B
Initiator (Allocation) (IEFSD162)	D
Direct Write (IDDWIJRN)	C
SJF Update SWB (IEFSJUPD)	E
SJF Delete SWB (IEFSJDEL)	E

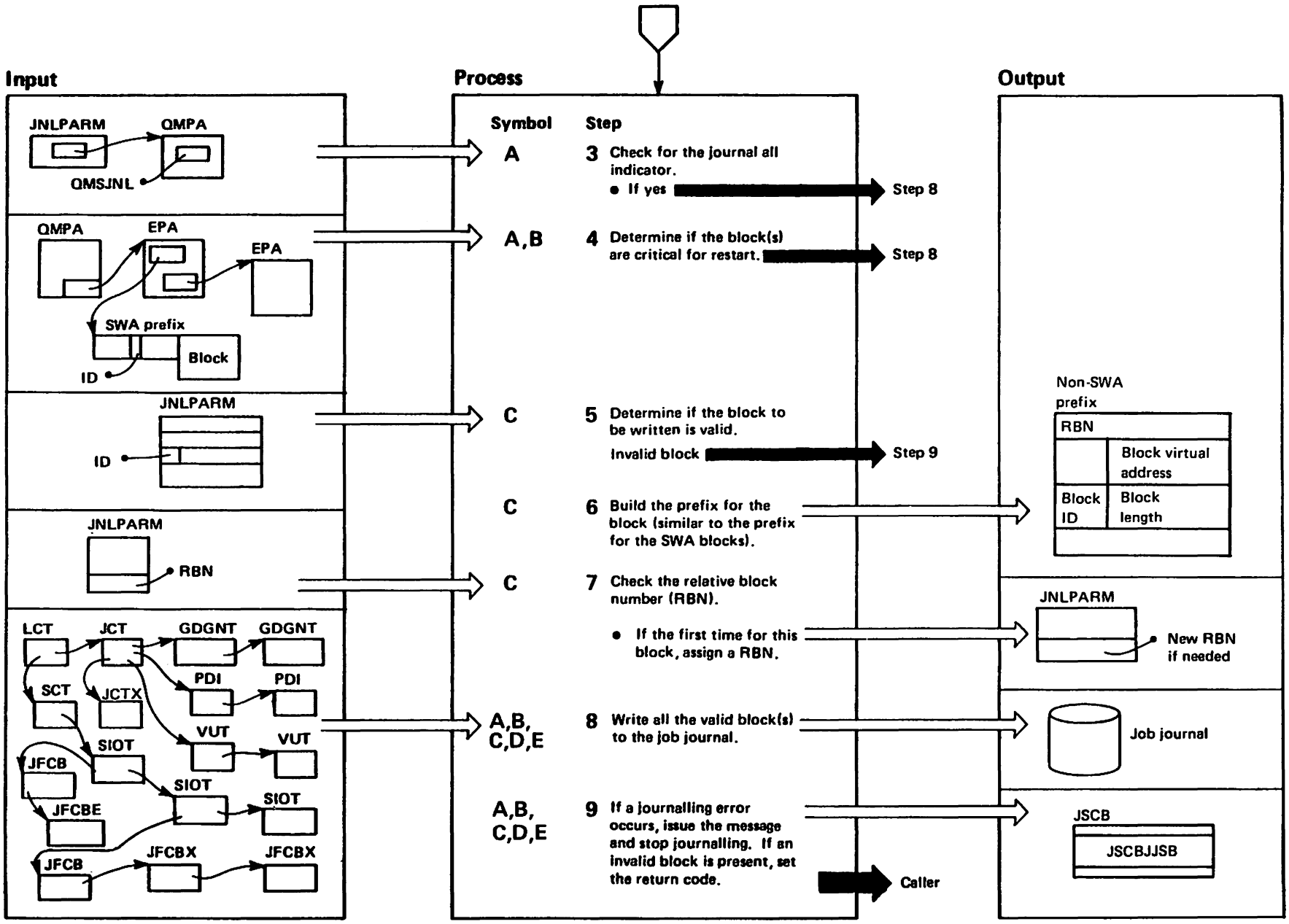
Writing Blocks to the Job Journal (IEFXB501) (Part 2 of 4)

Extended Description	Module	Label
This routine writes (updates) critical control blocks to the job journal for restart or termination preparation processing.		
1 Flags (JSCBJNLN, JSCBJNLF, and JSCBJNLE) in the JSCB field JSCBJNL indicate if a journal override condition exists, if a job journal exists, or if an error exists on the journal (from a previous 'write' situation).	IEFXB501	IEFXB501
2 The first field, JNLPCALL, of the parameter list at JNLPARM contains the indicator tested at this point.		

The second word of the journal parameter list contains the value indicated below:

Calling Routine	*Value*
SWA Write	QMPA address
SWA Write/Locate	External Parameter Area (EPA) chain address
Initiator Direct Write (of non-SWA blocks)	Linkage Control Table (LCT) address Block address (In this case, the third word contains the ID and length of the block.)
SJF Update and SJF Delete	Pointer to SWB chain address or block address

Writing Blocks to the Job Journal (IEFXB501) (Part 3 of 4)



Extended Description	Module	Label
3 *SWA Write (and Move):*		
If this indicator (QMSJNL) is on, the routine journals the control block regardless of the job state (see step 4). Only the terminator modules use this indicator.	IEFXB501	SWAWRT
4 The journal-write module contains a list of all critical control blocks for the four main job states: • Interpreter state • Allocation state • Problem program execution state • Termination state This list (or template) also indicates critical non-SWA (direct write) control blocks (e.g., step header record, VIO data sets such as data set page control table header and VIO data set control block).		SWAWRT SWAWRTL
The ID of each block that is scheduled to be journalled is found in the SWA prefix for the block. The ID is matched with the list in the template for the particular job state involved, to determine if the block is critical for restart.		
The format of the EPA appears below:		

For SWA locate:

↑ Block to be written or read		4
↑ SWA virtual address 3	Block ID	1
Length of block written or read		4
0 or ↑ next EPA		4

For SWA move:

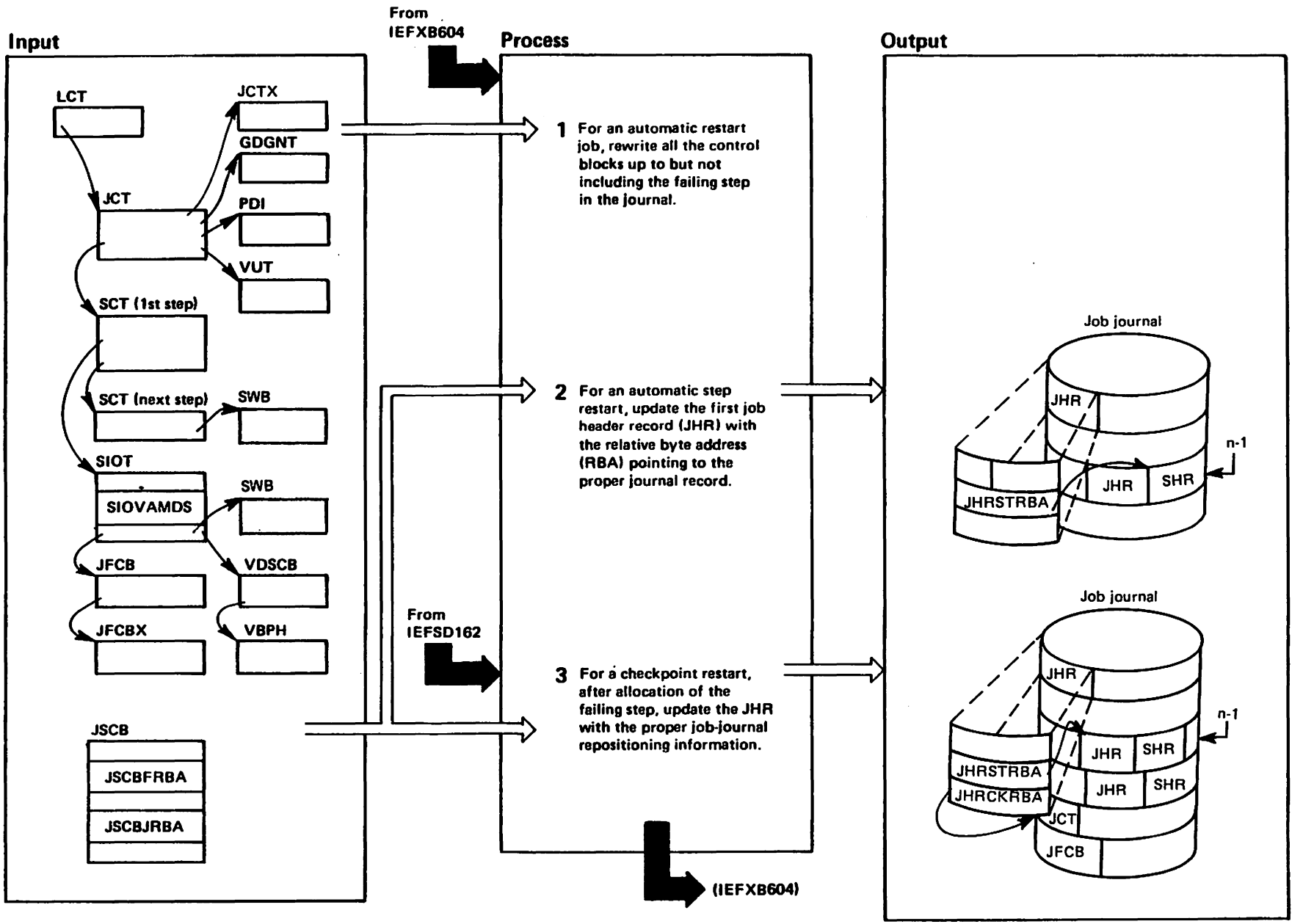
↑ Buffer to which block is read or from which block is written		4
↑ SWA virtual address from which block is read or to which block is written 3	Block ID	1

For SWA assign:

SWA virtual address (from SWA manager) 3	0	1
0		4

Extended Description	Module	Label
5 *Direct Write:*		
The parameter list contains the block ID. This ID is matched against the template as in step 4.		DRTWRT
6 The routine builds the prefix before it journals the block.		
7 For the first journalling of the block, the RBN (in the fourth word of the parameter list) is zero. This routine assigns a unique RBN that will be used if subsequent journalling of the block is required.		
8 Except when journalling control blocks at allocation-time, the order in which the blocks are journalled depends on the order in which they are updated on SWA. At allocation-time, the blocks are journalled in the following order: JCT, JCTX, SCT (of current step), one or more GDGNT (generation data group name table), one or more PDI (passed data set information), one or more VUT (volume unload table), SWBs off the SCT (via IEFSJGET), first SIOT plus SWBs off the SIOT (via IEFSJGET) plus JFCB plug JFCBX (one or more) or JFCBE, additional SIOT-JFCB-JFCBXs chains and SIOT-JFCB-JFCBE chains.	IEFXB501 IEFSJGET	RUNCHAIN
For the allocation-time journalling, the pointer chain beginning at the LCT gives block addresses.		
When writing to or getting information from the journal, the routine uses the request parameter list (RPL) that was built by the SWA create routine (IEFIB600). It passes the address of the RPL as a parameter and a specific number in register 0 (to indicate whether a get, put, or forced put should be issued) to the journal writer GET/PUT interface routine (IEFXBPUT). IEFXBPUT interrogates register 0 and, depending on its contents, issues the GET, PUT, or ENDREQ macro which results in a forced PUT. This allows JES to get information from the job journal or to put a control block on the journal.		JOURNAL
9 Error indicator in the JSCB is set, a WTP macro instruction is used to issue a message, a return code is placed in the parameter list JNLPARM, and control returns to the caller.		ERRMSG

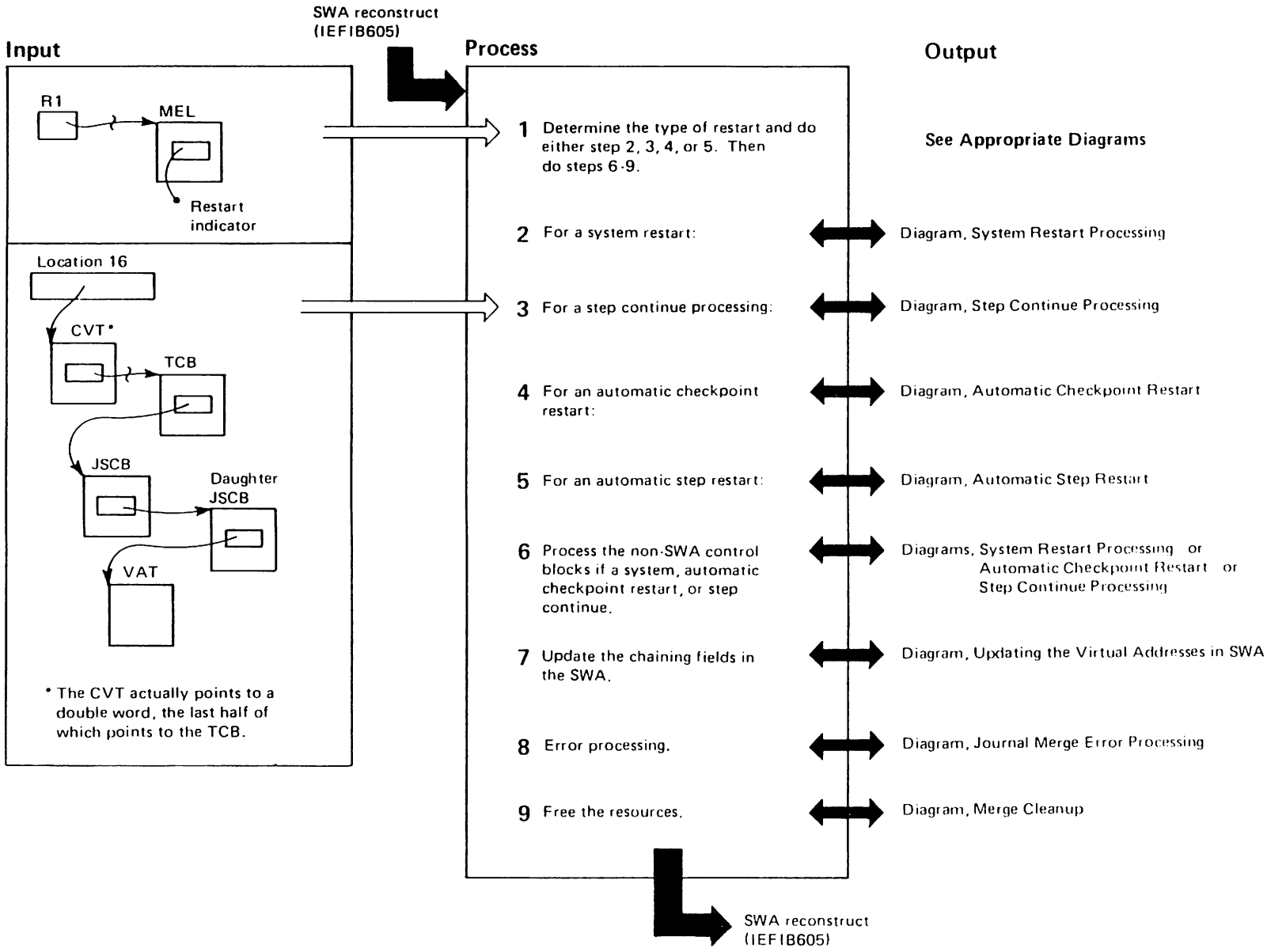
Journal for Restarted Jobs (IEFXB501) (Part 1 of 2)



Journal for Restarted Jobs (IEFXB501) (Part 2 of 2)

Extended Description	Module	Label
1 For an automatic step restart or a checkpoint restart, write all critical control blocks from step one up to but not including the failing step in the journal. Blocks are written as if they were all part of step n-1, where n is the failing step's number. Critical control blocks are: JCT, JCTX, PDI, GDGNT, VUT, SCT, SIOT, JFCB, JFCBX, and SWB, JFCBE. VIO blocks are also written if SIOVAMDS is on.	IEFXB501	RUNCHAIN RUNSIOT RUNSWB
2 For an automatic step restart, a GET macro with update is issued (via IEFXBPUT), using the relative byte address (RBA) saved in JSCBFRBA by the merge routine. The job header record (JHR) is then updated by inserting an RBA which was saved in JSCBJRBA by IEFXB604. This RBA points to the SHR for step n-1 and will be used by the merge routine after restart.		JHRUPDT
3 For a checkpoint restart, after allocation of the failing step, update the JHR by inserting JSCBJRBA and the RBA returned from the last PUT macro. This RBA will be used at restart-time to reposition the journal data set.		JHRUPDT

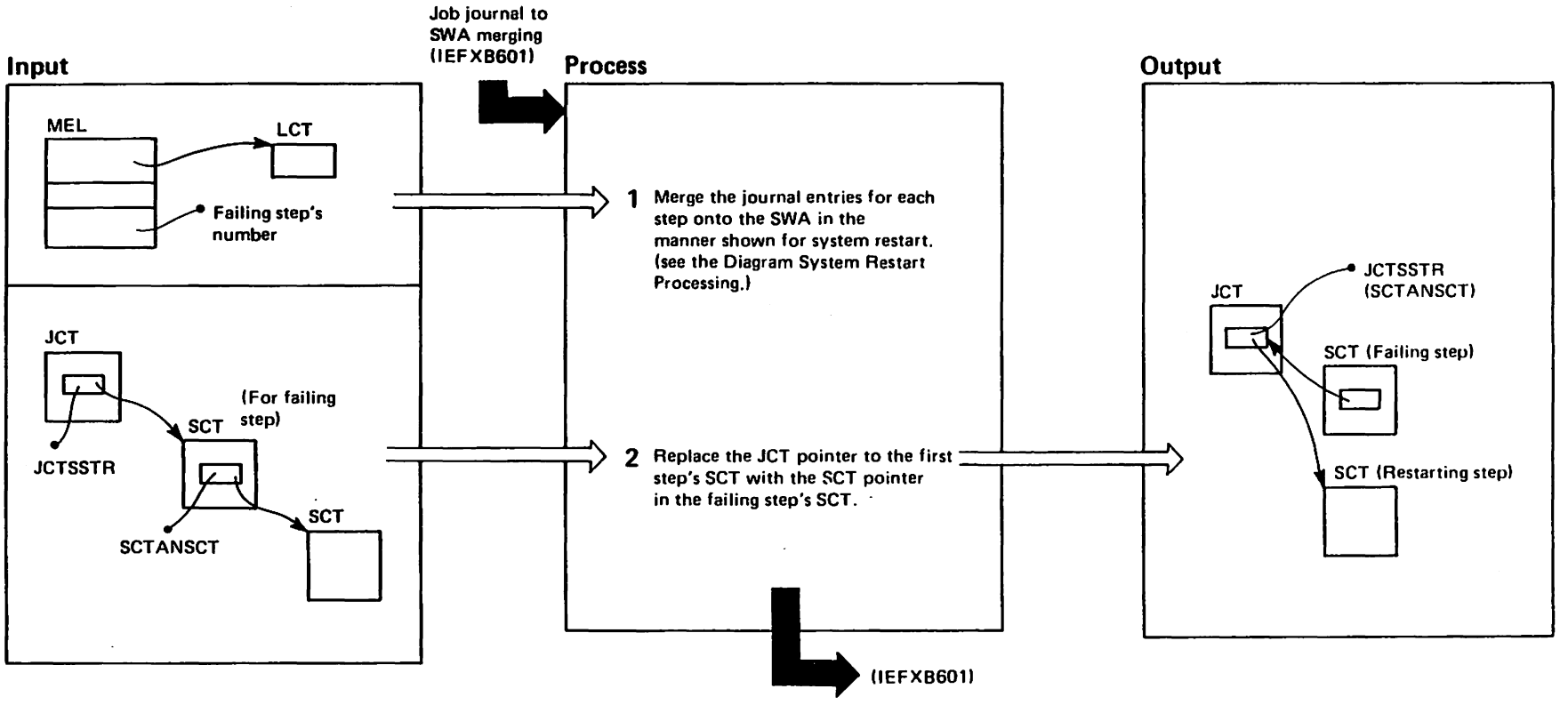
Job Journal to SWA Merging (IEFXB601) (Part 1 of 2)



Job Journal to SWA Merging (IEFXB601) (Part 2 of 2)

Extended Description	Module	Label
This routine reconstructs the SWA (from the job journal) so it has the control blocks in effect at the time indicated: • For automatic checkpoint restart: Control blocks at time checkpoint was taken. • For automatic step restart: Control blocks at beginning of the failing step. • For system failure: Control blocks at the point of failure. This diagram refers to several other diagrams covering the checkpoint/restart functions. Each of the latter diagrams represents a subroutine (within module IEFXB601) that has a given function to perform. This present diagram contains general module entry-information that is also applicable to these subsequent diagrams. (See also, the introduction to this section.)		
1 In the 6th byte of the merge entrance list (MEL) contains the restart indicator as follows: X'08' = system restart X'20' = step continue X'40' = automatic checkpoint restart X'80' = automatic step restart The MEL also contains the address of the LCT (in the first word) and the failing step's number (in the last two bytes).	IEFXB601	IEFXB601
2 For this case, a full merge of all control blocks for all steps is performed.		SYSMERGE
3, 4, 5 For each case, a full merge of the control blocks for the non-failing steps is performed, and selective merging of fields in critical control blocks for the failing step is performed.		SYSMERGE CKPTMRGE STEPMRGE
6 For each non-SWA control block on the job journal, an appropriate exit routine performs the required processing.		VATPUT VAMPROC
7 The routine updates the SWA control block chaining fields to reflect the new virtual addresses resulting from the SWA reconstruction.	IEFXB601	ADDRUPDT
8 The Routine sets a return code of X'24' in register 15 and sends an appropriate message to the programmer and/or the operator.		ERRPROC
9 The routine releases the virtual address table and any extensions to it. **Note:** There is one entry in the virtual address table (VAT) for each control block that the interpreter writes to the SWA. This entry points to the 16-byte prefix to the control block. When dynamic allocation routines cause the SWA's control block structure (that is, the relative control block addresses) to change during restart, the VAT updating routines insert the new control block addresses (of other journalled control blocks) into the appropriate fields of the control blocks in the SWA.		

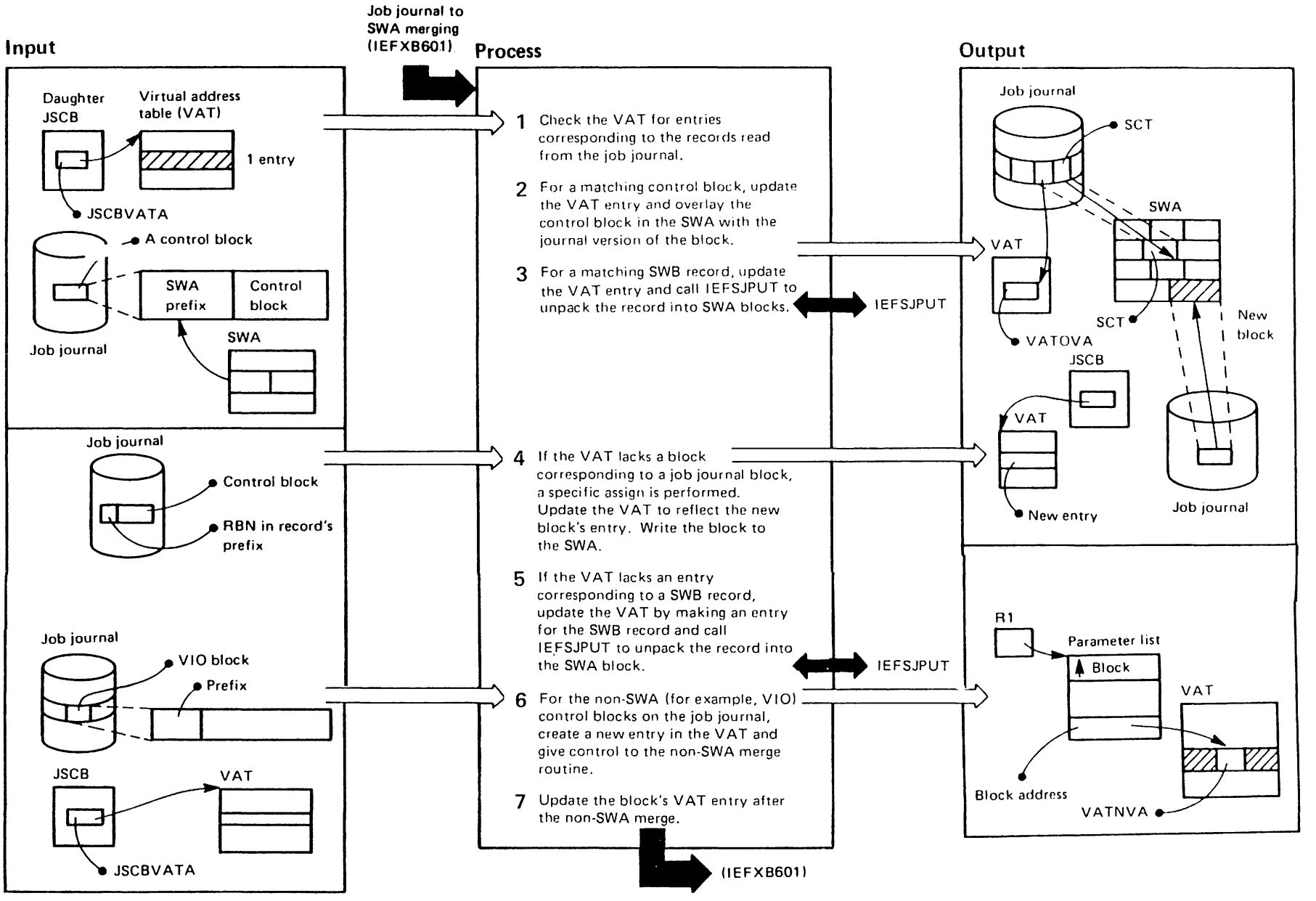
Step Continue Processing (IEFXB601) (Part 1 of 2)



Step Continue Processing (IEFXB601) (Part 2 of 2)

Extended Description	Module	Label
This routine handles the processing that allows a user's job to continue at the next step.		
1 This processing occurs when a step was being terminated at the time a system failure occurred. Since the job journal entries are complete, they are processed in the same manner as for system restarts.	IEFXB601	SYSMERGE
2 By resetting the JCT pointer (to the SCT), the restart will occur at the job step following the failing step.		CLEANUP

System Restart Processing (IEFXB601) (Part 1 of 2)

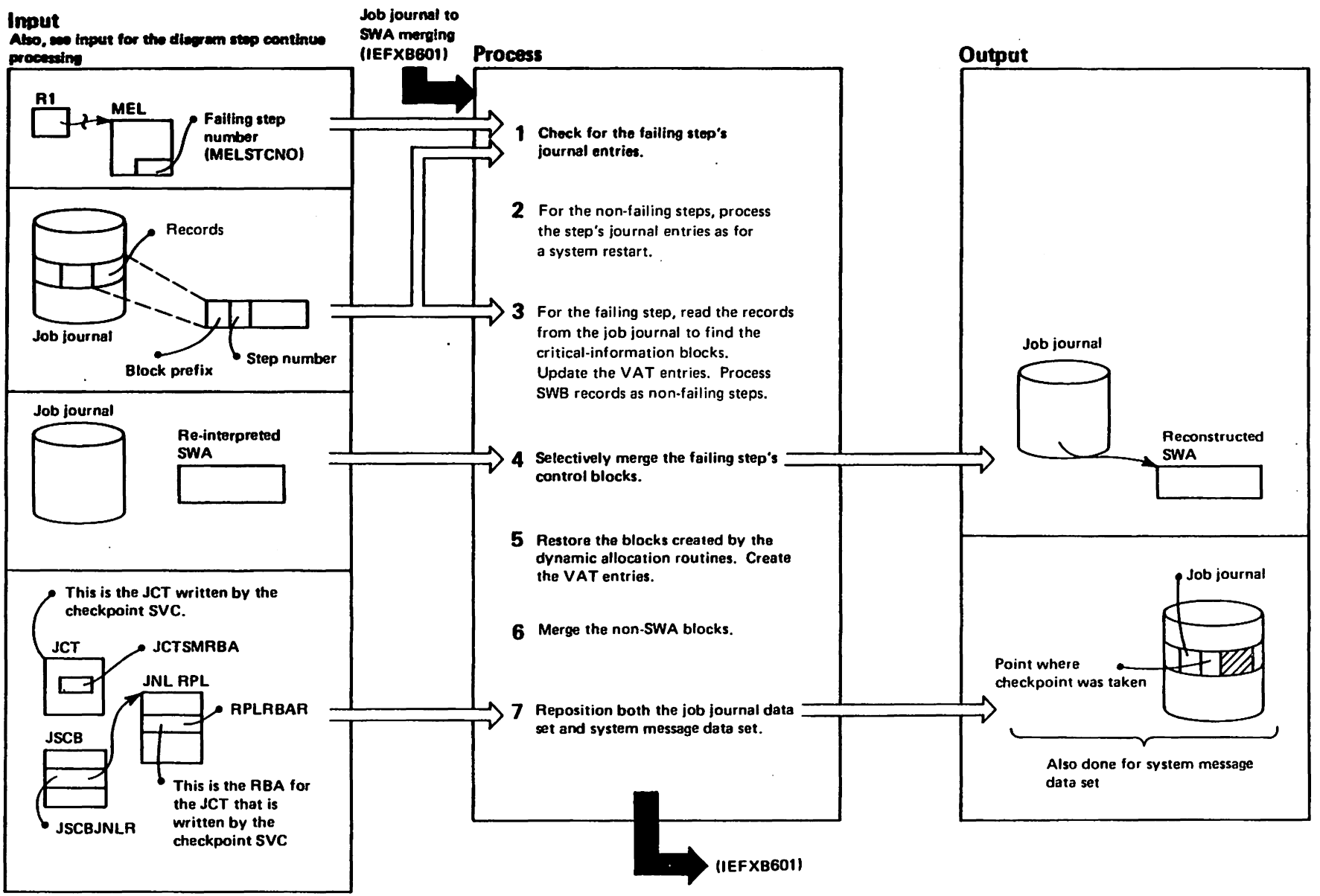


System Restart Processing (IEFXB601) (Part 2 of 2)

Extended Description	Module	Label
For a system restart, all control blocks for all steps in a job will be fully merged from the job journal to the SWA. (See also the diagrams Merge Cleanup and Updating the Virtual Addresses in SWA.)		
1 The prefix in the journal record contains record identifications. The VAT contains representations of all blocks in the SWA. The relative block number (RBN) and block ID field in the SWA prefix (for the record on the journal) are matched against entries in the VAT.	IEFXB601	VATPUT
2 If the RBN and ID fields of the prefix for a control block match those in the VAT, the old virtual address is placed in the VAT entry and the job journal form of the control block overlays the corresponding form in the SWA. (In the 'output' part of this diagram, the SCT is used as an example.)		VATPUT FLDMERGE
3 If the RBN and ID fields of the prefix for a SWB record match those in the VAT, the routine places the old virtual address in the VAT entry and IEFSJPUT unpacks the journal SWB record into SWA.	IEFSJPUT	IEFSJPUT
4 The SWA manager assign routine uses the assign function in the IEFQMREQ macro instruction to get storage for control blocks initially created by allocation routines and JFCB housekeeping routines. The assign routine uses the RBN in the block prefix. An entry for the new block is made in the VAT, and the corresponding block is written to the SWA.		ASGNRITE
5 For an unmatched SWB record, IEFSJPUT unpacks the record into SWA. The routine makes an entry for the new SWB in the VAT.	IEFSJPUT	IEFSJPUT
6 Based on the control block's ID (in the block prefix) the journal merge routine creates a new entry in the VAT and fills in the RBN, control block ID, and the old virtual address. The merge routine then calls a subroutine (IDDWIMRG or IDAVBPJ2) to merge the block to the SWA. The subroutine uses an interface parameter list to obtain the merge information.	IEFXB601	VATPUT VAMPROC

Extended Description	Module	Label
The parameter list used for this appears as follows: ↑ Block being merged (4) ID of block being merged* (1) | Length of block being merged (3) Relative block number (4) New virtual address** (4) ↑ GETMAIN storage area*** (4)		
*The block ID field has the following meanings: X'FE' — Data set page control table (DSPCT) header — VIO Routine Performing the Merge: Virtual Block Processor (VBP) X'FC' — Virtual data set control block (VDSCB) — VIO Routine Performing the Merge: Window Intercept (WI)		
**The new virtual address is that passed to the appropriate VIO merge routine for all except the first occurrence of the control block on the job journal.		
***The GETMAIN area address is passed back to the journal merge routine by a VIO merge routine when the VIO merge routine issues a GETMAIN macro instruction for the block to be merged.		
All merges subsequent to the first one (for this non-SWA block) use this information.		
The control block's ID given in the block prefix is compared against an internal table of block IDs in the SWA to determine if the block (on the journal) is also in SWA.		
The journal version of the block overlays the block as it resides in storage. After the block has been updated, the pointer fields in the block and the block's address (as given in the VAT) are updated.		
7 The information returned from the non-SWA merge routine indicates the location of the merged control block. This address is placed in the block's VAT entry.		VAMPROC VATPUT

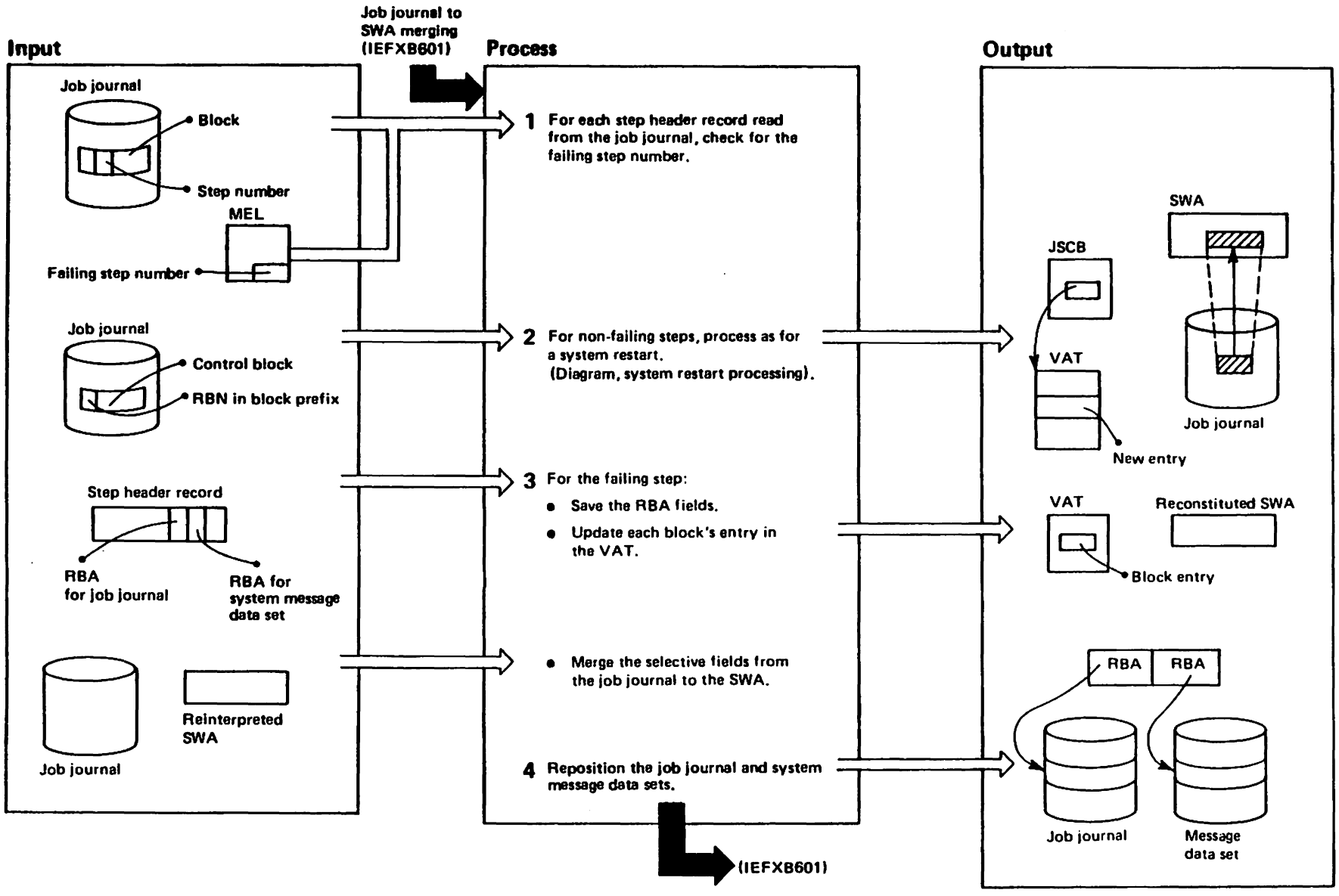
Automatic Checkpoint Restart (IEFXB601) (Part 1 of 2)



Automatic Checkpoint Restart (IEFXB601) (Part 2 of 2)

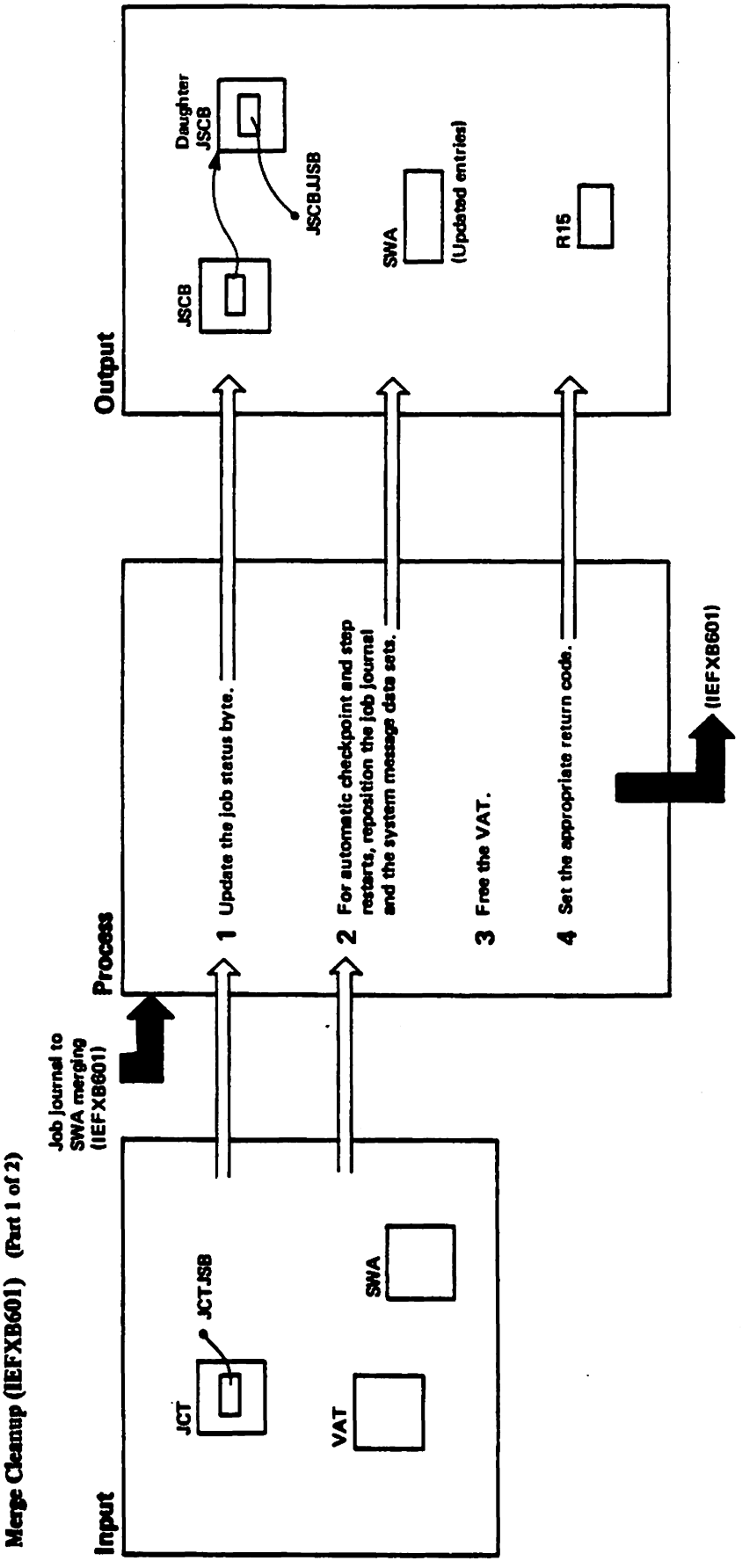
Extended Description	Module	Label
This routine merges control blocks from the job journal to the SWA for failed jobs that are eligible for an automatic checkpoint restart (checked via indicator in MEL). See also diagram, merge cleanup and updating the virtual addresses in SWA.		
1 Compare the step number in the step header record with the step number in the MEL. For a step header record, the SWPID field of the block prefix is X'C0'.	IEFXB601	
2 See the Diagram, system restart processing for processing of steps prior to the failing step.		
3 Examples of critical-information fields are the checkpoint and job status information fields of the JCT, the volume and label information fields of the JFCB, and the chain pointer fields of the SCT, SIOT, and JFCB. In addition, the routine updates the old virtual address field of the block's entry in the VAT.		CKPTMRGE VATPUT
4 The fields containing the critical information are merged from the job journal to the SWA.		FLDMERGE
5 See the diagram, system restart processing for assign details. The blocks include the SIOTs, JFCBs, JFCBEs, and JFCBXs created by dynamic allocation and JFCB housekeeping routines. The SWA manager assign and write routines specifically assigns these blocks and writes them to the SWA. The newly-created VAT entries for these blocks contain the RBN, ID, old virtual address, and new virtual address.		ASGNRITE VATPUT
6 See the diagram, system restart processing, step 3.		
7 From the job control table written by the checkpoint SVC routine, save the RBA (relative block address) field for the system message data set. From the journal RPL (JNL RPL), save the RBA of the JCT written by the checkpoint SVC.		CKPTMRGE

Automatic Step Restart (IEFXB601) (Part 1 of 2)



Automatic Step Restart (IEFXB601) (Part 2 of 2)

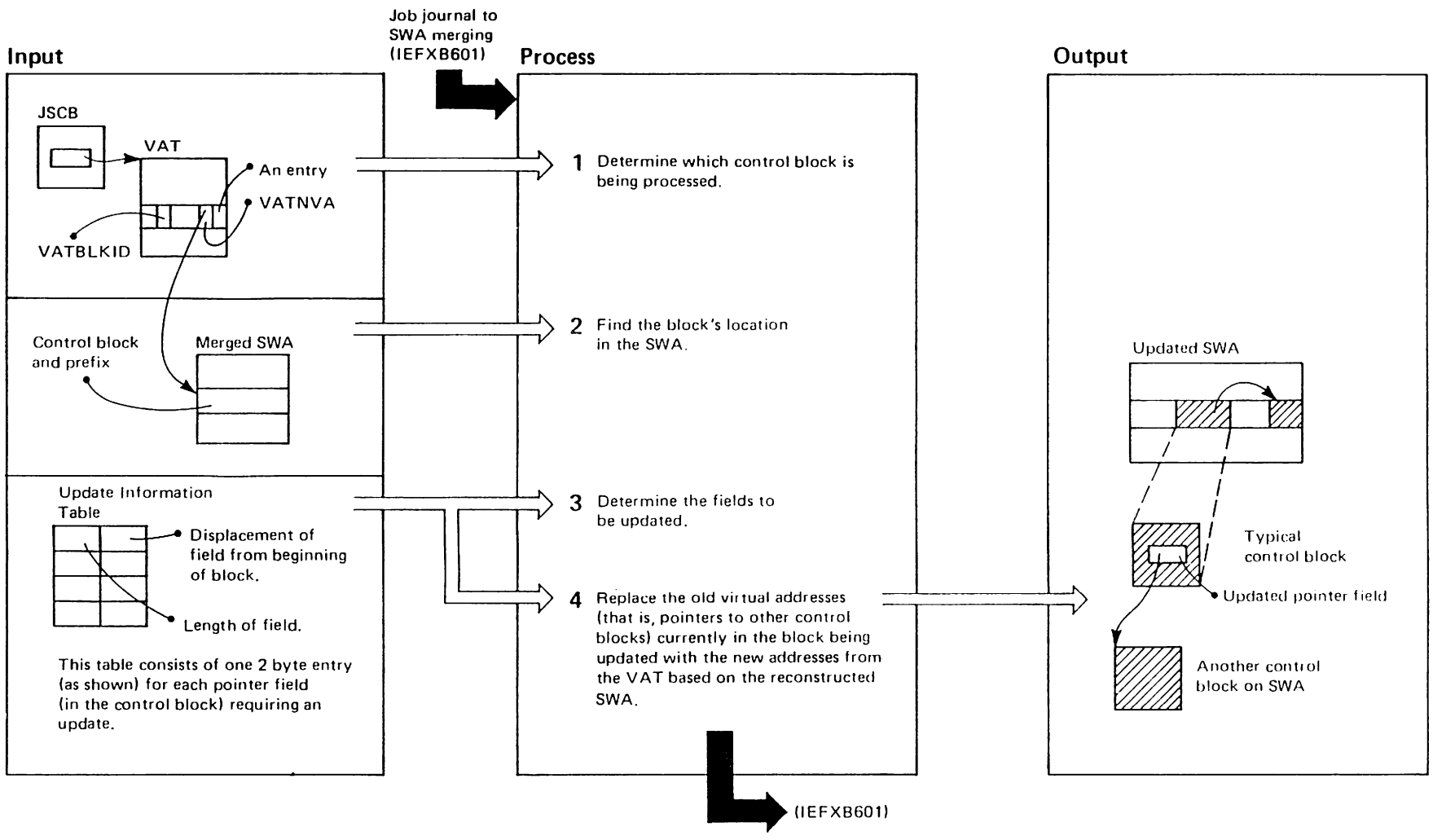
Extended Description	Module	Label
This routine merges control blocks from the job journal to the SWA for failed jobs that are eligible for an automatic step restart (checked via indicator in MEL). See also diagrams, merge cleanup and updating the virutal addresses in SWA.		
1 Check the step number field in the step header record and compare it against the failing step number given in the merge entrance list (MEL). (See diagram, automatic checkpoint restart.)	IEFXB601	STEPMRGE
2 See diagram, system restart processing.		
Note: If the step numbers do not match, the step is non-failing.		
3 The RBA fields saved are for the job journal and the system message data set. The fields are located in the step header records.		
For each critical control block associated with the step, the routine updates the old virtual address field in the VAT.		VATPUT
For example, selective merging involves the following fields in the indicated blocks:		FLDMERGE
JCT: job status information and restart switches.		
JFCB and JFCBX: volume information.		
JFCB: MOD data set information for TTR and track balance considerations.		
JFCBE: 3800 printer parameters.		
4 Pointers are established using the RBAs saved from the step header record. The pointers show the step's entry in each data set.		



Merge Cleanup (IEFXB601) (Part 2 of 2)

Extended Description	Module	Label
This routine does the clean-up functions for automatic checkpoint or step restart or for step continue processing.		
1 The latest version of this field information comes from the job journal (the JCT block). The JCTJSB information overlays that in the JSCBJJSB.	IEFXB601	CLEANUP
2 The relative block addresses used for repositioning the data sets are obtained from the step header record for automatic step restart or from the JCT and request parameter list (RPL) for automatic checkpoint restart.		
3 The VAT and any extensions to it are released.		
4 An error return code of X'24' causes the job to be purged from the system. A normal return code of X'00' permits restart to continue.		

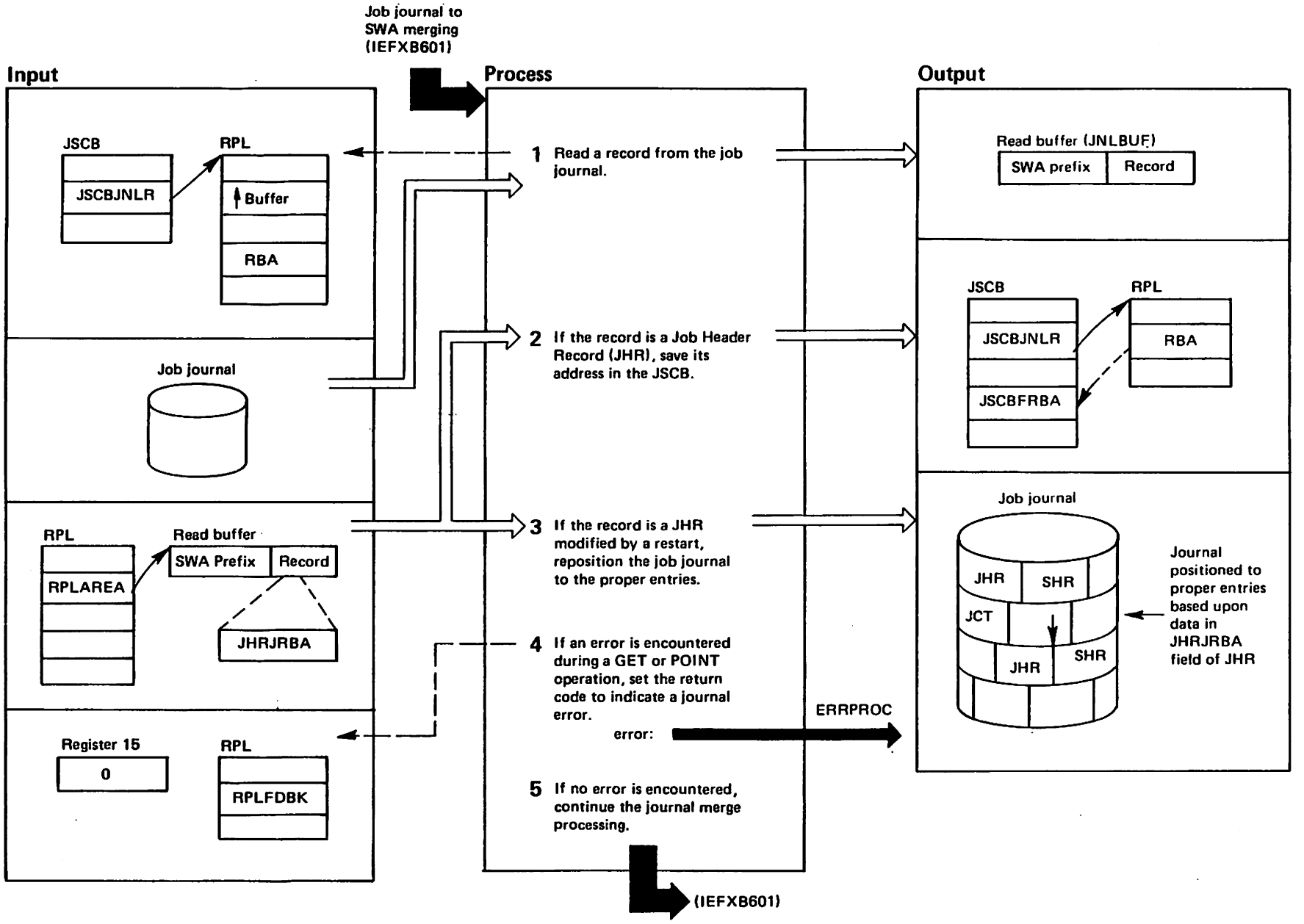
Updating the Virtual Addresses in SWA (IEFXB601) (Part 1 of 2)



Updating the Virtual Addresses in SWA (IEFXB601) (Part 2 of 2)

Extended Description	Module	Label
For each entry in the VAT, this routine updates the virtual address of all the block's fields that are changed.		
1 The block ID field in the VAT contains an indication of the control block being processed. The routine then processes consecutively all entries in the VAT.	IEFXB601	ADDRUPDT
2 The new virtual address field in the VAT entry for the block provides the new location in the merged SWA. The routine then reads the control block being processed. The SWPID field in the SWA prefix indicates the control block that is being updated.		
3 An internal table contains the necessary update information. This information includes the displacements and lengths of all fields that require updating. There is one table per control block being updated.		
4 For each address to be updated, the value in the new virtual address field (of the VAT entry for the changed control block field) replaces the existing old virtual address field in the control block.		UPDATE PTRUPDTE

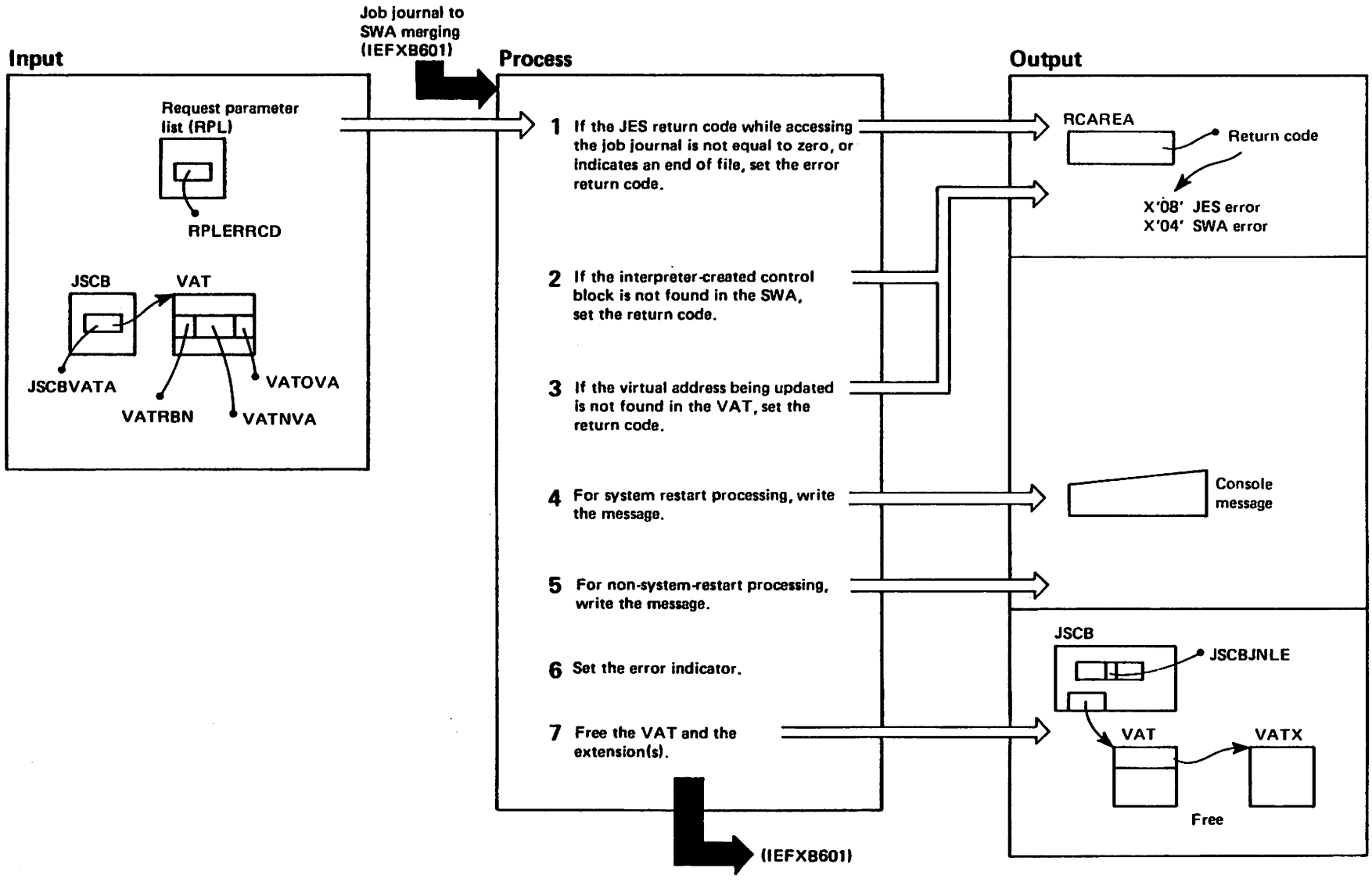
Journal Merge Reading (IEFXB601) (Part 1 of 2)



Journal Merge Reading (IEFXB601) (Part 2 of 2)

Extended Description	Module	Label
This routine is responsible for all reading from the job journal required for merge processing.		
1 A record is read from the job journal using the request parameter list (RPL) pointed to by the active JSCB (JSCBJNLR).	IEFXB601	READPROC
2 A job header record has a control block ID (X'C1') in the SWA prefix. Save the address, which was passed back in the RPL, (RPLRBAR) in the active JSCB (JSCBFRBA) for journal data set repositioning.		
3 A job header record written as a result of a restart contains job journal repositioning information in the field JHRJRBA. This value is used to issue the POINT macro to position the job journal to the proper entries.		
4 Any non-zero return code in register 15 (other than a logical error indicating end of file – R15=8, RPLERRCD=0004) is considered an error condition. An error return code is set and ERRPROC receives control. (Refer to Journal Merge Error Processing diagram). If normal return code, journal merge processing is continued.		

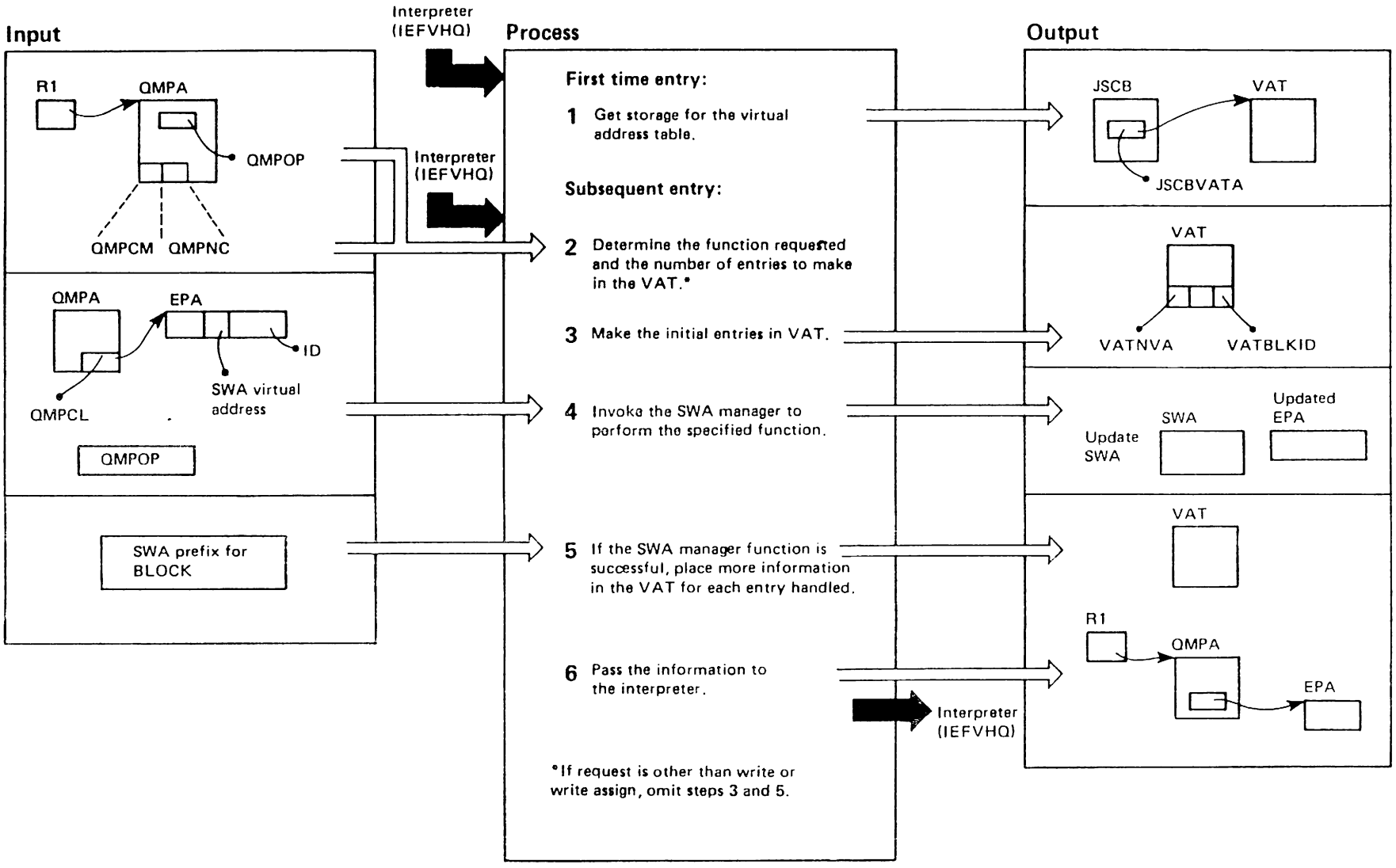
Journal Merge Error Processing (IEFXB601) (Part 1 of 2)



Journal Merge Error Processing (IEFXB601) (Part 2 of 2)

Extended Description	Module	Label
This processing handles errors that may be encountered during SWA reconstruction or in accessing the job journal. It issues an appropriate message and, for either automatic step or automatic checkpoint restart, it informs the operator that the job has been cancelled.		
1 The return code is set to X'08'.	IEFXB601	ERRPROC
2 The return code is set to X'04'.		
3 The return code is set to X'04'.		
4 This message is intended for the programmer and is written to the SYSOUT data set.		
5 The message is written to the programmer via the SYSOUT data set, and a message is written to the operator via the WTO macro instruction.		
6 The journal error bit in the JSCB is turned on.		
7 The routine releases the VAT resource, and returns a code of X'24' in register 15.		

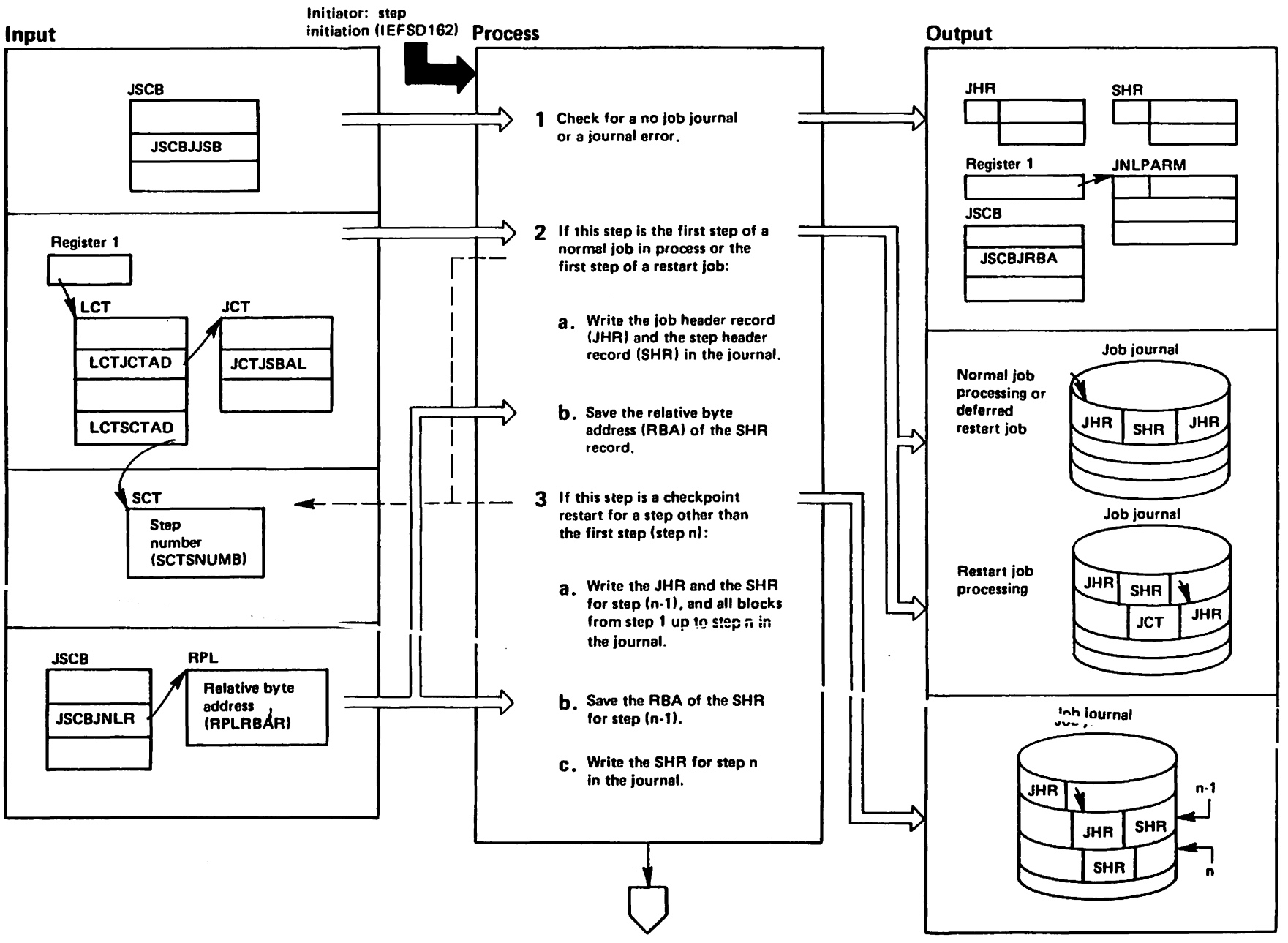
Move Mode Restart Interface Processing (IEFXB602) (Part 1 of 2)



Move Mode Restart Interface Processing (IEFXB602) (Part 2 of 2)

Extended Description	Module	Label
This routine builds a virtual address table (VAT) to be used by the journal merge routine during SWA reconstruction processing.		
1 The VAT is an 800-byte table. The JSCB pointer to the VAT is constructed.	IEFXB602	VATBUILD
2 If either a write or a write/assign function is requested, the routine determines the number of entries to be made in the VAT after the SWA manager performs its function.		
3 The routine uses the external parameter area (EPA) to get the SWA virtual address (used for the initial VATNVA field in the VAT) and the block ID if one exists.		
4 The routine uses the IEFQMREQ macro instruction to give control to module IEFQB551. The operation field, QMPOP, indicates whether the function is a write, a write/assign, an assign, a write all, a read all, or a read operation. The VAT updating occurs only for a write/assign, an assign operation, or a write all or write.		
5 The relative block number (RBN) is placed in the VAT for each entry, and the block ID field of the VAT is filled in if not already there.		
6 The routine returns control to the interpreter. The output to the interpreter is the same as the input from the interpreter but with additional information that was filled in by the SWA manager routine.		

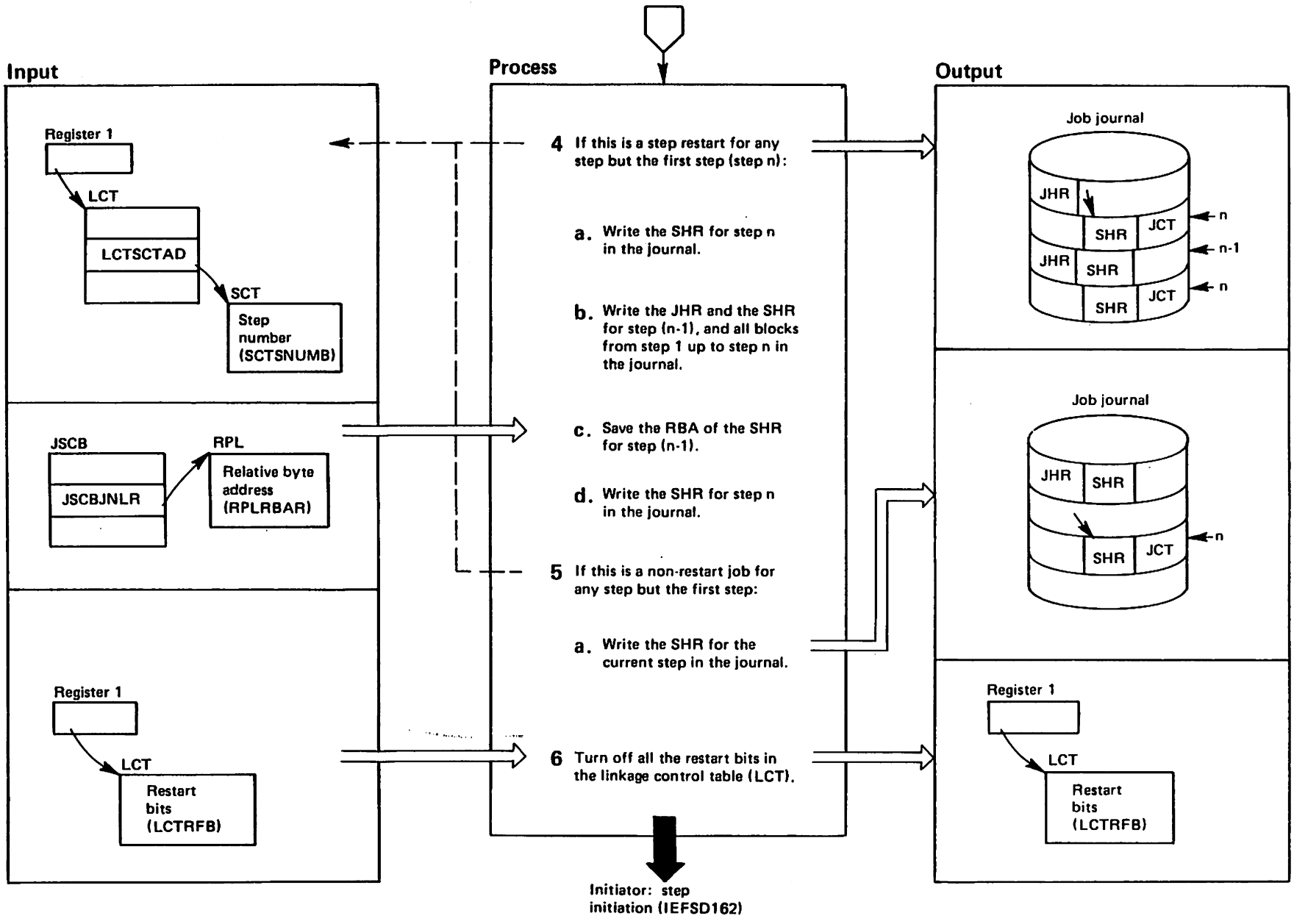
Building Step Header Record for Job Journal (IEFXB604) (Part 1 of 4)



Building Step Header Record for Job Journal (IEFXB604) (Part 2 of 4)

Extended Description	Module	Label
1 Check the JSCBJNLF and JSCBJNLE bits in the JSCBJJSB field to determine whether there is no job journal or there is a journal error. Change job state in the JSCB to in allocation.	IEFXB604	
2 If the failing step is the first step of the job or if it is the first step of a non-restart job, write the JHR and the SHR in the journal. Set JCTJSBAL to indicate that the job state is in allocation, and write the job control table (JCT) in the journal for all jobs except automatic checkpoint restart jobs, to record the in allocation status.		
3 If the failing step (step n) is any step but the first step of an automatic checkpoint job, write the JHR and the SHR for step n-1 in the journal, and all the control blocks of all previous steps up to but not including the failing step. This information must be saved to permit a possible subsequent restart. Finally, write the SHR for step n in the journal.		

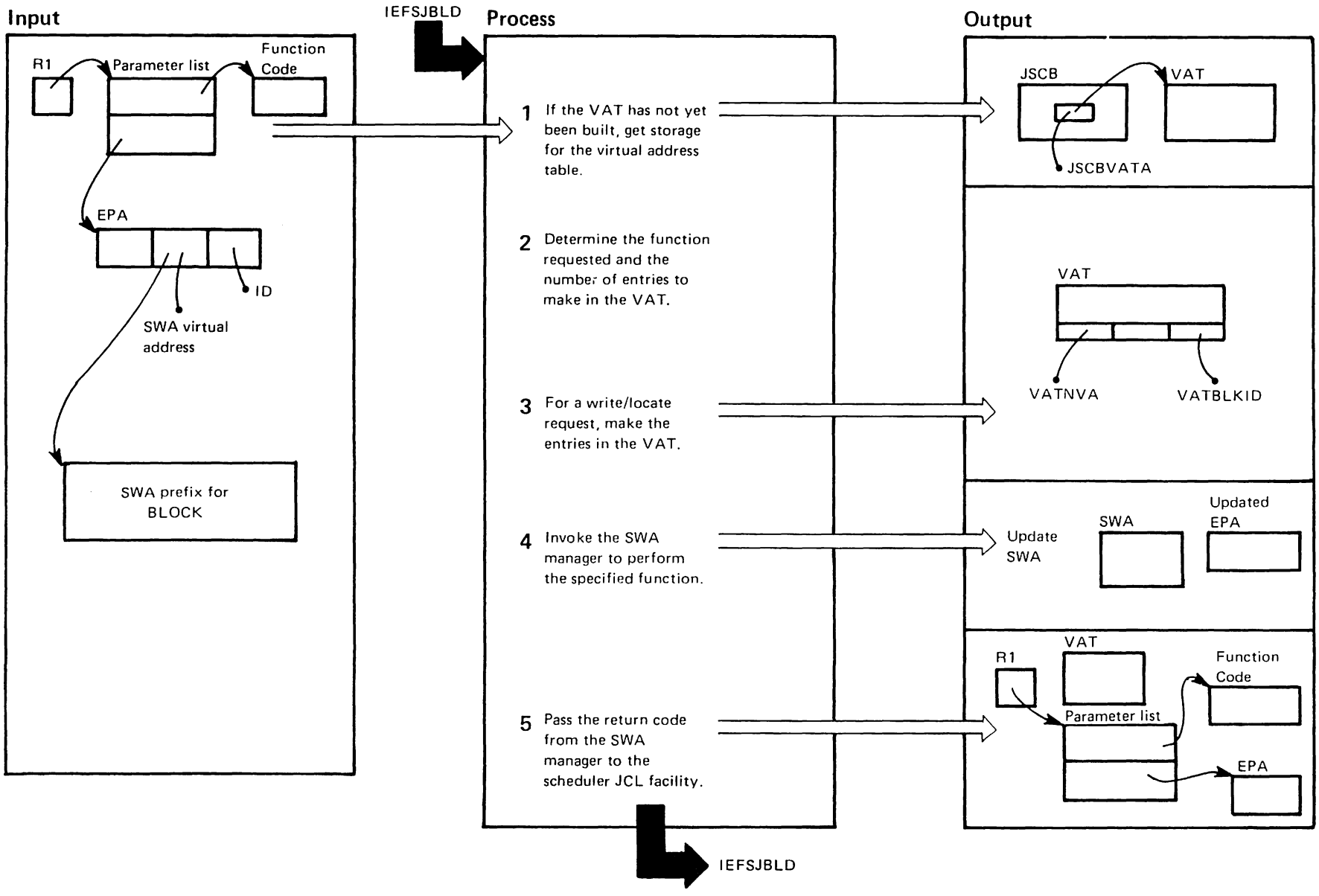
Building Step Header Record for Job Journal (IEFXB604) (Part 3 of 4)



Building Step Header Record for Job Journal (IEFXB604) (Part 4 of 4)

Extended Description	Module	Label
4 If the failing step is any step but the first step of a step restart job, write the SHR for step n, and then write the JHR and the SHR for step n-1 in the journal. Write all the control blocks for steps 1 thru n-1, and lastly write the SHR for step n in the journal once again. As in step 3, this information is saved to permit a possible restart.	IEFXB604	
5 If the failing step is any step but the first step of a non-restart job, write the SHR of the current step in the journal. Again, this information is saved to permit a possible restart.		
6 Turn off all restart bits (LCTRFBSM, LCTRFBCR, and LCTRFBDC) before exiting.		

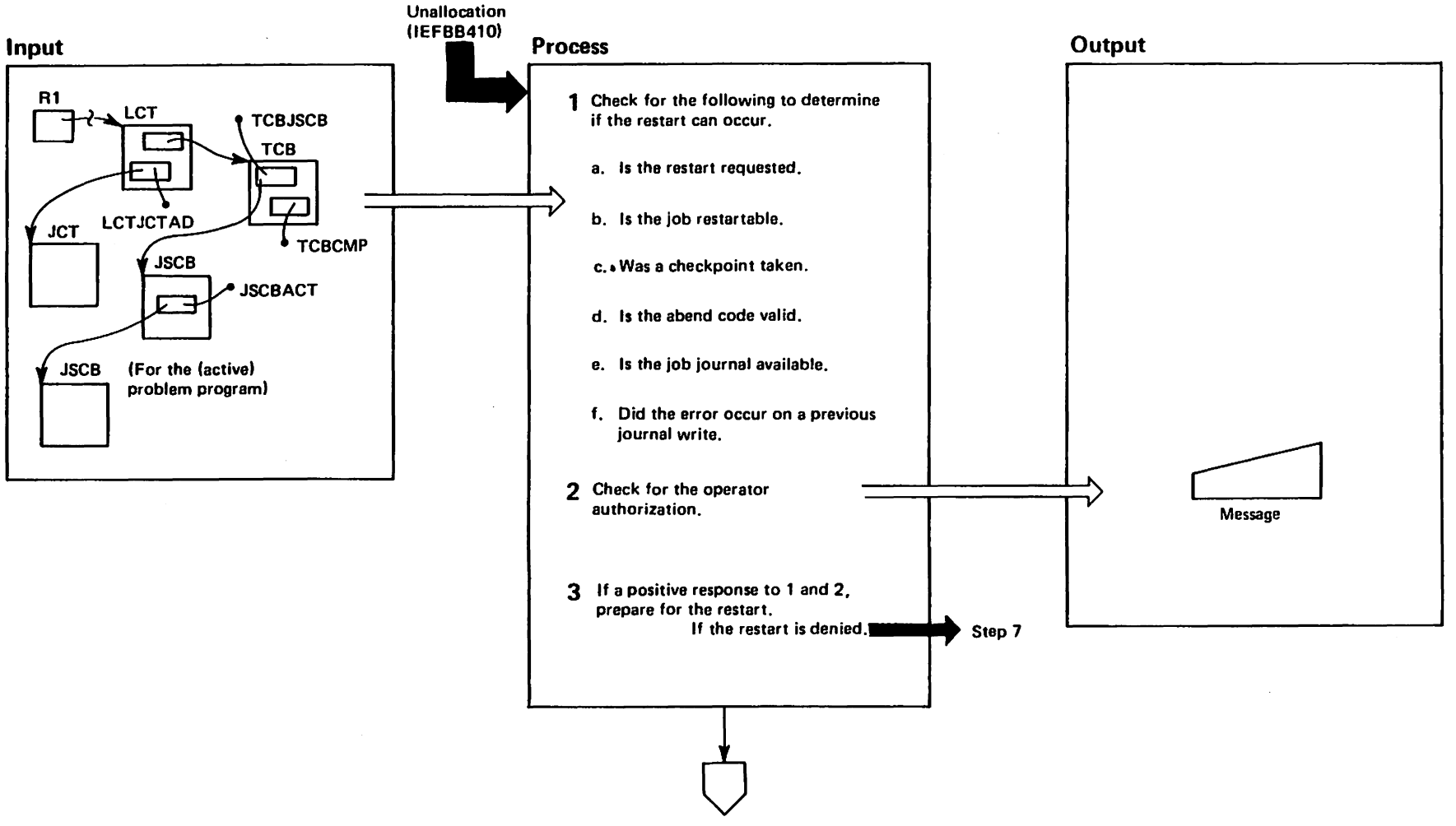
Locate Mode Restart Interface Processing (IEFXB611) (Part 1 of 2)



Locate Mode Restart Interface Processing (IEFXB611) (Part 2 of 2)

Extended Description	Module	Label
This routine builds a virtual address table (VAT) entry for each SWB when called by the scheduler JCL facility. This table is used by the journal merge routine during SWA reconstruction processing.		
1 The VAT is an 800-byte table. The JSCB pointer to the VAT is constructed.	IEFXB611	VATBUILD
2 If a write/locate function is requested, the routine determines the number of entries to be made in the VAT after the SWA manager performs its function. The relative block number (RBN) is placed in the VAT for each entry, and the block ID field of the VAT is filled in if not already there.		
3 The routine uses the external parameter area (EPA) to get the SWA virtual address (used for the initial VATNVA field in the VAT) and the block ID if one exists.		
4 The routine gives control to module IEFQB556, the SWA manager locate mode.		
5 The routine returns control to the scheduler JCL facility. The output to the scheduler JCL facility is the same as the input from the scheduler JCL facility but with additional information that was filled in by the SWA manager routine.		

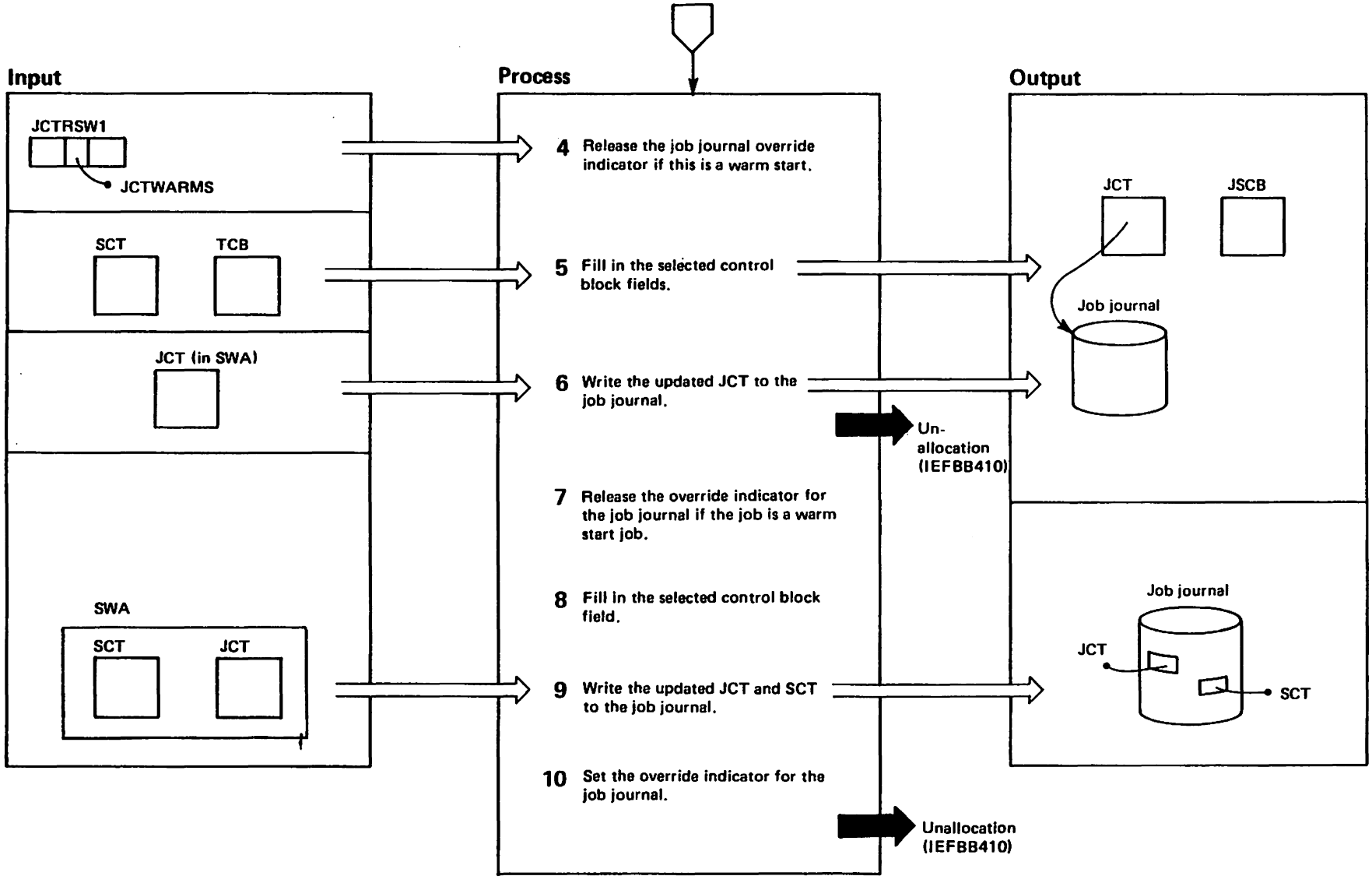
Preparing an Abended Job Step for Restart (IEFRPREP) (Part 1 of 4)



Preparing an Abended Job Step for Restart (IEFRPREP) (Part 2 of 4)

Extended Description	Module	Label
This routine determines if an abended (abnormally terminated) job step task can be restarted. If it can, the routine prepares the task for a restart.		
1 a) Test JCTNORST.	IEFRPREP	IEFRPREP
b) The job cannot restart if, after a checkpoint was taken, dynamic allocation routines have scratched a dynamically allocated data set that is used by the job.		
c) Test JCTCKFT.		
d) Check TCBCMP against the IEFYRCDS table of codes eligible for automatic restart.		JNL03
e) The job journal must have been specified as a SYSGEN option or at IPL time. Test JSCBJNLF.		JNL02
f) Test JSCBJNLE. The routine IEFXB500 will have set this flag if an error occurred during a previous writing of information to the job journal.		JNL01
2 Routine issues a WTOR macro instruction for the operator to give decision regarding a restart.		RP130
3 For any negative response in steps 1 and 2, go to step 7.		

Preparing an Abended Job Step for Restart (IEFRPREP) (Part 3 of 4)



Preparing an Abended Job Step for Restart (IEFRPREP) (Part 4 of 4)

Extended Description	Module	Label
4 The override indicator in the JSCB is set off to allow writing on the journal if the job is a warm start job. Otherwise, the indicator is already off at entry time.	IEFRPREP	
5 Field indicators affected: • JCTACODE (same information as TCBCMP). • JCTCKPTR (if checkpoint restart). • JCTSTEPR (if step restart). • JCTSCT (with value from SCTSNUMB). Based on operator reply, either hold job or re-enqueue job for immediate restart. Save restart step's SCT pointer (in field JCTSSTR). • JCTJSBEX and JSCBJSBX (same information; that is, the job is executing). • JCTJSBTM and JSCBJSBT (same information; that is, the job is terminating).		
6 This prohibits writing on the job journal (see step 4).	IEFXB500	
7 Set override bit as in step 4 to allow updating of job journal.	IEFRPREP	
8 Fields (indicators) affected: • JCTCKPTR (see step 5) • JCTSTEPR (see step 5) • JCTRESTT (no-restart indicator) • JCTACODE (see step 5) • JCTABEND • SCTONLYC (condition code, for use by allocation) • SCTABEND		
9 The routine writes the JCT and SCT to the job journal.	IEFRPREP IEFQB550 IEFXB500	CABEND2
10 The routine sets the override bit on to suppress further writing to the job journal until the job has restarted.	IEFRPREP	

IEFRCSTP - MODULE DESCRIPTION

DESCRIPTIVE NAME: Restart Codes Statement Processor

FUNCTION:

IEFRCSTP is used to customize the standard IBM supplied list of ABEND codes eligible for automatic restart. This list of codes is contained in IEFYRCDS, a data only module residing in SYS1.LINKLIB. IEFRCSTP is invoked by the Generalized Parmlib Scan Routine (IEEMB888).

Two entry points have been defined within IEFRCSTP:

- Entry point IEFRCADD is responsible for processing RESTART type statements in the SYS1.PARMLIB member SCHEDxx. The RESTART statement enables an installation to expand the IBM supplied list of codes from IEFYRCDS to include user defined ABEND codes.
- Entry point IEFRCDEL is responsible for processing NORESTART type statements in the SYS1.PARMLIB member SCHEDxx. The NORESTART statement enables an installation to delete codes from the list of codes eligible for automatic restart. This list includes both the IBM supplied and the user defined codes.

ENTRY POINT: IEFRCADD

PURPOSE: See Function

LINKAGE: CALL

CALLERS: IEEMB888

INPUT:

Statement Processor Parameter List (IEEZB821)

FIELD	LENGTH/MASK	DESCRIPTION
STMTP	36	
STMTID	4	Identifier 'STMT'
STMTVERS	1	Version number
STMTFLAG	1	Control flags
STMTENT	x'80'	STMT type entry
STMTEOP	x'40'	EOP type entry
STMTERR	x'20'	Error in statement processor routine
STMTLEN	2	Length of parameter list
STMTSUFX	2	Suffix of the SYS1.PARMLIB member being processed
STMTRECL	2	Length of STMT record
STMTRECD	4	Address of a complete logical record
STMTUSEP	4	Pointer to a user parameter area
STMTNVTP	4	Pointer to the NVT
STMTNUMB	4	Record number within SCHEDxx being processed
STMTRSV1	8	Reserved

OUTPUT:

Updated version of the table of restart codes eligible for automatic restart

EXIT NORMAL: Return to caller

EXIT ERROR: None. IEFRCSTP will not return if ABBENDED.

ENTRY POINT: IEFRCDEL

IEFRCSTP - MODULE DESCRIPTION (Continued)

PURPOSE: See Function

LINKAGE: CALL

CALLERS: IEEMB888

INPUT: same as for IEFRCADD

OUTPUT: same as for IEFRCADD

EXIT NORMAL: Return to caller

EXIT ERROR: None. IEFRCSTP will not return if ABBENDED.

EXTERNAL REFERENCES:

ROUTINES:
- IEEMB887 - Routine that parses the RESTART/NORESTART statement for valid keyword and abend codes

DATA AREAS:
- IEFYRCDS - Data only module containing the IBM supplied list of system completion codes eligible for automatic restart

CONTROL BLOCKS: None

TABLES:

- IEFRSTB - Table of codes eligible for automatic restart
- PARSETAB - Table of parse arguments to be passed to the generalized parser (IEEMB887)
- EBCDICX - Translate table used to convert EBCDIC to internal hex
- HEXTABLE - Translate table used to flag any non-hexadecimal character
- ALLTABLE - Translate table used to parse through an erroneous string so that all characters will pass

IEFRCSTP - MODULE OPERATION

This module processes RESTART/NORESTART statements in the SYS1.PARMLIB member SCHEDxx. It does the following:

1. On the first statement (STMT) entry to IEFRCSTP, the IBM supplied list of system completion codes eligible for automatic restart will be loaded into storage. This list is contained in IEFYRCDS, a data only module residing in SYS1.LINKLIB. IEFRCSTP will obtain storage from extended CSA to accomodate the default list plus an additional 50 entries. A copy of the IBM supplied list of codes eligible for automatic restart will be placed in the storage area obtained and anchored in the JESRSTRT field of the JESCT. The IBM supplied list will be used as a base and entries will be added or deleted, based on any RESTART/NORESTART statements that appear in the SYS1.PARMLIB member SCHEDxx, to form the table of codes eligible for automatic restart (IEFRSTB).

2. For each STMT entry, IEFRCSTP calls the Generalized Parse Routine (IEEMB887) to parse the statement for valid syntax.
   If the statement is valid, the codes specified are either deleted from the table of codes eligible for automatic restart (IEFRSTB), or added to the end of the user defined portion of the table.
   IEFRCSTP will continue processing codes until the end of the statement or an erroneous code is encountered.
   If an invalid code is encountered, IEFRCSTP will issue an informational message and stop processing the statement. The remainder of the statement will be ignored.

3. On the end of processing (EOP) entry, if no valid RESTART/NORESTART statements were specified, IEFRCSTP will obtain storage from extended CSA. IEFRCSTP will then load the IBM supplied list of system completion codes eligible for automatic restart, found in data only module IEFYRCDS, and copy it into the storage area obtained. In this case the table of codes eligible for automatic restart (IEFRSTB) will consist only of the IBM supplied list of codes eligible for automatic restart.

4. If no operands are specified on the RESTART/NORESTART statement, IEFRCSTP will issue an informational message stating that no operands were specifed. The statement will be ignored.

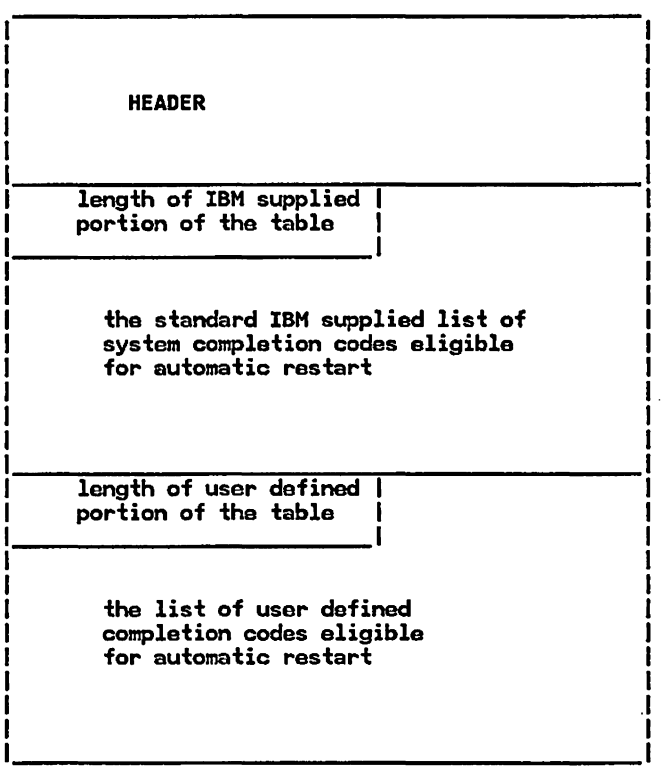
IEFRCSTP is entered at one of two points, IEFRCADD or IEFRCDEL. The point of entry is determined by the general parmlib scan routine, IEEMB888.

- When IEEMB888 invokes IEFRCSTP to process a RESTART statement type, control is passed to the RESTART codes statement processing entry point within IEFRCSTP (IEFRCADD). The restart statement processing will then search through the restart codes table (IEFRSTB) for a matching code. If a matching code is not found, the restart codes statement processing will add the valid code to the user defined portion of IEFRSTB.
If a matching code is found, the restart codes statement processing will issue an informational message stating that a duplicate code exists.

IEFRCSTP - MODULE OPERATION (Continued)

- When (IEEMB888) invokes IEFRCSTP to process a NORESTART statement type, control is passed to the NORESTART statement processing entry point within IEFRCSTP (IEFRCDEL). The NORESTART statement processing looks for a matching code in the table of restart codes (IEFRSTB). If a matching code is found, the code will be deleted from the table. If a matching code is not found, the NORESTART statement processing will issue an informational message stating that the code was not found.

Diagram of IEFRSTB:
(the table of eligible restart codes)



RECOVERY OPERATION:
The processing for this module takes place during nucleus initialization (NIP) time. Although an ESTAE environment could be established at this point, if an ABEND did occur, control would not be received because RTM is not fully initialized yet, therefore an ESTAE environment is not established.

IEFRCSTP - DIAGNOSTIC AIDS

ENTRY POINT NAMES: IEFRCADD
IEFRCDEL

MESSAGES:

IEF738I mem LINE num: <RESTART,NORESTART> STMT IGNORED.
NO OPERANDS SPECIFIED.

IEF734I mem LINE num: <RESTART,NORESTART> <CODE code,
STATEMENT> <IGNORED,ACCEPTED>. REASON= kyrc

Where:
mem - indicates which SCHEDxx SYS1.PARMLIB member contains the bad statement

num - indicates which line in the SCHEDxx SYS1.PARMLIB member is being processed

code - indicates the abend code (specified on the statement) in error

kyrc - represents a Keyword-reason code for the error encountered
ky - Keyword, the numerical representation of the keyword being processed (decimal)
rc - reason code, the numerical representation of the error encountered (decimal)

The following are valid keyword/reason code (kyrc) combinations:

ky: 01 - Unexpected data encountered

rc: 08 - Unrecognized keyword
28 - Keyword list not valid
44 - End delimiter in a keymord list (right parenthesis) missing
48 - Data following Keyword list ending delimiter (right parenthesis) encountered

ky: 02 - CODES(

rc: 16 - Value out of range
52 - Code already exists
56 - Code not found
60 - Duplicate keyword encountered

ABEND CODES: None

WAIT STATE CODES: None

RETURN CODES:

ENTRY POINT IEFRCADD:

EXIT NORMAL:

0 - Request successfully completed

ENTRY POINT IEFRCDEL:

EXIT NORMAL:

0 - Request successfully completed

REGISTER CONTENTS ON ENTRY:

ENTRY POINT IEFRCADD:

Register	0	= Undefined
Register	1	= Address of a word that points to a parameter list
Registers	2-12	= Undefined
Register	13	= Address of savearea
Register	14	= Return address
Register	15	= Entry point address

ENTRY POINT IEFRCDEL:

same as for IEFRCADD

REGISTER CONTENTS ON EXIT:

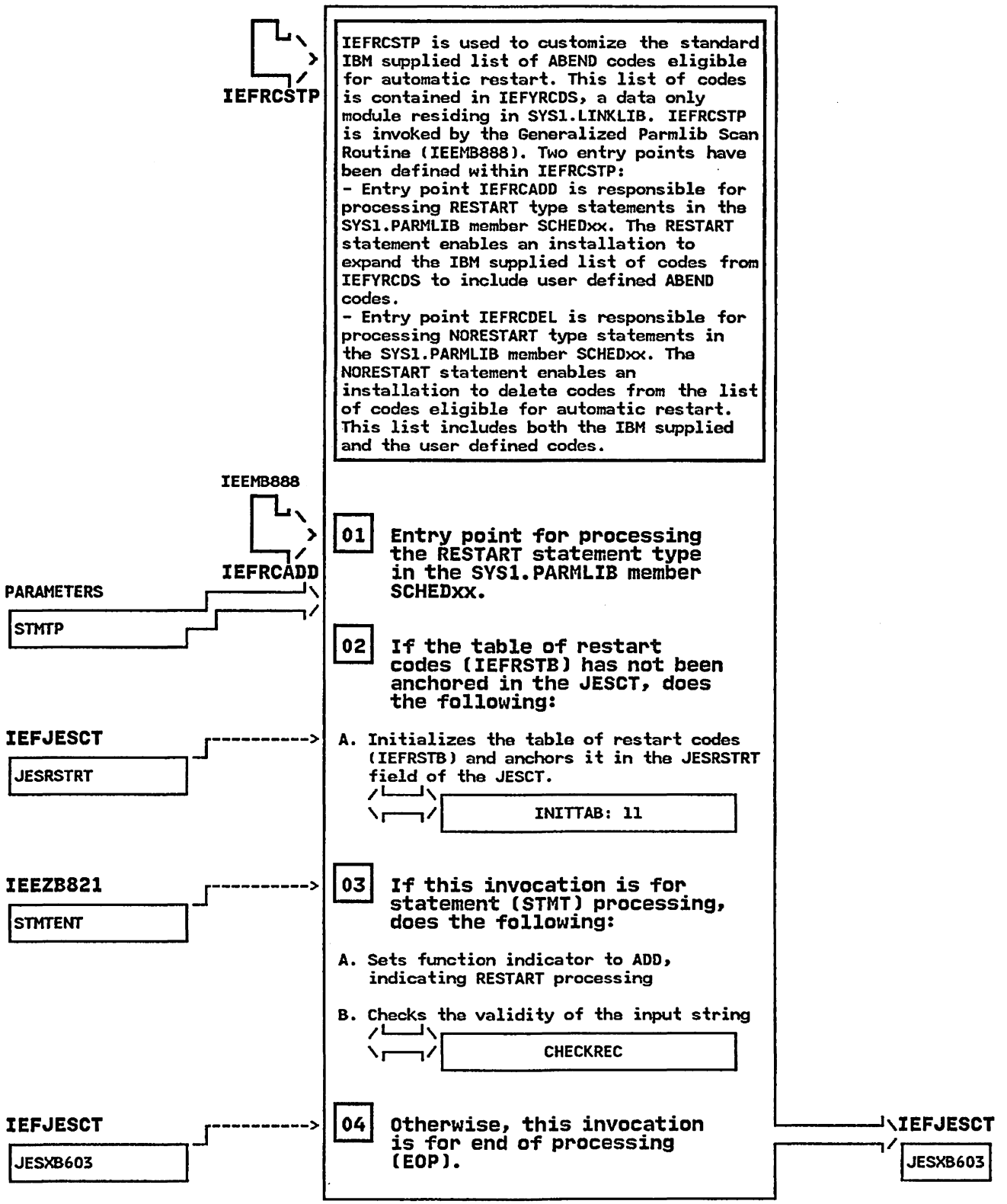
ENTRY POINT IEFRCADD:

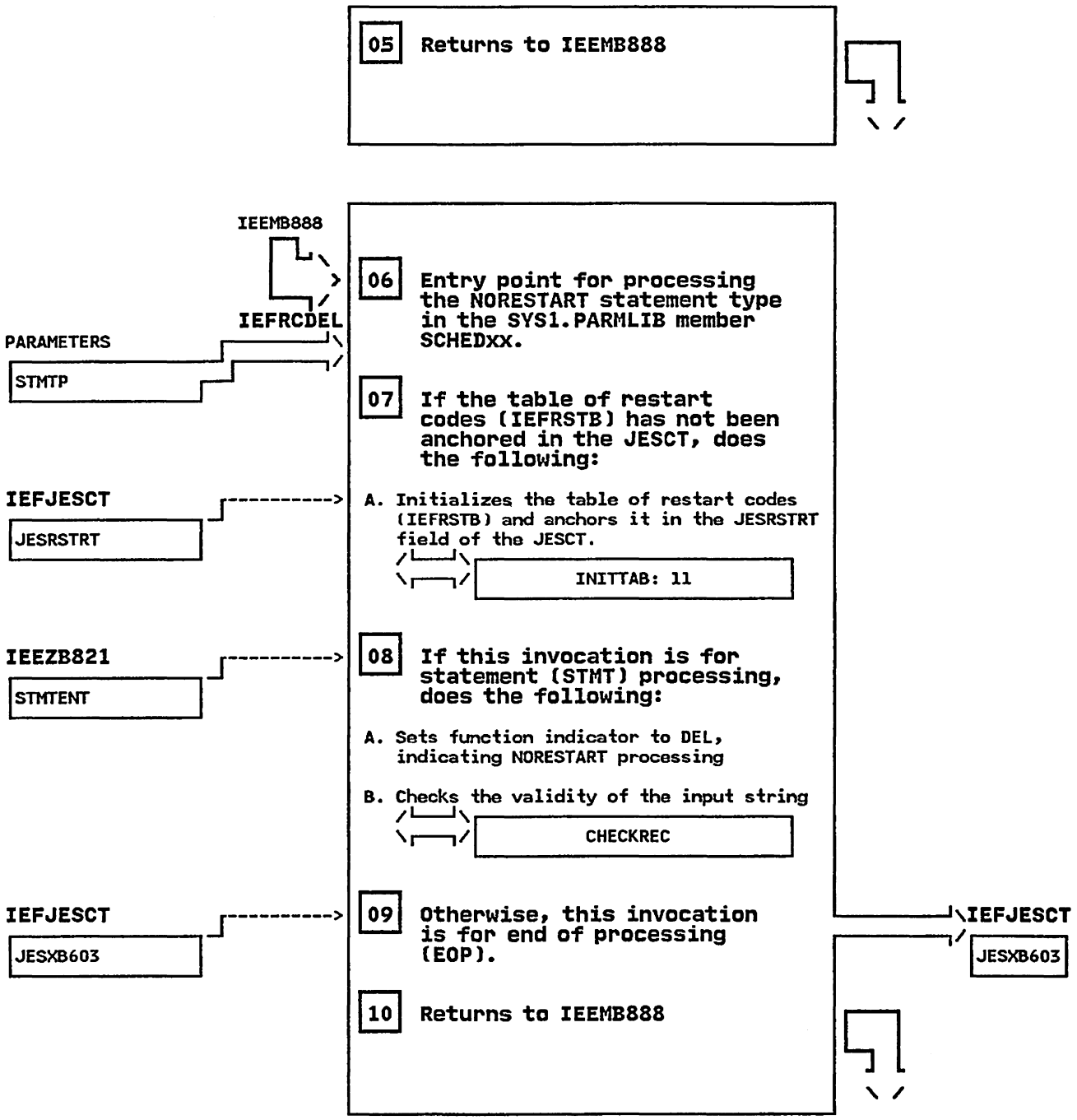
Registers	0-14	= Restored
Register	15	= Return Code

ENTRY POINT IEFRCDEL:

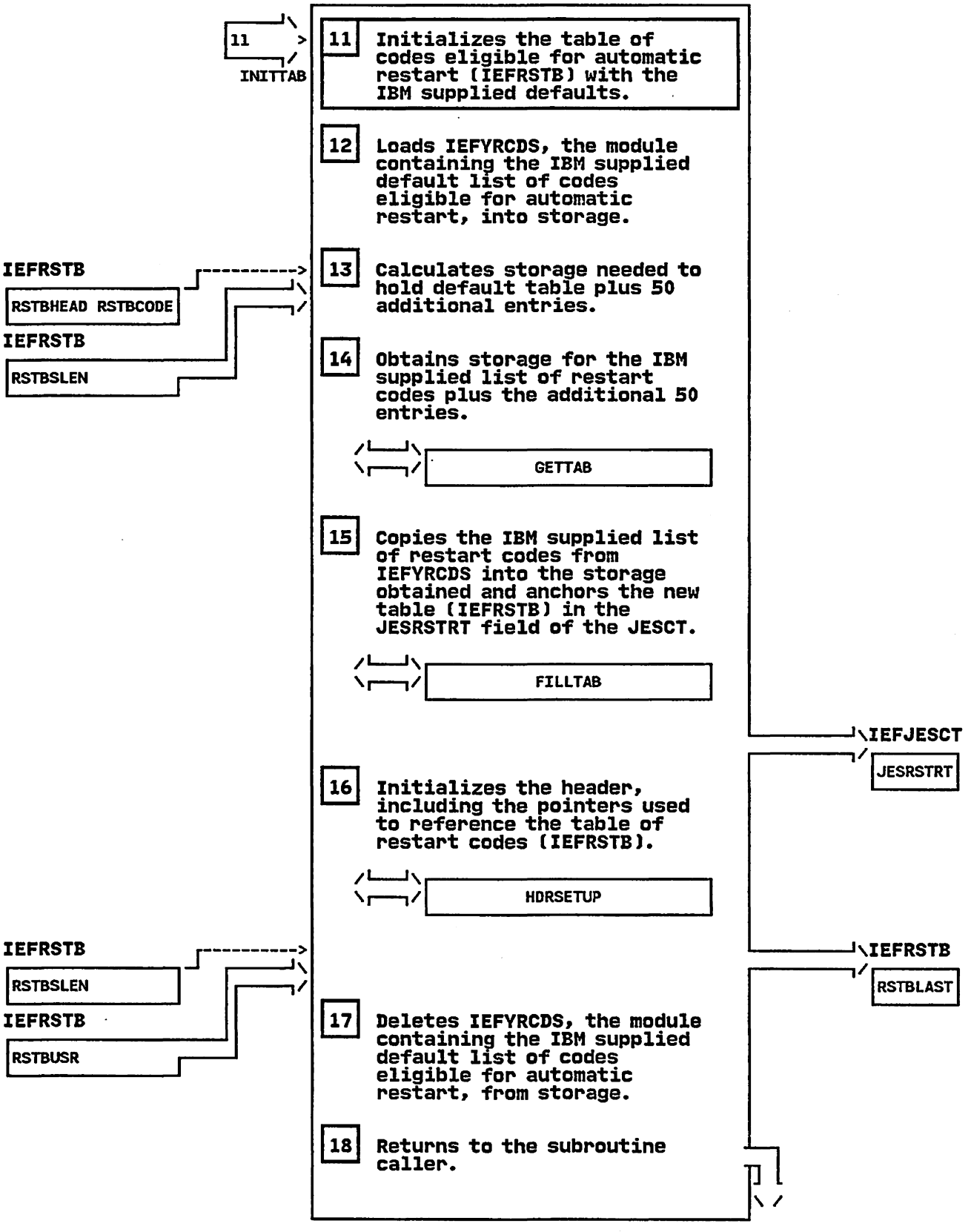
same as for IEFRCADD

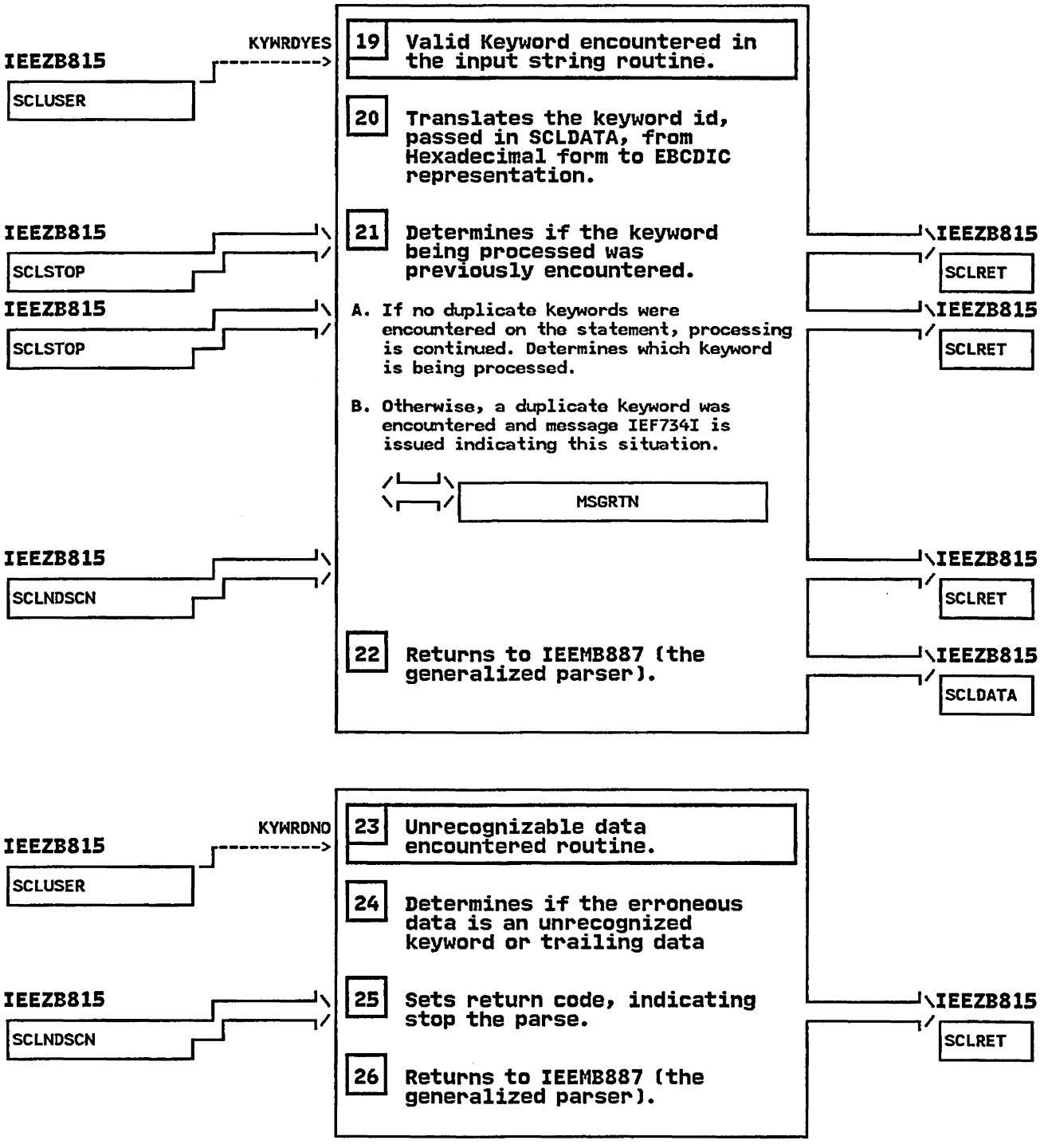
IEFRCSTP - Restart Codes Statement Processor STEP 01



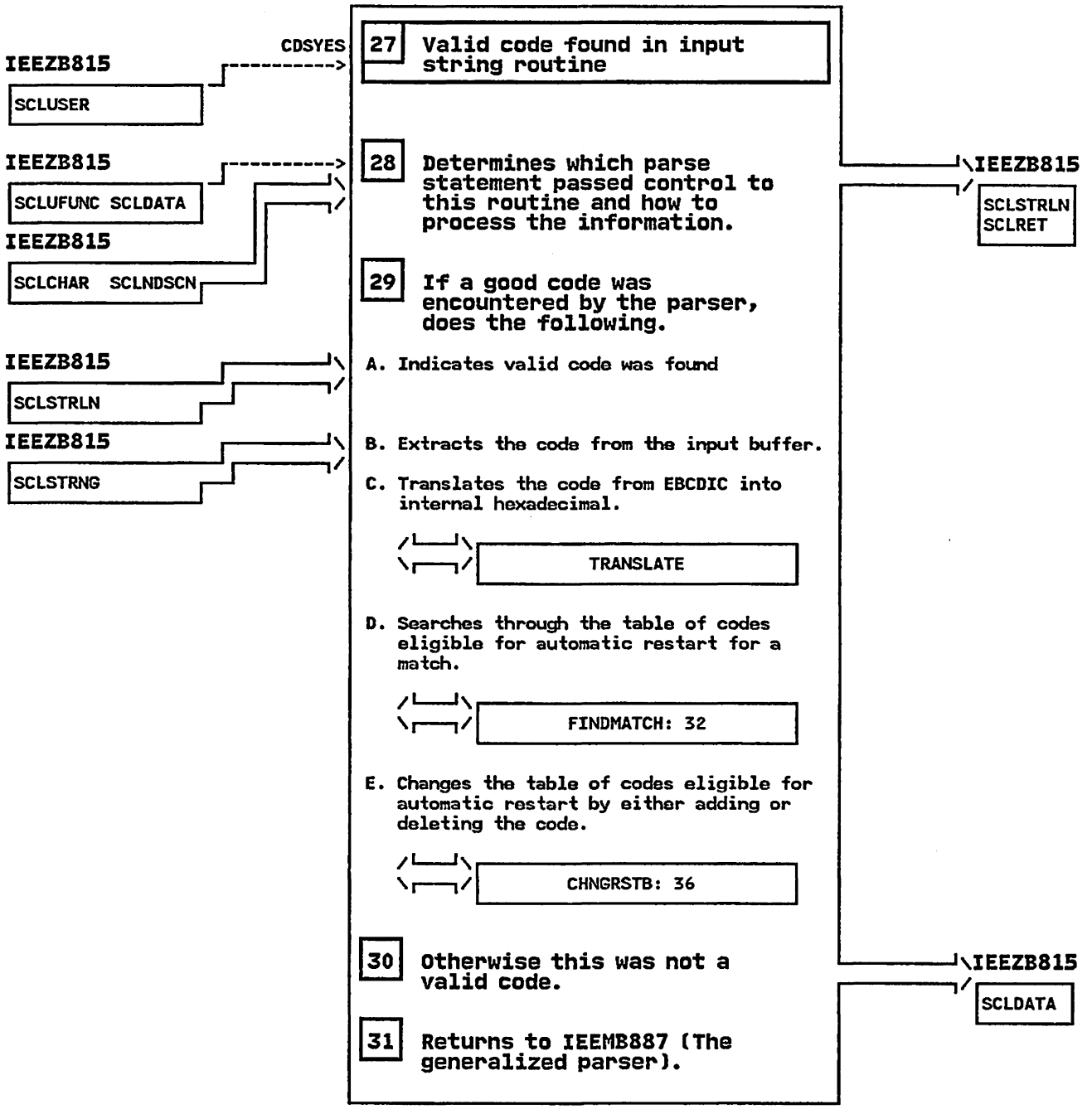


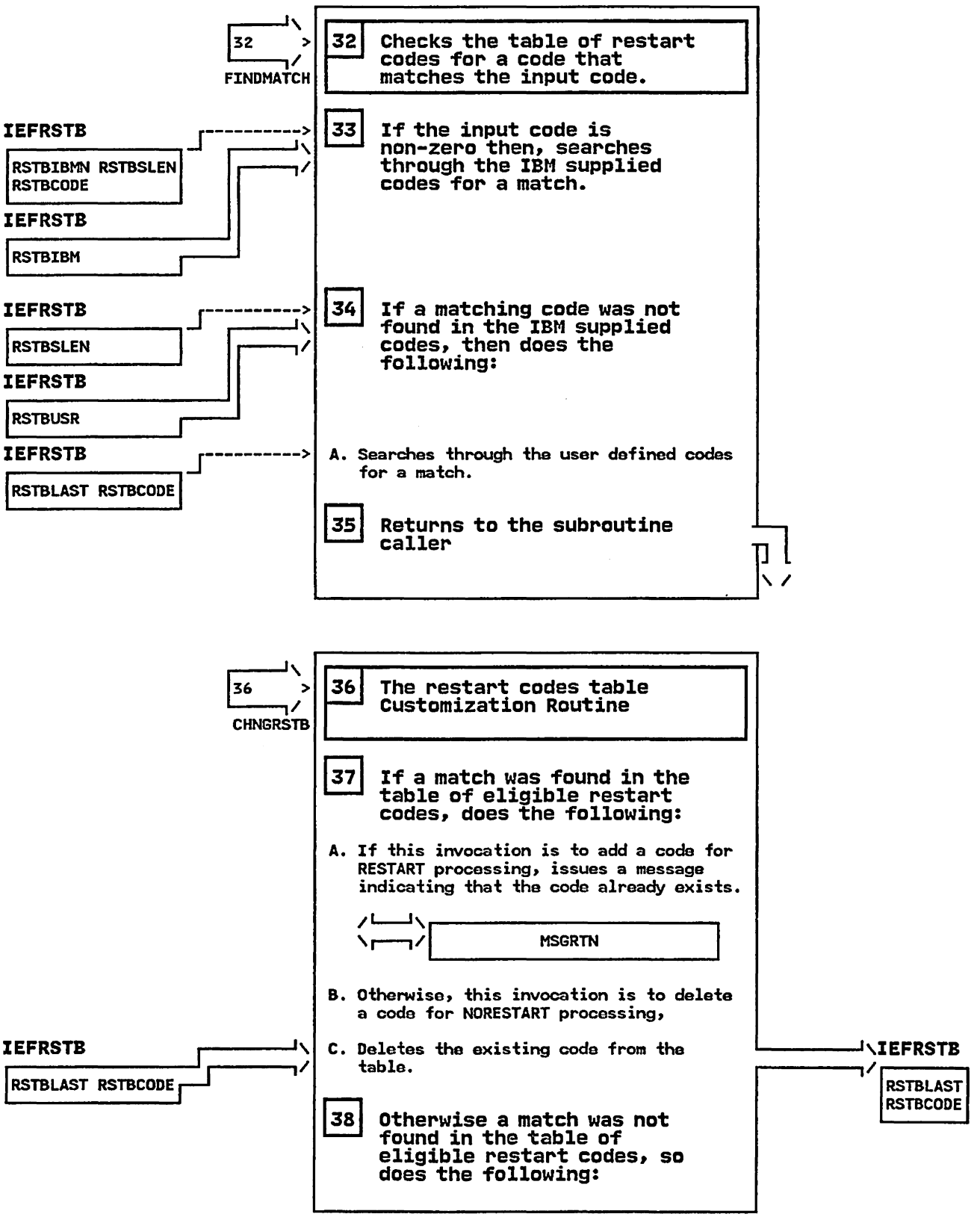
IEFRCSTP - Restart Codes Statement Processor STEP 11



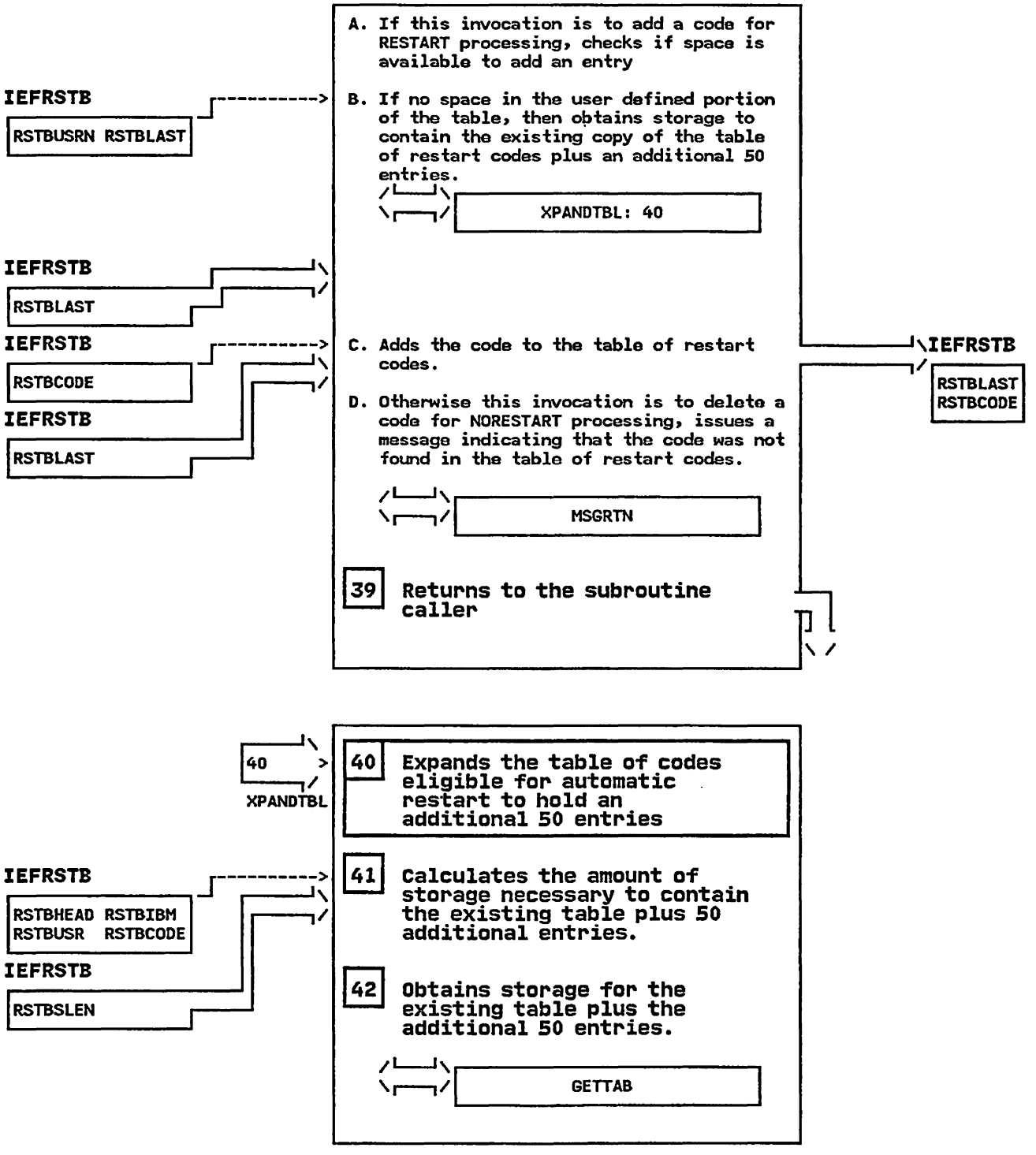


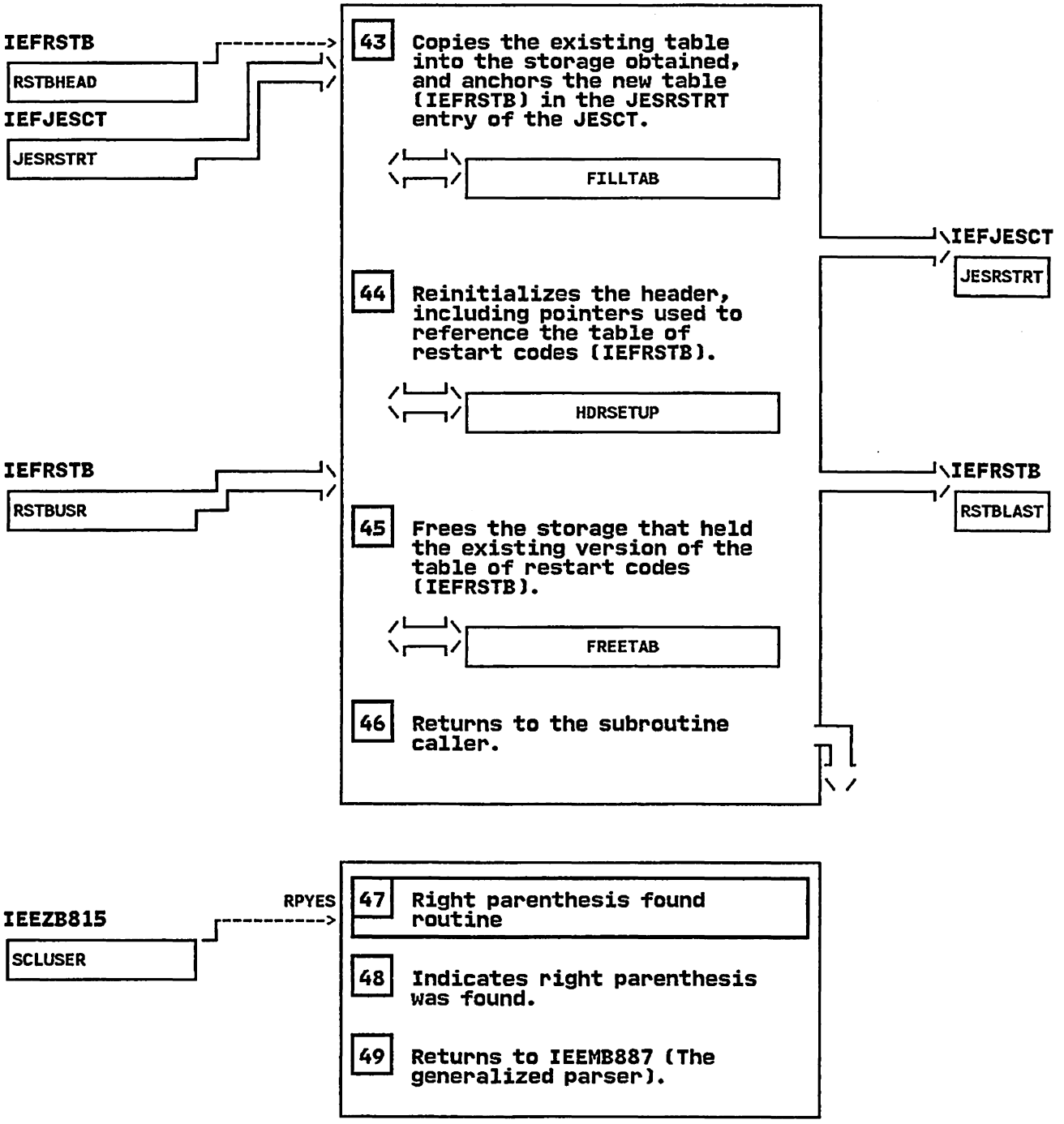
IEFRCSTP - Restart Codes Statement Processor STEP 27



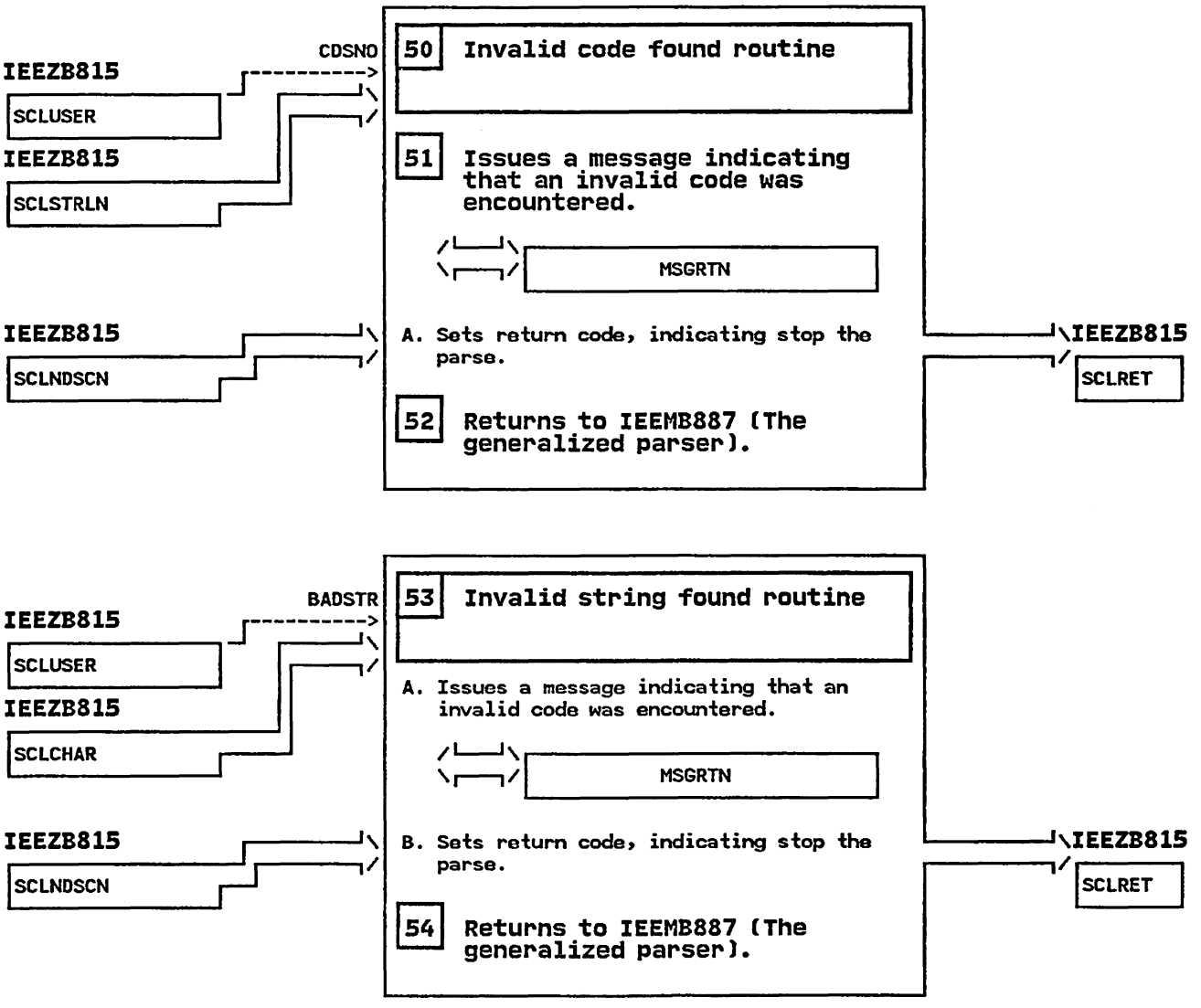


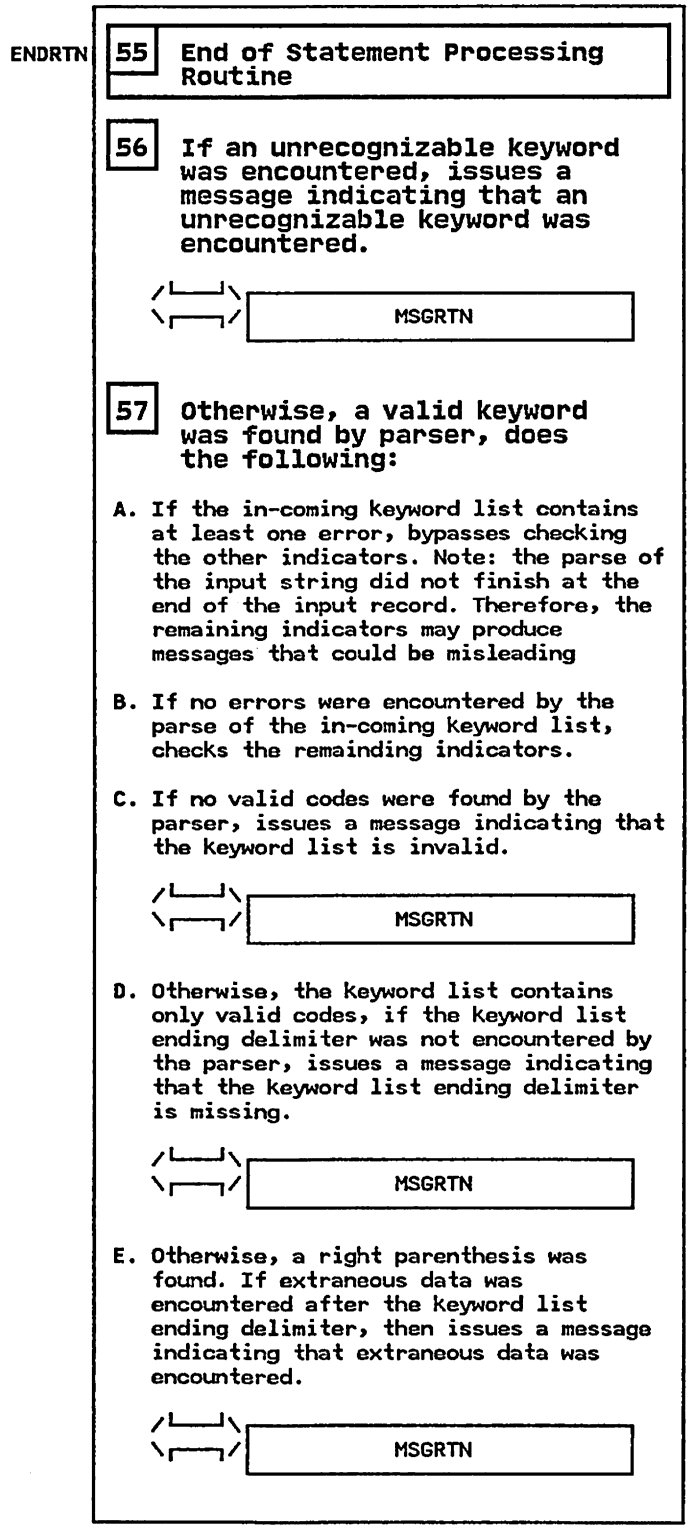
IEFRCSTP - Restart Codes Statement Processor STEP 38A





IEFRCSTP - Restart Codes Statement Processor STEP 50





IEFXBDYS - MODULE DESCRIPTION

DESCRIPTIVE NAME: Restart Dynamic SIOT Processing Routine

FUNCTION:
This module processes SIOTs which were dynamically allocated before the checkpoint was issued. The checkpoint processing will generate type 4 DSDR records in the checkpoint dataset for dynamically allocated DDs. These records contain a copy of the SIOT which is needed to successfully recreate those DDs which were dynamically allocated.

ENTRY POINT: IEFXBDYS

PURPOSE: See Function

LINKAGE: Call

CALLERS: IEFXB609

INPUT:
The checkpoint restart workarea specifically the following fields:

FIELD	LENGTH/MASK	DESCRIPTION
CRWADDRP	4	Pointer to type 4 DSDR (SIOT)
CRWACSIP	4	Pointer to current SIOT

OUTPUT:

FIELD	LENGTH/MASK	DESCRIPTION
CRWAREAS	3	Reason code
Updates the following fields in the current SIOT for SYSOUT		
SCTOUTNM	8	System output program name
SCTOUTNO	4	Output forms number
SCTOUTPN	1	Output class name
SIOTOUTC	1	Number of copies
SIOTDEST	8	Userid for routing output
SIOTSSNM	4	Name of subsystem to process this dataset
SIOTSSDS	'80'X	Subsystem dataset indicator
SIOTHOLD	'40'X	Dataset is to be placed on hold queue
For defered restarts only:		
SIOTTRKM	4	Have allocation indicate to subsystem that a new 'dataset' is needed for this DD.
SIOTDSNM	4	Have allocation indicate to subsystem that a new dataset name is needed for this DD.

EXIT NORMAL: Return to IEFXB609

EXIT ERROR: Return to IEFXB609

ENTRY POINT: DYSRETRY

PURPOSE:
Performs clean up processing when an ABEND occurs in the process of merging dynamic SIOT information.

IEFXBDYS - MODULE DESCRIPTION (Continued)

LINKAGE: SYNCH

CALLERS: RTM

INPUT: Estae Parameter List

OUTPUT: None

EXIT NORMAL:

EXIT ERROR: Return to caller

EXTERNAL REFERENCES:

ROUTINES: None

DATA AREAS:
IEFXBCRW - Checkpoint Restart Work Area
- Current SIOT
- type 4 DSDR for SIOT

CONTROL BLOCKS: SIOT - Step Input/Output Table

IEFXBDYS - MODULE OPERATION

IEFXBDYS merges dynamically allocated SIOTs.

1. IEFXBDYS determines if the DSDR represents a SIOT for a catalog, a GDG 'all' request, or a multi-unit request.
   These SIOTs are "system generated" (created by Allocation). Since these SIOT's will be recreated no further processing is done in this module for these cases.

2. Ensures the ddname for the DSDR matches the ddname of the current SIOT (see step 4 if not).

3. Determines if the DD represents a SIOT for a GDG all request.

   A. Removes the last nine characters from the dataset name so that allocation will recognize this as a GDG all request and recreate all the generations.

4. If the ddname of the DSDR SIOT was found in the SWA structure, then tests for a SYSOUT dataset.

   A. If this DSDR represents a SYSOUT dataset then merges the sysout fields of this SIOT.

   B. If this DSDR represents a SYSOUT dataset and this is a deferred restart then requests a new dataset and dataset name via allocation from the subsystem.

5. If the ddname of the DSDR SIOT did not match the current SWA SIOT then issues a reason code and return code.

6. Return to caller

RECOVERY OPERATION:
If an abend occurs in this module, the recovery point in IEFXB609 receives control from RTM. The recovery routine specifies to RTM the retry address, DYSRETRY, in the checkpoint restart workarea. When DYSRETRY receives control from RTM, it does the following:

1. Sets the return code to decimal 36 to indicate a checkpoint restart system error.

2. Performs the necessary cleanup.

3. Returns to the caller.

IEFXBDYS - DIAGNOSTIC AIDS

ENTRY POINT NAMES: IEFXBDYS
DYSRETRY

MESSAGES: None

ABEND CODES:

00B - Scheduler Restart Local Storage Error
Associated Reason Codes:
1 - A Scheduler Restart module failed to obtain enough storage from the preallocated storage area.
2 - A Scheduler Restart module attempted to free an area outside the preallocated storage area.

WAIT STATE CODES: None

RETURN CODES:

ENTRY POINT IEFXBDYS:

EXIT NORMAL:

(decimal)
Register 15 = 0 - Request completed successfully
Reason Code in CRWAREAS:
CRWANOER (0) - no error

EXIT ERROR:

(decimal)
Register 15 = 4 - Request processed unsuccessfully
Reason Code in CRWAREAS:
CRWADYNM (340) - Error in processing dynamically allocated SIOT (DSDR SIOT does not match current SIOT)

ENTRY POINT DYSRETRY:

EXIT ERROR:

(decimal)
Register 15 = 36 - Restart system error

REGISTER CONTENTS ON ENTRY:

ENTRY POINT IEFXBDYS:

Register 0 = Undefined
Register 1 = Address of a one word parameter list which contains the address of the checkpoint restart workarea.
Registers 2-12 = Undefined
Register 13 = Address of 18-word save area
Register 14 = Return address
Register 15 = Entry point address

ENTRY POINT DYSRETRY:

Register 0 = Undefined
Register 1 = Address of the ESTAE parameter list
Registers 2-14 = Undefined

IEFXBDYS - DIAGNOSTIC AIDS (Continued)

Register 15 = Entry point address

REGISTER CONTENTS ON EXIT:

ENTRY POINT IEFXBDYS:

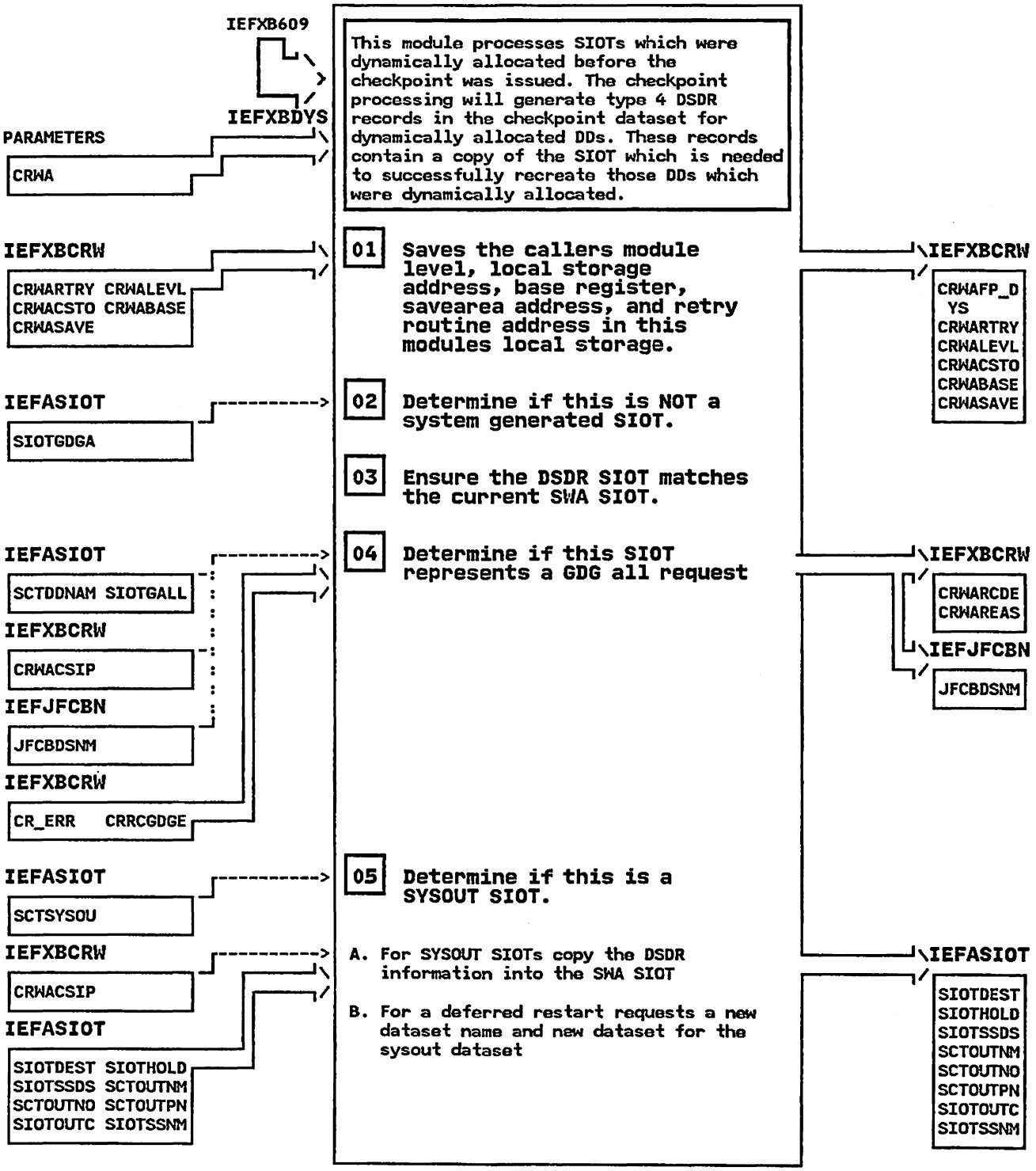
Register 0-14 = Restored
Register 15 = Return Code

ENTRY POINT DYSRETRY:

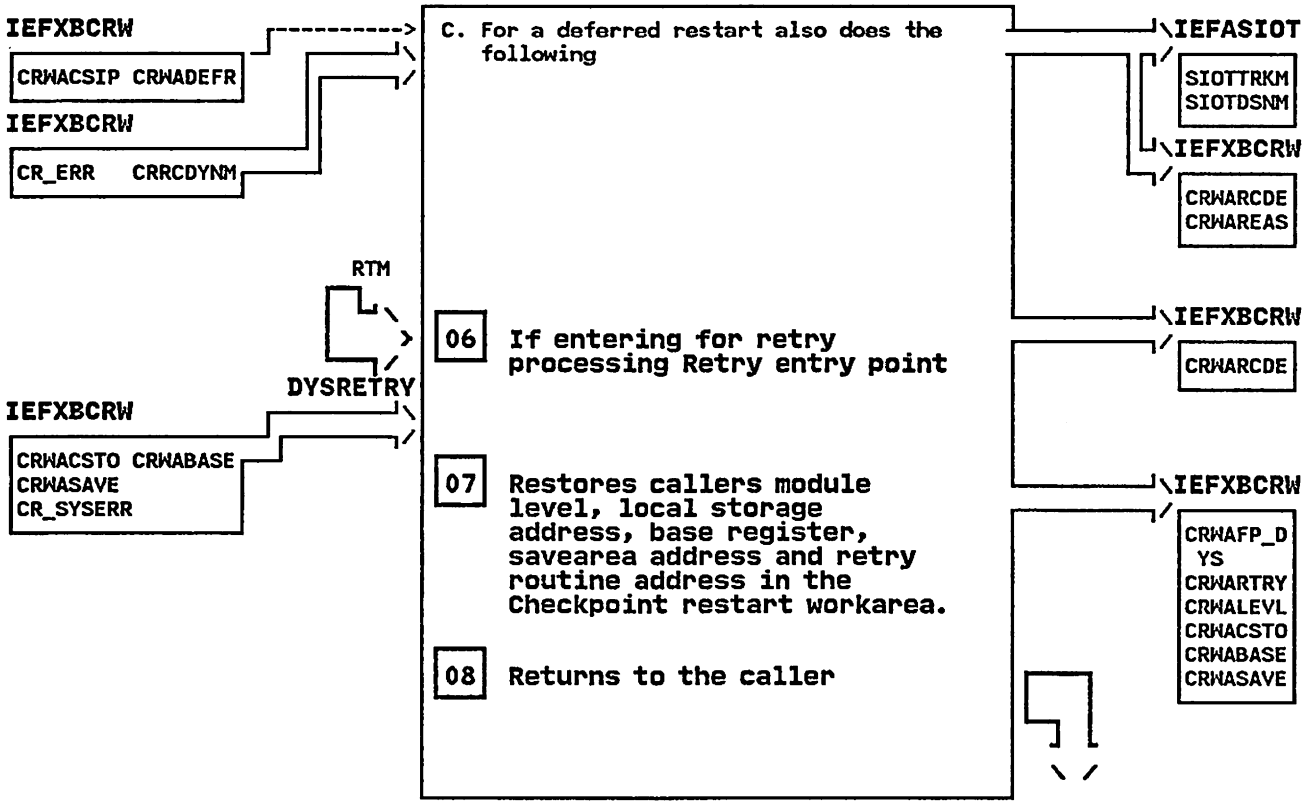
Registers 0-14 = Restored
Register 15 = Return code

IEFXBDYS - Restart Dynamic SIOT Processing Routine

STEP 01



IEFXBDYS - Restart Dynamic SIOT Processing Routine STEP 05C



IEFXBGDG - DIAGNOSTIC AIDS

ENTRY POINT NAMES: IEFXBGDG
XBGDGRTY

MESSAGES: None

ABEND CODES: None

WAIT STATE CODES: None

RETURN CODES:

ENTRY POINT IEFXBGDG:

EXIT NORMAL:

(decimal)
Register 15 = 0 - Processed request.

ENTRY POINT XBGDGRTY:

EXIT ERROR:

Register 15 = 36 (decimal) Restart System error

REGISTER CONTENTS ON ENTRY:

ENTRY POINT IEFXBGDG:

```
Register  0    = Undefined
Register  1    = Address of the input parameter list
Registers 2-12 = Undefined
Register  13   = Address of 18-word save area
Register  14   = Return address
Register  15   = Entry point address
```

ENTRY POINT XBGDGRTY:

```
Register  0    = Undefined
Register  1    = Address of the ESTAE parameter list
Registers 2-13 = Undefined
Register  14   = Return address to RTM
Register  15   = Entry point address
```

REGISTER CONTENTS ON EXIT:

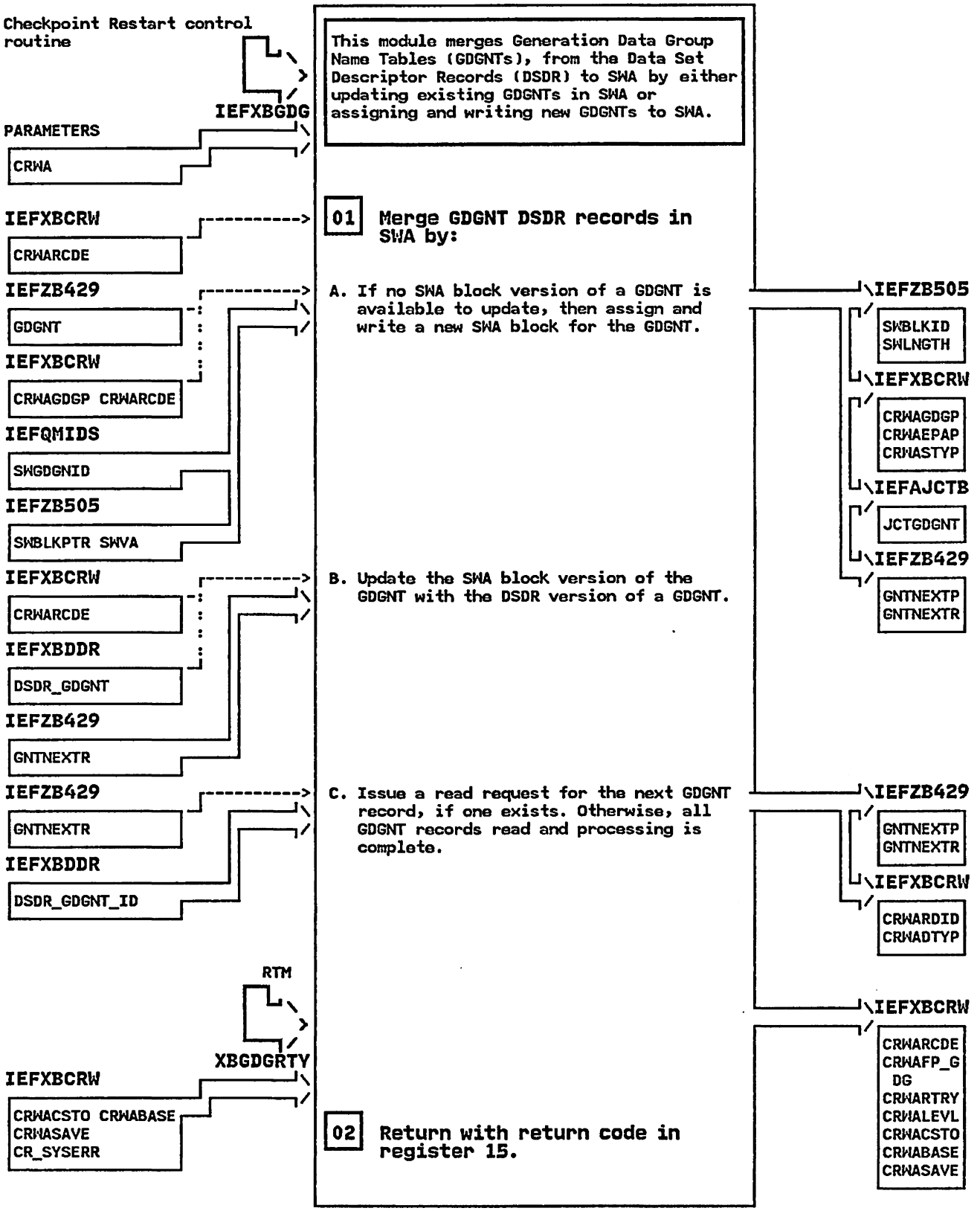
ENTRY POINT IEFXBGDG:

```
Register  0-14 = Restored
Register  15   = Return Code
```

ENTRY POINT XBGDGRTY:

```
Registers 0-14 = Restored
Register  15   = Return code
```

IEFXBGDG - Checkpoint Restart GDGNT Processing Routine STEP 01



IEFXBRDC - MODULE DESCRIPTION

DESCRIPTIVE NAME: Restart Read Checkpoint Dataset Routine

FUNCTION:
This module performs the reading of records from the checkpoint dataset and/or updating a pointer to point to the next record within the buffer.

ENTRY POINT: IEFXBRDC

PURPOSE: See Function

LINKAGE: Call

CALLERS: IEFXB609 and other restart processing routines

INPUT:
The checkpoint restart workarea specifically the following fields:

FIELD	LENGTH/MASK	DESCRIPTION
CRWADCBP	4	Pointer to the DCB for the checkpoint dataset
CRWACKID	16	ID of the checkpoint header record to be read
CRWARDID	2	ID of DSDR to be read
CRWADTYP	1	Type of DSDR to be read
CRWARANY	X'80'	Read any
CRWARCHR	X'40'	Read Checkpoint Header Record
CRWARSPC	X'20'	Read Specified DSDR type
CRWADDRP	4	Pointer to current record in the buffer

OUTPUT:

FIELD	LENGTH/MASK	DESCRIPTION
CRWARCDE	4	Return code
CRWAREAS	3	Reason code
CRWADDRP	4	Virtual address of DSDR within buffer

EXIT NORMAL: Return to caller

EXIT ERROR: Return to caller

ENTRY POINT: RDCRETRY

PURPOSE:
Performs clean up processing when an ABEND occurs in the process of reading the checkpoint dataset.

LINKAGE: SYNCH

CALLERS: RTM

INPUT: Estae Parameter List

OUTPUT: None

EXIT NORMAL:

EXIT ERROR: Return to caller

ENTRY POINT: SYNEXIT

IEFXBRDC - MODULE DESCRIPTION (Continued)

PURPOSE:
Entered when an uncorrectable input/output error occurred while reading the checkpoint dataset.

LINKAGE: Branch

CALLERS: Code supporting the READ macro

INPUT: Information in registers 0-1

OUTPUT: None

EXIT NORMAL: Return to caller

EXIT ERROR: Return to caller

EXTERNAL REFERENCES:

ROUTINES: None

DATA AREAS:
IEFXBCRW - Checkpoint Restart Work Area
- Checkpoint Dataset

CONTROL BLOCKS:
DCB - Data Control Block
JCT - Job Control Table
JSCB - Job Step Control Block
PSA - Prefixed Save Area
TCB - Task Control Block

IEFXBRDC - MODULE OPERATION

IEFXBRDC reads the type of record specified as input from the checkpoint dataset.

1. IEFXBRDC determines the type of record to be read from the checkpoint dataset.

2. If the type of record to be read is a checkpoint header record, the checkpoint dataset will be read until a record with the matching checkpoint ID is found.

3. If the type of record to be read is for a specific type of DSDR, the checkpoint dataset will be read or the pointer within the buffer will be incremented to the next record within the buffer. If the DSDR type is not the same as that requested, return code 4 with reason code 300 will be set.

4. If the type of record to be read is for any type of DSDR, the checkpoint dataset will be read or the pointer within the buffer will be incremented to the next record within the buffer.

5. If while reading the checkpoint dataset a record of undetermined type is found, return code 4 with reason code 235 will be set.

6. If while reading the checkpoint dataset an uncorrectable input/output error occurs, return code 4 with reason code 038 will be set.

7. Return to caller

RECOVERY OPERATION:

If an abend occurs in this module, the recovery point in IEFXB609 receives control from RTM. The recovery routine specifies to RTM the retry address RDCRETRY in the checkpoint restart workarea. When RDCRETRY receives control from RTM, it does the following:

1. Sets the return code to decimal 36 to indicate a checkpoint restart system error.

2. Performs the necessary cleanup.

3. Returns to the caller.

IEFXBRDC - DIAGNOSTIC AIDS

ENTRY POINT NAMES: IEFXBRDC
RDCRETRY
SYNEXIT

MESSAGES: None

ABEND CODES:

00B - Scheduler Restart Local Storage Error
Associated Reason Codes:
1 - A Scheduler Restart module failed to obtain enough storage from the preallocated storage area.
2 - A Scheduler Restart module attempted to free an area outside the preallocated storage area.

WAIT STATE CODES: None

RETURN CODES:

ENTRY POINT IEFXBRDC:

EXIT NORMAL:

(decimal)
Register 15 = 0 - Request completed successfully
Reason Code in CRWAREAS:
CRWANOER (0) - no error

EXIT ERROR:

(decimal)
Register 15 = 4 - Request processed unsuccessfully
Reason Code in CRWAREAS:
CRRCERIO (038) - Uncorrectable input/output error
CRRCUTYP (235) - Undetermined type of checkpoint record was encountered
CRRCEOF (237) - End of file reached
CRRCNTYP (300) - DSDR type read is not type requested

ENTRY POINT RDCRETRY:

EXIT ERROR:

(decimal)
Register 15 = 36 - Restart system error

ENTRY POINT SYNEXIT:

EXIT NORMAL:

Register 15 = 4 - Request processed unsuccessfully
Reason Code in CRWAREAS:
CRRCERIO (038) - Uncorrectable input/output error

EXIT ERROR:

Register 15 = 36 - Restart system error

REGISTER CONTENTS ON ENTRY:

IEFXBRDC - DIAGNOSTIC AIDS (Continued)

ENTRY POINT IEFXBRDC:

```
Register  0     = Undefined
Register  1     = Address of a one word parameter
                  list which contains the address
                  of the checkpoint restart workarea.
Registers 2-12  = Undefined
Register  13    = Address of 18-word save area
Register  14    = Return address
Register  15    = Entry point address
```

ENTRY POINT RDCRETRY:

```
Register  0     = Undefined
Register  1     = Address of the ESTAE parameter list
Registers 2-14  = Undefined
Register  15    = Entry point address
```

ENTRY POINT SYNEXIT:

```
Register  0-1   = Status information reflecting the
                  type of error
Registers 2-13  = Contents at the time the READ was
                  issued
Register  14    = Return address
Register  15    = Entry point address
```

REGISTER CONTENTS ON EXIT:

ENTRY POINT IEFXBRDC:

```
Register  0-14  = Restored
Register  15    = Return Code
```

ENTRY POINT RDCRETRY:

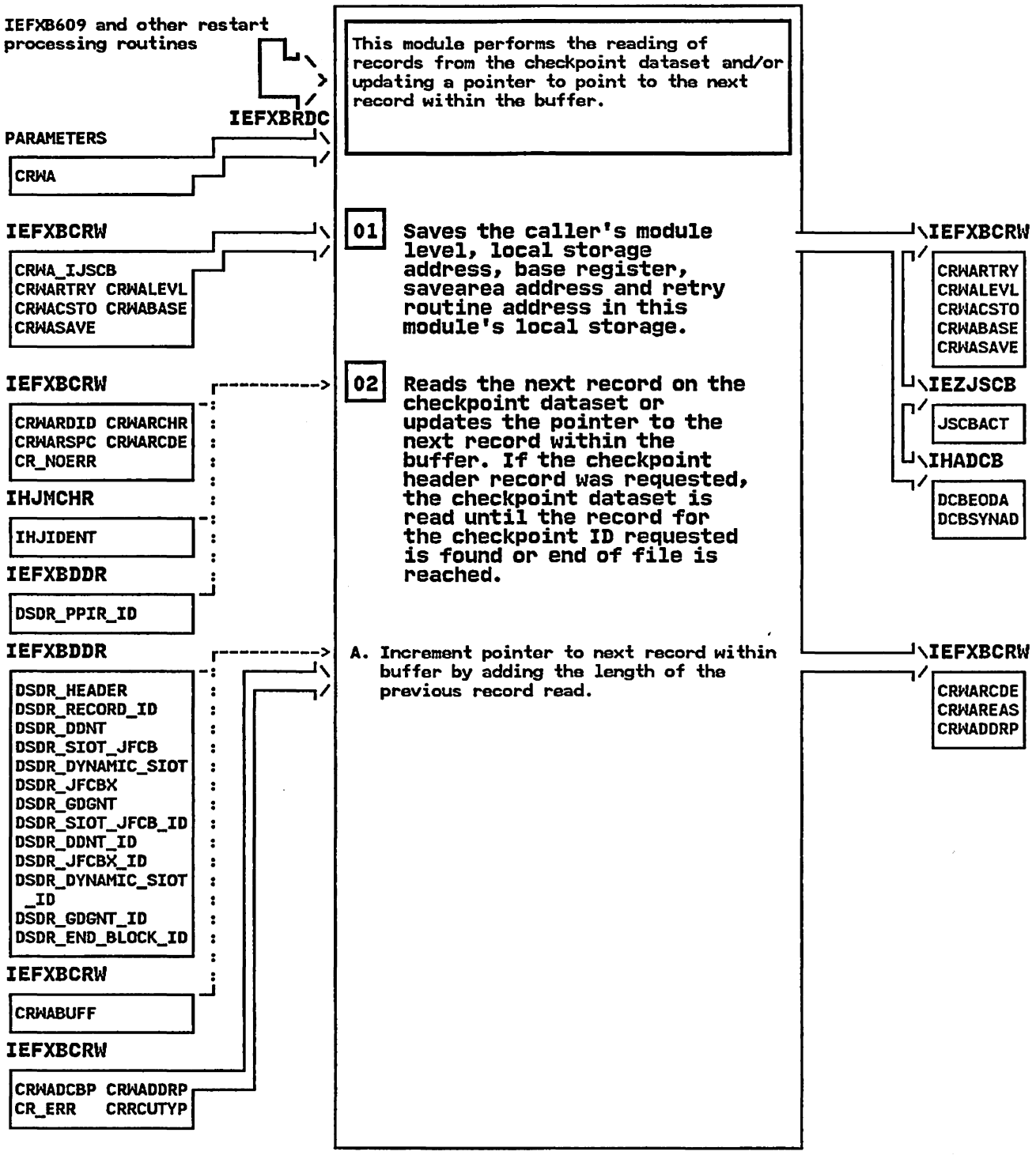
```
Registers 0-14  = Restored
Register  15    = Return code
```

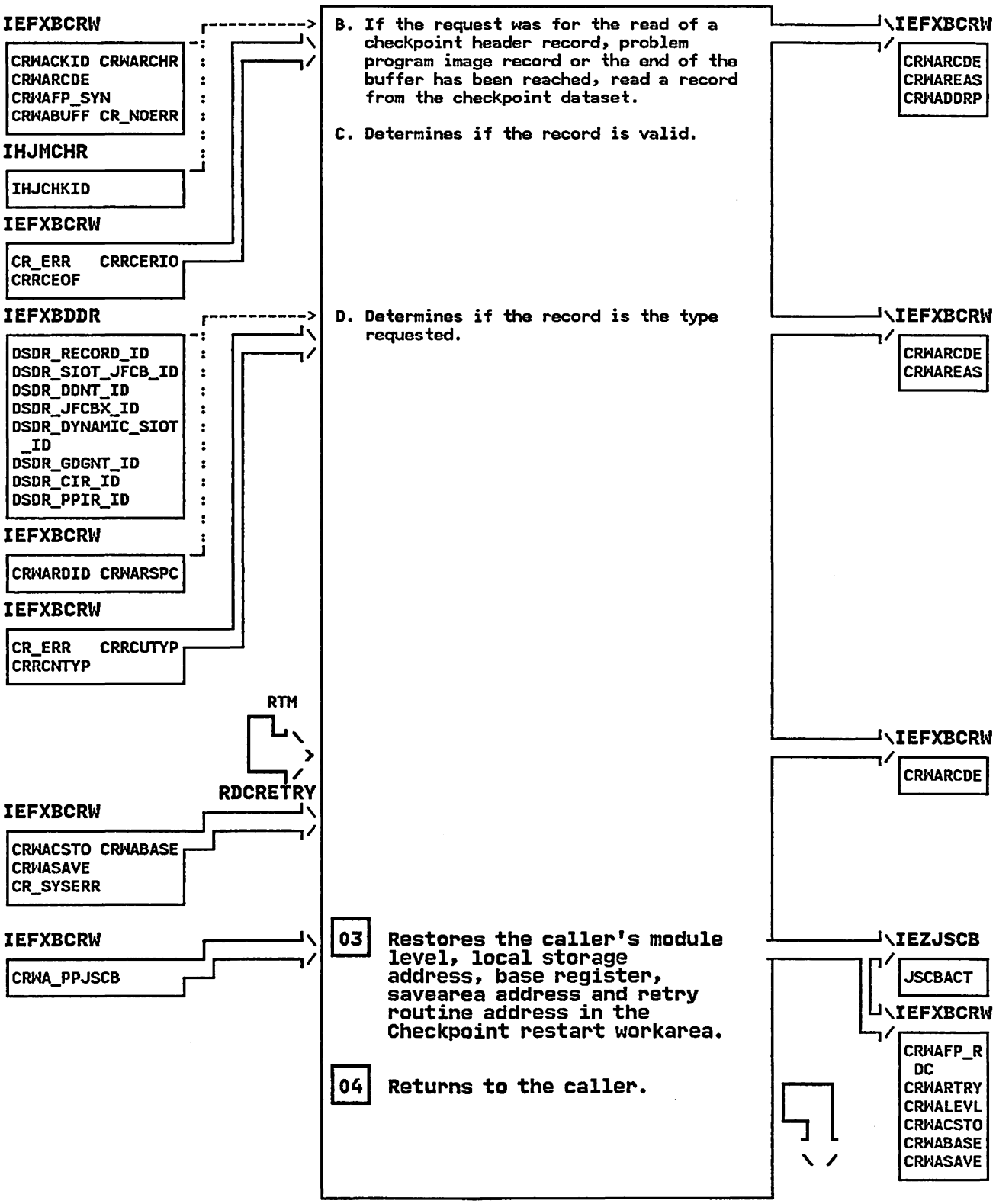
ENTRY POINT SYNEXIT:

```
Registers 0-14  = Same as at time of entry
Register  15    = Return code
```

IEFXBRDC - Restart Read Checkpoint Dataset Routine

STEP 01





IEFXBSWA - MODULE DESCRIPTION

DESCRIPTIVE NAME: Checkpoint Restart SWA Manager Interface Routine

FUNCTION:
This module performs the common SWA Manager READ/LOCATE, WRITE/LOCATE or ASSIGN/CONDITIONAL requests for the Checkpoint Restart control and processing routines.

ENTRY POINT: IEFXBSWA

PURPOSE: See Function

LINKAGE: Call

CALLERS:
Checkpoint Restart control and processing routines

INPUT: Register 1 points to a word pointing to CRWA

OUTPUT: SWA Manager call made as per caller's request.

EXIT NORMAL: Return to caller

ENTRY POINT: XBSWARTY

PURPOSE:
To perform cleanup processing when an abend occurred during SWA Manager processing.

LINKAGE: SYNCH

CALLERS: RTM

INPUT: Estae parameters

OUTPUT: None

EXIT NORMAL:

EXIT ERROR: Return to caller

EXTERNAL REFERENCES:

ROUTINES: SWA Manager

DATA AREAS: IEFXBCRW - Checkpoint Restart Work Area

CONTROL BLOCKS:
None
CVT - Communications Vector Table . . . . . (U)
JESCT - Job Entry Subsystem Communications Table. . (U)

(C=create, M=modify, U=use, D=delete)

IEFXBSWA - MODULE OPERATION

The following steps are involved in the process of invoking the SWA Manager:

1. If the pointer for the External Parameter Area (EPA) in the Checkpoint Restart Work Area (CRWA) field CRWAEPAP is zero then return to the caller with a return code of 4 in register 15 and reason code 281 (decimal) in CRWAREAS.

2. If the SVA field SWVA for a READ LOCATE or WRITE LOCATE is zero, then return to the caller with a return code of 4 in register 15 and reason code 283 (decimal) in CRWAREAS.

3. For request types READ LOCATE and WRITE LOCATE, check for multiple EPAs. Check each EPA for a valid SVA. For an invalid EPA, set the return code to 4 in register 15 and reason code 280 (decimal) in CRWAREAS.

4. Select the SWA Manager call type using the CRWA field CRWASTYP. For each call type invoke the SWA Manager. If the type is not recognized, then return with a return code of 4 in register 15 and reason code 282 (decimal) in CRWAREAS. If the request type is ASSIGN and the return code from SWA Manager is non zero then return with a return code of 4 in register 15 and reason code 280 (decimal) in CRWAREAS.

5. Return to caller

IEFXBSWA - DIAGNOSTIC AIDS

ENTRY POINT NAMES: IEFXBSWA
XBSWARTY

MESSAGES: None

ABEND CODES: None

WAIT STATE CODES: None

RETURN CODES:

ENTRY POINT IEFXBSWA:

EXIT NORMAL:

(dec)
Register 15 = 0 - Processed request.

(dec)
Register 15 4 - Request not processed.
Reason codes with return code 4
280 - Error while processing a SWA Manager request.
281 - No EPA specified for a SWA Manager Request.
282 - Invalid SWA Manager request type.
283 - Invalid SVA specified in EPA.

ENTRY POINT XBSWARTY:

EXIT ERROR:

Register 15 = 36 (decimal) Restart System error

REGISTER CONTENTS ON ENTRY:

ENTRY POINT IEFXBSWA:

Register 0 = Undefined
Register 1 = Address of the input parameter list
Registers 2-12 = Undefined
Register 13 = Address of 18-word save area
Register 14 = Return address
Register 15 = Entry point address

ENTRY POINT XBSWARTY:

Register 0 = Undefined
Register 1 = Address of the ESTAE parameter list
Registers 2-13 = Undefined
Register 14 = Return address to RTM
Register 15 = Entry point address

REGISTER CONTENTS ON EXIT:

ENTRY POINT IEFXBSWA:

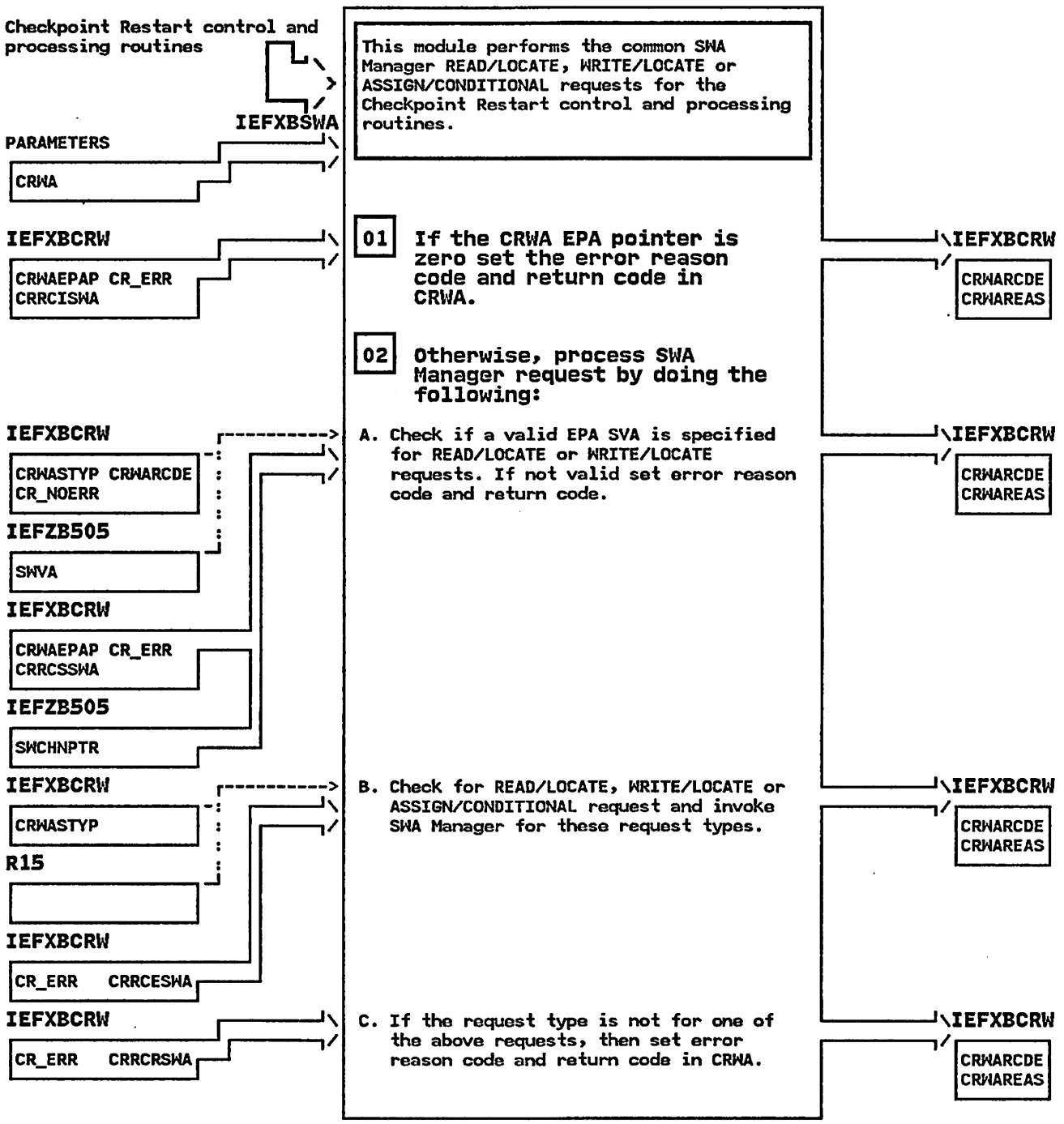
Register 0-14 = Restored
Register 15 = Return Code

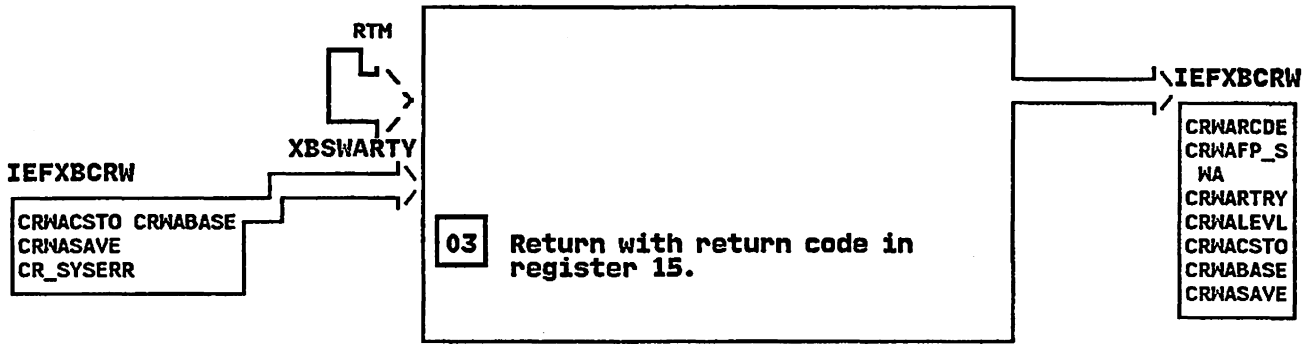
IEFXBSWA - DIAGNOSTIC AIDS (Continued)

ENTRY POINT XBSWARTY:

Registers 0-14 = Restored
Register 15 = Return code

IEFXBSWA - Checkpoint Restart SWA Manager Interface Routine STEP 01





IEFXBSJX - MODULE DESCRIPTION

DESCRIPTIVE NAME: Restart SIOT/JFCB and Extension Processing Routine

FUNCTION:
This module processes the SIOTs, JFCBs, and JFCB extensions corresponding to the DSDR read from the checkpoint dataset.

ENTRY POINT: IEFXBSJX

PURPOSE: See Function

LINKAGE: Call

CALLERS: IEFXB609

INPUT:
The checkpoint restart workarea and a function code indicating the type of processing to be performed.

OUTPUT:
If the function code was for SIOT/JFCB processing, the DSDR copies of the SIOT/JFCB and any JFCB extensions are merged to the SWA copies.

If the function code was for the initialization of a new SIOT/JFCB, SWA blocks are assigned and initialized for the SIOT and JFCB.

CRWACSIP - Address of new SIOT

EXIT NORMAL: Return to IEFXB609

EXIT ERROR: Return to IEFXB609

ENTRY POINT: SJXRETRY

PURPOSE:
Performs clean up processing when an ABEND occurs in the process of reading the checkpoint dataset.

LINKAGE: SYNCH

CALLERS: RTM

INPUT: Estae Parameter List

OUTPUT: None

EXIT NORMAL:

EXIT ERROR: Return to caller

EXTERNAL REFERENCES:

ROUTINES:
IEFXBMRG - Restart Generalized Merge Routine
IEFXBRDC - Restart Read Checkpoint Dataset Routine
IEFXBSWA - Restart SWA Manager Routine
IEFGB4DC - Dataset Reservation/Release
IEFEB4UV - Enhanced Unit Verification Interface

CONTROL BLOCKS:
ATCA - Allocation Communication Area
CVT - Communications Vector Table
JCT - Job Control Table
JFCB - Job File Control Block

IEFXBSJX - MODULE DESCRIPTION (Continued)

JFCBE - Job File Control Block Extension for 3800
JFCBX - Job File Control Block Extension
SCT - Step Control Table
SIOT - Step Input/Output Table

IEFXBSJX - MODULE OPERATION

IEFXBSJX processes the SIOTs, JFCBs, and JFCB extensions corresponding to the DSDR on the checkpoint dataset as follows:

1. If the type of processing requested is for SIOT/JFCB processing, this routine does the following:

   a. The SIOT fields in the DSDR are merged to the SIOT in SWA,
   b. The DSDR copy of the JFCB is merged to the SWA version of the JFCB by invoking routine IEFXBMRG. After the merge by template, several fields in the JFCB are merged seperately.
   c. If the JFCB indicates that any JFCBEs or JFCBXs exist for this SIOT/JFCB, the DSDR copy of the SWA block is read from the checkpoint dataset and IEFXBMRG is invoked to merge the DSDR copy of the JFCB extension to the SWA copy.

2. If the type of processing requested is for the initialization of a new SIOT/JFCB, this routine does the following:

   a. A new SWA block will be assigned for the SIOT and JFCB.
   b. The fields which would not be initialized by normal SIOT/JFCB merge processing are initialized.
   c. The newly assigned and initialized SIOT and JFCB are then written to SWA.

3. Return to caller

RECOVERY OPERATION:

If an abend occurs in this module, the recovery point in IEFXB609 receives control from RTM. The recovery routine specifies to RTM the retry address SJXRETRY in the checkpoint restart workarea. When SJXRETRY receives control from RTM, it does the following:

1. Sets the return code to decimal 36 to indicate a checkpoint restart system error.

2. Performs the necessary cleanup.

3. Returns to the caller.

IEFXBSJX - DIAGNOSTIC AIDS

ENTRY POINT NAMES: IEFXBSJX
SJXRETRY

MESSAGES: None

ABEND CODES:

00B - Scheduler Restart Local Storage Error
Associated Reason Codes:
1 - A Scheduler Restart module failed to obtain enough storage from the preallocated storage area.
2 - A Scheduler Restart module attempted to free an area outside the preallocated storage area.

WAIT STATE CODES: None

RETURN CODES:

ENTRY POINT IEFXBSJX:

EXIT NORMAL:

(decimal)
Register 15 = 0 - Request completed successfully
Reason Code in CRWAREAS:
CRRCNOER (0) - no error

EXIT ERROR:

(decimal)
Register 15 = 4 - Request processed unsuccessfully
Reason Code in CRWAREAS:
CRRCDUMY (031) - SWA SIOT indicates that the DD is not a dummy and DSDR indicates the dataset is a dummy
CRRCEVIO (239) - Deferred restart and VIO dataset
CRRCXSJX (320) - Modules does not support the function requested
CRRCUVLU (321) - Error return from IEFEB4UV for a request to convert a device type to a look-up value.
CRRCXMRG (420) - Error occured while processing a merge request.

ENTRY POINT SJXRETRY:

EXIT ERROR:

Register 15 = 36 - Restart system error

REGISTER CONTENTS ON ENTRY:

ENTRY POINT IEFXBSJX:

Register 0 = Undefined
Register 1 = Address of a two word parameter list which contains the address of the checkpoint restart workarea and the address of the function code.
Registers 2-12 = Undefined
Register 13 = Address of 18-word save area

IEFXBSJX - DIAGNOSTIC AIDS (Continued)

Register 14 = Return address
Register 15 = Entry point address

ENTRY POINT SJXRETRY:

Register 0 = Undefined
Register 1 = Address of the ESTAE parameter list
Registers 2-14 = Undefined
Register 15 = Entry point address

REGISTER CONTENTS ON EXIT:

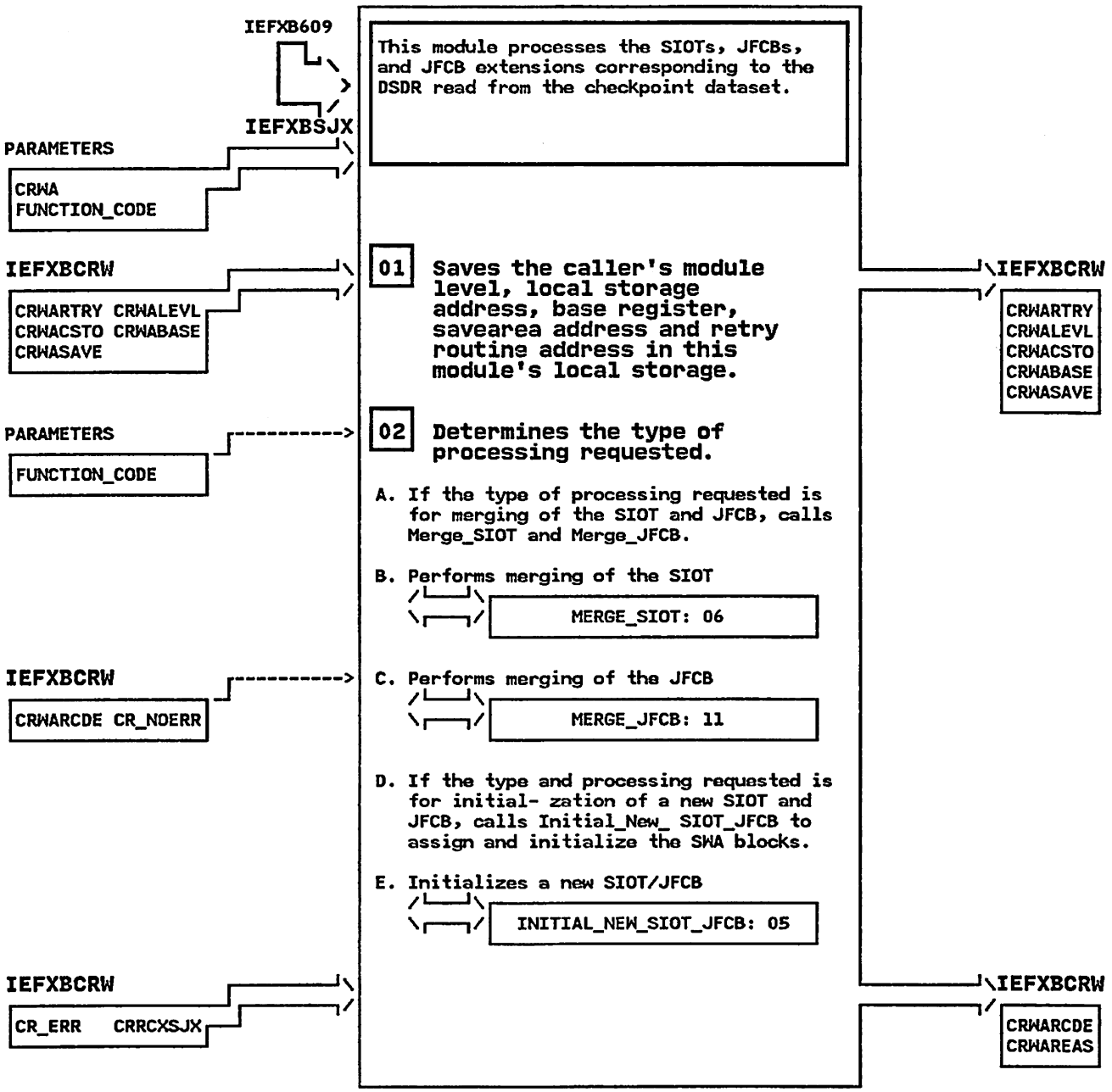
ENTRY POINT IEFXBSJX:

Register 0-14 = Restored
Register 15 = Return Code

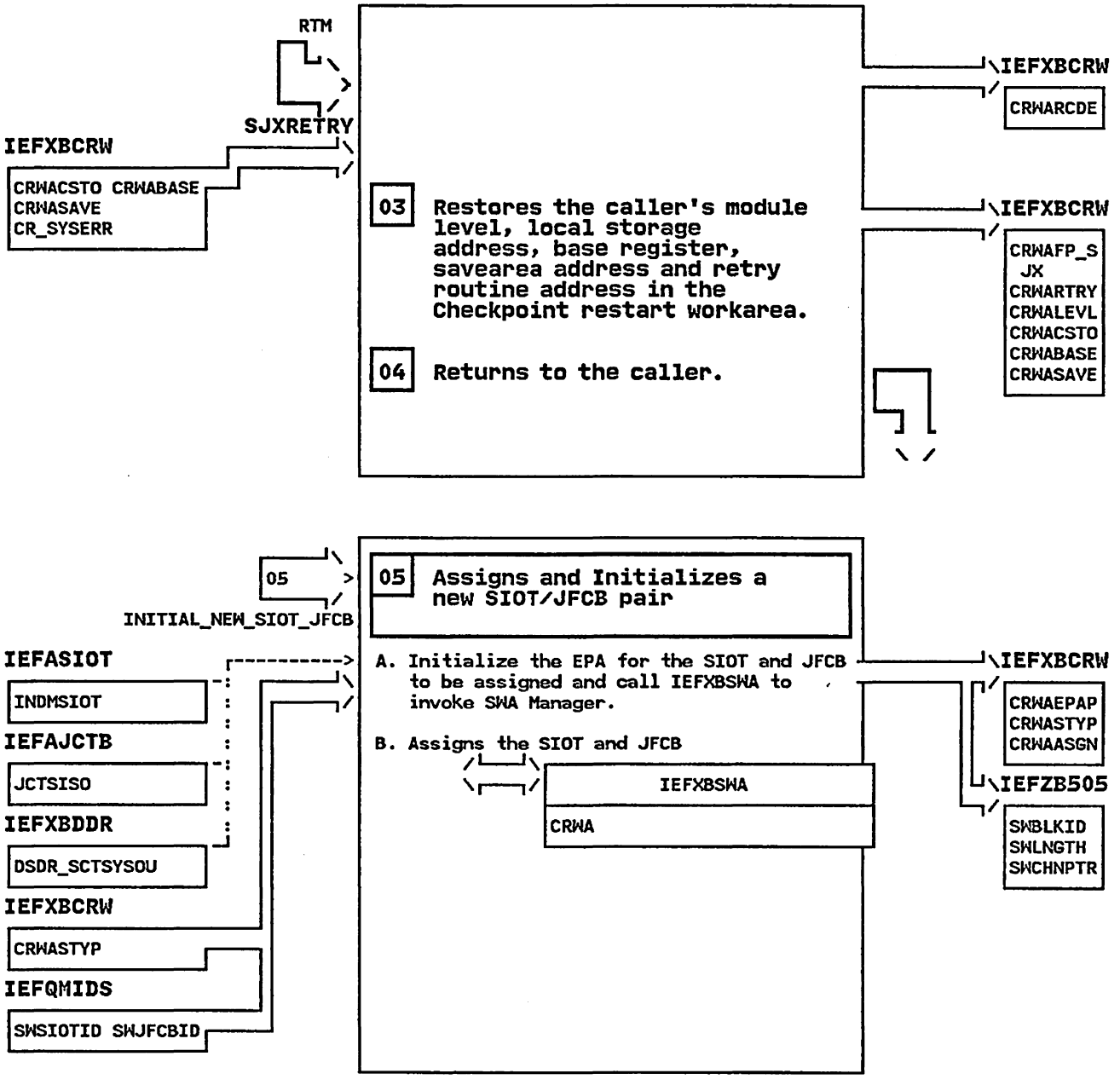
ENTRY POINT SJXRETRY:

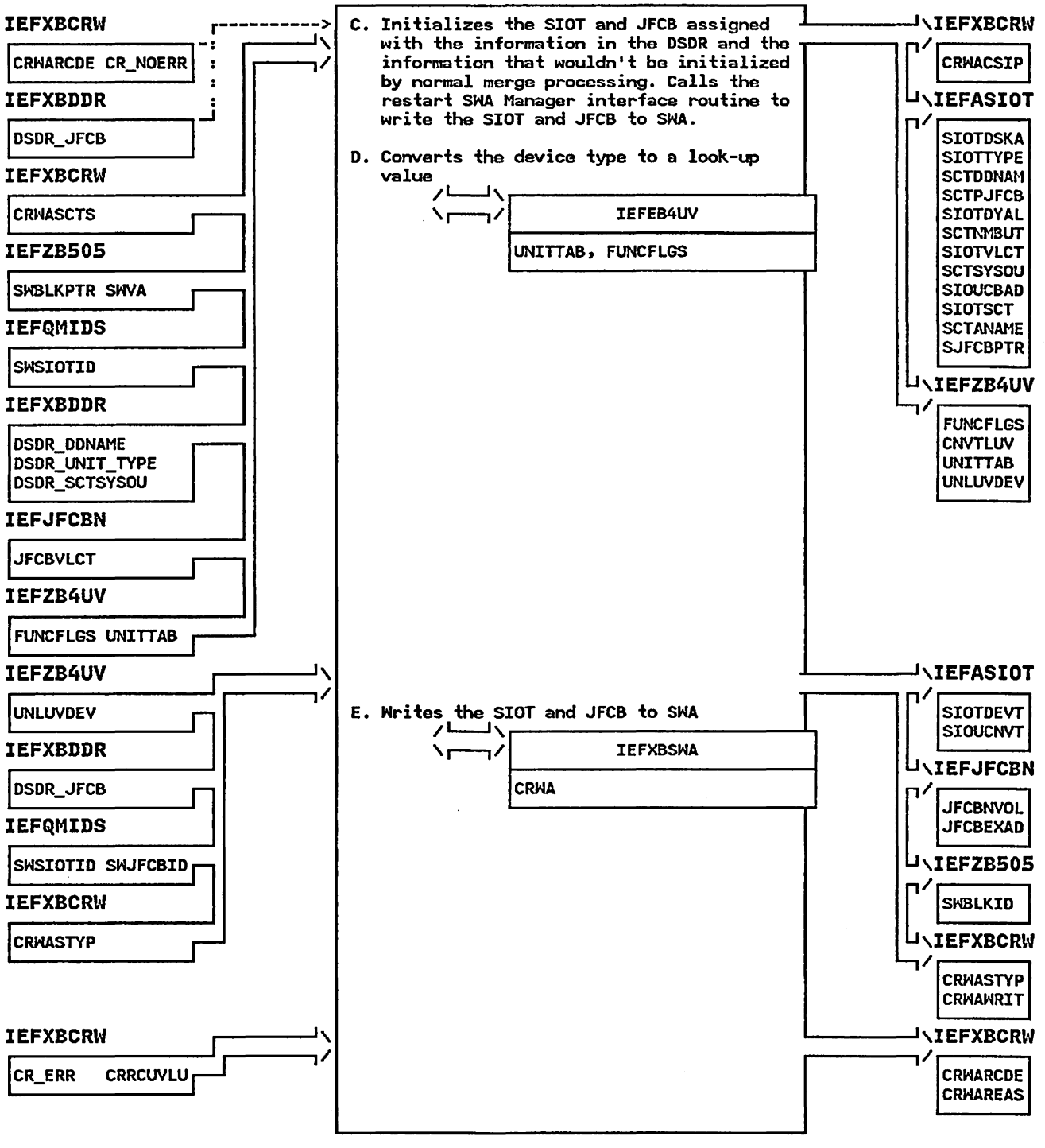
Registers 0-14 = Restored
Register 15 = Return code

IEFXBSJX - Restart SIOT/JFCB and Extension Processing Routine STEP 01

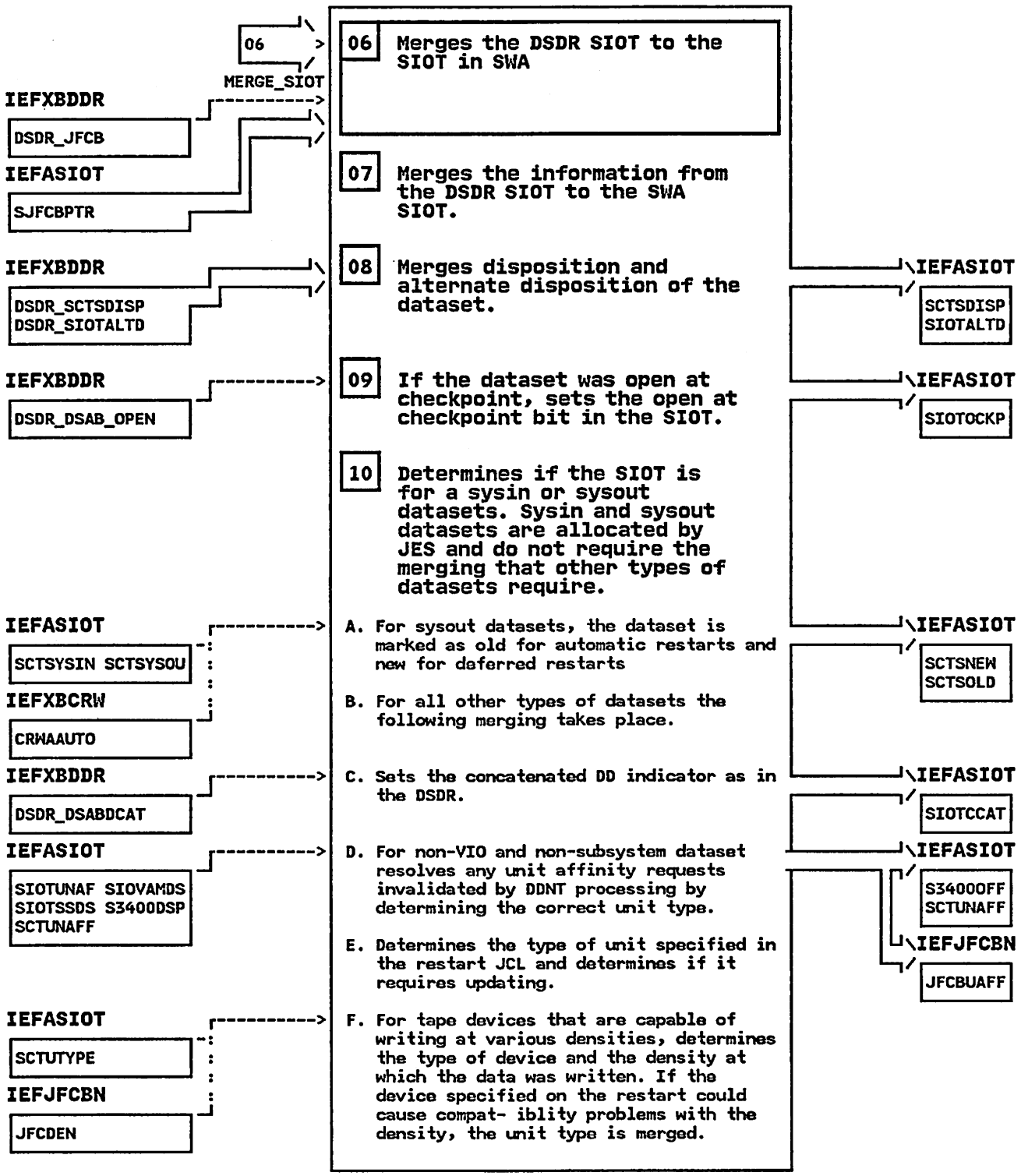


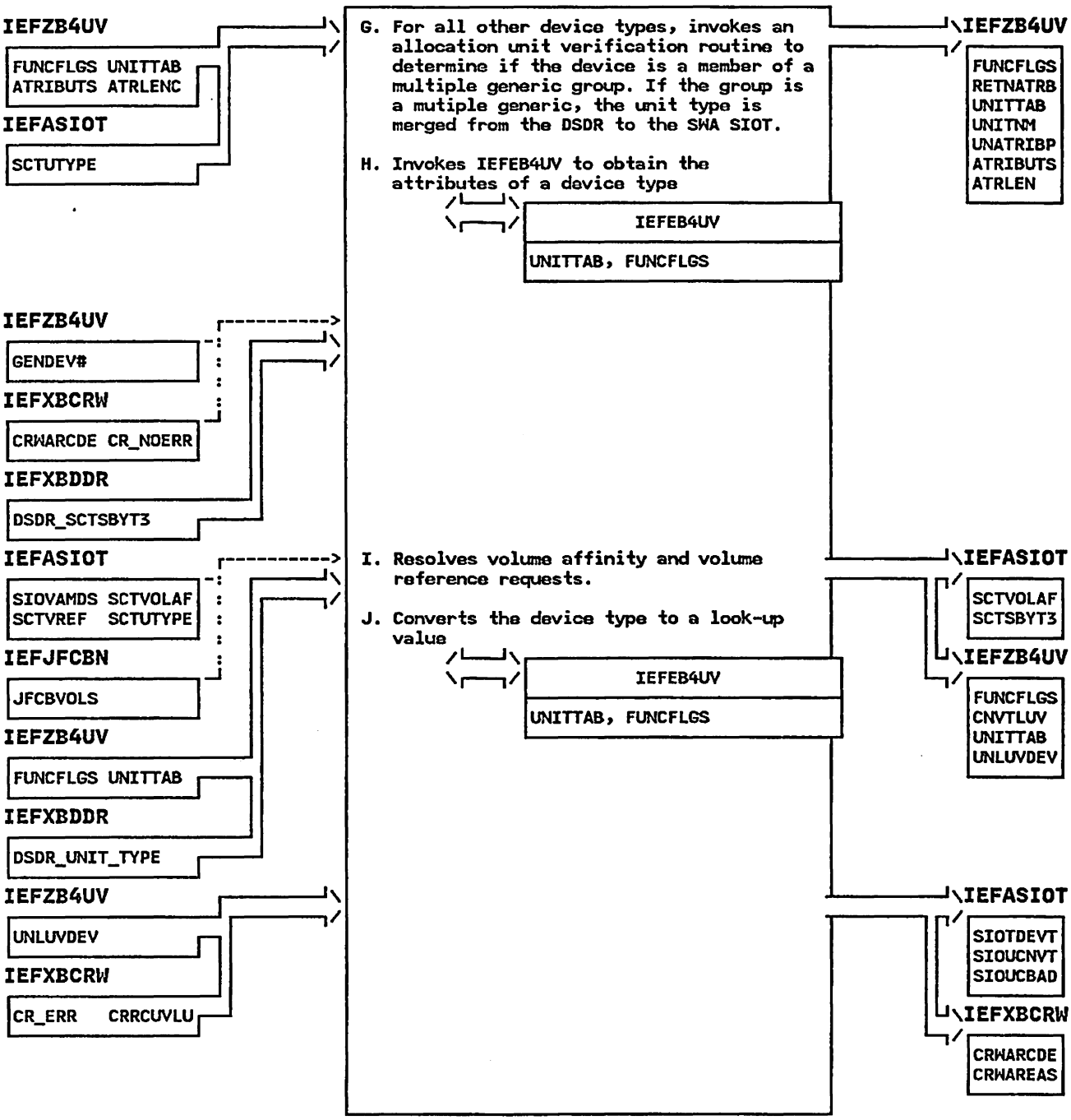
IEFXBSJX - Restart SIOT/JFCB and Extension Processing Routine STEP 03



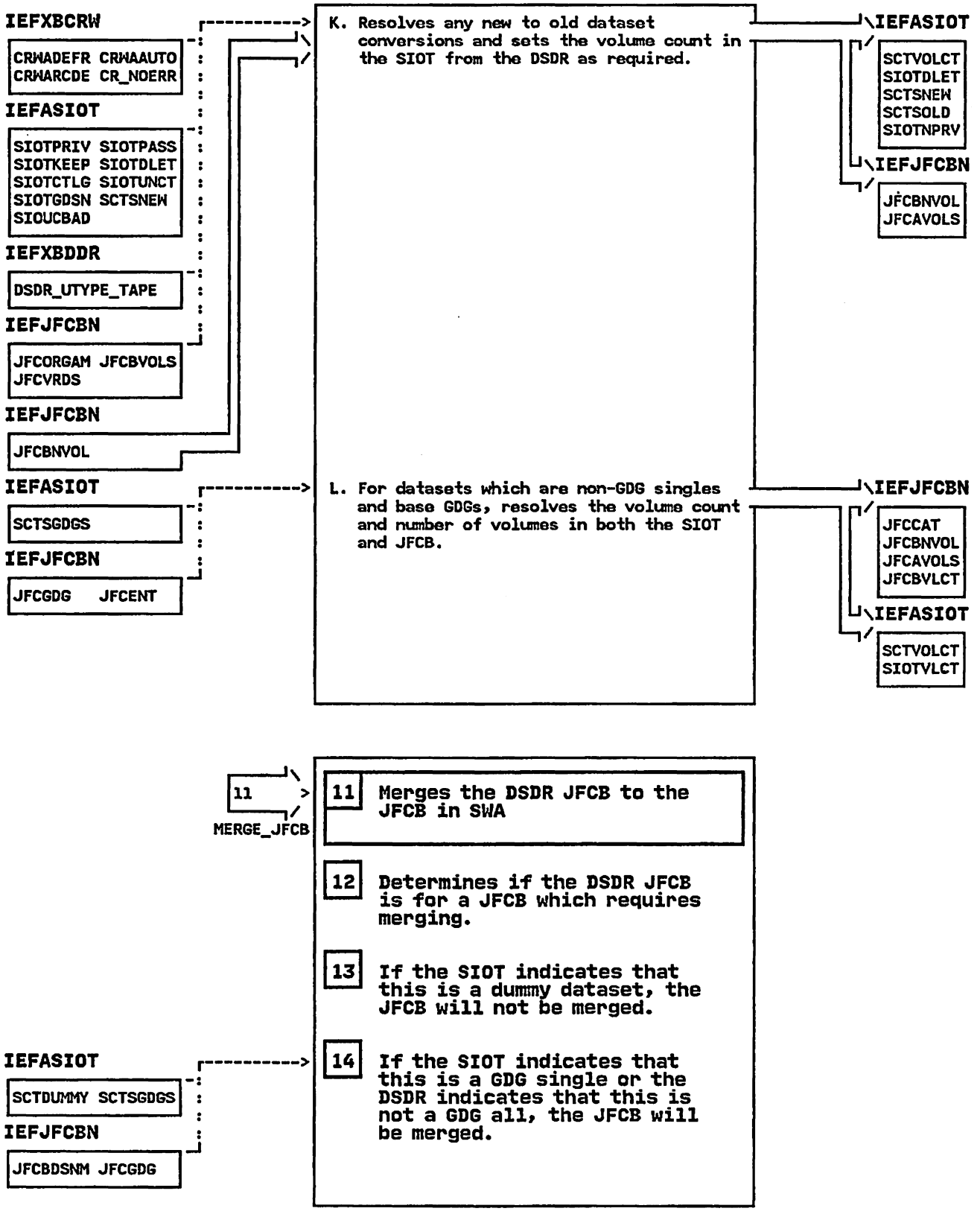


IEFXBSJX - Restart SIOT/JFCB and Extension Processing Routine STEP 06

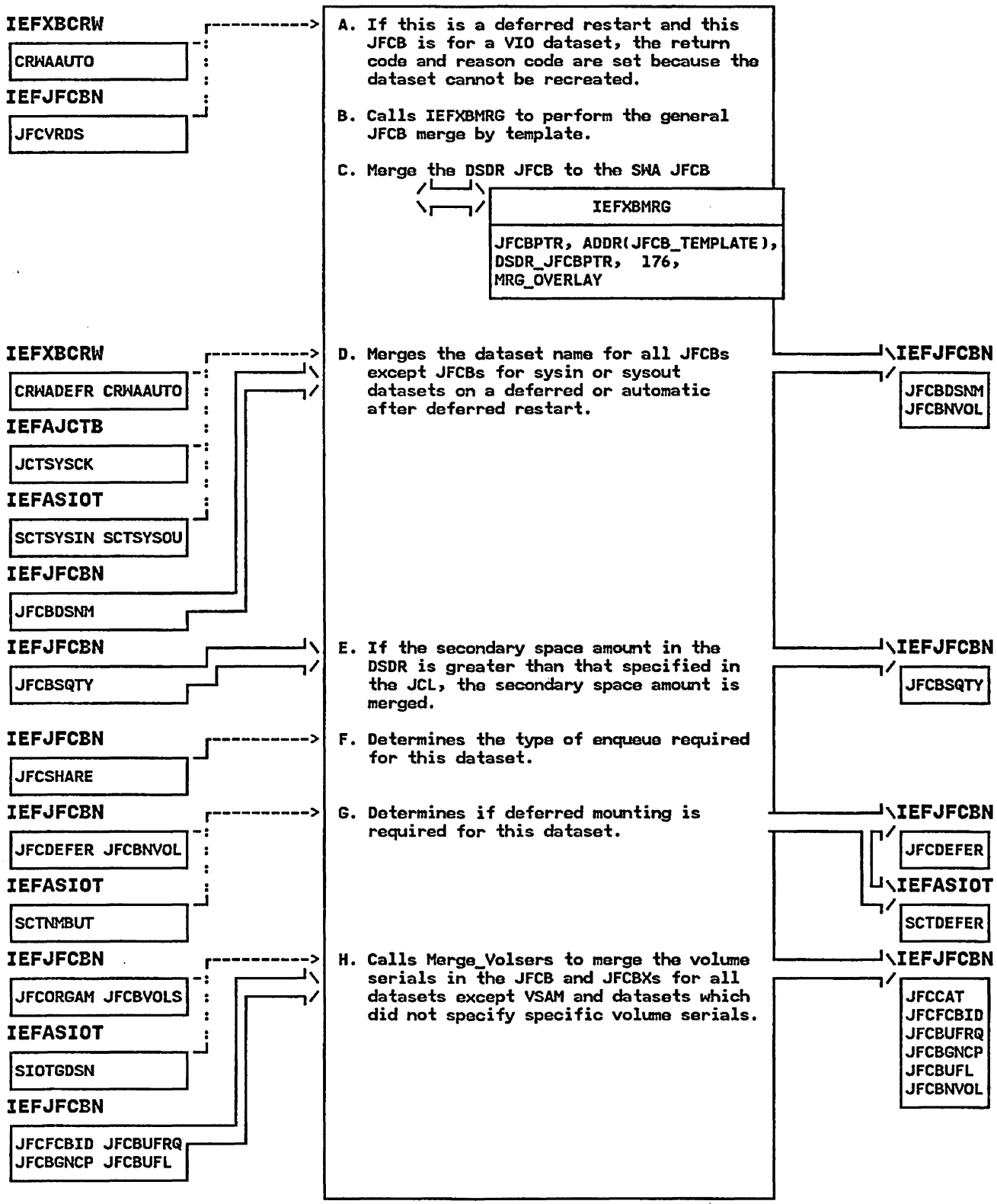




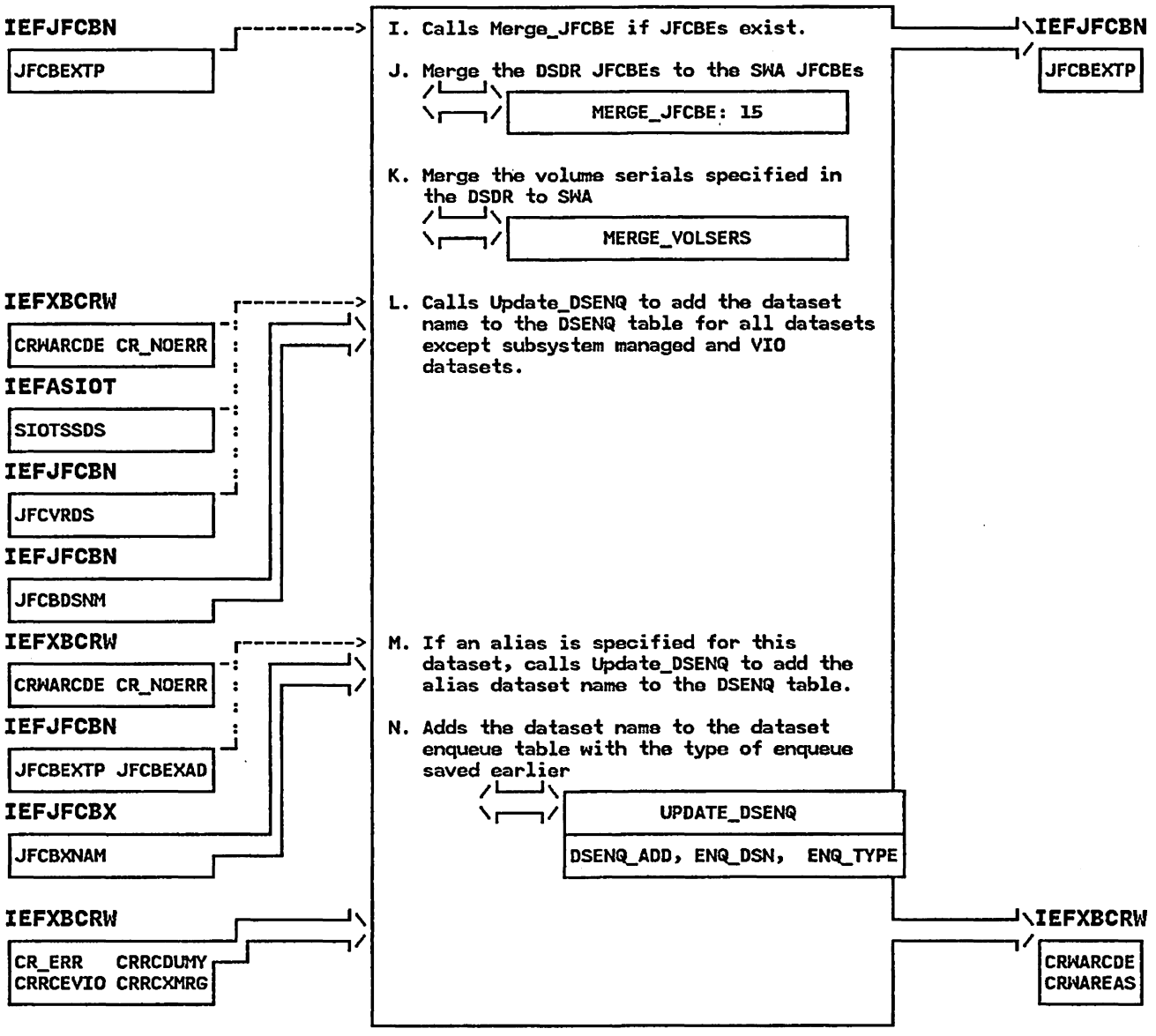
IEFXBSJX - Restart SIOT/JFCB and Extension Processing Routine STEP 10K



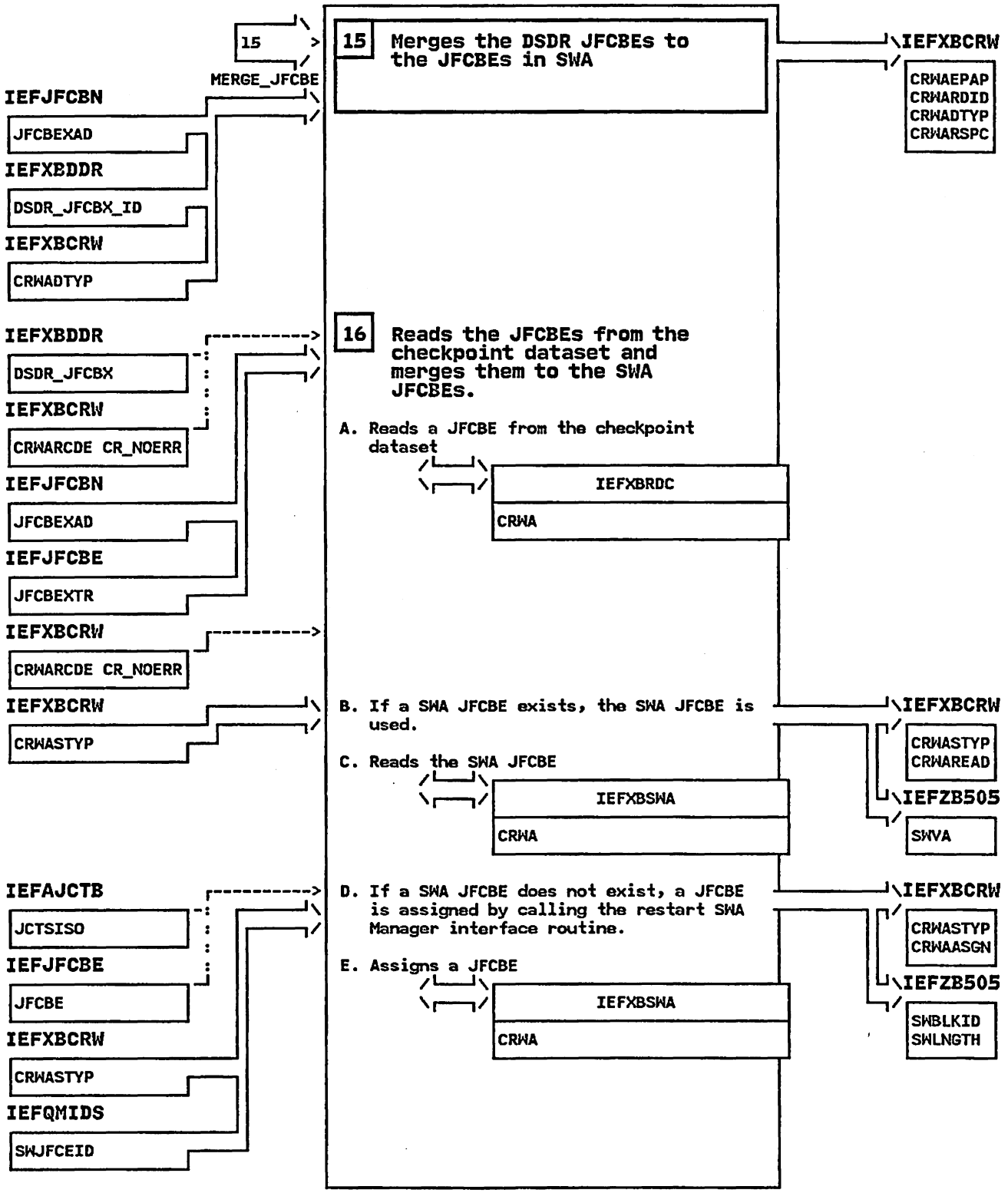
IEFXBSJX - Restart SIOT/JFCB and Extension Processing Routine STEP 14A



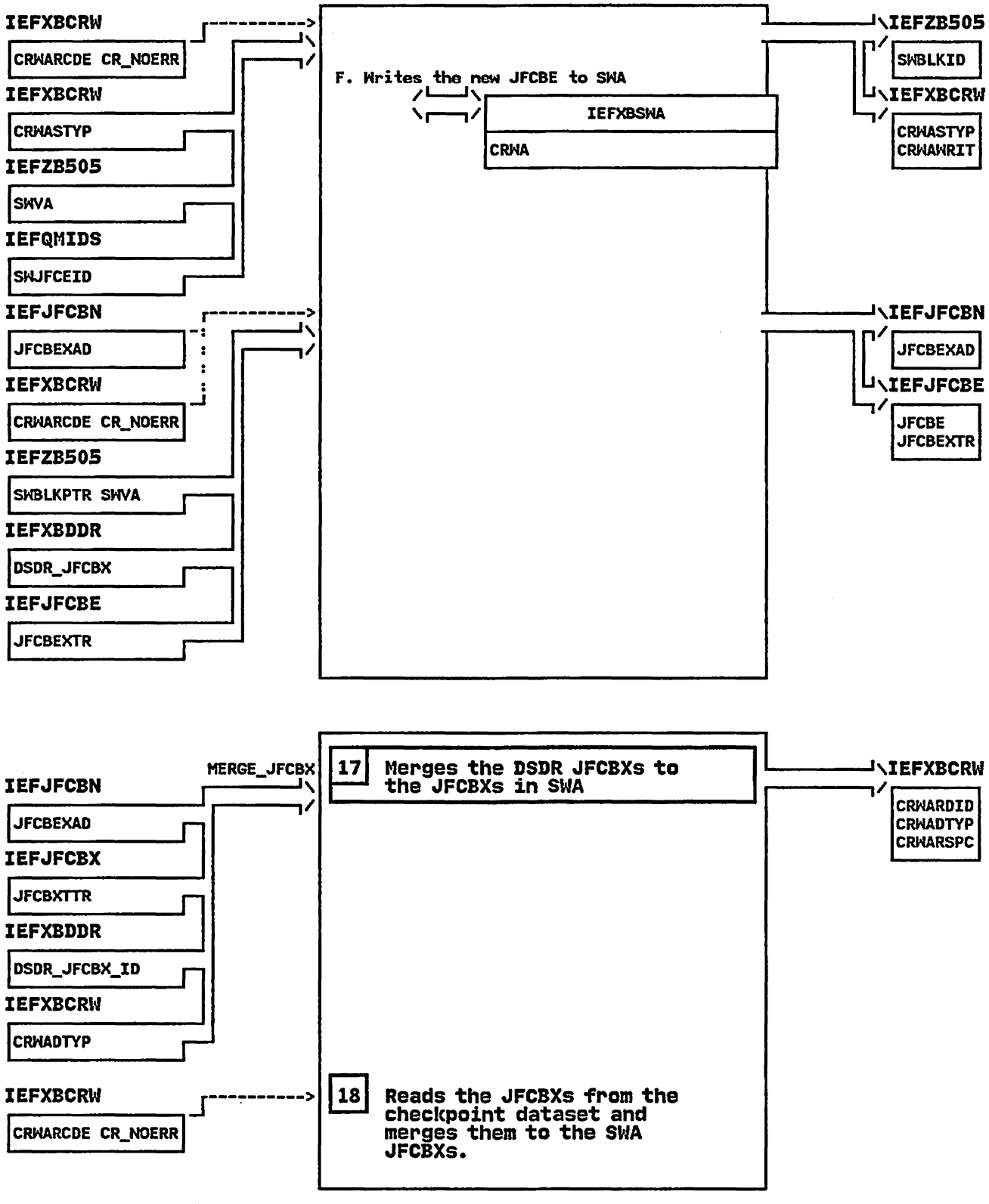
IEFXBSJX - Restart SIOT/JFCB and Extension Processing Routine STEP 14I



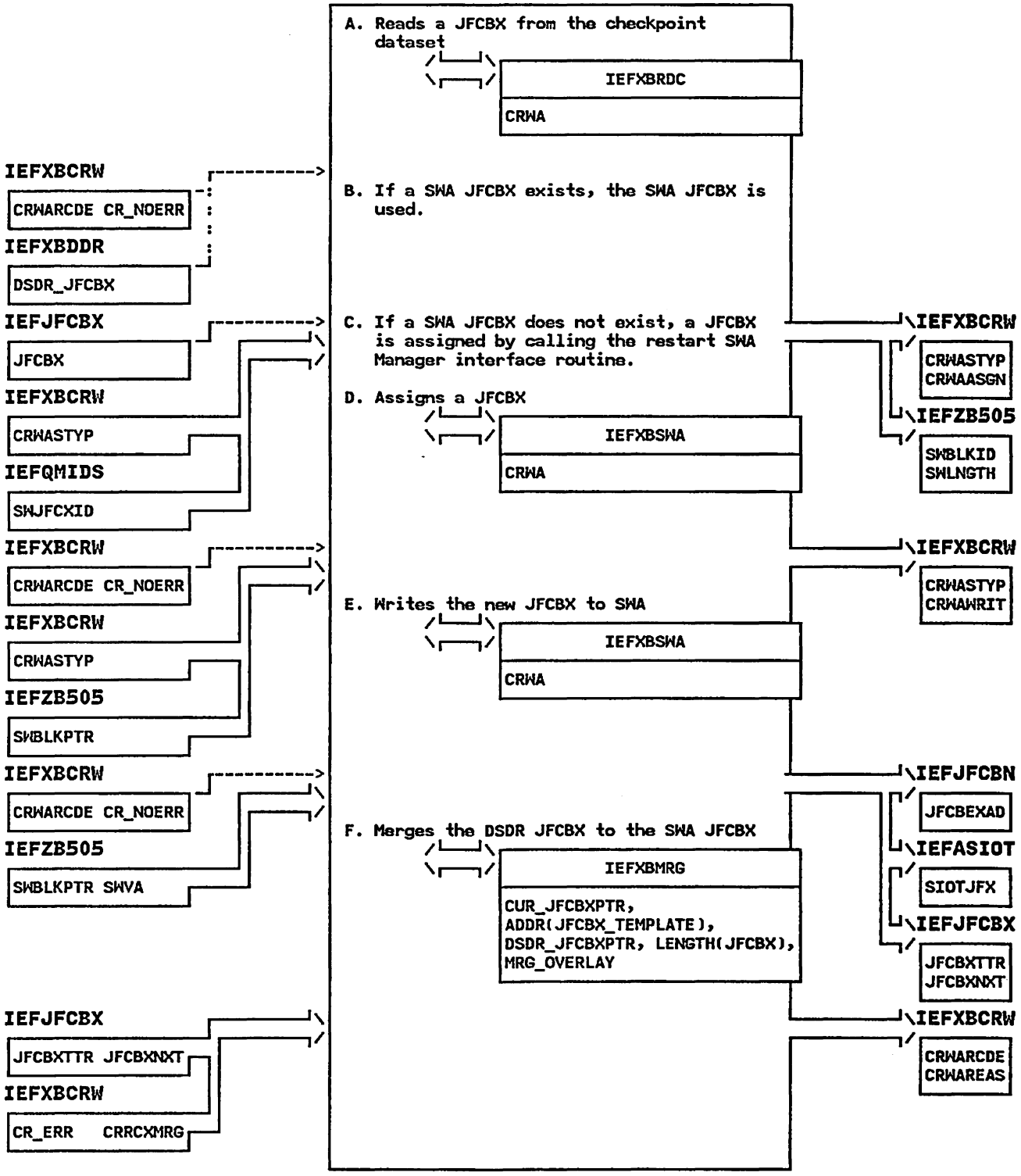
IEFXBSJX - Restart SIOT/JFCB and Extension Processing Routine STEP 15



IEFXBSJX - Restart SIOT/JFCB and Extension Processing Routine STEP 16F



IEFXBSJX - Restart SIOT/JFCB and Extension Processing Routine STEP 18A



IEFXBUSJ - MODULE DESCRIPTION

DESCRIPTIVE NAME: Restart Unmatched SIOT/JFCB processing Routine.

FUNCTION:
This module processes SIOT/JFCB pairs in SWA that did not have a corresponding checkpoint DSDR record. These DDs may represent the following:

For deferred restarts - DDs added to the JCL

For automatic restarts -
System generated DD's (ie. GDG 'all' cases or private catalogs, these are not merged by checkpoint/restart: they will be recreated by allocation.)
DD's dynamically allocated (created and possibly updated) after the checkpoint was taken. (See following description)

The SIOT/JFCB in SWA is altered by three different methods for automatic restarts:

1. Interpreter recreation of the original JCL
2. Journal Merge (IEFXB601) may update or add DDs in SWA.
3. Checkpoint/restart will recreate the DDs as they looked at the time of the checkpoint ('un-does' some of journal merge processing)

Due to the way the processing is done, dynamic allocations which follow the checkpoint will be placed back into SWA by Journal Merge (IEFXB601). These DDs represent resources that were "allocated" to the job by code following the checkpoint. Since the job will be restarted at the instruction following the checkpoint these resources must be freed in order to place the job into the same state as it was in at the time of the checkpoint.
In order to free these resources a call is made to the allocation routine IEFAB4A0 - common unallocation to perform disposition processing. This routine will scratch/uncatalog these resources so that they may be reused.

ENTRY POINT: IEFXBUSJ

PURPOSE: See Function

LINKAGE: Call

CALLERS: IEFXB609

INPUT:
The checkpoint restart workarea, specifically the following fields:

FIELD	LENGTH/MASK	DESCRIPTION
CRWACSIP	4	Pointer to current SIOT (last matched SIOT at entry)
CRWALSIP	4	Pointer to last SIOT in SIOT chain

OUTPUT:

FIELD	LENGTH/MASK	DESCRIPTION
CRWARCDE	4	Return code

IEFXBUSJ - MODULE DESCRIPTION (Continued)

CRWAREAS 3 Reason code

Alters the SIOT/JFCB chain in SWA to include or exclude the leftover SIOTs.

Frees unneeded resources (ie. uncatalogs and scratches the unneeded datasets via IEFAB4A0)

EXIT NORMAL: Return to IEFXB609

EXIT ERROR: Return to IEFXB609

ENTRY POINT: USJRETRY

PURPOSE:
Performs clean up processing when an ABEND occurs in the process of resolving unmatched SIOTs.

LINKAGE: SYNCH

CALLERS: RTM

INPUT: Estae Parameter List

OUTPUT: None

EXIT NORMAL:

EXIT ERROR: Return to caller

EXTERNAL REFERENCES:

ROUTINES:
IEFAB4A0 - Common unallocation for disposition processing

DATA AREAS: IEFXBCRW - Checkpoint Restart Work Area

CONTROL BLOCKS:
JSCB - Job Step Control Block
TCB - Task Control Block
PSA - Prefixed Save Area
SCT - Step Control Table
SIOT - Step Input/Output Table

IEFXBUSJ - MODULE OPERATION

IEFXBUSJ processes unmatched SIOT/JFCB pairs

1. IEFXBUSJ determines if this is a deferred restart. For a deferred restart the DD's left are treated as being added to the JCL. The following is done:
   - Update the last SIOT pointer to point to the last SIOT.
2. For an automatic restart, loop through the unmatched SIOT's and do the following:
   - If this DD represents a new dataset or a converted from new dataset and it was dynamically allocated and the dataset is not a subsystem dataset then does the following:
     - Fill in the parameter list to IEFAB4A0 common unallocation and call this routine.
     - Check the return code from common unallocation and set the reason code.
3. Return to caller

RECOVERY OPERATION:
If an abend occurs in this module, the recovery point in IEFXB609 receives control from RTM. The recovery routine specifies to RTM the retry address USJRETRY in the checkpoint restart workarea. When USJRETRY receives control from RTM, it does the following:

1. Sets the return code to decimal 36 to indicate a checkpoint restart system error.
2. Performs the necessary cleanup.
3. Returns to the caller.

IEFXBUSJ - DIAGNOSTIC AIDS

ENTRY POINT NAMES: IEFXBUSJ
USJRETRY

MESSAGES: None

ABEND CODES:

00B - Scheduler Restart Local Storage Error
Associated Reason Codes:
1 - A Scheduler Restart module failed to obtain enough storage from the preallocated storage area.
2 - A Scheduler Restart module attempted to free an area outside the preallocated storage area.

WAIT STATE CODES: None

RETURN CODES:

ENTRY POINT IEFXBUSJ:

EXIT NORMAL:

(decimal)
Register 15 = 0 - Request completed successfully
Reason Code in CRWAREAS:
CRRCNOER (0) - no error

EXIT ERROR:

(decimal)
Register 15 = 4 - Request processed unsuccessfully
Reason Code in CRWAREAS:
CRRCXUSJ (380) - Error in processing a SIOT/JFCB that had no matching DSDR

ENTRY POINT USJRETRY:

EXIT ERROR:

(decimal)
Register 15 = 36 - Restart system error

REGISTER CONTENTS ON ENTRY:

ENTRY POINT IEFXBUSJ:

Register 0 = Undefined
Register 1 = Address of a one word parameter list which contains the address of the checkpoint restart workarea.
Registers 2-12 = Undefined
Register 13 = Address of 18-word save area
Register 14 = Return address
Register 15 = Entry point address

ENTRY POINT USJRETRY:

Register 0 = Undefined
Register 1 = Address of the ESTAE parameter list
Registers 2-14 = Undefined
Register 15 = Entry point address

IEFXBUSJ - DIAGNOSTIC AIDS (Continued)

REGISTER CONTENTS ON EXIT:

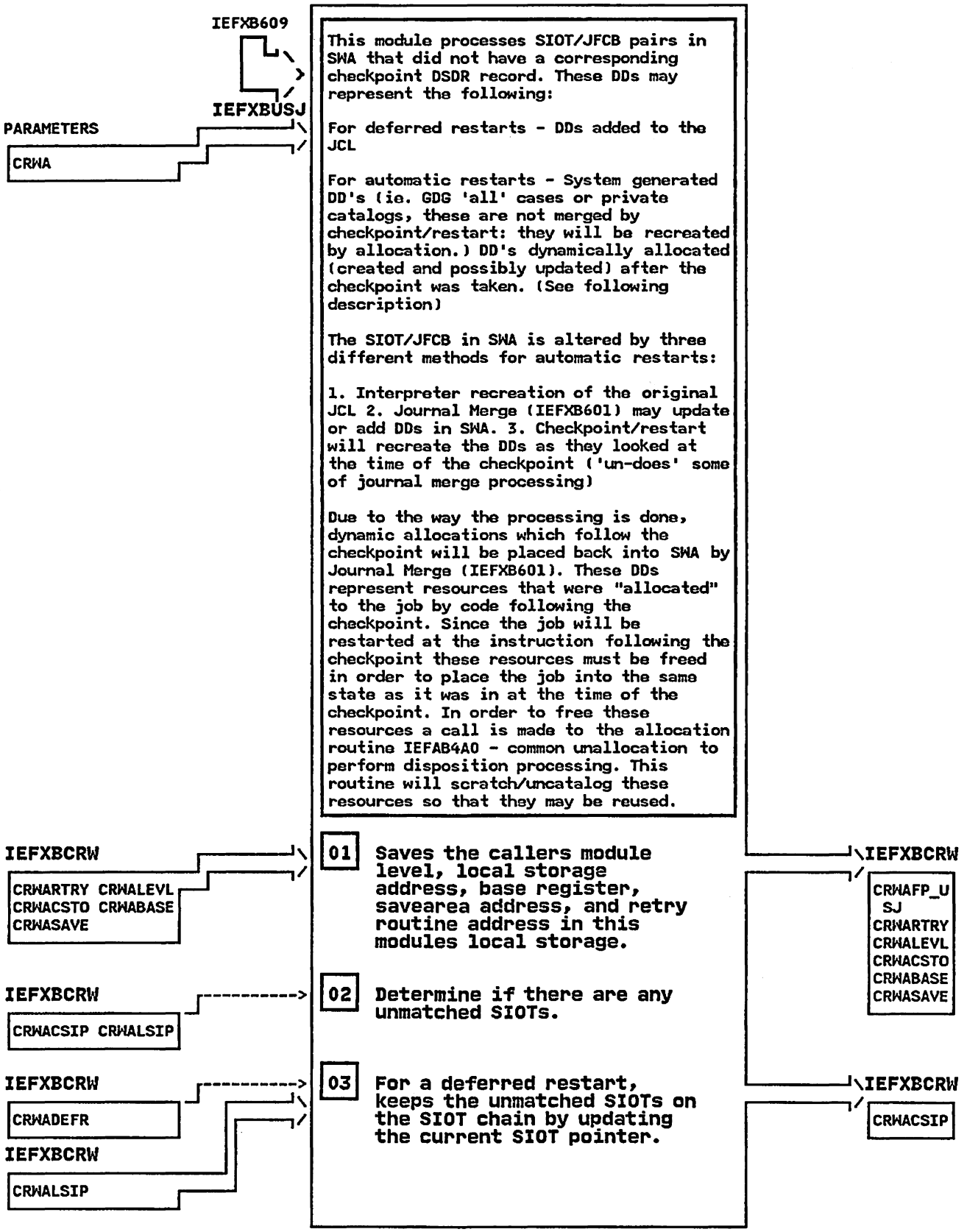
ENTRY POINT IEFXBUSJ:

Register 0-14 = Restored
Register 15 = Return Code

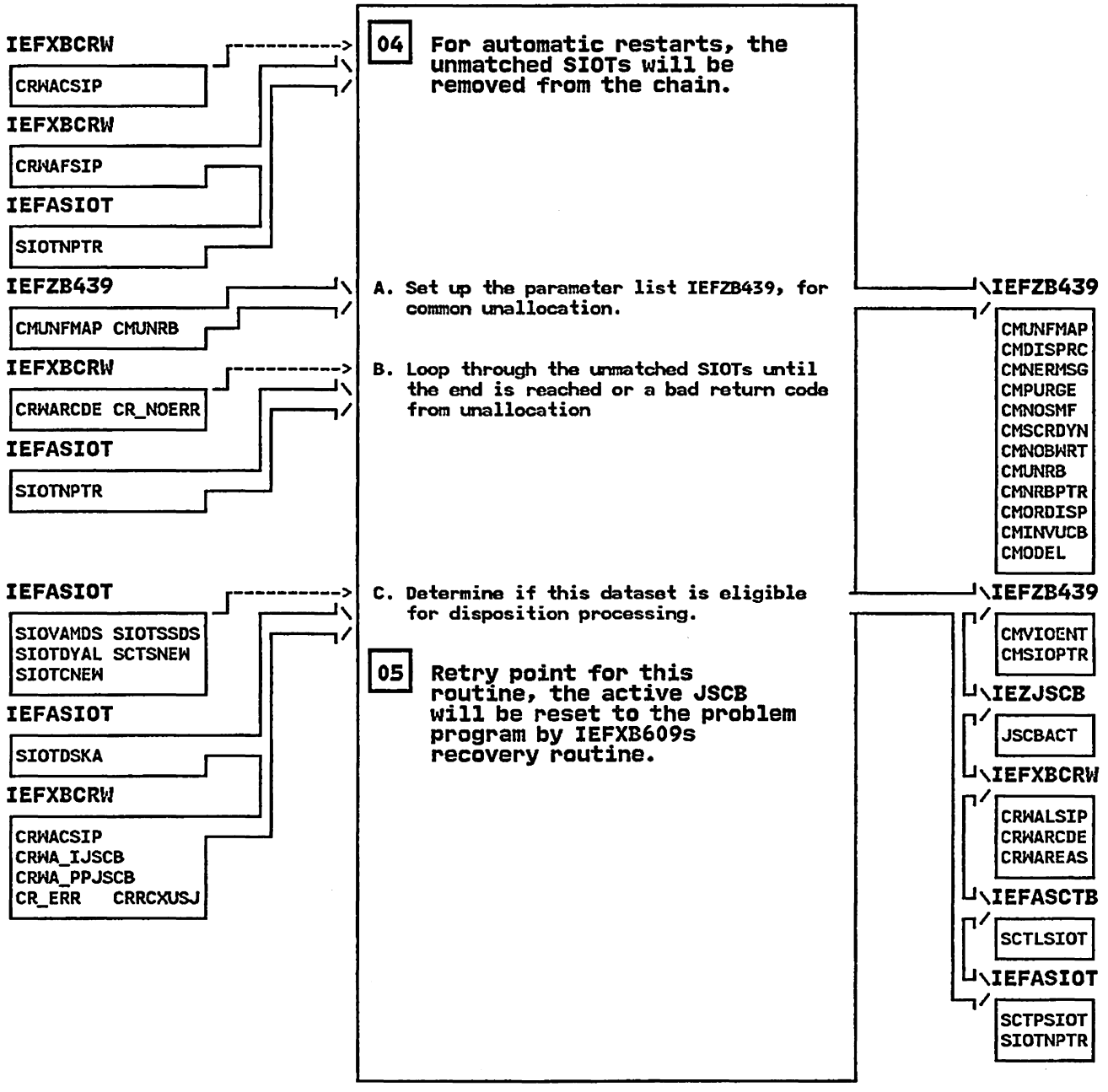
ENTRY POINT USJRETRY:

Registers 0-14 = Restored
Register 15 = Return code

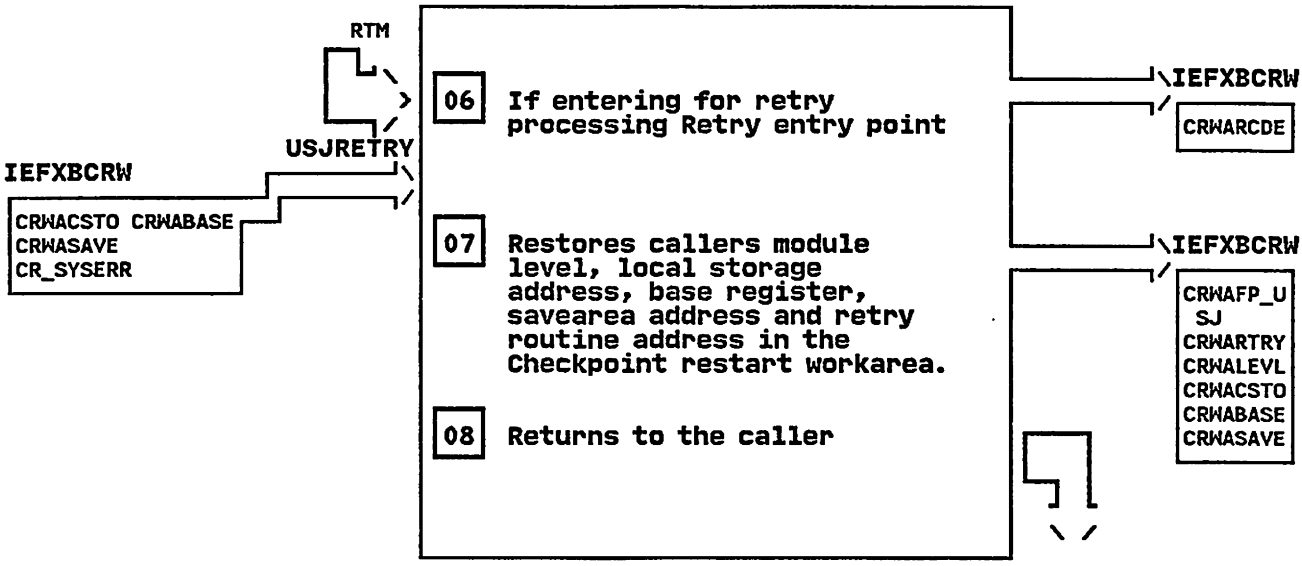
IEFXBUSJ - Restart Unmatched SIOT/JFCB processing Routine. STEP 01



IEFXBUSJ - Restart Unmatched SIOT/JFCB processing Routine. STEP 04



IEFXBUSJ - Restart Unmatched SIOT/JFCB processing Routine. STEP 06



IEFXB609 - MODULE DESCRIPTION

DESCRIPTIVE NAME: Checkpoint Restart Dataset Descriptor Record Processor Control Routine

FUNCTION:
This module performs common initial processing for the Checkpoint Restart processing routines, controls the processing of SWA information in the DSDR records on the checkpoint dataset to recreate the SWA environment at the time of a checkpoint for a restarting job and performs common cleanup processing.

ENTRY POINT: IEFXB609

PURPOSE: See Function

LINKAGE: Call

CALLERS: IEFIB605

INPUT: LCTPTR = address of LCT (Linkage Control Table)

OUTPUT:
SWA block environment reflects the environment at the time of the checkpoint.

EXIT NORMAL: Return to IEFIB605

EXIT ERROR: Return to IEFIB605

ENTRY POINT: RECOVERY

PURPOSE:
To recover from an error that caused the exit to RTM.

LINKAGE: SYNCH

CALLERS: RTM

INPUT:
Estae parameters
System diagnostic work area (SDWA)

OUTPUT: SVC dump and a record written in LOGREC data set.

EXIT NORMAL:
Return to RTM specifying the retry address stored in the Checkpoint Restart workarea.

EXIT ERROR:
Percolate to the caller's recovery routine if the abend did not occur while checkpoint restart was processing or a previous ABEND occurred.

ENTRY POINT: XB609RTY

PURPOSE:
To perform cleanup processing when an abend occurred during Checkpoint restart processing.

LINKAGE: SYNCH

CALLERS: RTM

INPUT: Estae parameters

IEFXB609 - MODULE DESCRIPTION (Continued)

OUTPUT: None

EXIT NORMAL:

EXIT ERROR: Return to IEFIB605

EXTERNAL REFERENCES:

ROUTINES:
- IEFXBDYS - Restart Dynamic SIOT Processing Routine
- IEFXBGDG - Restart GDGNT Processing Routine
- IEFXBRDC - Restart Read Checkpoint Dataset Routine
- IEFXBSJX - Restart SIOT/JFCB and Extension Processing Routine
- IEFXBSWA - Restart SWA Manager Routine
- IEFXBUSJ - Restart Unmatched SIOT/JFCB Processing Routine
- IEFXB610 - Restart Checkpoint Dataset Open Processor
- IEFAB4F5 - Allocate Catalog Control
- IEFGB4DC - Dataset Reservation/Release
- IEFDB476 - Dynamic Allocation Error Message Processor
- IEFEB4UV - Enhanced Unit Verification Interface

DATA AREAS:
- IEFXB603 - Restart Message Module
- IEFXBCRW - Checkpoint Restart Work Area
- - Checkpoint Dataset

CONTROL BLOCKS:
- ATCA - Allocation Communication Area
- CSCB - Command Scheduling Control Block
- CVT - Communications Vector Table
- DACA - Device Allocation Communication Area
- DCB - Data Control Block
- DSAB - Dataset Association Block
- GDA - Global Data Area
- GDGNT - GDG Name Table
- JCT - Job Control Table
- JESCT - Job Entry Subsystem Communications Table
- JFCB - Job File Control Block
- JFCBX - Job File Control Block Extension
- JSCB - Job Step Control Block
- LCT - Linkage Control Table
- PSA - Prefixed Save Area
- QDB - Queue Descriptor Block
- SCT - Step Control Table
- SCTX - Step Control Table Extension
- SDWA - System Diagnostic Work Area
- SIOT - Step Input/Output Table
- TCB - Task Control Block
- TIOT - Task Input/Output Table
- UCB - Unit Control Block

IEFXB609 - MODULE OPERATION

The following steps are involved in the process of merging the SWA information in the DSDR records on the checkpoint dataset to the SWA blocks in SWA:

1. Acquire the dynamic storage to be used by the DSDR processing and service routines of IEFXB609. Initialize the Checkpoint Restart Work Area to be used by the DSDR processing and service routines.

2. Establish the recovery environment.

3. Locate the SIOT and JFCB for the checkpoint dataset, dynamically allocate and open the checkpoint dataset.

4. If the type of restart is deferred, create the proper SCT structure. Ensure that the restart SVC gets control when the job is initiated.

5. Read the Checkpoint Header Record for this checkpoint ID, process the pertinent information contained in it and save the record for passing to the Restart SVC.

6. Read the checkpoint dataset to determine if any DDNAME Tables (DDNTs), representing JCL defined DDNAMEs that were dynamically unallocated prior to the checkpoint, exist. If any exist, the SIOTs representing these DDNAMEs will be removed from the SIOT chain.

7. Read the DSDRs on the checkpoint dataset, find the matching SWA block and call the appropriate processing routine to handle the DSDR.
   - IEFXBSJX for SIOTs, JFCBs, JFCBEs and JFCBXs.
   - IEFXBDYS for dynamically allocated SIOTs.
   - IEFXBGDG for GDGNTs.

8. If any unmatched SIOTs/JFCBs remain on the chain after processing all DSDRs, module IEFXBUSJ is called to process these blocks appropriately.

9. Build the restart parameter list.

10. Close the checkpoint dataset, cancel the recovery environment, free the local storage and perform any other cleanup as required.

11. Return to caller

RECOVERY OPERATION:

The recovery segment (RECOVERY) provides recovery for the Checkpoint Restart control and all processing routines as follows:

If this is the first ABEND:

1. Stores diagnostic information in the system diagnostic work area (SDWA).

2. Writes an entry in the LOGREC dataset and takes an SVC dump.

3. Specifies the retry address stored in the checkpoint restart workarea to RTM.

4. Returns to RTM.

IEFXB609 - MODULE OPERATION (Continued)

If a previous ABEND occurred:

1. Frees the local storage that was obtained.
2. Specifies that RTM percolate to the caller's recovery routine.
3. Returns to RTM.

If the error occurred in the Checkpoint Restart control routine, the retry segment (XB609RTY) in the Checkpoint Restart control routine receives control from RTM and does the following:

1. Sets the return code to indicate a Checkpoint Restart system error.
2. Cancels the recovery environment and frees the local storage that was obtained.
3. Returns to the caller.

IEFXB609 - DIAGNOSTIC AIDS

ENTRY POINT NAMES: IEFXB609
RECOVERY
XB609RTY

MESSAGES:

IEF006I jjj RESTARTING AT xxxxxx yyyyyy
IEF007I RESTART NOT SUCCESSFUL FOR jjj (xxx)
IEF010I CHECKPOINT RESTART OF JOB jjj ABENDED
IEF209I VIRTUAL STORAGE UNAVAILABLE FOR jjj.sss.ppp

The reason codes associated with message IEF007I are defined in the IEFXBCRW.

ABEND CODES: None

WAIT STATE CODES: None

RETURN CODES:

ENTRY POINT IEFXB609:

EXIT NORMAL:

(decimal)
Register 15 = 0 - DSDR records successfully merged from the checkpoint dataset to SWA.

EXIT ERROR:

(decimal)
Register 15 = 36
- Error return code from a processing routine
- Error return code from a supporting allocation routine
- Error while opening or reading the checkpoint dataset

ENTRY POINT RECOVERY:

EXIT NORMAL:

Register 15 = 4 - Retry to mainline cleanup processing

EXIT ERROR:

Register 15 = 0 - Do not retry

ENTRY POINT XB609RTY:

EXIT ERROR:

Register 15 = 36 (decimal)

REGISTER CONTENTS ON ENTRY:

ENTRY POINT IEFXB609:

Register 0 = Undefined
Register 1 = Address of a word that contains

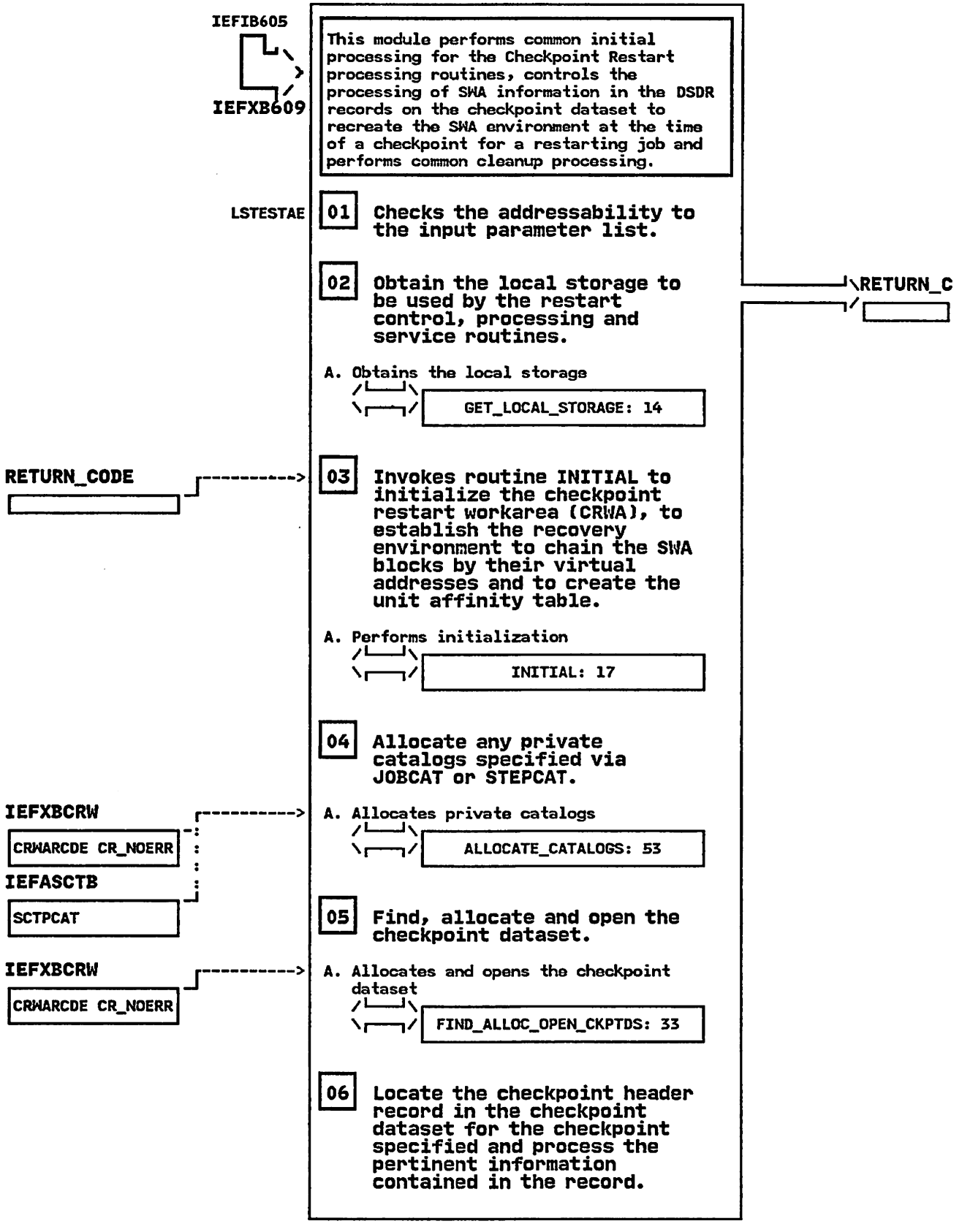
IEFXB609 - DIAGNOSTIC AIDS (Continued)

```
                    the address of the input
                    parameter list
   Registers 2-12 = Undefined
   Register  13   = Address of 18-word save area
   Register  14   = Return address
   Register  15   = Entry point address

 ENTRY POINT RECOVERY:

   Register  0    = Indicates whether a SDWA was obtained
   Register  1    = Pointer to the SDWA if a SDWA was
                    obtained
   Register  2    = Pointer to the ESTAE parameter list if
                    a SDWA was not obtained
   Registers 3-13 = Undefined
   Register  14   = Return address to RTM
   Register  15   = Entry point address

 ENTRY POINT XB609RTY:

   Register  0    = Undefined
   Register  1    = Address of the ESTAE parameter list
   Registers 2-14 = Undefined
   Register  15   = Entry point address
```

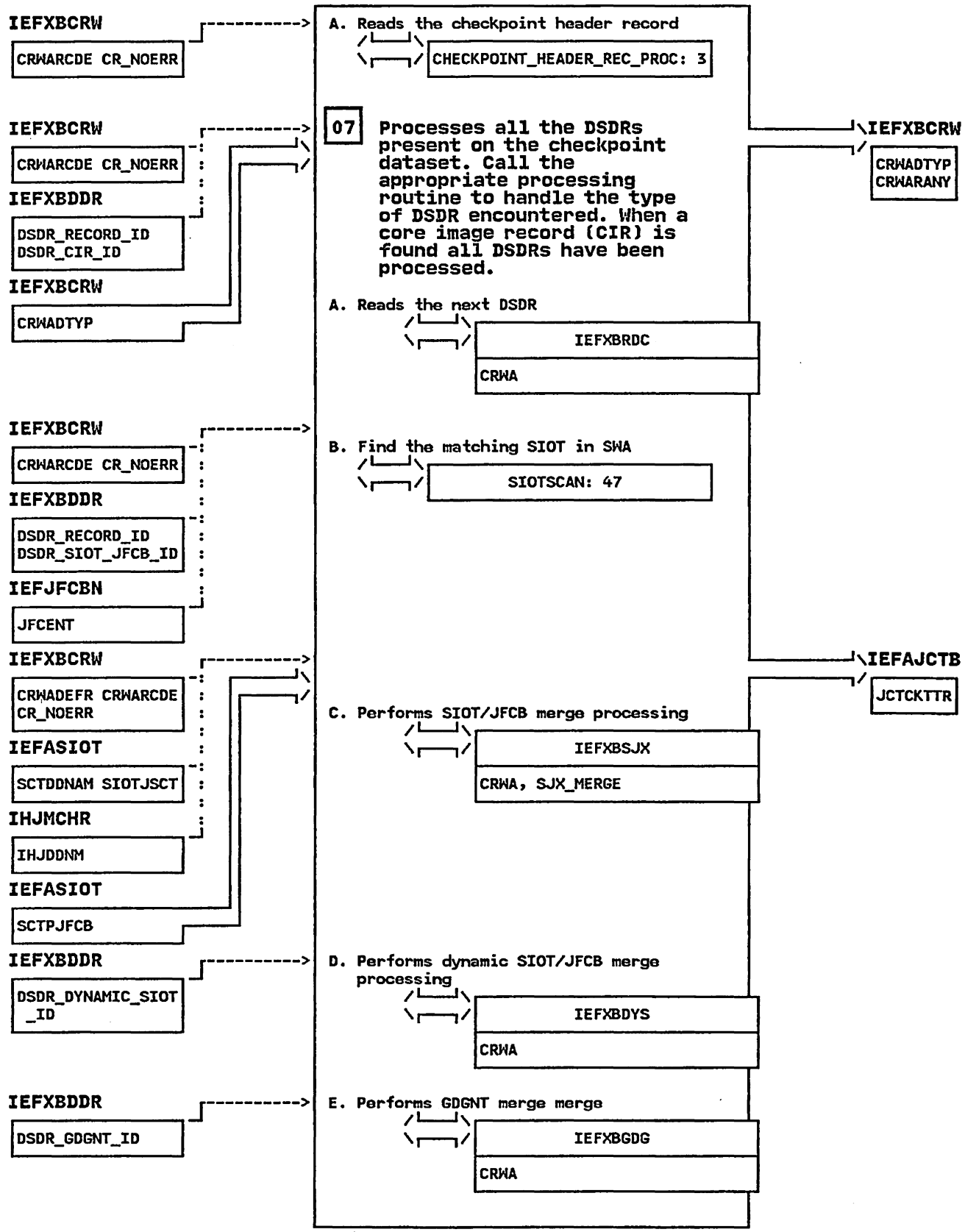
REGISTER CONTENTS ON EXIT:

```
 ENTRY POINT IEFXB609:

     Register  0-14 = Restored
     Register  15   = Return Code

 ENTRY POINT RECOVERY:

     Registers 0-13 = Undefined
     Register  14   = Return address
     Register  15   = Retry address

 ENTRY POINT XB609RTY:

     Registers 0-14 = Restored
     Register  15   = Return code
```

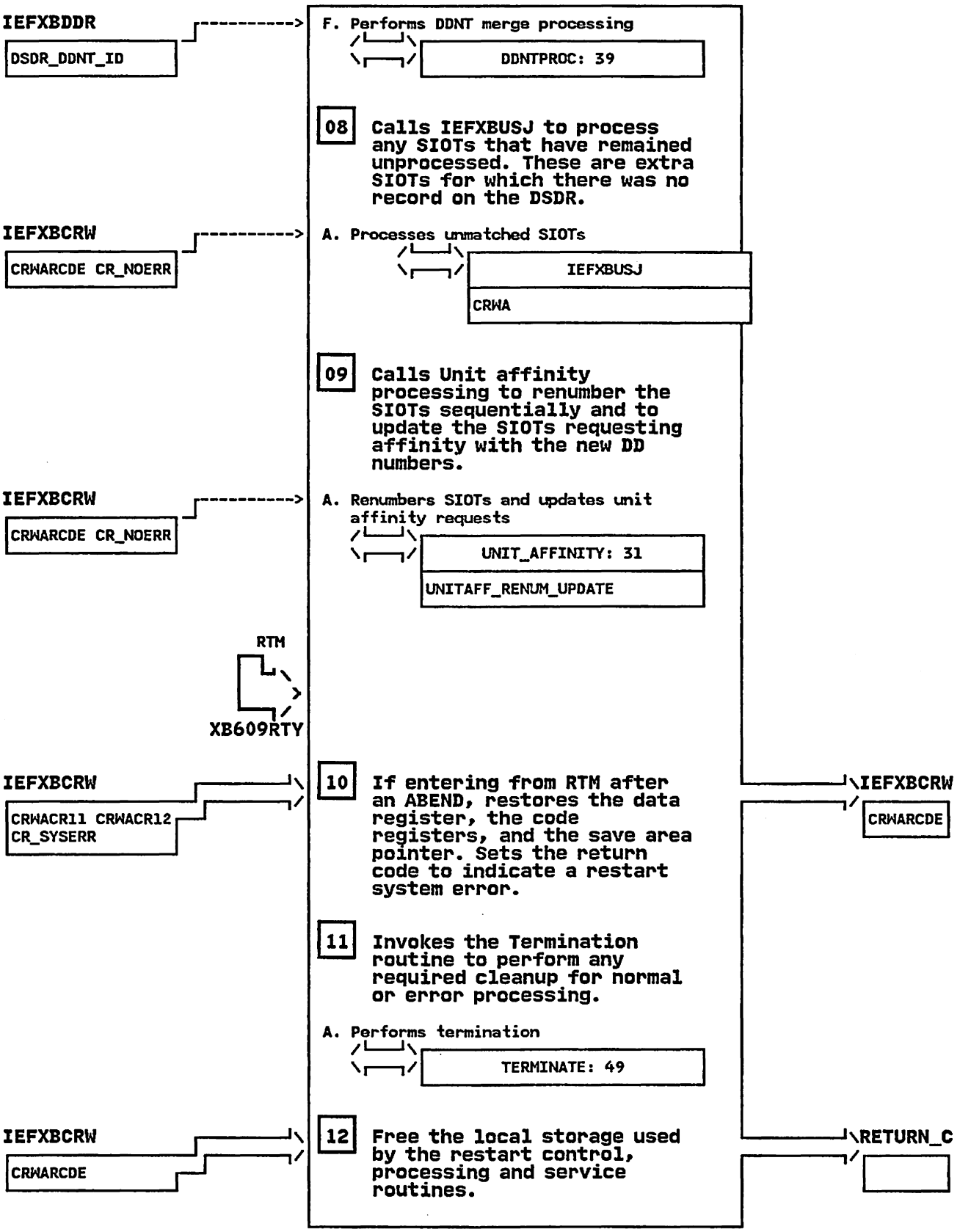
IEFXB609 - Checkpoint Restart Dataset Descriptor Record Processor ContSTEP 01

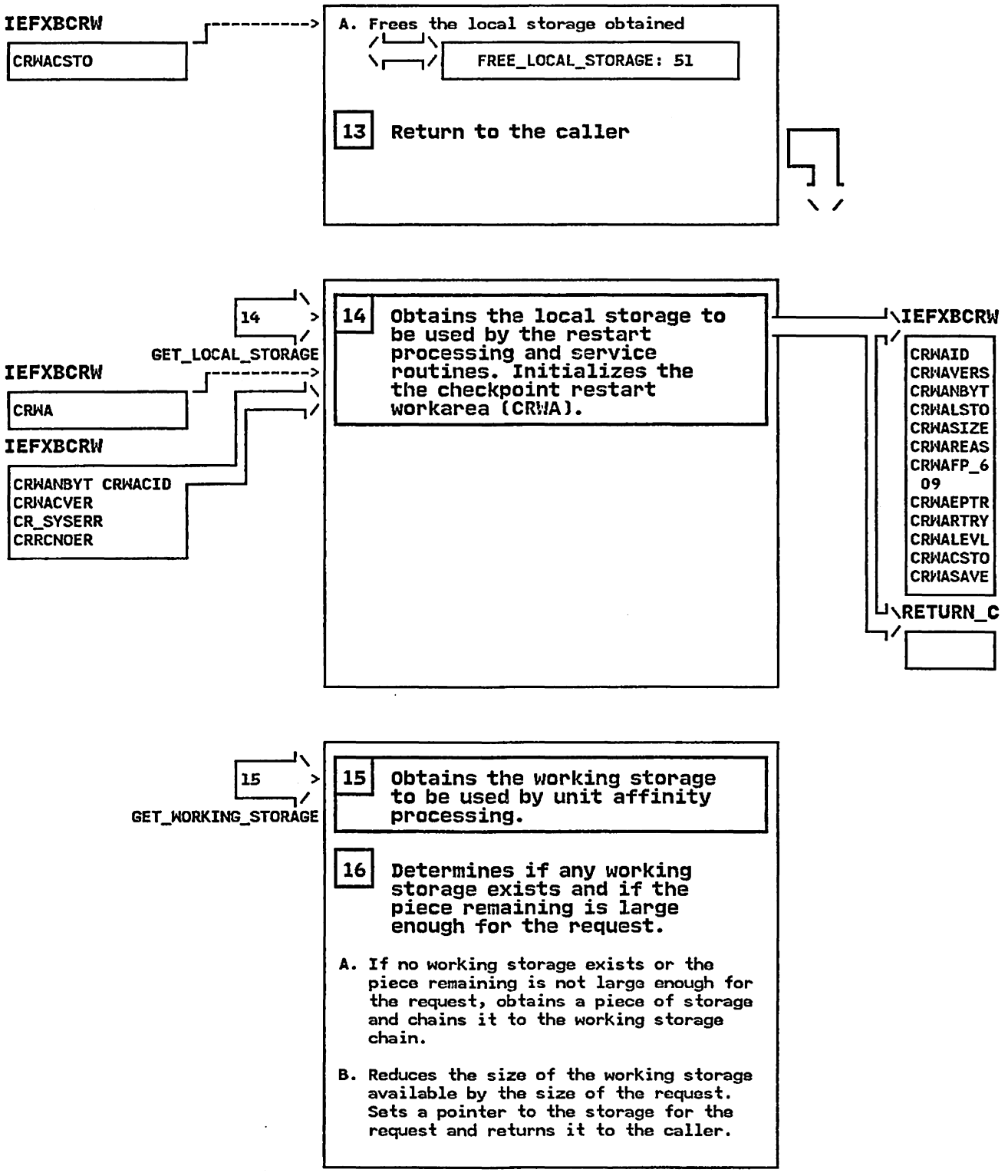


IEFXB609 - Checkpoint Restart Dataset Descriptor Record Processor ContSTEP 06A

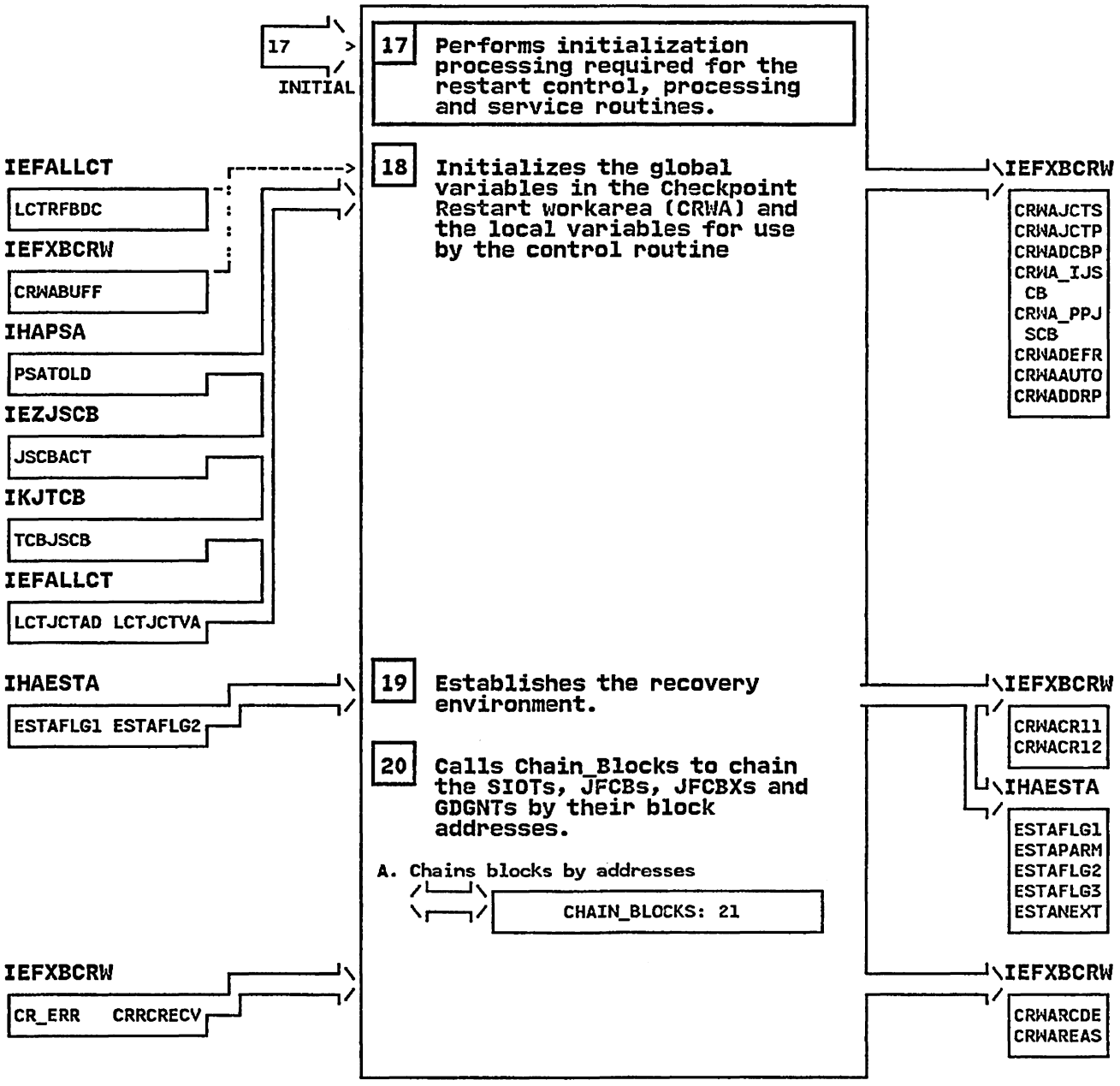


IEFXB609 - Checkpoint Restart Dataset Descriptor Record Processor ContSTEP 07F

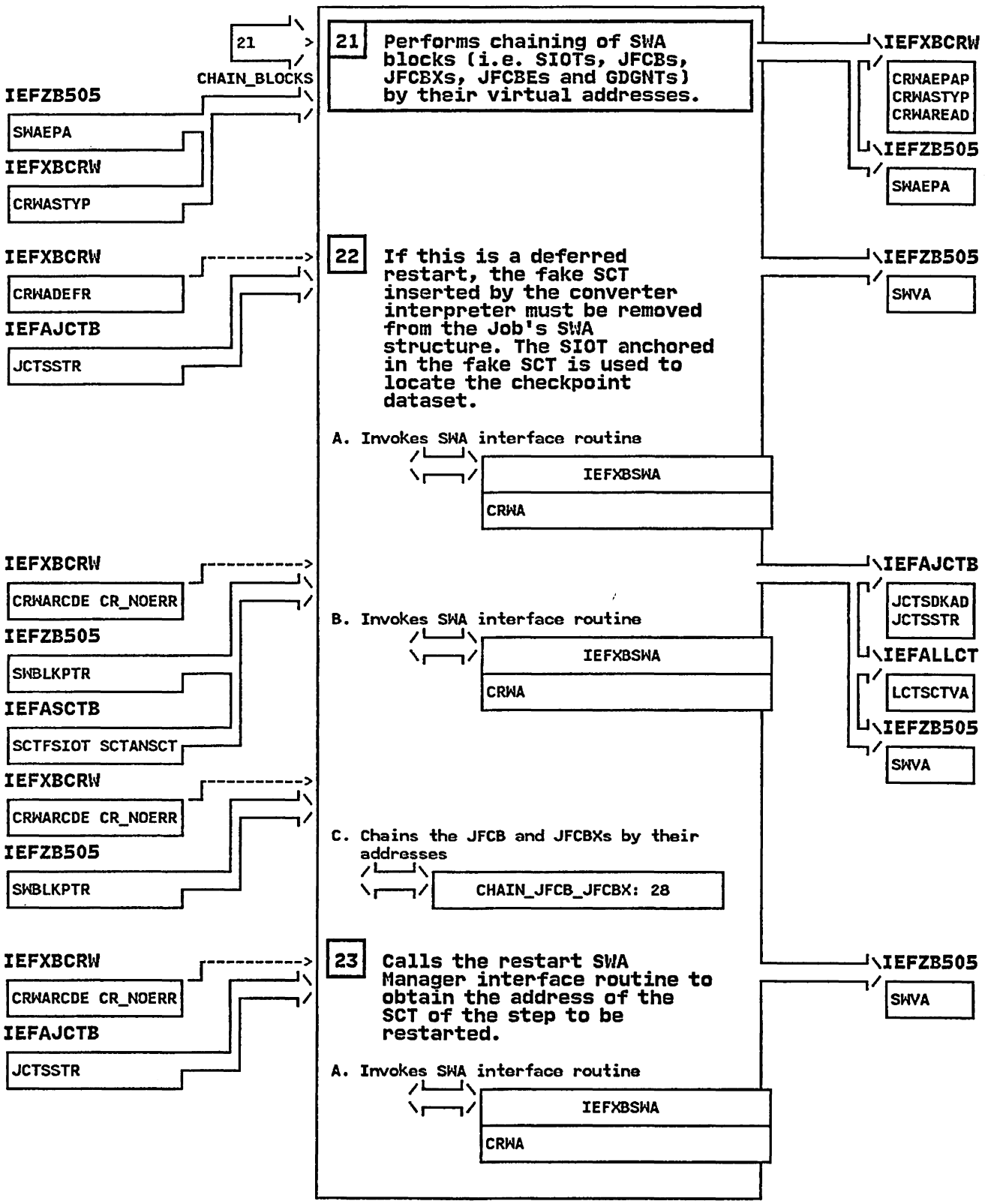




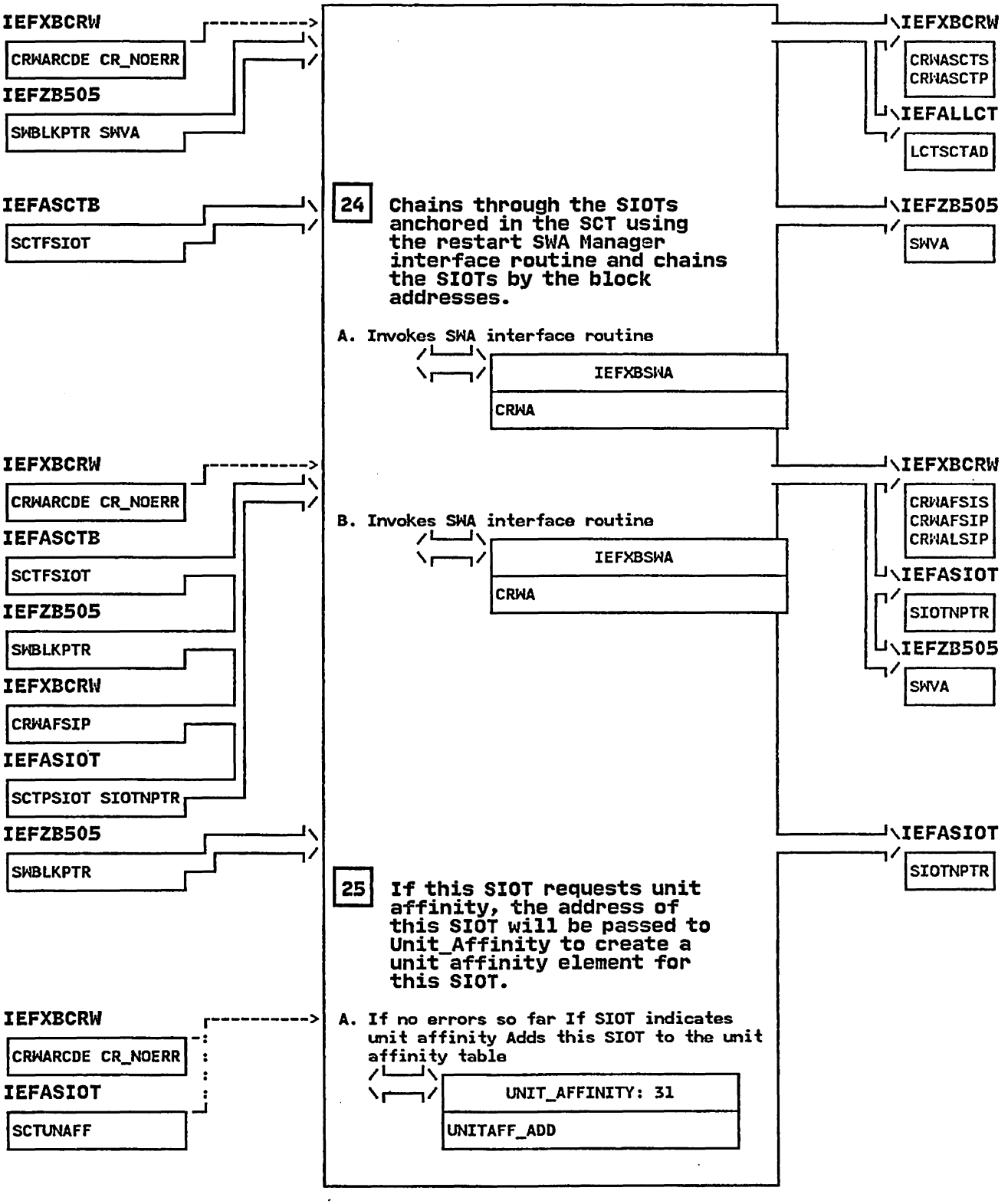
IEFXB609 - Checkpoint Restart Dataset Descriptor Record Processor ContSTEP 17

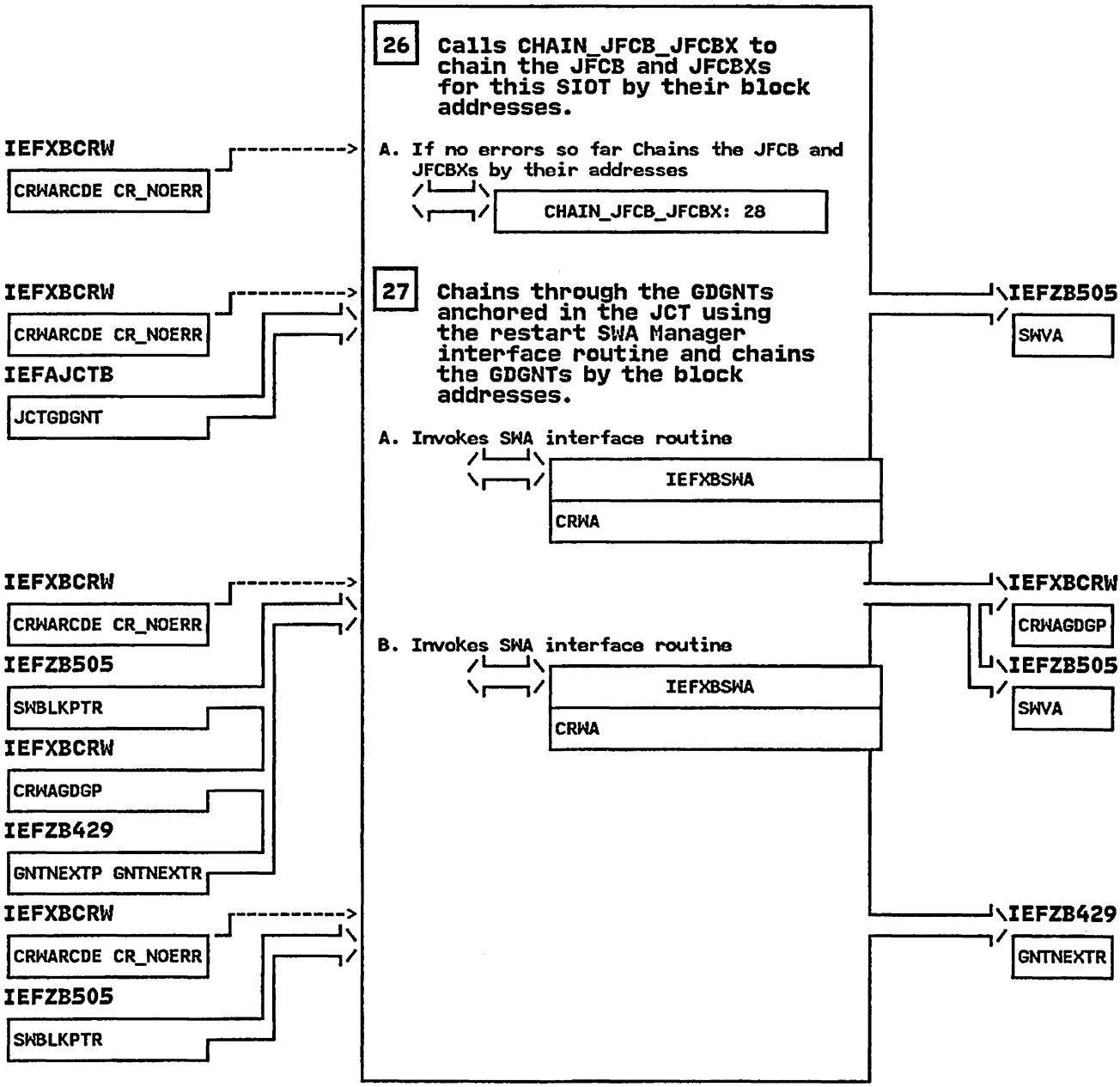


IEFXB609 - Checkpoint Restart Dataset Descriptor Record Processor ContSTEP 21

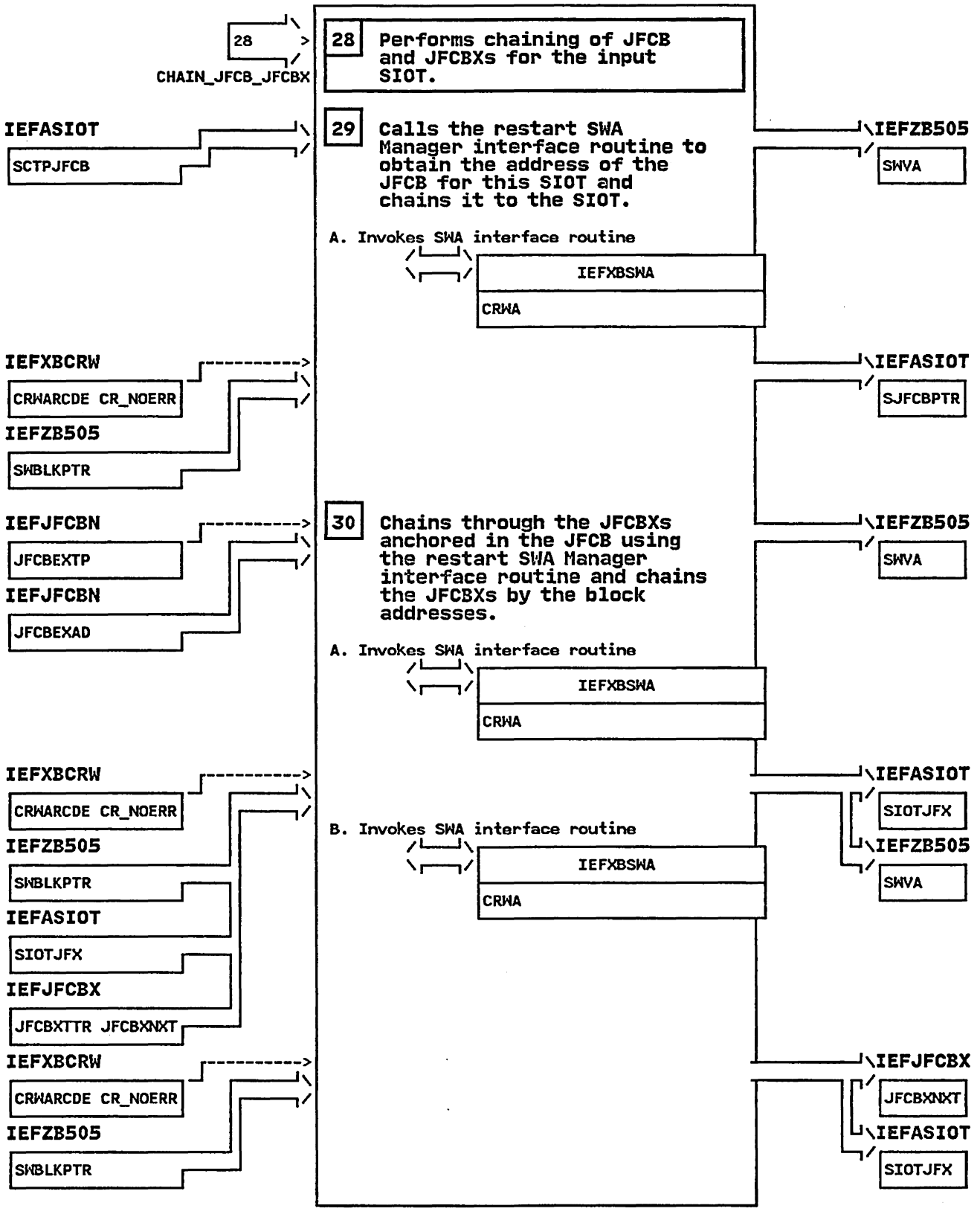


IEFXB609 - Checkpoint Restart Dataset Descriptor Record Processor ContSTEP 24

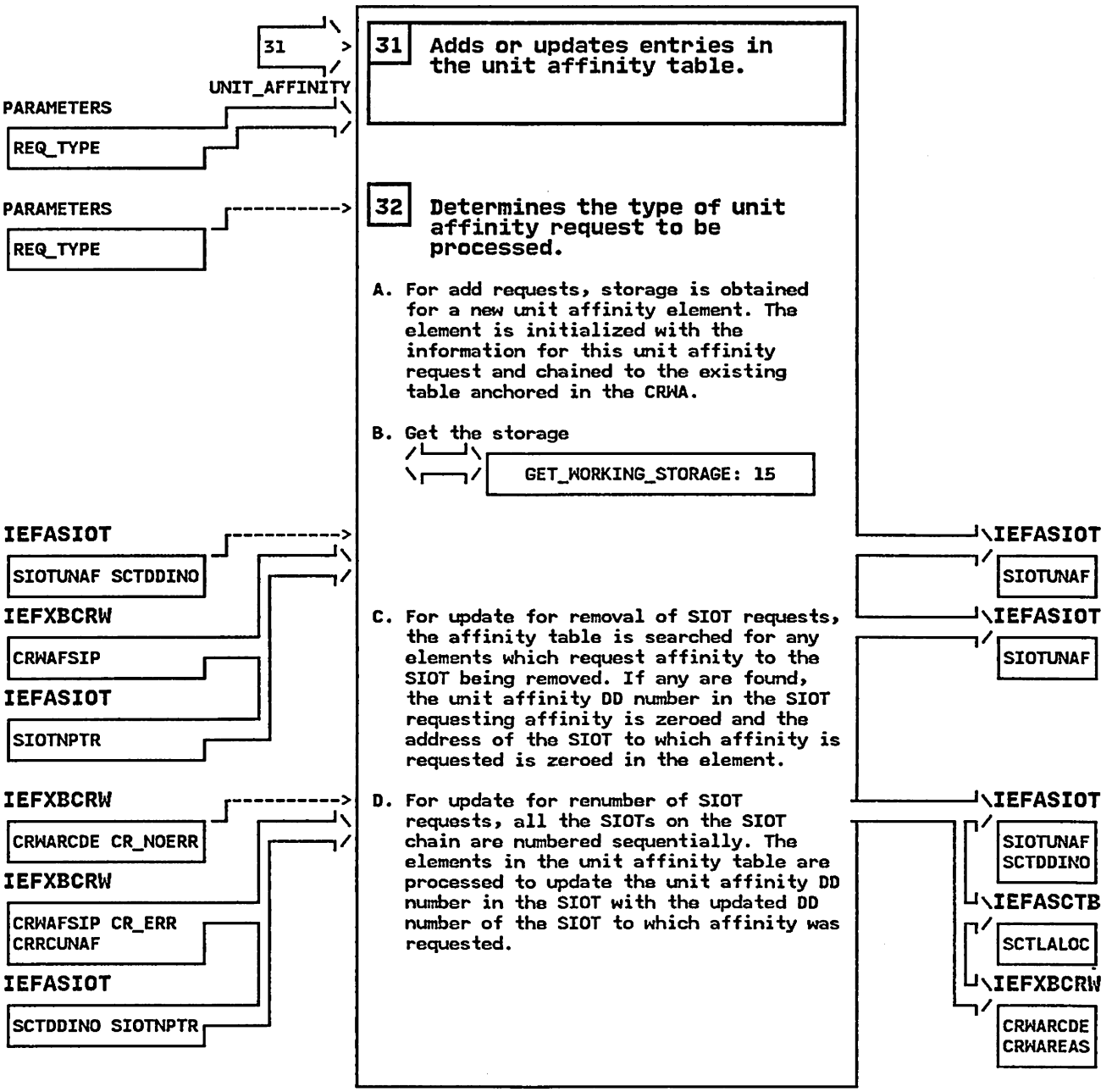




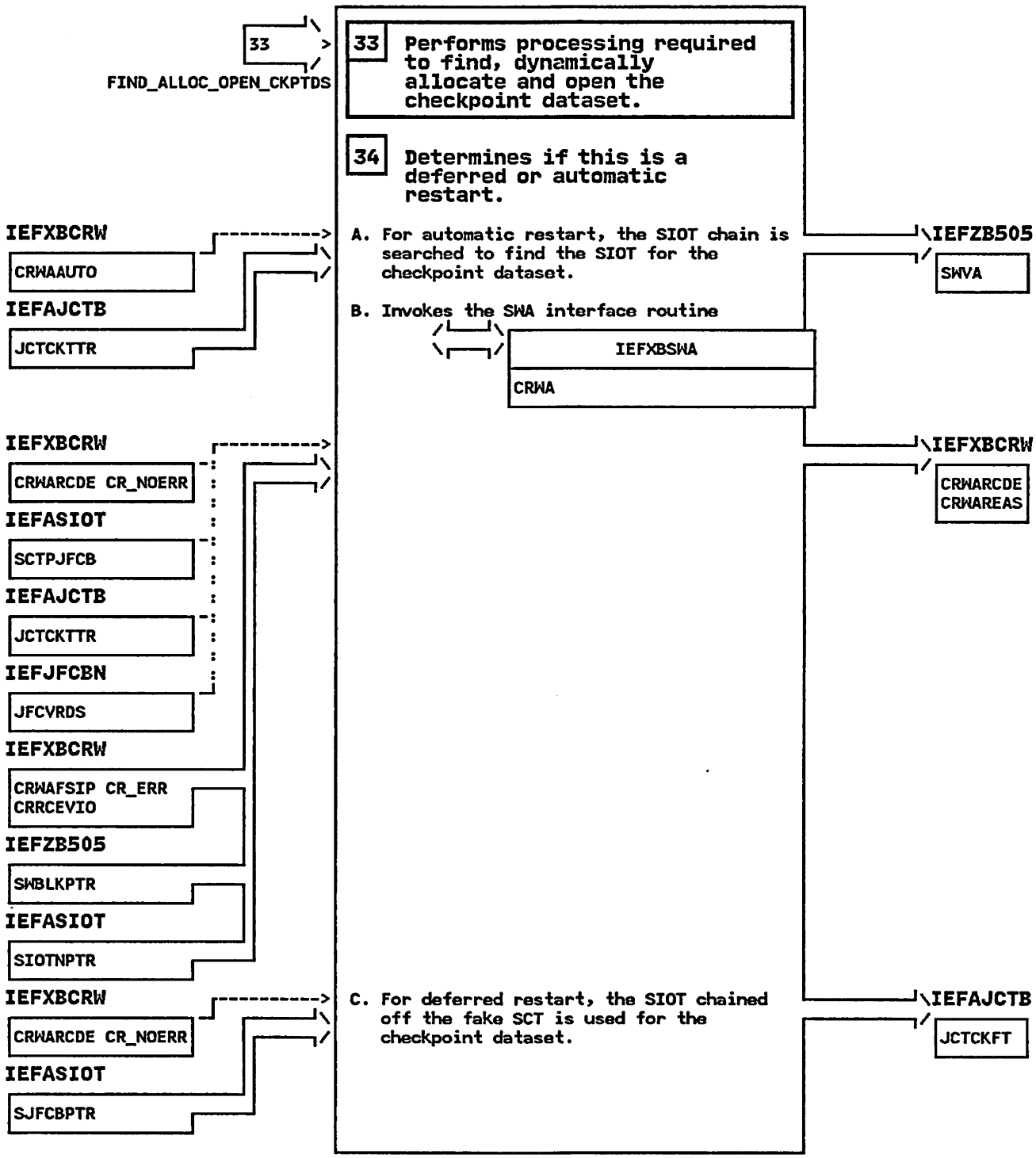
IEFXB609 - Checkpoint Restart Dataset Descriptor Record Processor ContSTEP 28

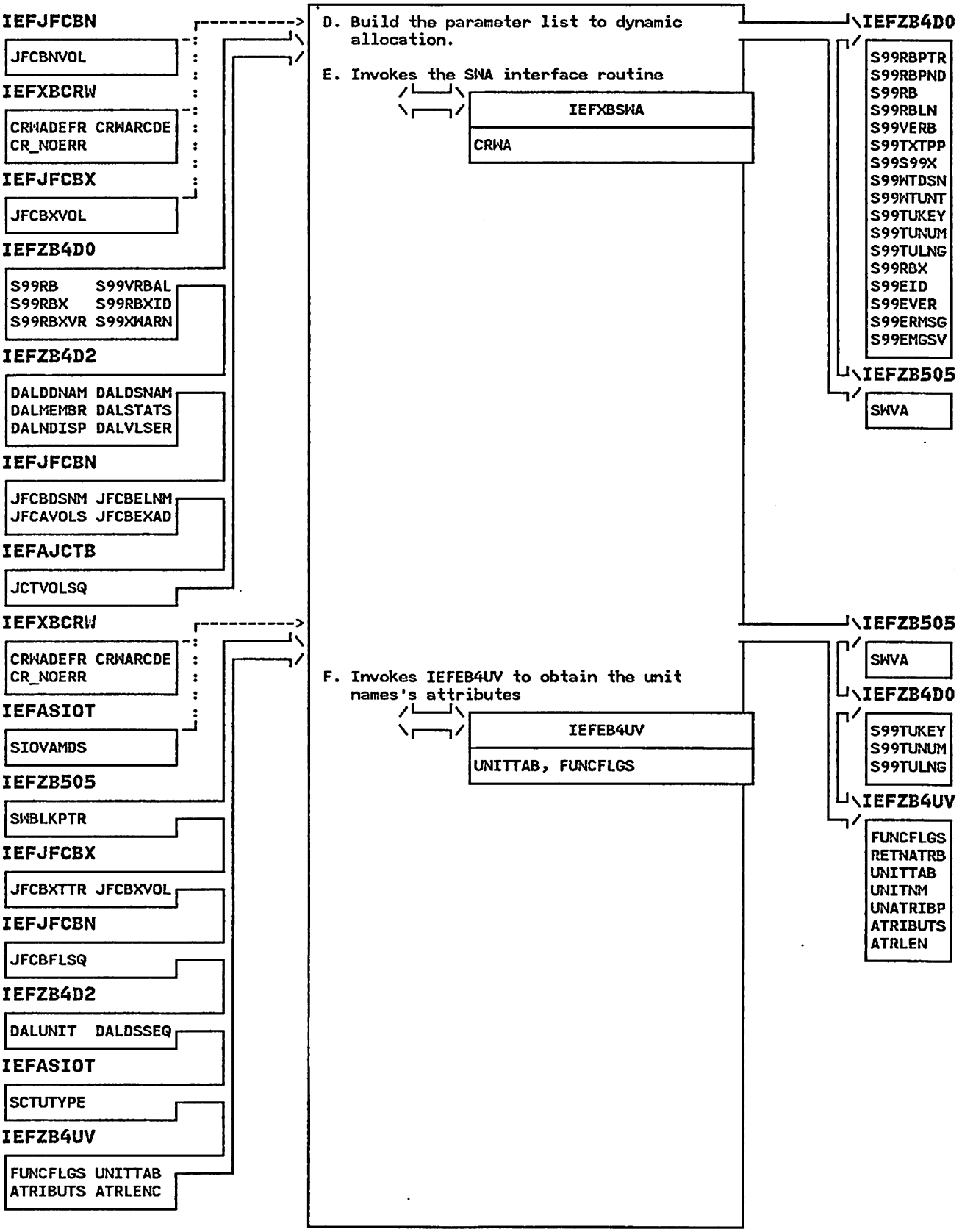


IEFXB609 - Checkpoint Restart Dataset Descriptor Record Processor ContSTEP 31

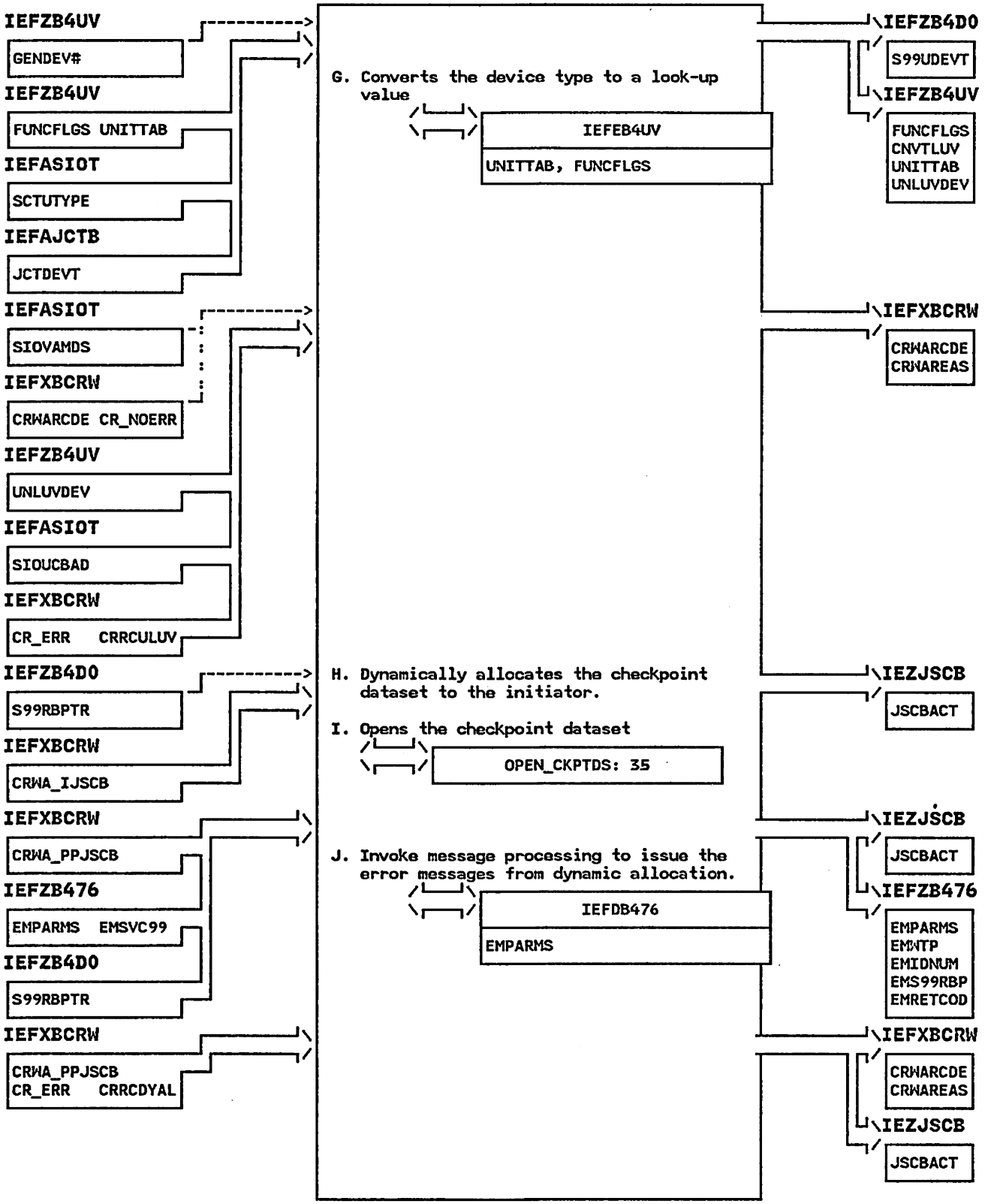


IEFXB609 - Checkpoint Restart Dataset Descriptor Record Processor ContSTEP 33

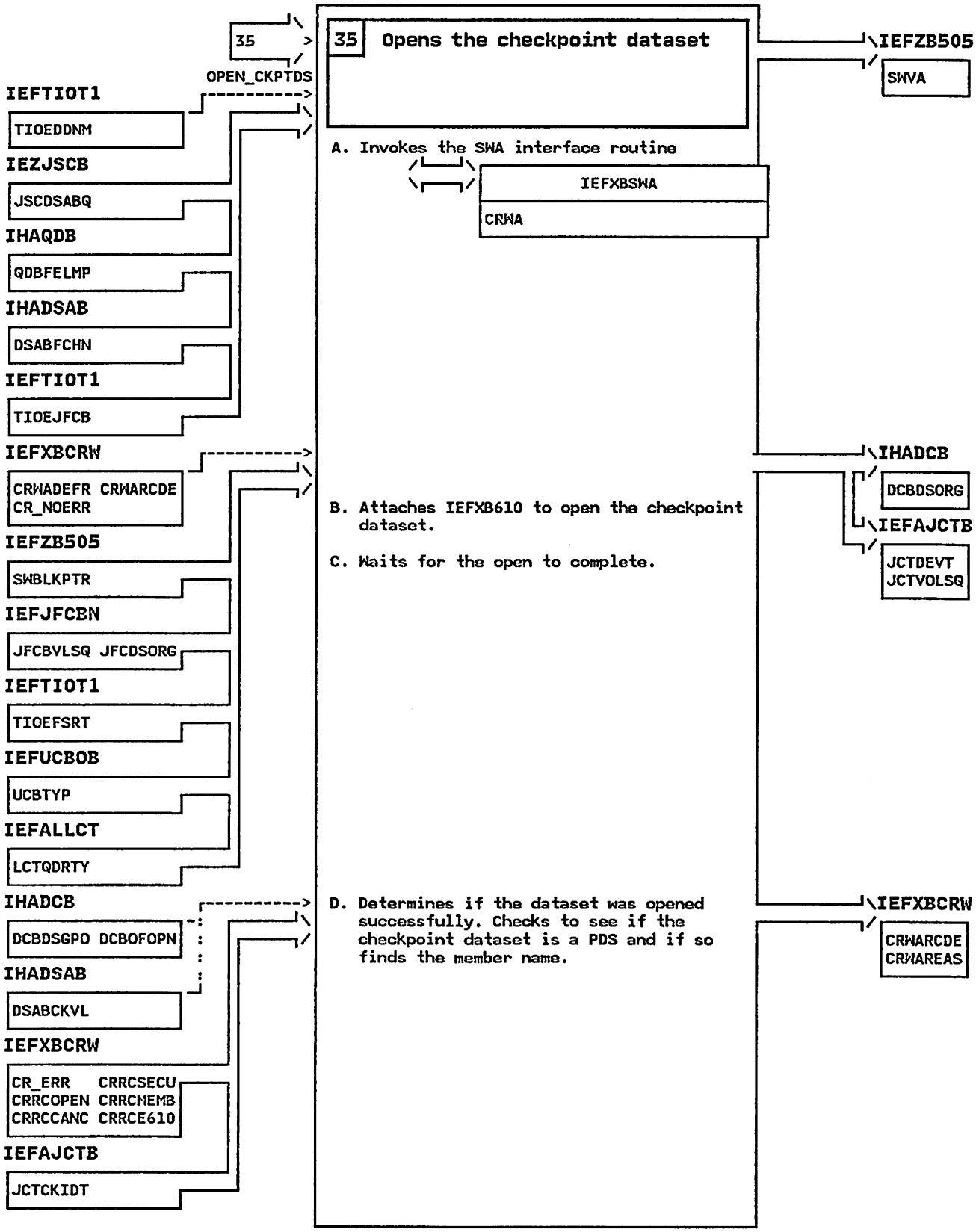




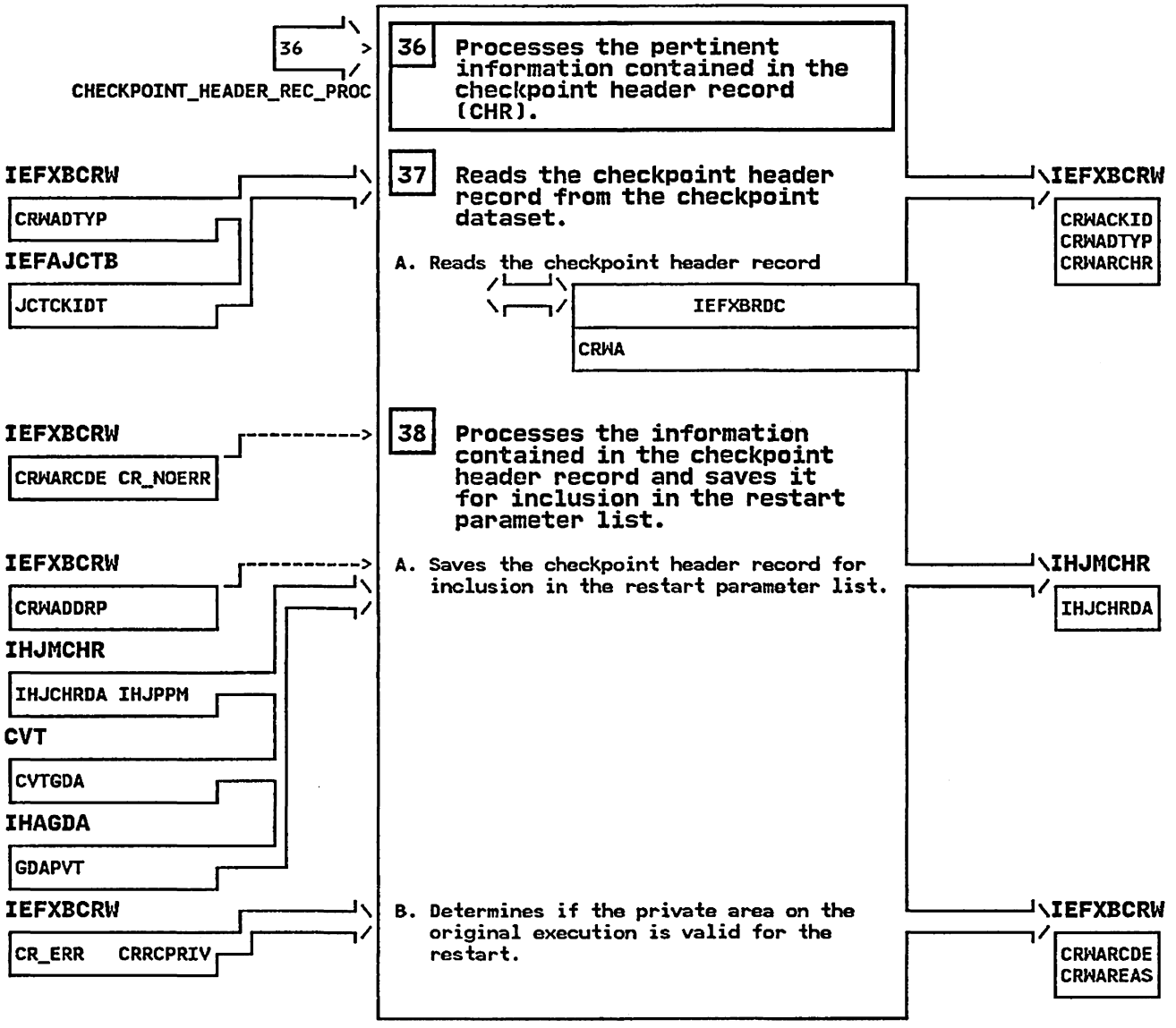
IEFXB609 - Checkpoint Restart Dataset Descriptor Record Processor ContSTEP 34G

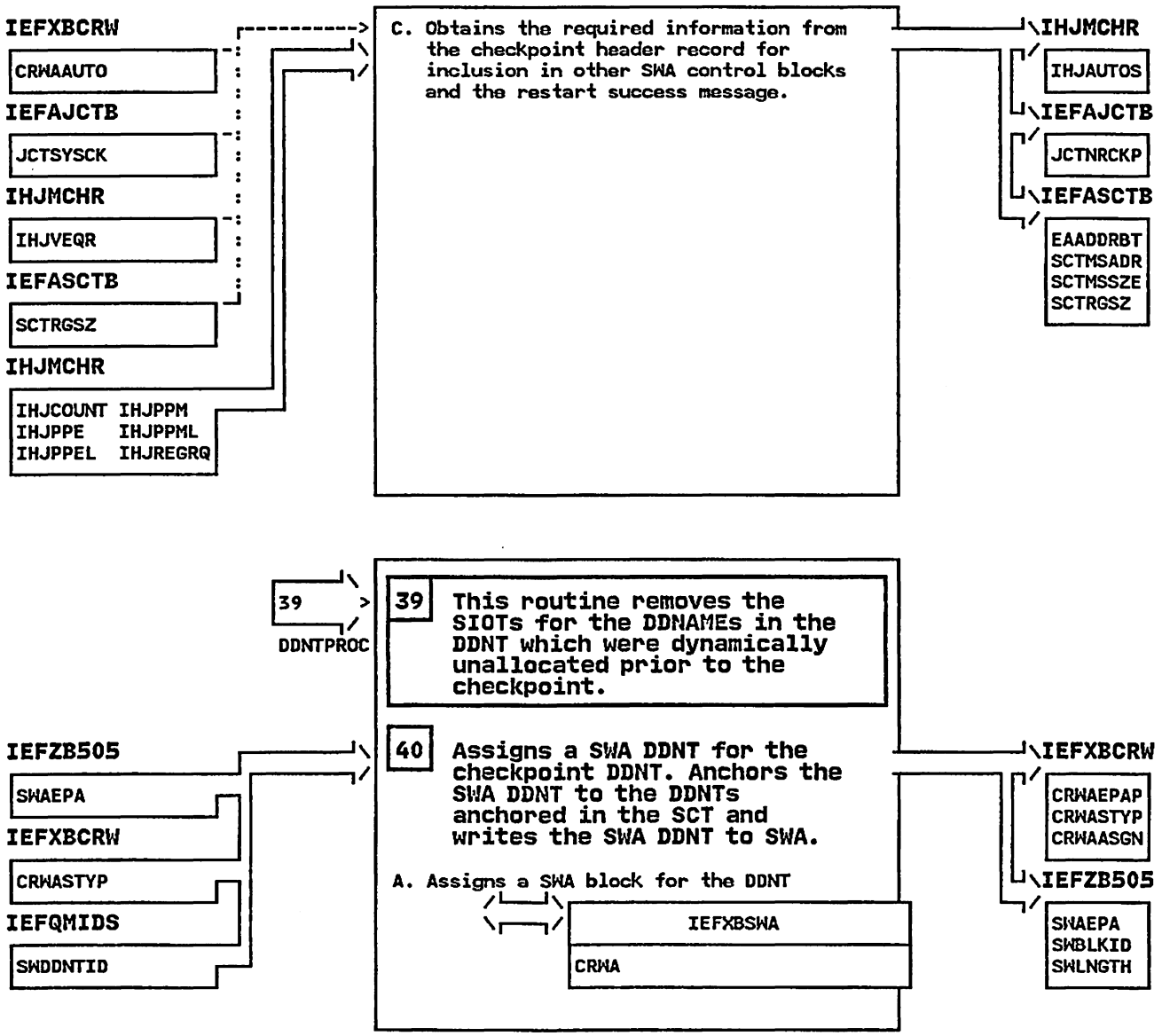


IEFXB609 - Checkpoint Restart Dataset Descriptor Record Processor ContSTEP 35

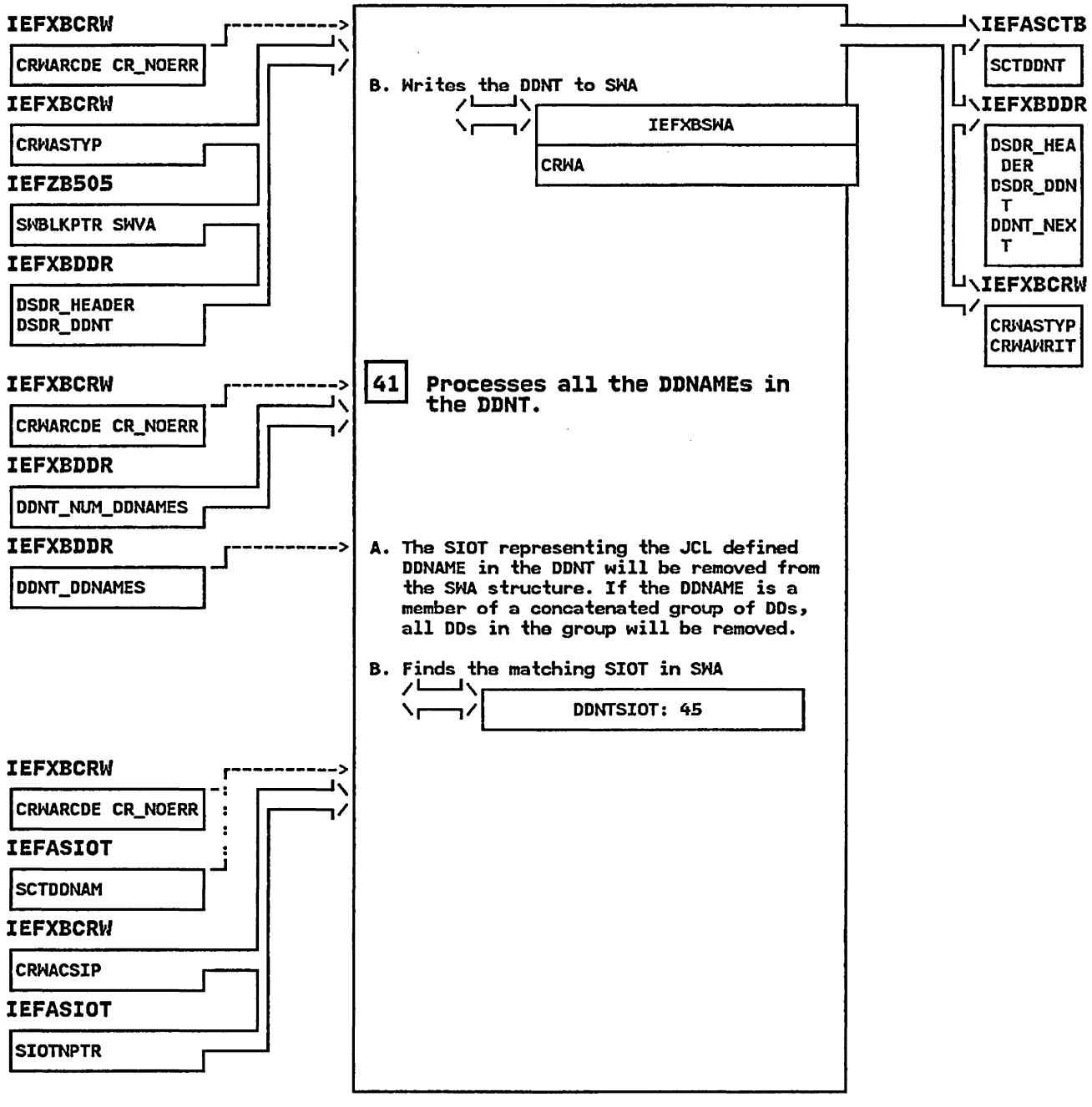


IEFXB609 - Checkpoint Restart Dataset Descriptor Record Processor ContSTEP 36

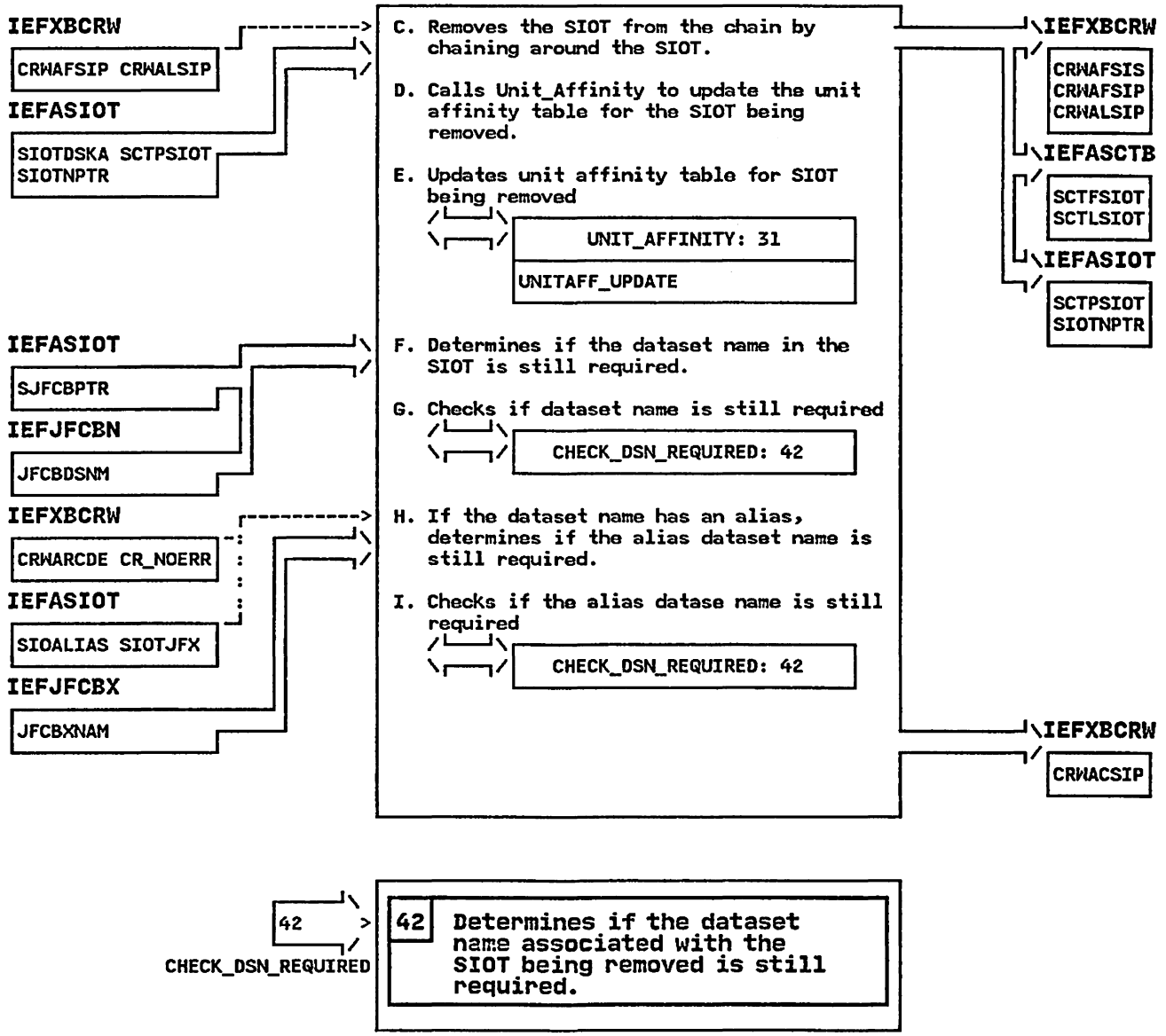




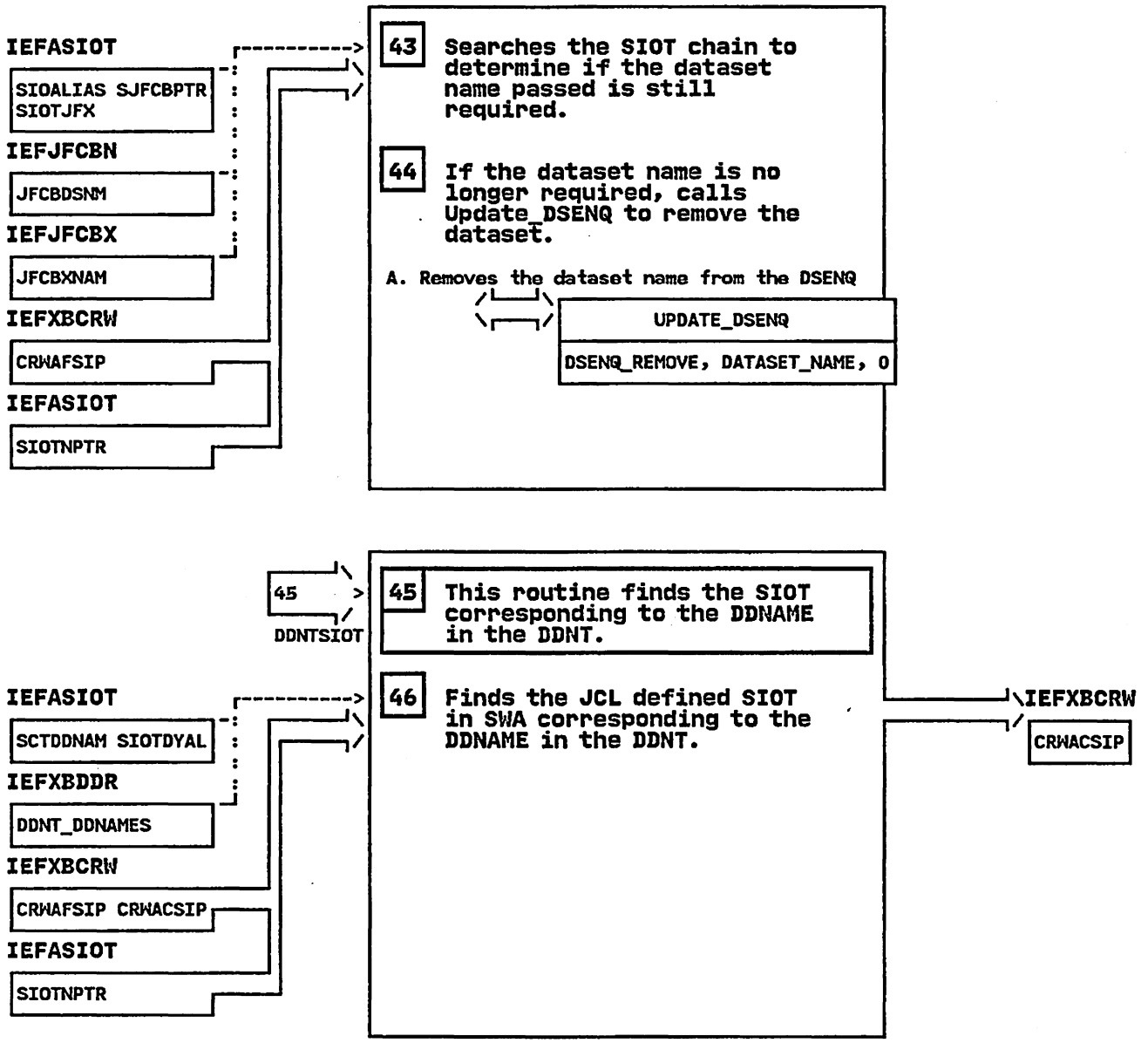
IEFXB609 - Checkpoint Restart Dataset Descriptor Record Processor ContSTEP 40B



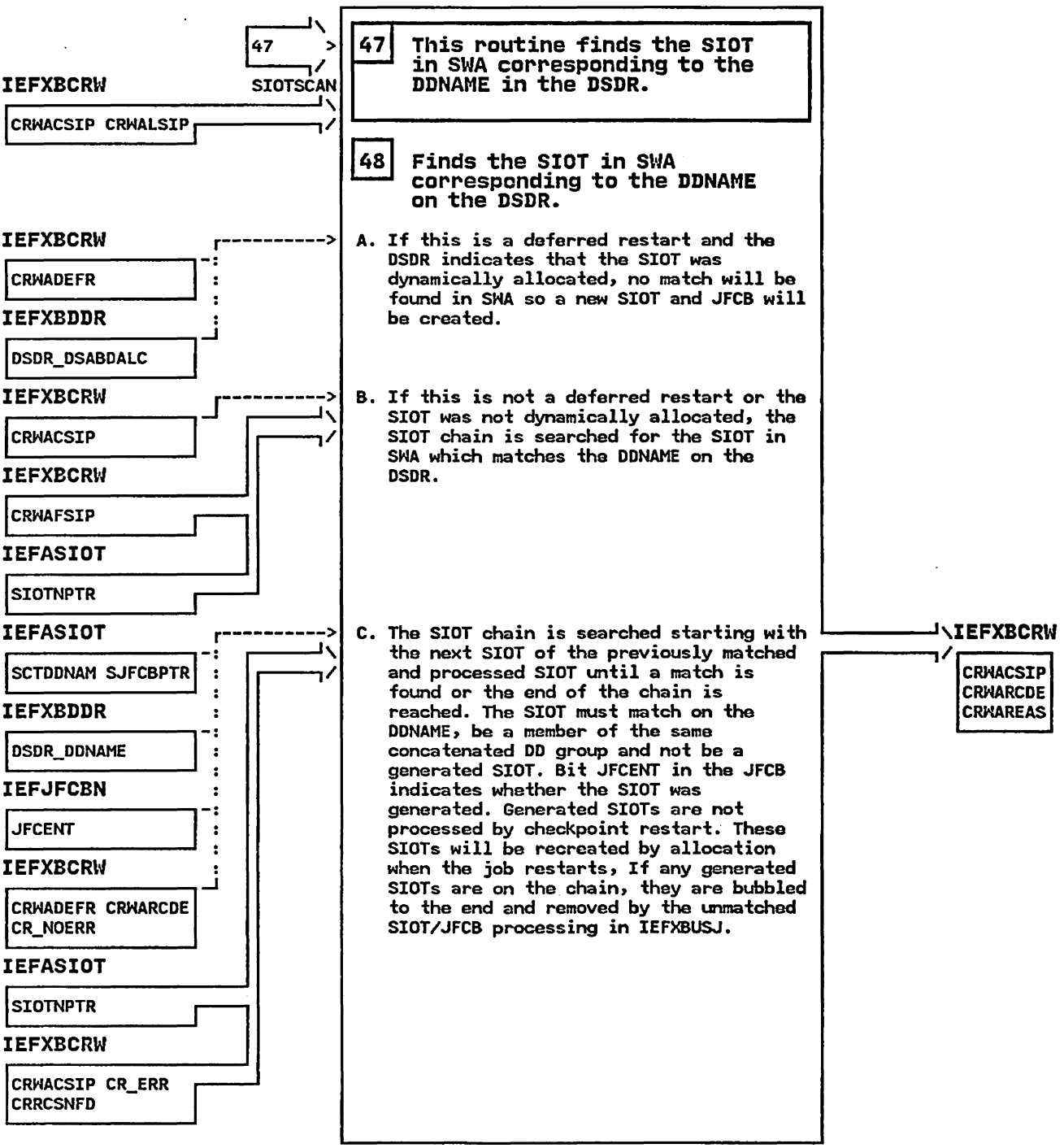
IEFXB609 - Checkpoint Restart Dataset Descriptor Record Processor ContSTEP 41C



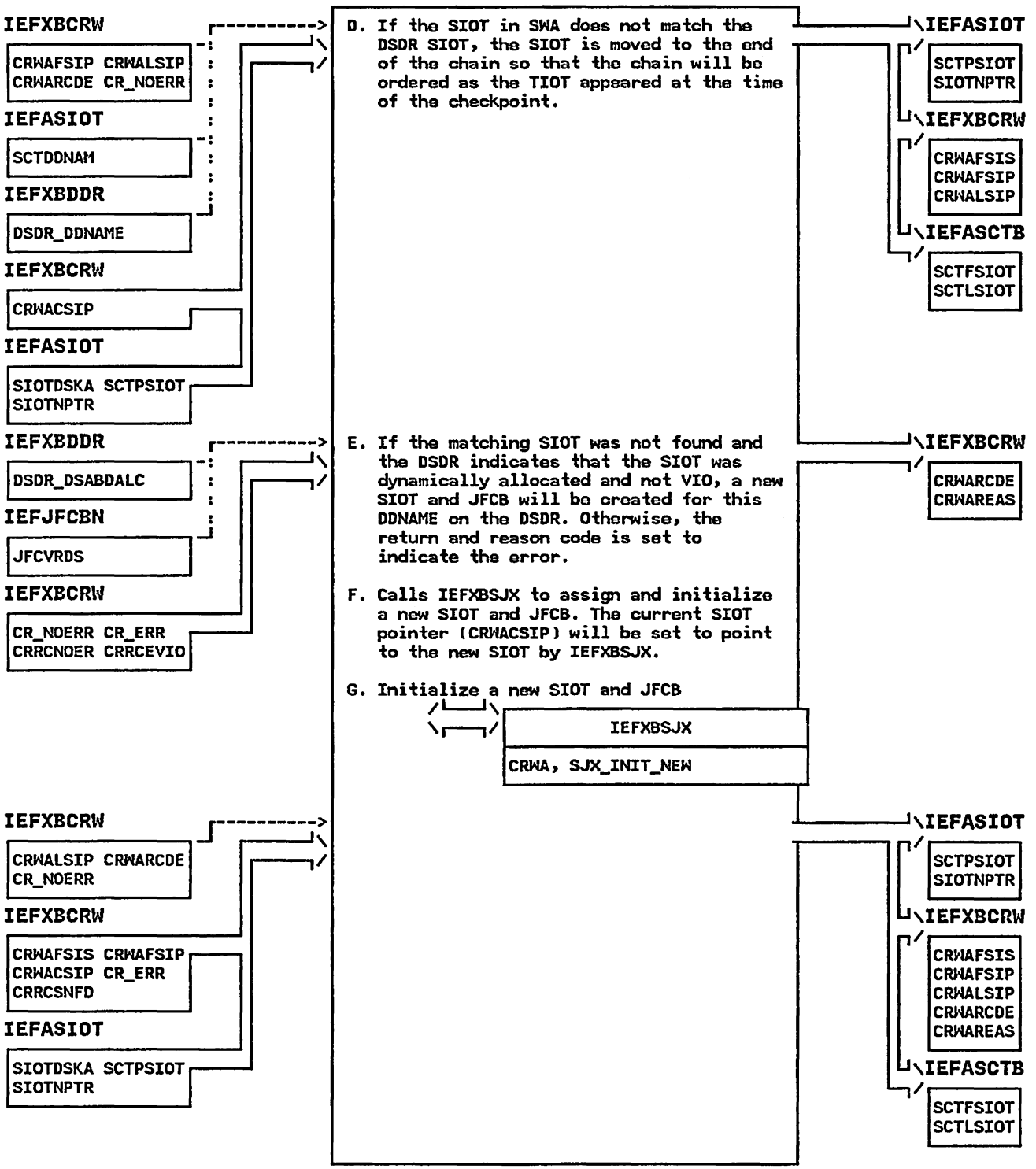
IEFXB609 - Checkpoint Restart Dataset Descriptor Record Processor ContSTEP 43



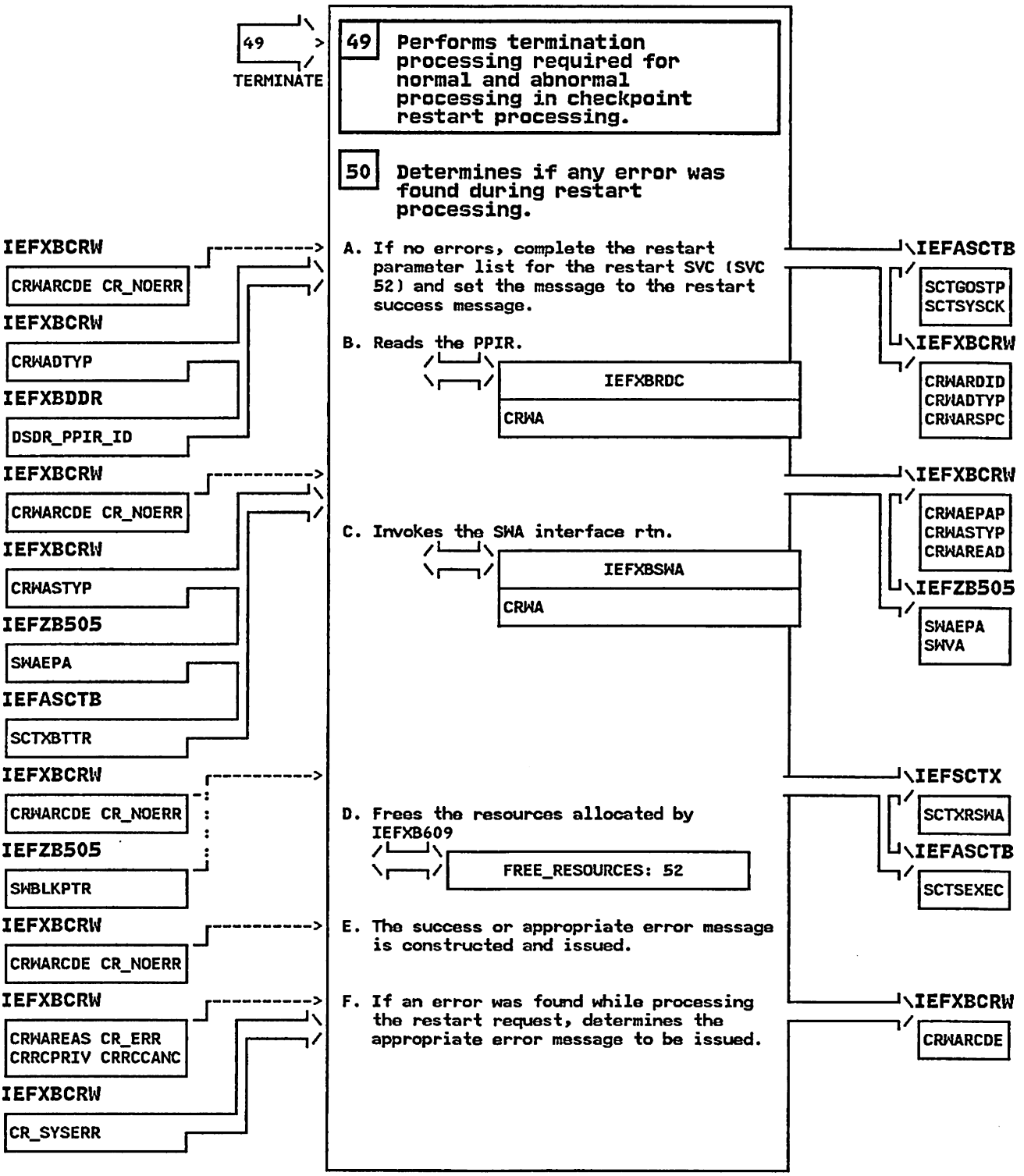
IEFXB609 - Checkpoint Restart Dataset Descriptor Record Processor ContSTEP 47



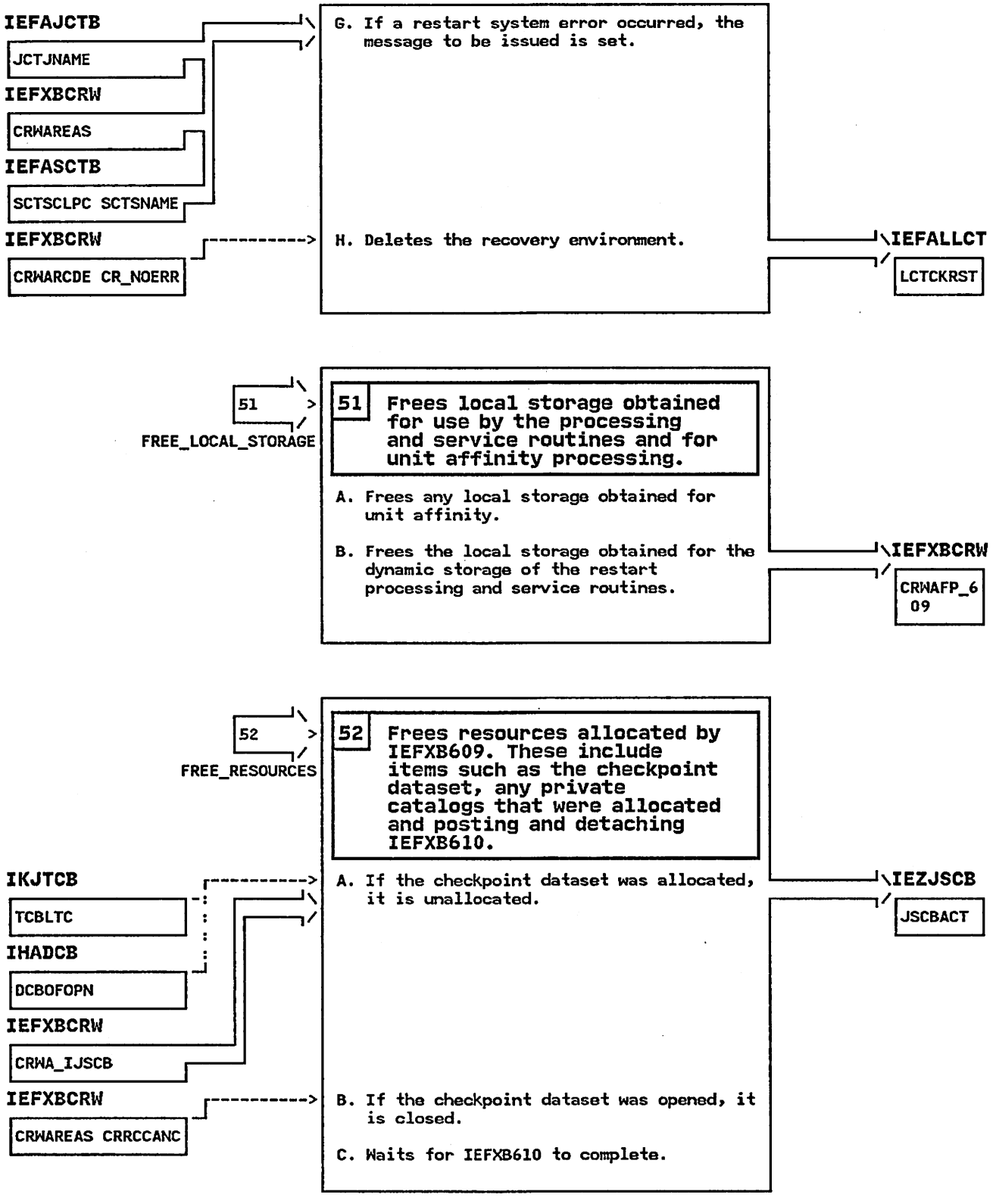
IEFXB609 - Checkpoint Restart Dataset Descriptor Record Processor ContSTEP 48D

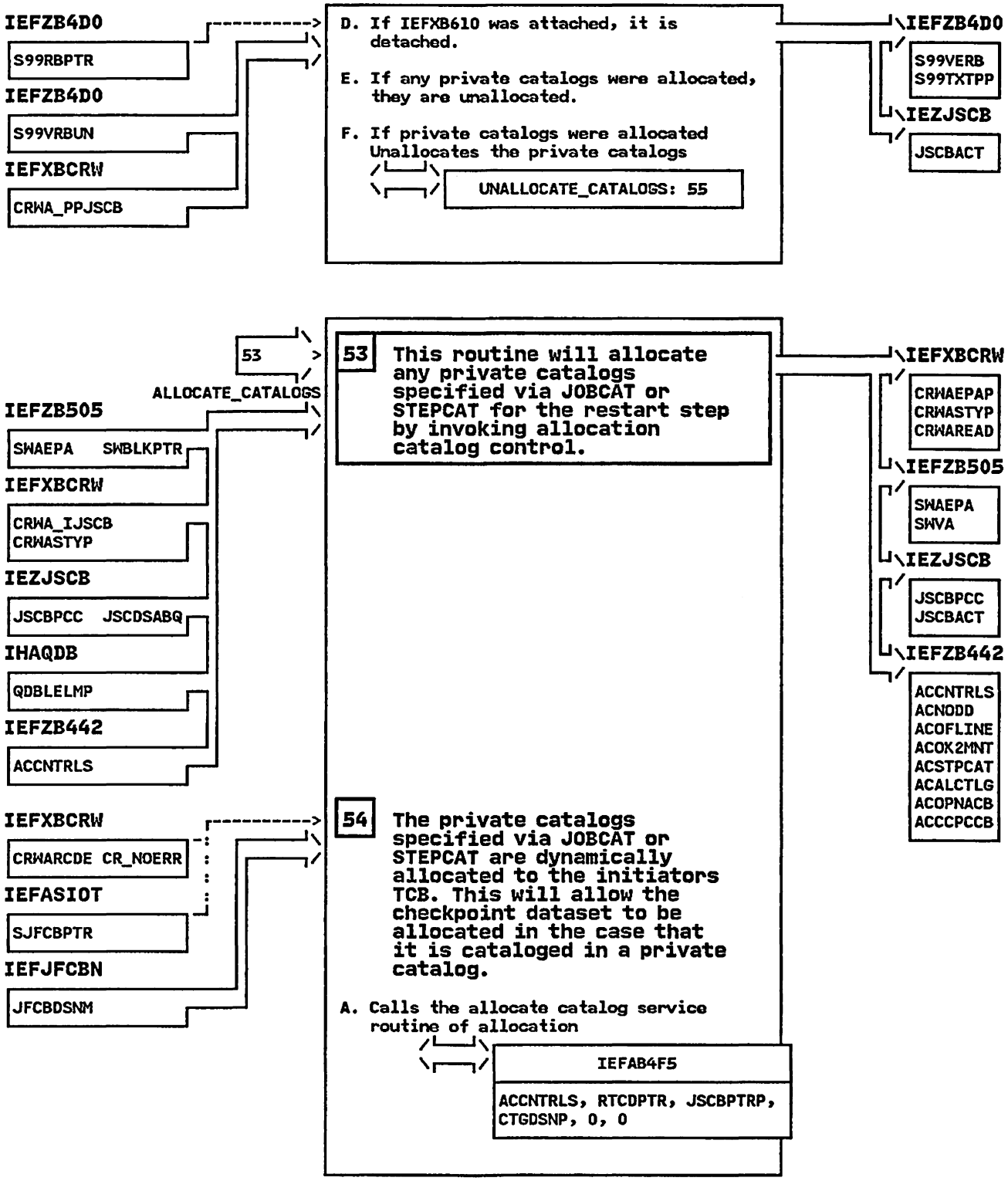


IEFXB609 - Checkpoint Restart Dataset Descriptor Record Processor ContSTEP 49

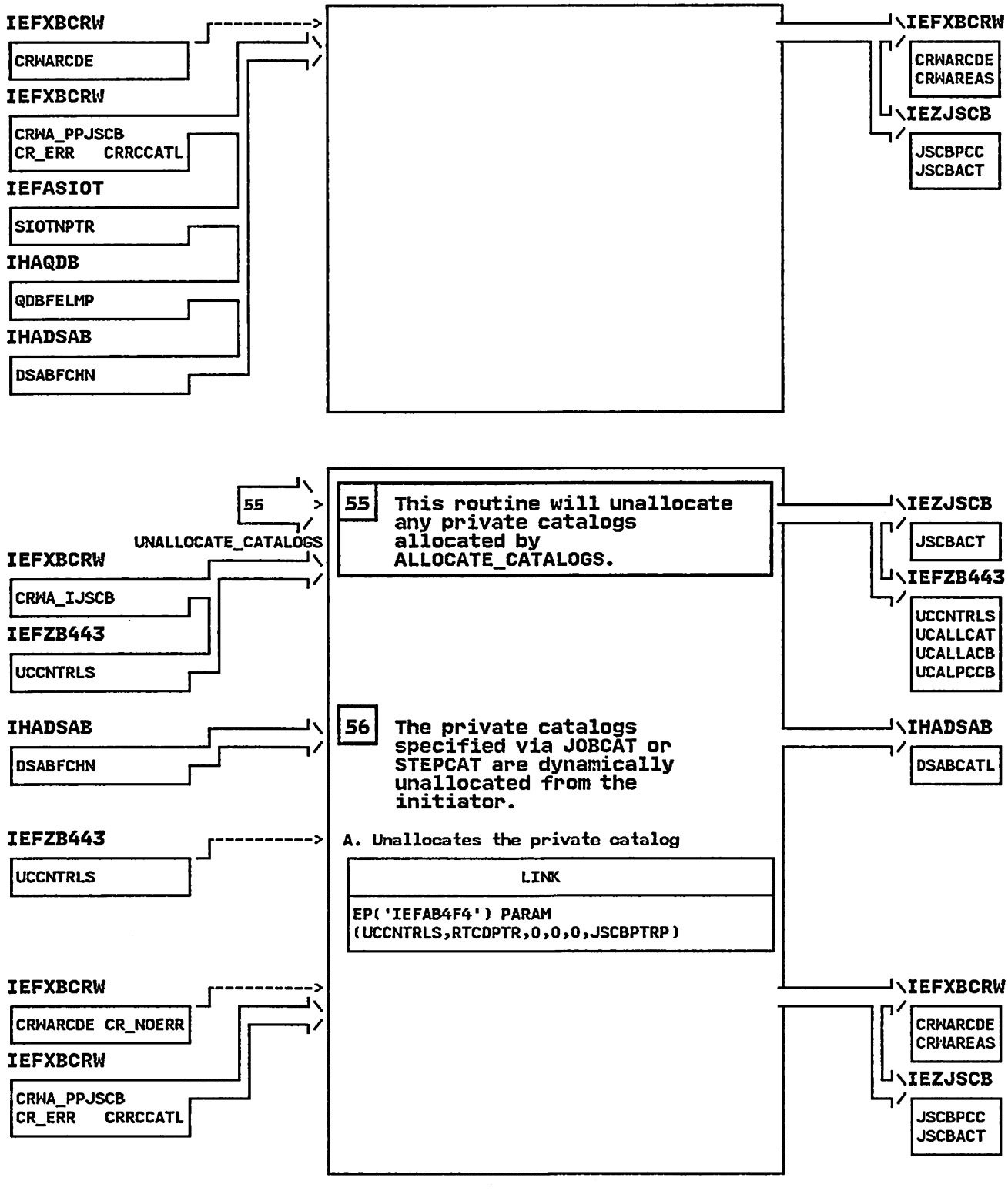


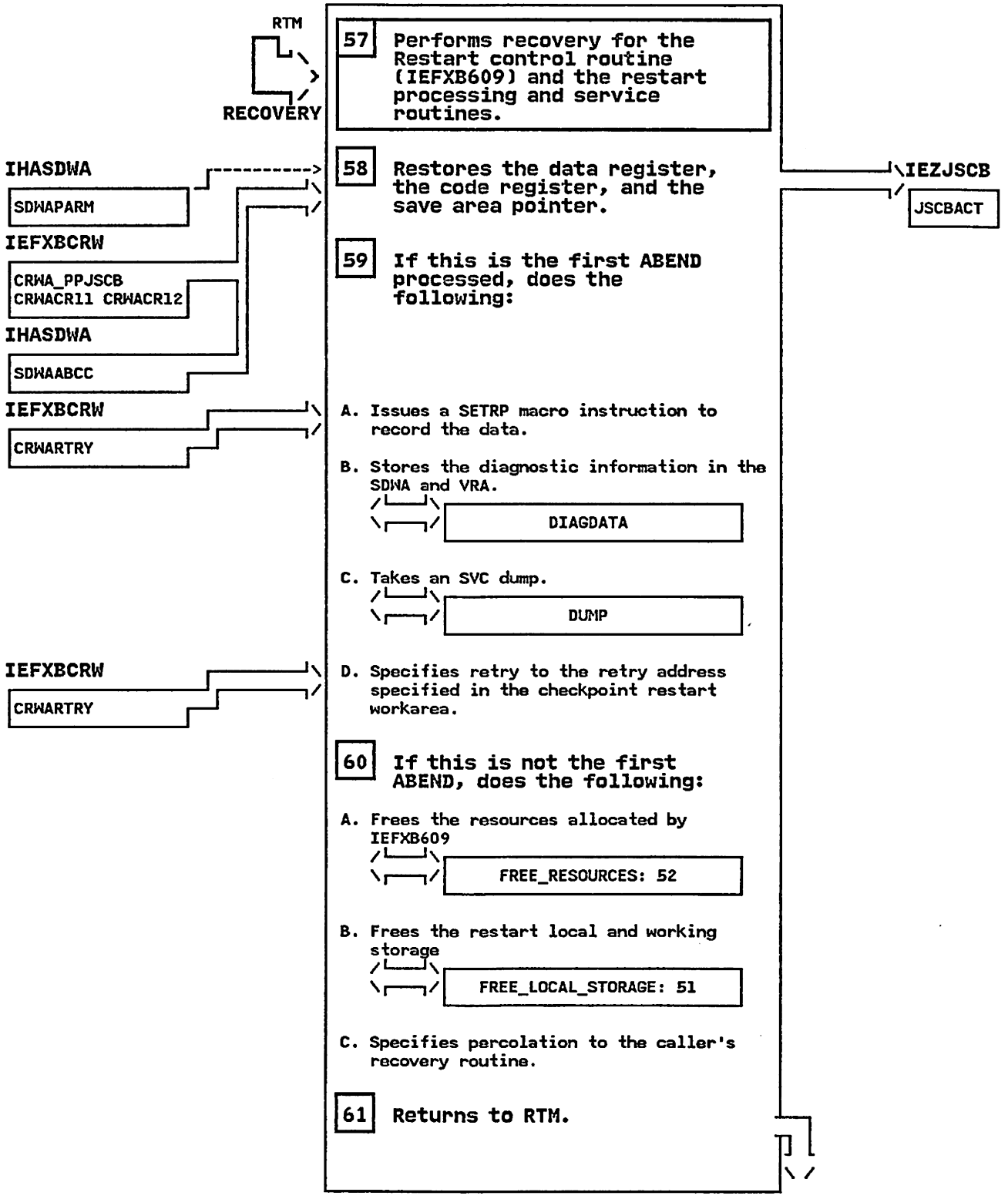
IEFXB609 - Checkpoint Restart Dataset Descriptor Record Processor ContSTEP 50G





IEFXB609 - Checkpoint Restart Dataset Descriptor Record Processor ContSTEP 55





INDEX

MVS/Extended Architecture
System Logic Library:
Scheduler Restart

READER'S
COMMENT
FORM

LY28-1745-1

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

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

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

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

_____

_____

_____

_____

What is your occupation? _____

How do you use this publication? _____

Number of latest Newsletter associated with this publication: _____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

MVS/Extended Architecture System Logic Library: Scheduler Restart

LY28-1745-1 S370-36

Reader's Comment Form

Cut or Fold Along Line

Fold and tape Please Do Not Staple Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department D58, Building 921-2
PO Box 950
Poughkeepsie, New York 12602

Fold and tape Please Do Not Staple Fold and tape

Printed in U.S.A.

IBM®

LY28-1745-01

MVS/Extended Architecture System Logic Library: Scheduler Restart

LY28-1745-1

S370-36

Printed in U.S.A.