"Restricted Materials of IBM"
All Rights Reserved
Licensed Materials - Property of IBM
©Copyright IBM Corp. 1987
LY28-1695-0
File No. S370-36

Program Product

MVS/Extended Architecture System Logic Library: Global Resource Serialization

MVS/System Product:

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

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

This publication supports MVS/System Product Version 2 Release 2.0, and contains information that was formerly presented in MVS/Extended Architecture System Logic Library Volume 7, LY28-1230-4, which applies to MVS/System Product Version 2 Release 1.7. See the Summary of Amendments for more information.

First Edition (June, 1987)

This edition applies to Version 2 Release 2.0 of MVS/System Product 5665-291 or 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.

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 390, 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.

(c) Copyright International Business Machines Corporation 1987

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 Table of Contents and Index.

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 Table of Contents and Index contains the master preface, the master table of contents, and the 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 and an index.

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 Table of Contents and 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.

 (c) Copyright IBM Corp. 1987

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 to group the books by related functions.

The Table of Book Titles in the Master Preface in MVS/Extended Architecture System Logic Library: Master Table of Contents and 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 Table of Contents and Index. For the component books, the page number in an index entry consists of the mnemonic for the component and the page number; for System Logic Library: Module Descriptions, the page number consists of the mnemonic "MOD" and the page number.

For example:

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

MOD-245 refers to MVS/Extended Architecture System Logic Library: Module Descriptions, page MOD-245.

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.

 LY28-1695-0 (c) Copyright IBM Corp. 1987

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 Table of Contents and Index.

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

 LY28-1695-0 (c) Copyright IBM Corp. 1987

CONTENTS

 (c) Copyright IBM Corp. 1987

 LY28-1695-0 (c) Copyright IBM Corp. 1987

FIGURES

42. Key to Method-of-Operation Diagrams GRS-76

LY28-1695-0 (c) Copyright IBM Corp. 1987

Summary of Amendments
for LY28-1695-0
for MVS/System Product Version 2 Release 2.0

This publication is new for MVS System Product Version 2 Release 2.0. It contains information that was reorganized from the GRS section in MVS/XA System Logic Library Volume 7, LY28-1230-4, which applies to MVS/XA System Product Version 2 Release 1.7.

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

- Changes supporting storage management, including the extended resource queue area, the resource queue area, and the pool extent block.
- Minor technical and editorial changes throughout the publication.

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

LY28-1695-0 (c) Copyright IBM Corp. 1987

INTRODUCTION

This introduction provides background information necessary to understand the purpose and processing of the modules that comprise global resource serialization. The first topic briefly describes the functions and interfaces provided by global resource serialization: the ENQ, DEQ, and RESERVE macro instructions; the use of exit routines and resource name lists to convert serialization requests; the role of the GRSCNFxx and IEASYSxx PARMLIB members and of operator commands in initializing and controlling a global resource serialization complex; the GQSCAN macro; and the GRSQ parameter on the SDUMP macro, and the GRSTRACE parameter for print dump (PRDMP), and the GRACETRACE keyword, or the VERBEXIT subcommand, for the interactive problem control system (IPCS). Readers familiar with these interfaces can skip this topic. Subsequent topics describe the key concepts and terminology of the subcomponents of global resource serialization.

THE FUNCTIONS AND INTERFACES OF GLOBAL RESOURCE SERIALIZATION

Global resource serialization serializes the use of both local and global serially reusable resources, as requested by ENQ, DEQ, and RESERVE macro instructions. Local resources are accessible by only one system; global resources reside on shared direct access devices and are accessible by more than one system in a loosely coupled or shared spool multiprocessing environment.

Formerly, without the services provided by global resource serialization, the only means of serializing global resources was a hardware RESERVE instruction, generated by the RESERVE macro instruction. The hardware RESERVE instruction reserved the entire volume containing the requested resource for use by one system, until that system relinquished control of the resource by means of DEQ.

Global resource serialization serializes the use of global resources without using the hardware RESERVE instruction. By communicating global requests to all systems included in the global resource serialization complex (defined by the installation), global resource serialization serializes the use of resources on the volume, not the entire volume. More than one system can enqueue concurrently on different resources on a single shared volume; and more than one system can enqueue concurrently on the same resource if all the requests specify shared control.

To serialize use of a global resource among systems in the global resource serialization complex, a program issues the ENQ macro (and, subsequently, the DEQ macro) with a scope of SYSTEMS. A scope of STEP or SYSTEM requests local serialization. To allow the installation to run existing programs without changing them (for example, programs that contain RESERVE), global resource serialization provides three exit routines that check three resource name lists (defined by the installation or IBM-supplied defaults): an inclusion exit and SYSTEM inclusion resource name list; an exclusion exit and SYSTEMS exclusion resource name list; and a RESERVE conversion exit and resource name list. The SYSTEM inclusion resource name list contains names of resources to be serialized globally. The SYSTEMS exclusion list contains names of resources to be serialized locally (including data sets to be excluded from generic names in the SYSTEM inclusion list). The RESERVE conversion resource name list contains names of

 (c) Copyright IBM Corp. 1987

global resources for which the hardware RESERVE instruction is to be suppressed. Which exits are invoked depends on the request and the scope it specified:

- For ENQ or DEQ requests that specify SYSTEM (local serialization), global resource serialization invokes the inclusion exit and, if the requested resource is named in the SYSTEM inclusion list, the SYSTEMS exclusion exit. If the requested resource is named in the inclusion list and not in the exclusion list, global resource serialization changes the scope to SYSTEMS (global serialization).

- For ENQ or DEQ requests that specify SYSTEMS (global serialization), global resource serialization invokes the exclusion exit. If the requested resource is named in the SYSTEMS exclusion list, global resource serialization changes the scope to SYSTEM (local serialization).

- For RESERVE requests, global resource serialization invokes the exclusion exit. If the requested resource is named in the SYSTEMS exclusion list, global resource serialization will issue a SYSTEM (local) ENQ for the resource and will not suppress the hardware RESERVE instruction. If the requested resource is not named in the exclusion list, global resource serialization invokes the RESERVE conversion exit:

  - If the resource is named in the RESERVE conversion list, global resource serialization issues a SYSTEMS (global) ENQ and suppresses the hardware RESERVE instruction.

  - If the resource is not named in the RESERVE conversion list, global resource serialization issues a SYSTEMS (global) ENQ for the resource but does not suppress the hardware RESERVE instruction.

The systems in a global resource serialization complex must be connected using dedicated CTCs (channel to channel adapters). The installation defines the global serialization complex by (1) defining the systems that are to participate in the complex in the GRS= parameter in an IEASYSxx member of SYS1.PARMLIB; and (2) defining the CTCs to be used by the systems in the GRSCNFxx member of SYS1.PARMLIB.

To allow the operator to monitor and modify the global resource serialization complex, global resource serialization provides the DISPLAY GRS and VARY GRS operator commands. The VARY GRS command allows the operator to suspend or resume a system's participation in a global resource serialization ring (the active global resource serialization systems in the complex, also called the main ring); rebuild a disrupted global resource serialization ring; or terminate a system's participation in the complex. The DISPLAY GRS command allows the operator to display the status of the systems in the global resource serialization complex and the channel-to-channel adapters (CTCs) assigned to global resource serialization and attached to the system on which the command is issued. The DISPLAY GRS command allows the operator to display resource contention information, the contents of the RNLs, the resource qnames, and resource name information.

In addition, global resource serialization provides the following:

- the GQSCAN macro, which allows users to obtain information about resources without directly accessing internal control blocks

- the GRSQ parameter on the SDUMP macro to request the inclusion of global resource serialization control blocks in an SVC dump

- and the GRSTRACE parameter for print dump (PRDMP) and the GRACETRACE keyword, or the VERBEXIT subcommand, for the

 LY28-1695-0 (c) Copyright IBM Corp. 1987

interactive problem control system (IPCS) to format global resource serialization data.

THE SUBCOMPONENTS OF GLOBAL RESOURCE SERIALIZATION

The global resource serialization component can be divided into several subcomponents, each of which is responsible for part of the processing necessary to provide the functions and interfaces described in the preceding topic. The structure of global resource serialization is reflected in the names of the modules that comprise it: the first three characters, ISG, identify the modules as part of global resource serialization; the fourth character identifies the function or service within the component or subcomponent that the module supports.

Figure GRS-1 summarizes the module naming conventions for global resource serialization modules. Figure GRS-2 shows the organization of the subcomponents that make up the global resource serialization component.

Module names: ISGzxxxx
ISG = global resource serialization

z=	Function
B	ring processing
C	command processing
D	dump support
G	mainline ENQ/DEQ/RESERVE processing
J	CTC processing
L	fast path ENQ/DEQ processing
M	WTO/WTOR message processing (ISGMSG00)
N	initialization
Q	queue scan (GQSCAN macro)
S	storage management

Note: Initialization modules (ISGNxxxx) are described in System Initialization Logic.

Figure 1. Global Resource Serialization Module Naming Conventions

 (c) Copyright IBM Corp. 1987

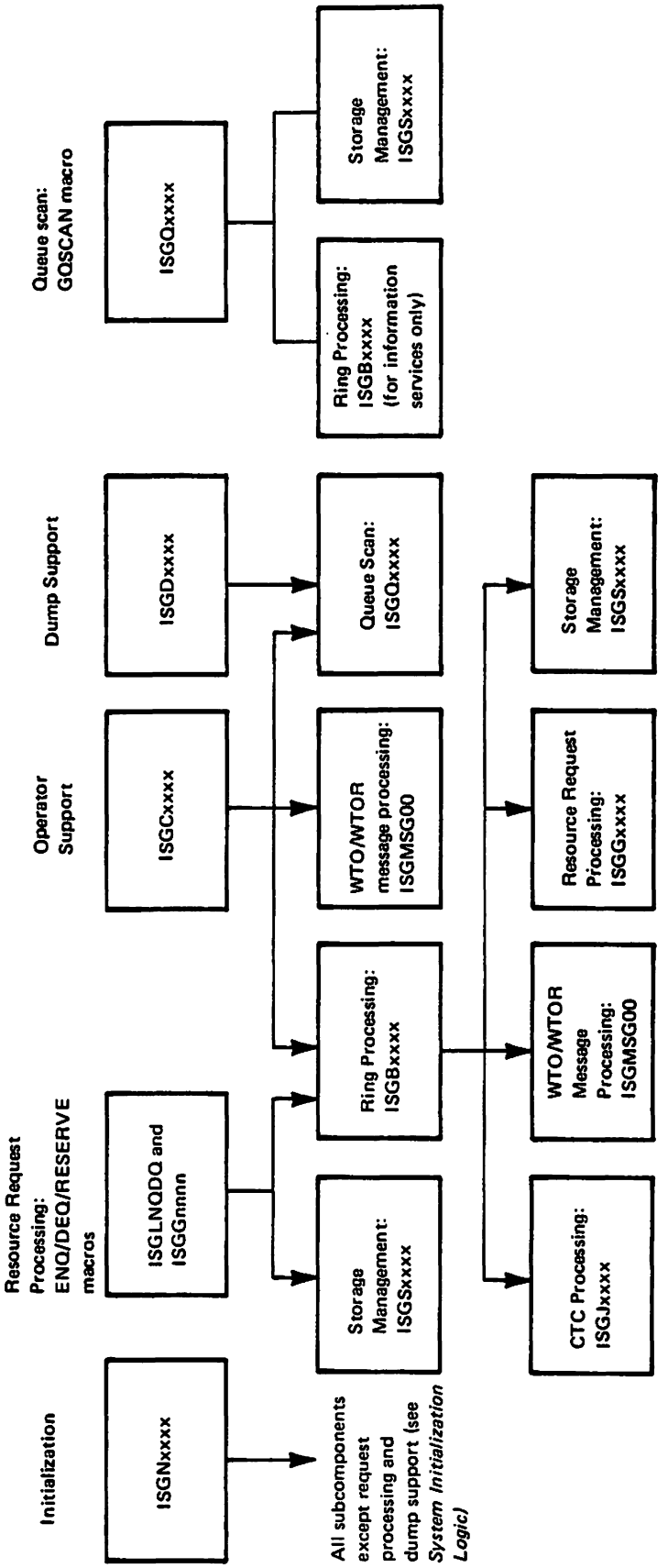


Figure 2. Subcomponents Invoked for Primary Functions/Interfaces of Global Resource Serialization

LY28-1695-0 (c) Copyright IBM Corp. 1987

The following describes each subcomponent, the functions or services it provides, and its relationship to other subcomponents.

- Initialization. The initialization subcomponent has two primary responsibilities:

  - Creating and initializing the global resource serialization address space. Global resource serialization has its own address space, accessible by means of PC/AUTH cross memory services, in which many of its modules execute and in which it keeps most of its data. Global resource serialization receives control in the address space that requests its services and then transfers control (via a PC instruction) to the global resource serialization address space to process the request.

  - Establishing the global resource serialization complex, as defined by the installation in the GRSCNFxx and IEASYSxx members of SYS1.PARMLIB.

  The initialization subcomponent invokes all subcomponents of global resource serialization except for resource request processing and dump support. System Initialization Logic describes the global resource serialization initialization modules, whose names follow the format ISGNxxxx.

- Command Processing. This subcomponent (module names of the format ISGCxxxx) supports the VARY GRS and DISPLAY GRS operator commands. The global resource serialization command interface (ISGCMDI) executes in the master scheduler address space and receives control from the command service processor (IEECB808) when a VARY GRS or DISPLAY GRS is detected. ISGCMDI posts the global resource serialization command router (ISGCMDR) in the global resource serialization address space. ISGCMDR routes control to the appropriate request processor:

  - ISGCQSC - QUIESCE processing. ISGCQSC removes an active system from the global resource serialization ring. Requests for global resources made prior to the QUIESCE will remain intact. ISGCQSC processes QUIESCE requests for the system on which the command is issued or for any other system in the global resource serialization ring.

  - ISGCPRG - PURGE processing. ISGCPRG removes a quiesced system from the global resource serialization complex. PURGE processing releases all global resources owned by the system being purged and deletes all outstanding requests for global resources made by that system. A PURGE request can only be processed on an active system in the global resource serialization complex.

  - ISGCRST - RESTART processing. ISGCRST rejoins a quiesced system with the global resource serialization ring or rebuilds a ring that has been disrupted. ISGCRST processes RESTART requests for the system issuing the command, for a specific quiesced system in the complex, or for all inactive systems in the complex. The topic "Adding a System to the Main Ring" (under "Ring Processing" later in this introduction) describes the processing that occurs to add a system to the global resource serialization (main) ring. This processing is shared between modules of the command processing subcomponent and the ring processing subcomponent.

  - ISGCDSP - DISPLAY processing. ISGCDSP displays the status of

    1. each system known to the global resource serialization complex

2. the CTCs that are assigned to global resource serialization and that are attached to the system on which the command was issued, including the status of the systems attached to this system via the CTCs

3. both 1 and 2

4. any resource contention

5. the RNLs

6. the resource qnames

7. the resources and their requestors

8. or the combination of 1, 2, 4, and 5

The command processing subcomponent invokes the following subcomponents:

- The queue scan subcomponent to obtain information about global resources.

- The ring processing subcomponent to obtain information to be displayed on the status of systems in the complex and CTCs assigned to global resource serialization and attached to the system on which the command was issued; to remove a system from the complex; and to vary the participation of systems in the global resource serialization ring.

- The WTO/WTOR message processing module (ISGMSG00) to communicate with the operator.

- Resource Request Processing. This subcomponent processes ENQ, DEQ, and RESERVE macro instructions. It also receives control during termination to purge all local and global resources acquired by the terminating task or address space.

  Processing of ENQ, DEQ, and RESERVE requests is divided into fast path and mainline processing: fast path processing (ISGLNQDQ) handles local ENQ and DEQ requests that meet certain eligibility requirements (requests that do not require specialized processing); mainline processing (module names of the format ISGGxxxx) handles local ENQ and DEQ requests ineligible for fast path processing, all global ENQ and DEQ requests, and all RESERVE requests. Global resource requests require communication among all active global resource serialization systems in the complex before the request can be satisfied; this subcomponent invokes the ring processing subcomponent to communicate with those systems. Both fast path (in some cases) and mainline processing invoke the storage management subcomponent to obtain the control blocks that represent the request.

- Ring Processing. The active systems in a global resource serialization complex are called a main ring or a global resource serialization ring. Ring processing modules (module names of the format ISGBxxxx) are responsible for (1) passing to all systems in the main ring the information they require to serialize global resource requests across all the systems in the main ring; and (2) adding or deleting systems from the main ring as specified in initialization parameters or requested by the operator via the VARY GRS command. Ring processing also provides information about the systems and CTCs in the global resource serialization complex - for example, their status or the system name associated with a particular sysid. (The topic "Ring Processing," later in this introduction, provides more information on how ring processing provides its functions.)

  The ring processing subcomponent invokes the CTC processing subcomponent to actually initiate I/O on the CTCs; the WTO/WTOR message processing module (ISGMSG00) to communicate

 LY28-1695-0 (c) Copyright IBM Corp. 1987

with the operator (for example, when a ring processing module needs to notify the operator of a problem in the main ring); the resource request processing subcomponent; and the storage management subcomponent to allocate control blocks.

- CTC Processing. This subcomponent (module names of the format ISGJxxxx) builds control blocks for the CTCs that connect systems in the global resource serialization complex (based on information specified in the GRSCNFxx PARMLIB member); initiates I/O on the CTCs; and handles interrupts on the CTCs. It invokes the ring processing subcomponent when it receives an interrupt on a CTC.

- Storage Management. The storage management modules (module names of the format ISGSxxxx) are responsible for managing the resource queue area (RQA) and the extended resource queue area (ERQA) of the global resource serialization address space. ISGNASIM allocates the RQA from the private area below 16 megabytes, and the ERQA from the private area above 16 megabytes, of the global resource serialization address space during initialization. The storage management modules allocate and deallocate storage in the RQA or in the ERQA, in one-page blocks called PEXBs (pool extent blocks). PEXBs in the RQA contain QWB, MRB, CRB, TWKA and HWKA cell types while PEXBs in the ERQA contain QCB, QEL, QXB and PQCB cell types.

  Each PEXB is divided into cells. There are different types of cells, each type accommodating a particular control block. In addition, different cell types are defined for a single control block that can vary in size. For example, three cell types are defined for QCBs: one to accommodate QCBs for resource names of 1-24 bytes; one for resource names of 25-52 bytes; and one for resource names of 53-255 bytes. Therefore, there is one cell type for each particular control block (or size range of a control block) allocated from the private area of the global resource serialization address space. A single PEXB contains only one type of cell and, therefore, only one particular control block, thereby reducing the amount of information required to assign or free cells in a PEXB. In addition, if a cell type is associated with a control block that exists for both global and local resources, a PEXB containing that cell type is used only for local resources or only for global resources, not for both.

  PEXBs containing cells associated with local resources are allocated from the low-address end of the RQA or the ERQA; PEXBs containing cells associated with global resources are allocated from the high-address end of the RQA or the ERQA. Resource pool tables (RPTs), a local RPT and a global RPT, are used to keep track of the allocated PEXBs. The local RPT contains an entry for each type of cell associated with local resources; the global RPT contains an entry for each type of cell associated with global resources. PEXBs that have been allocated for a single cell type are chained together and the RPT entry for that cell type contains pointers to the first and last PEXB in the chain.

  The storage management routines assign and release cells in PEXBs, allocating another PEXB if no PEXB of the requested cell type contains an available cell or deallocating the PEXB if the cell just released was the last assigned cell in the PEXB. When the number of deallocated PEXBs reaches a certain value (defined by global resource serialization), global resource serialization releases (via PGRLSE) the real storage associated with the deallocated PEXBs. Modules executing in the global resource serialization address space invoke the storage allocation routine (ISGSALC) and storage deallocation routine (ISGSDAL) directly. An interface module (ISGSMI) provides the interface to ISGSALC and ISGSDAL for routines not executing in the global resource serialization address space.

In addition, the storage management subcomponent provides hashing routines to expedite searches of queues for a requested resource or for the requests from a particular address space in a particular system. (The topic "Control Blocks Representing Serialization Requests," later in this introduction, illustrates the hash tables used by the hashing routines.)

- Queue Scan. Queue scan modules (ISGQSCAN and its recovery module ISGQSCNR) process the GQSCAN macro instruction. The queue scan module returns to the issuer of GQSCAN a collection of data from multiple sources. To do this, it invokes the following subcomponents:

  - Storage management to hash resource names to expedite the search for more information and to allocate and deallocate PQCBs (place holder QCBs), QELs, QXBs, and HWKAs (huge workareas), which contain the RIBs (resource information blocks) and RIBEs (RIB extensions) used to collect the required information.

  - The information services of the ring processing subcomponent to convert system names to sysids and sysids to system names.

- Dump Support. Because most of its key control blocks are in its own address space, global resource serialization provides its own dump support to dump the control blocks. The dump support modules (module names of the format ISGDxxxx) obtain and format information about local and global resources for SNAP dump, print dump (PRDMP), or interactive problem control system (IPCS), and provide a dump of most global resource serialization control blocks when the GRSQ parameter is specified on an SDUMP macro. The dump support subcomponent invokes the queue scan module (ISGQSCAN), via the GQSCAN macro instruction, to obtain data about local and global resources for a SNAP dump. Figure GRS-2 illustrates the subcomponents invoked for each interface/function global resource serialization provides.

Resource request processing is the primary function of global resource serialization, and ring processing is one of the more complex functions. The next topics describe the control blocks built to represent serialization requests (necessary background information for understanding request processing); the processing of ENQ, DEQ, and RESERVE requests; and ring processing.

CONTROL BLOCKS REPRESENTING SERIALIZATION REQUESTS

Global resource serialization receives the information it requires to process a request in a PEL (parameter element list). From data in the PEL, global resource serialization builds a QWB (queue work block) in SQA to represent the ENQ, DEQ, or RESERVE request. If the request is for a global resource, global resource serialization subsequently copies the QWB from SQA to the private area of the GRS address space. (Note that two routines copy QWBs: ISGGQWB0 and ISGGQWBC. ISGGQWB0 copies QWBs into or out of other data areas, such as from the ring system authority (RSA) message received via the CTC from another system. ISGGQWBC copies QWBs to QWBs, as from the SQA QWB to a QWB in the private area of the global resource serialization address space.)

From information in the QWB, global resource serialization creates the control blocks - QCBs, QELs, and QXBs - that it uses to satisfy the request. The ring processing modules pass to every system in the ring the QWBs for global resource requests from each system. (See the topic "Ring Processing" later in this introduction for more detail.) As a result, each system creates and chains QCBs, QELs, and QXBs that represent all global requests in the main ring and creates and chains QCBs, QELs, and QXBs for the local requests of this system only. The

 LY28-1695-0 (c) Copyright IBM Corp. 1987

following describes the role of each control block and the ways in which they are chained.

- A QCB (queue control block) describes the resource being requested; global resource serialization builds a QCB if one does not already exist for the resource. The QCB contains pointers to the previous and next QCBs that are accessible via a single entry (the QCB synonym chain) in the queue hash table (QHT). There are two queue hash tables: a local queue hash table and a global queue hash table. QCBs for global resources are chained from the global queue hash table; QCBs for local resources are chained from the local queue hash table.

- A QEL (queue element) describes the requestor (the ASID of the requestor and whether the requestor requires shared or exclusive control of the resource) and contains pointers that define the various queues of QELs:

  - The queue of QELs that represent requests for a single resource, pointed to by the QCB for that resource

  - The queues of QELs that represent the requests of a single address space. If the requests originated on this system, there is one queue for QELs requesting global resources and one queue for QELs requesting local resources for each address space. The ASCB for the address space is the anchor for both queues. If the requests originated on another system in the main ring (they represent global requests for an address space executing on another system), the queue of QELs is located by means of an entry in the SYSID/ASID hash table. Each entry in the SYSID/ASID hash table points to a QEL; that QEL points to (1) other QELs that have the same SYSID/ASID combination, and (2) the next QEL with a different SYSID/ASID combination that is accessible via this entry.

- A QXB (queue extension block) describes the ENQ request - for example, the address of the requestor's TCB; the ECB or SVRB to be posted when the request is satisfied; and, if the request specified more than one resource, the number of resources requested and the number of QELs waiting to receive control of requested resources.

Figure GRS-3 illustrates QCBs, QELs, and QXBs for global resources and the various ways in which they are chained.

Global resource requests represented by control blocks in system SYS1:

Request A	Request B	Request C	Request D
ASID 123	ASID 567	ASID 567	ASID 789
SYSID SYS1	SYSID SYS2	SYSID SYS2	SYSID SYS2
Resources requested: X,W	Resource requested: W	Resource requested: Y	Resource requested: Z

Control blocks for requests A-D in system SYS1:

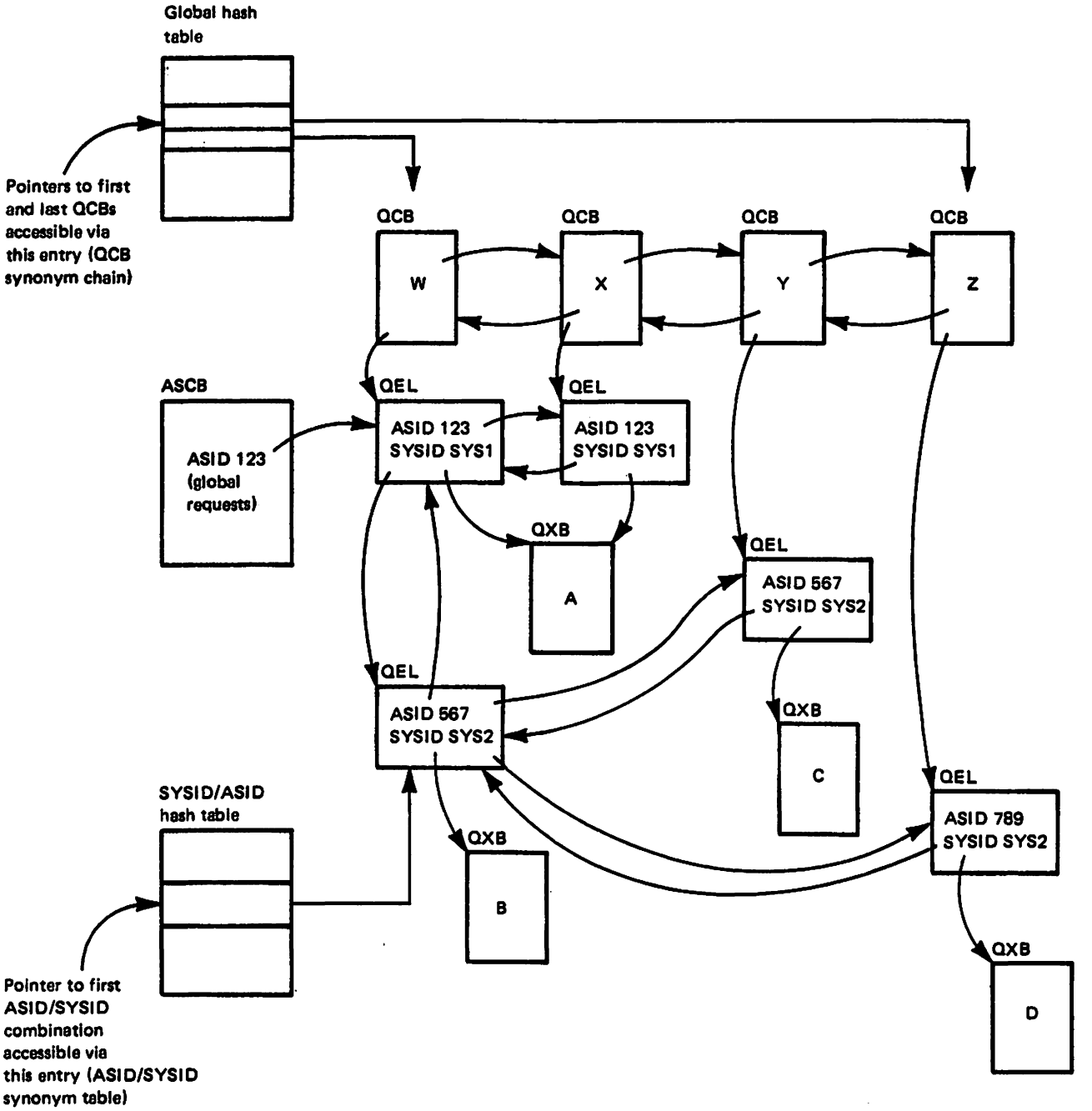


Figure 3. Example of Control Blocks for Global Serialization Requests

 LY28-1695-0 (c) Copyright IBM Corp. 1987

PROCESSING ENQ, DEQ, AND RESERVE REQUESTS

The general (and simplified) processing global resource serialization does to satisfy an ENQ request (whether the request is local or global) or a RESERVE request consists of the following steps:

1. Copies the PEL into a QWB.
2. Checks the resource name lists to determine if the resource requested is global or local.

   (For a global resource, delays the requestor and communicates with other systems in the ring before continuing.)
3. Builds and chains, if necessary, a QCB, QEL, and QXB to represent the request.
4. If the QEL is the first QEL on that QCB's QEL chain or if the QEL requested shared control and prior QELs also request shared control, grants the request. Otherwise, delays the requestor (by issuing WAIT) until the task for the QEL just created is posted (see the DEQ processing steps, described next); and then grants the request.
5. Returns to the issuer of the ENQ or RESERVE macro instruction via EXIT prolog.

The general (and simplified) processing done to satisfy a DEQ request (whether local or global) includes the following steps:

1. Copies the PEL into a QWB.
2. Checks the resource names lists to determine if the resource is local or global.

   (For a global resource, delays the requestor and communicates with other systems in the ring before continuing.)
3. Finds the QCB, QEL, and QXB that represent the request to be dequeued.
4. Frees the QEL and, if this is the last QEL associated with the request, also frees the QXB.

   If this is the last QEL for the QCB, frees the QCB. Otherwise, posts the TCB for the next QEL chained from the QCB (or, if the next and one or more subsequent QELs request shared control, posts the TCBs for those QELs).
5. Returns to the issuer of the DEQ macro instruction by means of EXIT prolog.

These steps expand for the variations in processing that occur for fast path versus mainline processing, for global versus local resources, and for special situations. For example:

- In step 2, fast path processing (which handles requests that require only streamlined processing) checks only the inclusion list and passes the request to mainline processing if it finds the resource name in the inclusion list, without checking the exclusion list. Mainline processing checks all applicable resource name lists.
- For global requests, ENQ/DEQ/RESERVE processing copies the SQA QWB to a QWB in the private area of the global resource serialization address space.
- Steal processing occurs in one exceptional case. When a resource is requested by a task that is part of an ABENDing task structure, and the resource is owned by another task in this same task structure, ENQ/DEQ/RESERVE processing

initiates a resource steal because the ABENDing task is not able to release the resource.

Variations such as these are described in the method-of-operation diagrams for the resource request processing modules (ISGLNQDQ and ISGGxxxx). Figure GRS-4 illustrates the module flow of the primary modules that receive control to process ENQ/DEQ/RESERVE requests.

By far the most significant variation in the simplified steps listed above is the necessity to communicate with other systems for global resource requests. Ring processing, which controls the communication, is described in the next topic.

LY28-1695-0 (c) Copyright IBM Corp. 1987

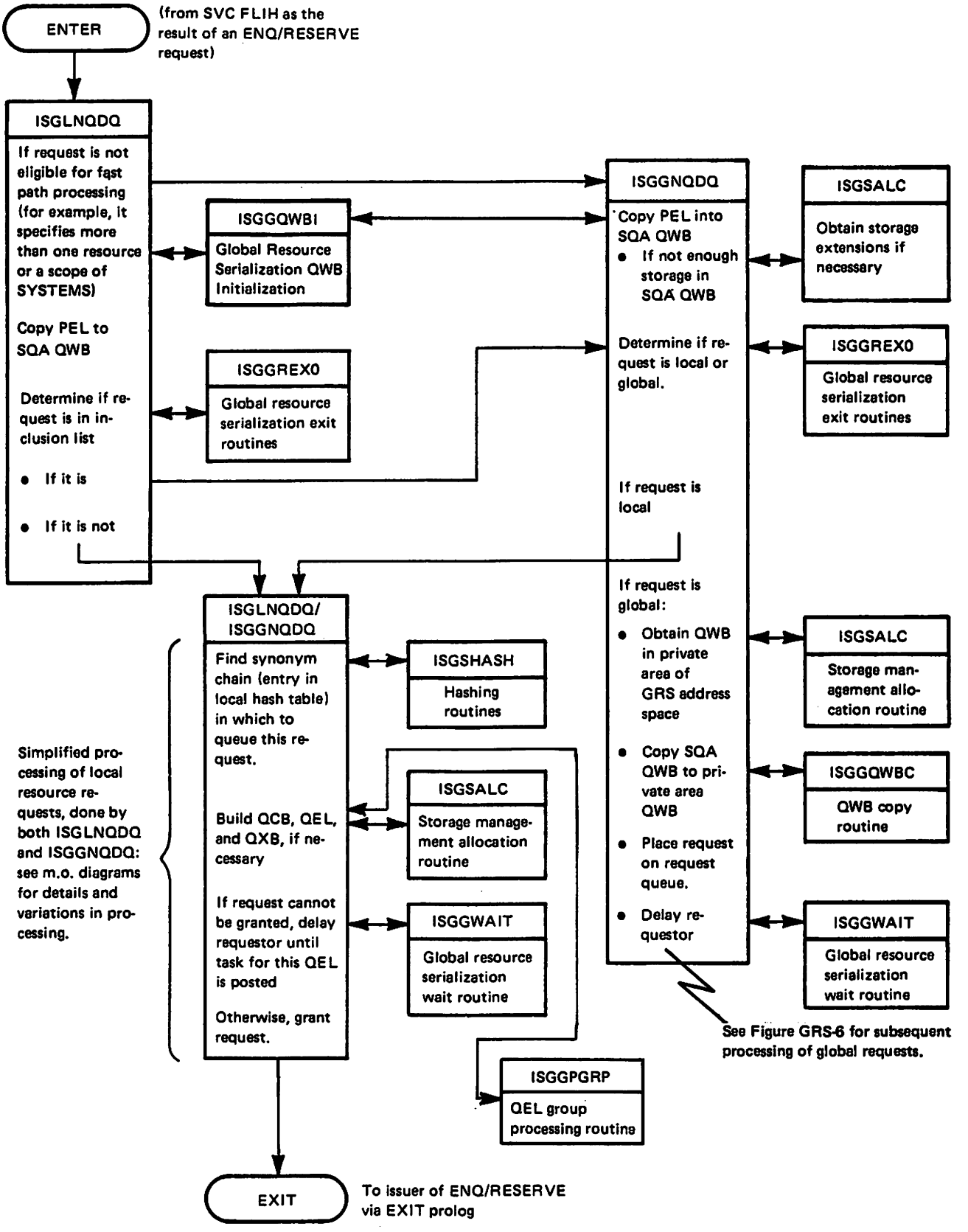


Figure 4. Simplified Process Flow for ENQ/RESERVE Processing

RING PROCESSING

Ring processing provides three main functions:

- It passes to all systems in the main ring the information they require to serialize global resource requests across all the systems in the main ring. See the topic "Serializing Global Resources."
- Working with the command processing subcomponent, it adds or deletes a system to or from the main ring, as specified in initialization parameters or requested by the operator via the VARY GRS command. The topic "Adding a System to the Main Ring" is a simplified overview of the add function.
- It provides to other subcomponents information about the systems in the main ring. See the topic "Providing Informational Services."

SERIALIZING GLOBAL RESOURCES

Global resource serialization achieves the serialization of global resources by duplicating the control blocks that represent requests for global resources in every system in the main ring. Every system contains QCBs, QELs, and QXBs, queued in identical order, that reflect every request made by a system for a global resource. Therefore, system A cannot grant a request to a requestor from system A if another QEL, representing a request from system B, precedes the QEL for system A - until it receives the DEQ request from system B for that QEL or unless both requests specify shared control.

Ring processing passes requests for global resources to all systems in the main ring by passing a message called the RSA (ring system authority) from system to system so that the RSA makes a complete circuit of the ring. Each system places its global requests, in the form of QWBs, in the RSA using one of two methods:

1. Compression level 1. Determined by the value 1 found in the QPLFCPRS field of the Queue Work Block Parameter list (QPL). It indicates the QWBs in the RSA are copies of the system QWBs and can contain unused bytes (non-compressed QWB).
2. Compression level 3. (Level 2 is not currently used.) Determined by the value 3 found in the QPLFCPRS field of the QPL. It indicates the basic section of the QWBs placed in the RSA is shortened and the Storage Management Parameter list (SMPL) section is shortened to accomodate only those fields which will vary (SMPCNUM). Thus more requests can fit in the RSA. (Compressed QWB)

(The RSA can also contain a command area that is used to send data or commands for the command processing subcomponent.) There is only one RSA, containing batches of QWBs placed there by each system; at any time, the RSA is either between systems on a CTC or at one of the systems. Basically, when the RSA arrives at a system, ring processing does the following:

1. Sets an RSA residency interval, the amount of time the RSA will reside in that system. (The RSA residency interval allows for the varying speeds of different processors in the ring and, therefore, prevents a faster processor from driving a slower processor.)
2. Invokes ISGGQWB to reproduce from the RSA copies of QWBs that this system placed in the RSA the last time the RSA resided in this system. These QWBs are copies of QWBs that originated on this system: they have made a complete circuit of the main ring and have been seen by all systems in the main ring. Therefore, this system can now remove them from the RSA.

 LY28-1695-0 (c) Copyright IBM Corp. 1987

3. Reproduce QWBs placed in the RSA by other systems into QWBs that this system obtains from storage management. These reproduced QWBs represent global requests that originated on other systems in the main ring.

4. Adds to the RSA QWBs for global requests from this system that have accumulated since the last RSA residency.

5. When the RSA residency interval expires, sends the RSA to the next system in the ring; that system then performs these same steps.

This processing actually involves four queues of QWBs. Each system contains the four queues and uses them as follows:

- The request queue. When mainline ENQ/DEQ/RESERVE processing determines that an ENQ, DEQ, or RESERVE specifies a global resource, it obtains a QWB for the global request in the private area of the global resource serialization address space. It then chains this global QWB on the request queue and delays the requestor (by issuing WAIT). The request queue is serialized by compare-and-swap logic and is last-in/first-out.

- The ring processing internal queue. Ring processing moves the QWBs placed on the request queue to its own internal queue (pointed to by the RSVQWBIF field in the ring processing system vector table (RSV)). The internal queue is first-in/first-out. Ring processing re-orders QWBs as necessary when it moves them from the request queue to the internal queue. When the RSA resides in this system, ring processing reproduces the QWBs from the internal queue into the output RSA (the RSA to be sent to the next system in the ring) and moves them to the sent queue, described next.

- The sent queue (also called the staging queue). To aid recovery if the RSA is lost, the sent queue provides a record of the QWBs sent in the output RSA to the next system in the ring. Ring processing places in the sent queue:

  1. QWBs from other systems that arrived at this system in the input RSA. Ring processing invokes ISGGQWB1 (INSYS-COPY) to reproduce other systems' QWBs from the input RSA. Ring processing places the QWBs on the sent queue. These QWBs remain in the output RSA and are sent to the next system as part of the output RSA.

  2. QWBs that this system places in the output RSA (therefore, QWBs that originated on this system and are being sent to the next system). When ring processing invokes ISGGQWB1 (OUTSYS-COPY) to reproduce QWBs from the internal queue to the output RSA. Ring processing moves those QWBs to the sent queue.

  Ring processing does these two steps each time the RSA resides in this system but after it moves the sent queue created during the previous RSA residency to the process queue, described next. (Once the RSA has made a complete circuit of the ring, there is no need to keep a record of the QWBs contained in the RSA that started that circuit of the ring.)

- The process queue. When the RSA arrives at a system, ring processing moves the sent queue created during the previous RSA residency to the process queue. Because of the role of the sent queue (described above), the QWBs on the process queue have made a complete circuit of the main ring. The process queue is the output from ring processing: the requests on the process queue are now ready for processing (building QCBs, QELs, and QXBs to represent those requests and attempting to grant those requests).

  QWBs appear on the process queue in the same order in which they were passed through the RSA; QWBs will appear in the

same order on the process queues of all systems in the main ring. This ensures that global resource QELs, created from the QWBs, are in the same order on all systems in the main ring.

Combining the updating of the queues with the simplified steps (listed earlier) that occur when the RSA arrives at system results in the following sequence. (Figure GRS-5 illustrates the queues and the input and output RSA; the circled numbers in Figure GRS-5 refer to the following steps.)

1. The RSA arrives and ring processing sets the RSA residency interval.

2. Ring processing moves the current sent queue (created during the previous RSA residency) to the process queue.

3. Ring processing removes from the input RSA this system's QWBs, which were reproduced into the RSA during the previous RSA residency.

4. Ring processing invokes ISGGQWB1 (INSYS-COPY) to reproduce other systems' QWBs from the input RSA. Ring processing places QWBs in the sent queue.

5. Ring processing moves QWBs from the request queue (global requests that originated on this system since the previous RSA residency) to the internal queue.

6. Ring processing invokes ISGGQWB1 (OUTSYS-COPY) to reproduce QWBs on the internal queue to the output RSA. Ring processing moves QWBs to the sent queue.

7. When the RSA residency expires, ring processing sends the output RSA to the next system in the ring.

Once ring processing has built the process queue, it posts ISGGRP00, which processes the requests represented by the QWBs on the process queue and, therefore, builds QCBs, QELs, and QXBs for all global requests in the ring. Figure GRS-6 illustrates the modules that receive control to process requests for global resources.

During ring processing the following exceptional conditions can occur that cause a main ring failure and require the ring processing exception handling task (code that is part of ISGBTC):

- Condition A

  The RSA fails to complete a full circuit of the main ring within the time allowed. (Entry point ISGBDRM of ISGBDR gets control through periodic timer interrupts to detect this condition.)

- Condition B

  An I/O error occurs on a CTC assigned to the global resource serialization main ring. (The CTC processing subcomponent of global resource serialization detects this I/O error.)

- Condition C

  A status inquiry request arrives from a system at the opposite end of a global resource serialization CTC. (The CTC processing subcomponent of global resource serialization detects this event; a SNAPSHOT, performed by the system at the opposite end of the CTC, causes the status inquiry to occur.)

When any of these conditions occur, the GVTXBECB ECB in the GVTX is posted. This post activates the ring processing exception handling task; this task processes these exceptional conditions as follows:

 LY28-1695-0 (c) Copyright IBM Corp. 1987

- Condition A Response

  The ring processing exception handling task writes messages to the operator that report the main ring failure and issues a VARY GRS command to automatically rebuild the disrupted ring.

- Condition B Response

  The ring processing exception handling task writes a message to the operator that identifies the I/O error and issues a VARY OFFLINE command to vary offline the CTC that encountered the I/O error. (This I/O error can cause the condition A main ring failure to occur and subsequently cause the condition A response described above.)

- Condition C Response

  This task sends an RSAIRCD to the system at the opposite end of the CTC; this RSAIRCD contains the name and status of the sending system. (In some cases, the ring processing exception handling task issues a VARY ONLINE for the CTC that received the status inquiry request.)

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

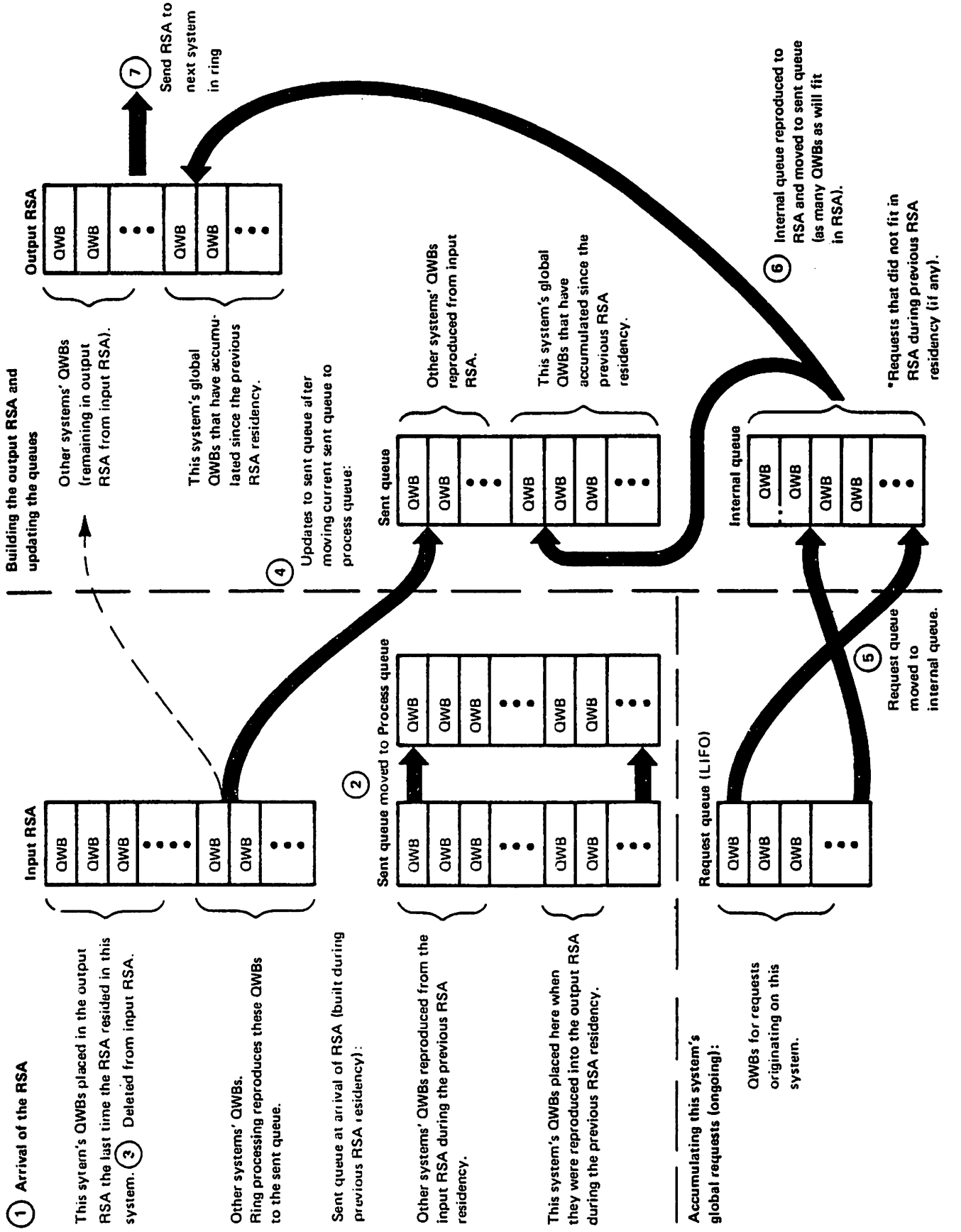


Figure 5. Updating the RSA and Ring Processing Queues

LY28-1695-0 (c) Copyright IBM Corp. 1987

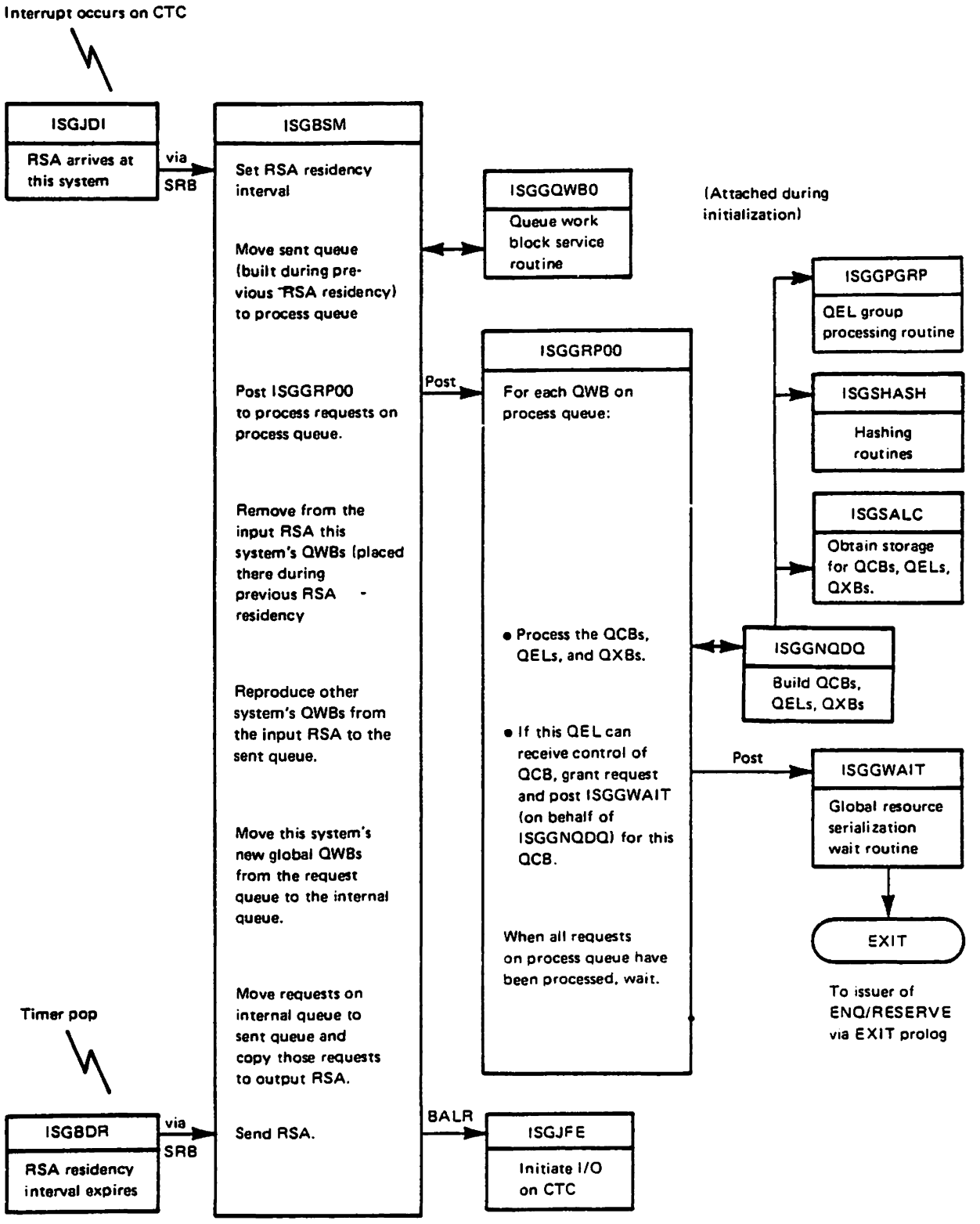


Figure 6. Simplified Process Flow for Global ENQs (Ring Processing)

ADDING A SYSTEM TO THE MAIN RING

When adding a system to the main ring, global resource serialization must ensure that the global resource queues in the system entering the ring are identical to the global resource queues in the other systems in the main ring. One active system in the main ring (subsequently called the adding system) is responsible for adding to the main ring the system that wants to join the main ring (subsequently called the entering system). Understanding the processing that occurs on the adding system and the entering system requires an understanding of the following:

- The RSAIRCD (ring system authority identity record). The RSAIRCD is a small record of control information that is passed back and forth across a CTC that connects the adding system and entering system. The RSAIRCD can be sent across CTCs that are not used to pass the RSA. The RSAIRCD is used only to pass commands and status information needed to add a system to the main ring; it cannot be used to pass global serialization requests.

- The RSVENTY table (ring processing system vector table entry table, mapped by the mapping macro for the RSV). The RSVENTY table contains an entry for every system defined to the global resource serialization complex. Each entry contains a flag that indicates if the system is part of the main ring.

- Save-QWB mode and the hold queue. When a system enters save-QWB mode, it (1) stops placing global requests that originate on that system into the output RSA (the requests remain on the internal queue); and (2) moves the sent queue to a hold queue instead of to the process queue. The system does not create QCBs, QELs, and QXBs for QWBs on the hold queue until the system leaves save-QWB mode; at that time, the system moves the QWBs from the hold queue to the process queue.

The processing done on the adding system and entering system includes the following steps; responsibility for executing these steps is shared between the command processing subcomponent and ring processing. (Note that this processing is simplified; it focuses on the steps necessary to ensure that the entering system's global resource queues will match the queues in other systems in the main ring.)

1. The entering system enters save-QWB mode. This step is part of the SENDCMD-RSCRADDS function of ISGBCI. ISGBCI invokes ISGBRF (entry point ISGBRFNM) to handle the SENDCMD-RSCRADDS function. Once the system has entered the main ring and starts to receive and send the RSA (step 6), it will move the sent queue to the hold queue, not to the process queue.

2. Using the command area of the RSA, the adding system sends each system currently in the main ring an RSVENTY table entry for the entering system.

3. The adding system instructs all systems currently in the main ring to stop adding requests (global QWBs) to the RSA.

4. When the RSA is empty of QWBs, the adding system sends the RSVENTY table, one entry at a time in the RSAIRCD, to the entering system.

 LY28-1695-0 (c) Copyright IBM Corp. 1987

5. The adding system enters save-QWB mode. When systems resume adding QWBs to the RSA, it will move the sent queue to the hold queue, not to the process queue.

   Note: Steps 2, 3, 4, and 5 are part of the ADDSYS function of ISGBCI.

6. Once steps 3 and 4 complete, the entering system and all systems currently in the main ring have an RSVENTY table that defines the new main ring (including the entering system). The entering system begins to receive and send the RSA. All systems in the new main ring, except for the adding and entering systems (which are still in save-QWB mode), resume sending QWBs in the RSA.

7. Because the entering system is still in save-QWB mode (step 1), it places the QWBs it receives in the RSA on its hold queue. Although it is receiving new global requests (assuming there are other systems in the ring other than the entering and adding systems), its existing global resource queues (QCBs, QELs, and QXBs) might not match the other systems' queues. (If this is the first time the entering system has entered the main ring, its queues will be empty.) However, because the adding system has also entered save-QWB mode (step 5), its queues represent the global queues current at the time the entering system entered the ring. The adding system issues the GQSCAN macro instruction for all global resources and sends the results (using the BUFSEND function of ISGBCI) to the entering system.

8. The entering system (1) issues a GQSCAN macro instruction to search its own global resource queues for each global resource identified in the data received from the adding system; and (2) compares the results to the data received from the adding system (the results of the GQSCAN macro issued on the adding system). The entering system generates QWBs to eliminate differences in the data (and, therefore, in the global resource queues) and places the generated QWBs at the beginning of the process queue.

9. Both the adding system and the entering system leave save-QWB mode. Requests placed on the hold queue move to the process queue (after any generated QWBs on the entering system's process queue). When the entering system creates QCBs, QELs, and QXBs for the requests on its process queue, the resulting global resource queues will match the queues of other systems in the main ring.

PROVIDING INFORMATIONAL SERVICES

Some global resource serialization modules call ring processing modules for information only:

- To convert a sysid to a system name or vice versa.
- To obtain the status of systems in the complex and of the CTCs that are assigned to global resource serialization and attached to the system that requested the information.

A sysid is a numerical synonym for a sysname (system name). Sysids range from 1 through 255 and are associated with every global resource. (The sysid for a local resource is 0.) The sysid occurs in certain global resource serialization control blocks (such as QELs and QWBs). Ring processing maintains the correspondence between sysnames and sysids and provides routines to convert a sysname to a sysid and vice versa.

Ring processing records the status of CTCs in RSLs (ring processing system link blocks). There one RSL in each system for each CTC attached to that system and assigned to global resource serialization. Ring processing records the status of systems in the RSVENTY table. Ring processing coordinates each system's updates to its RSVENTY table so that the RSVENTY table

in each main ring system provides the same status information. Ring processing achieves this coordination by passing information in the command area of the RSA.

 LY28-1695-0 (c) Copyright IBM Corp. 1987

DIAGNOSTIC TECHNIQUES

The following topics contain diagnostic aids to help you solve problems with global resource serialization.

DEBUGGING HINTS

CHECK ON ENABLED WAIT DURING IPL

If an enabled wait occurs during IPL processing, you can make the following check to determine if the wait was due to missing entries in the SYSTEMS exclusion RNL.

- Check the request queue in the GVT (CVTREQQ) for QWBs.
- Compare the resource name identified in the PEL portion of the QWB to the entries in the SYSTEMS exclusion RNL and SYSTEM inclusion RNL.
- If the RNLs indicate that the resource name identifies a global resource, the requester of that resource must wait until master scheduler initialization completes before the requester is granted control of the resource.
- If the requester must complete processing prior to master scheduler initialization completing, the resource name must be added to the SYSTEMS exclusion RNL.

PROBE POINTS

The following probe points are useful to help you debug global resource serialization problems or set SLIP traps.

1. Probe point for obtaining the RSA message that this system received:

   Module: ISGBSM
   Label: RECVTP1
   Data: - RSAPTR (register 6) points to the RSA.
   - Register 4 contains the length of the RSA.
   - Register 13 points to the RSV.
   - RSVIBFOR (RSV+X'8C') points to the received RSA.

2. Probe point for obtaining the RSA message that this system sent:

   Module: ISGBSM
   Label: SENDTP1
   Data: - Register 13 points to the RSV.
   - RSVOBFOR (RSV+X'90') points to the sent RSA.

3. Probe point for obtaining the QWB that is to be processed (the first QWB on the process queue):

   Module: ISGGRP00
   Label: GRPNXTPQ
   Data: - Register 3 points to the GVT.
   - GVTPRCQF (GVT+X'40') points to the QWB to be processed.

USEFUL FIELDS IN THE GVT AND THE GCL

The following indicators, when set to one, have these meanings:

GVT indicators:

GVTGRSNA - Global resource serialization is not active. (Only local requests can be processed.)

GVTNCMDR - Global resource serialization commands cannot be processed.

GVTGQDMG - Global resource queues have been damaged. This system will reject VARY GRS,RESTART commands.

GVTNCOMM - CTC-driver and ring processing functions are not operative.

GVTNREQS - Requests cannot be put on the command request queue.

GCL indicators:

GCLINOP - CTC processing will not allow use of this CTC because a software error occurred and the control blocks of this CTC (GCL or RSL) might be damaged.

GCLIOERR - CTC processing will not allow use of this CTC because an I/O error occurred on this CTC.

GCLOFFLN - CTC processing will not allow use of this CTC because the CTC has been varied offline.

CTC PROCESSING DEBUGGING HINTS

The following debugging hints help you isolate problems in the CTC processing subcomponent.

1. Field GCLWGCQF of the GCL is the write queue of the corresponding GCL (representing a CTC) and points to a write GCQ when the write queue is not empty. GCLWGCQF is zero when the write queue is empty.

2. Field GCLCNTS is bumped by one before the STARTIO for a SENDBUF or SENDBUF-IMMEDIATE. Field GCLCNTC is bumped by one when the SENDBUF or SENDBUF-IMMEDIATE completes. Therefore, by comparing these two count fields you can determine if a write operation is in progress.

3. Field GCLRGCQF is the read queue of the corresponding GCL and points to a read GCQ when the read queue is not empty. GCLRGCQF points to a dummy GCQ (located in the GCV) when the read queue is empty.

4. The address in field GCLRGCQF is a word-multiple address when the GCL does not have a read channel program in progress. The address is bumped by one when a read channel program is started. Therefore, by checking the low order bit in GCLRGCQF you can determine if a read channel program is in progress.

5. Field GCLTRACE contains the last 15 CCW operation codes sensed from the corresponding CTC. In a dump, the acronym TRC1 appears a short distance before this field. The occurrence of an EE or ED operation code in this area indicates that the system taking the dump sensed a broken channel program that was started by the system at the opposite end of the CTC.

 LY28-1695-0 (c) Copyright IBM Corp. 1987

RING PROCESSING DEBUGGING HINTS

The following debugging hints help you isolate problems in the ring processing subcomponent.

1. Field RSVIBFOR points to the RSA input buffer. Field RSVMRLRL contains the length of the last RSA received.

2. Field RSVOBFOR points to the RSA output buffer. Field GCBLNBUF of the RSA output GCB contains the length of the last RSA sent or the length of the RSA that soon will be sent. Field RSVGCBOP points to the RSA output GCB.

3. Field RSARCSEQ of the RSA is the RSA send count, which is a number that is bumped by one each time the RSA is sent. By comparing RSARCSEQ in the input buffer to RSARCSEQ in the output buffer, you can determine if the system that took the dump was holding the RSA at the time of the dump. Also, by comparing RSARCSEQ values in dumps taken by different systems, you can determine which system last received the RSA before a failure.

4. When a system is in the main ring, field RSVRSASC contains the RSA send count of the last RSA sent by this system (if the system is not holding the RSA) or the send count of the RSA that will soon be sent by this system (if the system is holding the RSA). RSVRSASC is set to zero when a system does main ring cleanup.

5. Subroutine CLNUFAIL (in module ISGBCI) does the main ring cleanup. When a system does main ring cleanup after a main ring disruption, CLNUFAIL copies field RSVRSASC to an entry in the RSVENTY table, and also marks entries in the RSVENTY table to show which systems were in the main ring at the time of the disruption and which RSA was last received before the disruption. Because main ring cleanup is serialized by the ISGBCI-ENQ-resource, cleanup might not occur immediately after the main ring disruption because another task might be holding the ISGBCI-ENQ-resource at the time of the disruption.

ENQ/DEQ/RESERVE PROCESSING DEBUGGING HINTS

The following debugging hints help you isolate problems in the ENQ/DEQ/RESERVE processing subcomponent.

1. The queue work areas (QWAs) used by ENQ/DEQ mainline processing contain information that is useful in solving ENQ/DEQ/RESERVE problems. There are two QWAs: one for local resource processing (the local QWA pointed to by GVTLQWA), and the other for global resource processing (the global QWA pointed to by GVTGQWA).

   The QWA is divided into the following major areas:

   QWABASIC - This is the basic section of the QWA. It contains the information required by the mainline routine to process the resource request. For example, it indicates whether or not the request is authorized, whether global resources are part of the request, and whether the request is an ENQ or DEQ. This is also the only section of the QWA that can be mapped to the SVRB extended save area or the RMPL work area.

   QWARSA - This is the first request save area section of the QWA. It contains the information required to process a global or local resource request. This section is moved to the QWBHRSA field and later restored to the QWARSA field by module ISGGRP00. It exists in the QWABASIC section of the QWA.

QWARSA2 - This is the second request save area section of the QWA. It contains the information needed to process a global or local resource request. This section contains the requester's job name, SYSID, ASID, and ASCB address. This data is moved to the QWBHRSA2 field and later restored to the QWARSA2 field by module ISGGRP00. It exists in the QWARSA section of the QWA.

QWARDA - This is the request data area section of the QWA. It contains the counts of the types of resources being processed, and the addresses of internal control blocks.

Work/Save

areas - This series of general work/save areas follows the QWARDA area in the QWA and are used by the resource request processing routines. These areas are used to save register contents.

QWATRMRM - This work area section of the QWA is used by the termination resource manager. It contains information used by ISGGTRM0 and ISGGTRM1 to process a termination request.

When a local resource is being processed, the QWABASIC section of the QWA is moved to the SVRB extended save area when the requester of the resource must be suspended because the resource is not immediately available. QWABASIC information is then referenced in the SVRB extended save area following the notification that the resource is available.

When a global resource is being processed, the QWABASIC section of the QWA is always moved to the SVRB extended save area because the global resource requester is always suspended.

After the requester is notified (via cross memory post) that the requested resource is available, the data in the SVRB extended save area is copied back to the QWABASIC section of the QWA. This information in QWABASIC is then used to complete the processing of the request.

The main point to consider about the QWA is that whenever an ENQ/DEQ/RESERVE requester is suspended, the SVRB extended save area contains useful information that can be used in debugging. An important piece of information in the QWABASIC section of the SVRB extended save area is the QWB address used to define a global resource request. By locating this QWB (pointed to by QWAQWBA), you can find the data presented to ENQ/DEQ/RESERVE processing in the original request. If this field in the QWA is zero, then a local resource is being processed.

2. ENQ/DEQ/RESERVE processing uses two types of QWBs to process resource requests: the SQA QWB (pointed to by GVTSQWB), and the global resource serialization address space QWBs (pointed to by QXBQWB, GVTREQQ, and GVTPRCQF).

   When a local resource is being processed, the SQA QWB is used. When a global resource is being processed, the SQA QWB is used only until the global resource serialization private area QWBs are constructed. The following shows the process in which the resource data is passed between ISGGNQDQ and ISGGRP00.

   - The requester's PEL is moved to the SQA QWB.
   - The local QWA is initialized.
   - Information in the QWA and SQA QWB is moved to the global resource serialization private area QWBs.

 LY28-1695-0 (c) Copyright IBM Corp. 1987

- The QWABASIC section of the local QWA is moved to the SVRB extended save area.
- The global resource serialization private area QWBs are placed on the request queue. (These QWBs are subsequently moved to the process queue by ring processing routines.)
- The ring processing function notifies ISGGRP00 that work (QWBs) is now available on the process queue.
- ISGGRP00 moves the QWBHRSA and QWBHRSA2 fields to the global QWARSA and QWARSA2 fields respectively.
- ISGGRP00 processes the requests and notifies the requester (ISGGNQDQ SVRB) when the resource request is satisfied.
- ISGGNQDQ restores the local QWA from the QWABASIC section of the SVRB extended save area. It then locates the global resource serialization private area QWBs defining this request from the restored QWABASIC section. This address is then used to restore the QWARSA from the QWBHRSA.

3. Prior to master scheduler initialization completing, any global resource requests placed on the request queue that are required for IPL processing will cause an enabled wait state. To prevent this from occurring, any global resource requests required during IPL processing before master scheduler initialization has completed should be placed in the SYSTEMS exclusion RNL.

ENQ/DEQ/RESERVE TERMINATION RESOURCE MANAGER DEBUGGING HINTS

The following debugging hints help you isolate problems in the ENQ/DEQ/RESERVE termination resource manager function:

1. For normal and abnormal task termination, ISGGTRM0 receives control from RTM in either the address space of the terminating task or the address space of the master scheduler. In either case, ISGGTRM0 issues a PC to ISGGTRM1 in the global resource serialization address space to process the request. The input resource manager parameter list (RMPL, which is pointed to by register 1 on entry) defines the type of termination request.

2. ISGGTRM0 uses the local QWA to store information related to its processing. QWABASIC is initialized with common resource processing information and QWATRMRM is initialized with information related to the task or address space being purged. For the format of this data, refer to the QWA in the Debugging Handbook.

3. If only local resources are being purged, the ENQ/DEQ cross memory services lock (CMSEQDQ) is held to provide serialization for the local QWA.

4. If global resources need to be purged, then the data stored in the QWA must be preserved during this process. ISGGTRM1 saves this data in the dynamic area before calling ISGGQWB5. Register 9 in ISGGTRM1 points to the dynamic area. The information in the dynamic area includes the QWARSA, QWAASCB, QWATRMRM, QWAJOBNM, GVTXLSMP, and RUB (register updated block).

STORAGE MANAGEMENT DEBUGGING HINTS

The following debugging hints help you isolate problems in the storage management subcomponent.

1. Most global resource serialization control blocks reside in the global resource serialization address space. Pools of control blocks are maintained in resource pools as defined by two resource pool tables (RPTs), the local RPT and the global RPT. RPTs, in turn, address pool extension blocks (PEXBs) that define the control blocks (cells) for global resource serialization. (For an overview of these control blocks, see Figure GRS-15.)

   Each PEXB is 4K bytes in length and contains multiple cells for control blocks of the same type and size. PEXBs of QWB, MRB, CRB, TWKA, and HWKA cell types are contained in the RQA, while PEXBs of QCB, QEL, QXB, and PQCB cell types are contained in the ERQA. Listed below are the global resource serialization control blocks that are defined within a PEXB. (The RPT indexes are described in the following hint.)

Control Block	RPT Index	Name	Attributes
QCB	1	queue control block size 1	local or global
QCB	2	queue control block size 2	local or global
QCB	3	queue control block size 3	local or global
QEL	4	queue element	local or global
QXB	5	queue extension block	local or global
QWB	6	queue work block	global only
HWKA	6	huge work area	local only
TWKA	7	tiny work area	local or global
PQCB	8	placeholder QCB	local or global
MRB	9	message request block	local or global
CRB	10	command request block	global only

   The RPT header contains either the acronym LRPT (local RPT) or GRPT (global RPT). Also, in the PEXB headers, the PEXBs addressed by each RPT contain the acronym PEXB as well as the acronym for one of the control blocks listed above. This information is useful when you are scanning the RQA or the ERQA in a dump listing to locate a particular control block, or when you find an address of an unknown control block. From the information in the PEXB, you can determine the type of control block (defined by the acronym) and whether or not the control block is in use by global resource serialization. The control block is in use if it is not chained to the available cell chain in the PEXB header.

   The available chain is double-headed (PEXFRST and PEXLAST) and single-threaded (PEXNCELL). Note that the first four bytes of each cell are used to chain available cells together.

2. A storage manager parameter list (SMPL) is the input to the storage manager allocation (ISGSALC) and deallocation (ISGSDAL) routines. The SMPL describes the number and type of control blocks requested. The type of control block is defined by an RPT index value in the SMPL. The RPT indexes (defined in the ISGRPT and ISGSMPL mapping macros) are used to index into the RPT to locate the RPT entry (RPTE) for the control block in question.

3. The QCB is defined in three sizes: size 1 for those with an RNAME of 24 bytes or less, size 2 for those with an RNAME of 52 bytes or less, and size 3 for those with an RNAME of 255 bytes or less. Each QCB has a unique index corresponding to the three sizes.

 LY28-1695-0 (c) Copyright IBM Corp. 1987

4. The sequence in which the storage manager allocates control blocks is:

   - When the request is received, ISGSALC checks that the caller has the proper lock needed to allocate the cells. If the caller does not hold the proper lock then the storage manager issues ABEND 09A with a reason code of 8110 if the global resource serialization lock is not held, and a reason code of 810C if the CMS enq/deq lock is not held.

   - If the global resource serialization address space is initialized, ISGSALC checks if the caller is in 24 bit mode and the request is to allocate cells in the ERQA. If so, the storage manager issues an ABEND of 09A with a reason code of 8114.

   - ISGSALC attempts to satisfy the request from the queue of active PEXBs that are chained from RPTEFPXB and RPTELPXB. If, while scanning the active PEXB queue, ISGSALC finds a PEXB with no available cells, the PEXB is rechained to the end of the active PEXB queue.

   - If sufficient PEXBs are not available on the active queue, ISGSALC searches the inactive PEXB queue that is chained from RPTEIAPQ. If available, the inactive PEXB is moved to the front of the active PEXB queue and the required cells are obtained from this PEXB.

   - If the inactive PEXB chain is empty and the request is still not satisfied, an additional page is obtained from the RQA for QWB, HWKA, TWKA, MRB, or CRB cell type request, or from the ERQA for QCB, QEL, QXB, or PQCB cell type request. A new PEXB is then constructed and chained to the front of the active queue.

   - If the RQA has been completely assigned, then the storage manager issues ABEND 09A with a reason code of 8104. If the ERQA has been completely assigned, then the storage manager issues ABEND 09A with a reason code of 8108.

5. A bit map in the RQA defines each page of the RQA, and a bit map in the ERQA defines each page of the ERQA. When the storage manager attempts to allocate a control block and no active or inactive PEXB is found, the RQA/ERQA bit map is searched for an available page. (The address of the RQA bit map is in GVTXBTMP and the length of the RQA bit map is in GVTXBTML. The address of the ERQA bit map is in GVTXEBMP and the length of the ERQA bit map is in GVTXEBML). The storage manager allocates control blocks from the high end of the RQA/ERQA for global resources and the low end for local resources. Therefore, for global resources, the search proceeds from the high order bit in the bit map to the low order bit. For local resources, the search proceeds from the low order bit in the bit map to the high order bit. When a page is allocated in the RQA/ERQA, the corresponding bit in the bit map is set to 1. When a page is deallocated from the RQA/ERQA such as a PEXB, the corresponding bit in the bit map is set to 0. By scanning the bits in the bit map, you can determine the number and locations of all allocated control blocks in the RQA/ERQA. (The address of the RQA/ERQA is in GVTXRQA/GVTXERQA.)

6. You can locate a PEXB header by zeroing the low order 12 bits of the cell (or control block) address. The PEXB header contains the addresses of the first and last available cells in this PEXB. The header also contains pointers to the previous and next PEXBs for this control block. By scanning the queue of available cells (pointed to by PEXFRST), you can determine if a particular control block is allocated to a function or has been released.

When cells are returned to the storage manager, they are placed at the end of the available chain. When cells are assigned by the storage manager, they are assigned from the front of the queue. This ensures that a history of cell usage is maintained within the PEXBs because the oldest are used first.

When all cells within a PEXB have been freed, the PEXB is moved to the front of the chain of available PEXBs (that is, the inactive PEXBs pointed to by RPTEIAPQ). Therefore, a history of PEXBs is not maintained. Whenever the count of inactive PEXBs (maintained in GVTXIACT) equals the count in RPTIACNT, all inactive PEXBs defined by this PRT are released. The storage manager deallocation routine (ISGSDAL) schedules ISGSPRLS to perform the page release function (via the PGSER macro).

7. Control blocks in the RQA/ERQA are not fixed. Instead, global resource serialization relies on the storage isolation function of SRM to ensure that the real frames associated with these virtual pages remain in storage until a critical storage shortage is encountered. (Refer to Initialization and Tuning for information about storage isolation.)

8. With the exception of the QWB, all global control blocks are serialized with the global resource serialization local lock. All local resources and the QWB are serialized with the ENQ/DEQ cross memory services lock (CMSEQDQ).

SDWA AND SDWAVRA CONTENTS

All global resource serialization recovery routines (except ISGGEST0) record the following information in the SDWA:

SDWAMODN - Load module name
SDWACSCT - CSECT name
- Date of compilation
- Product/PTF number
SDWACID - Component identifier (SCSDS)
- Subcomponent identifier
SDWAREXN - Recovery routine name

Additional information is recorded in the variable recording area (SDWAVRA) in the key-length-data format as described in the following topics.

Recorded by ISGBERCV

ISGBERCV records the following in the SDWAVRA:

- The REPL and its address. (The REPL contains execution footprints. Also, if the failing module was working with a particular RSL, the REPL contains the address of the that RSL.)

- The RSC being processed at the time of failure and its address. (Recorded only if (1) ISGBC1 and (2) ISGBRF or ISGBSF was the failing module.)

- Six words copied from the UCB of the CTC that encountered the timeout condition. (Recorded only if ISGBCI is the failing module and the ABEND reason code is 620C.)

Recorded by ISGBFRCV

ISGBFRCV records the following in the SDWAVRA:

- The RVR and its address. (The RVR contains execution footprints. Also, if the failing module was working with a particular RSL, the RVR contains the address of that RSL.)

 LY28-1695-0 (c) Copyright IBM Corp. 1987

- The ISGBSR or ISGBSM entry point that encountered the failure.
- The addresses of the RSLs used to receive and send the RSA.
- Field RSVCRSAT of the RSV, which indicates whether a ring processing function was being performed at the time of the failure. Also. field RSVCPHNO, which indicates the phase of the function being performed.
- The addresses of the RSA input buffer and output buffer, plus six words from the beginning of each buffer.

If the failure occurred for entry point ISGBSRRI, the following is also recorded:

- The address of the RSL.
- The device address of the CTC represented by that RSL.
- The RSL flags: RSLLKSF, RSLLKIF, and RSLBFCTC.

Recorded by ISGCRCV

ISGCRCV records the following in the SDWAVRA:

- The contents of the CRWALEIB field (LOGREC error information) when ISGCRCV beings recovery processing.
- The parameter list passed to ISGBCI if the error exit routine determined that the failure occurred during a call to ISGBCI. (ISGCRCV invokes exit routines in failing modules as a part of its recovery processing.)
- The contents of the CRWALEIB field when ISGCRCV completes processing.

For each CRWA on the chain, ISGCRCV repeats the recording noted above. Therefore, multiple CRWALEIB fields might be recorded.

Recorded by ISGCRET0

ISGCRET0 (at entry point ISGCR0RV) records the following in the SDWAVRA:

- The FRR parameter list. (Refer to the PARMAREA structure in module ISGCRET0.)

Recorded by ISGCRET1

ISGCRET1 (at entry point ISGCR1RV) records the following in the SDWAVRA:

- The FRR parameter list. (Refer to the PARMAREA structure in module ISGCRET1.)

Recorded by ISGDSDMP

ISGDSDMP (at entry point ISGDSDRV) records the following in the SDWAVRA:

- The contents of the DEPL (ESTAE parameter list for SDUMP).

Recorded by ISGDSNAP

ISGDSNAP (at entry point ISGDSNRV) records the following in the SDWAVRA:

- The ESTAE parameter list. (Refer to the PARMAREA structure in module ISGDSNAP.)

Recorded by ISGGEST0

ISGGEST0 does not request recording to SYS1.LOGREC. Nothing is copied into SDWAVRA.

Recorded by ISGGFRR0

ISGGFRR0 records the following in the SDWAVRA:

- The contents of the QFPL (ENQ/DEQ FRR parameter list).
- The contents of the output data area (ODA) if the queue verifier routine detects queue damage. (Refer to module IEAVEQV0 for the mapping of the ODA.)
- Internal processing flags. (Refer to the FLAGS structure in module ISGGFRR0.)
- Resource damage flags. (Refer to the DAMAGE structure in module ISGGFRR0.)

Recorded by ISGGQSRV

ISGGQSRV (at entry point ISGGRECV) records the following in the SDWAVRA:

- The error information block (EIB) (local to ISGGQSRV).

Recorded by ISGJRCV

ISGJRCV records the following in the SDWAVRA:

- The CTC unit address.
- The address of the IOSB.
- The IOSB fields: IOSFLA, IOSFLB, IOSFLC, IOSCOD, IOSCSW, IOSSNS, and IOSUSE.
- The address of the GCQ.
- The first five words of the GCQ.
- The contents of GCL.

Recorded by ISGQSCNR

ISGQSCNR records the following in the SDWAVRA:

- The contents of QFPL1 (queue scanning services FRR parameter list).
- The input parameter list (built by the GQSCAN macro) to ISGQSCAN, if it is available.
- The original system completion code and reason code describing the error.
- The control block cell type and address, if the control block was found not valid.
- Internal recovery status flags. (Refer to the RCVYSTFG structure in module (ISGQSCNR.)

Note: ISGQSCNR does not record the 09A ABEND code issued by ISGQSCAN.

Recorded by ISGSMI

ISGSMI (at entry point ISGSMIFR) records the following in the SDWAVRA:

- The FRR parameter list. (Refer to the PARMAREA structure in module ISGSMI.)

 LY28-1695-0 (c) Copyright IBM Corp. 1987

- The original system completion code and reason code (in SDWAGR15) describing the error.

GENERAL INFORMATION USEFUL FOR GLOBAL RESOURCE SERIALIZATION ANALYSIS

RECOVERY CONSIDERATIONS

The recovery routines for the global resource serialization subfunctions are:

Recovery Routine	Subfunction
*ISGBERCV - ESTAE	Ring processing
*ISBFRRCV - FRR	
*ISGCRCV - ESTAE	Command Processing
ISGCRET0 - FRR	
ISGCRET1 - FRR	
ISGDSDMP (EP-ISGDSDRV) - ESTAE	Dump support
*ISGDSNAP (EP-ISGDSNRV) - ESTAE	
ISGGEST0 - ESTAE	Request (ENQ/DEQ/RESERVE) processing
ISGGFRR0 - FRR	
*ISGGQSRV (EP-ISGGRTRY)-FRR	Global queue services
*ISGJRCV - FRR	CTC processing
*ISGCRCV - ESTAE	WTO/WTOR message processing
*ISGCRCV - ESTAE	Initialization
*ISGQSCNR - FRR	Queue scanning services
*ISGGFRR0 - FRR	Storage management
*ISGSMI (EP-ISGSMIFR) - FRR	

---

* This routine suppresses duplicate dumps via DAE and its default dump-suppression criteria.

SERIALIZATION

When GRS=NONE is specified, all required global resource serialization resources are serialized with the CMSEQDQ lock.

When GRS=START or GRS=JOIN is specified, the following chart summarizes the serialization of the resources used by global resource serialization.

CMSEQDQ	Local	CS	Resource
X			Local hash table
	X		Global hash table
	X		SYSID/ASID hash table
X			Local ASCB QEL queue
	X		Global ASCB QEL queue
X			Local storage management pools
	X		Global storage management pools
X			Storage management QWB pools
		X	Request queue
	X		Process queue
X			Local QWA
	X		Global QWA

Legend:

CMSEQDQ - ENQ/DEQ cross memory services lock
Local - Global resource serialization local lock
CS - Compare and Swap instruction

LY28-1695-0 (c) Copyright IBM Corp. 1987

CONTROL BLOCKS

Global resource serialization uses the following control blocks. For the format of these data areas, refer to the Debugging Handbook and Data Areas (microfiche).

Data Area Description

CEPL — Command ESTAE parameter list - anchors the LIFO queue of CRWAs and contains an error recording areas for requested functions.

CRB — Command request block - contains information required to process a DISPLAY GRS or VARY GRS command.

CRWA — Command recovery work area - contains the error information used by the command recovery routine to handle errors.

DEPL — SDUMP ESTAE parameter list - contains information used by the global resource serialization dump support subcomponent to process an SDUMP request.

DPL — DEQ purge list - contains the information needed to complete processing for a DEQ SYSID, DEQ ASID, or DEQ TCB purge request.

DSPL — Dump sort parameter list - contains information for the global resource serialization dump sort routine.

ERQA — Extended resource queue area - contains PEXBs that define QCBs, QELs, QXBs and PQCBs.

GCB — Global resource serialization CTC-driver request block - is the parameter list required by the CTC-driver for all functions (except extracting area lengths).

GCC — Global resource serialization CTC-driver control card table - contains the information from the global resource serialization SYS1.PARMLIB member for this system.

GCL — Global resource serialization CTC-driver link control block - contains information related to each CTC in the system.

GCP — Global resource serialization CTC-driver buffer prefix - contains message length and validity checking data.

GCQ — Global resource serialization CTC-driver queueing element - contains information used by CTC processing when sending or receiving a message or an unusual-event notification.

GCT — Global resource serialization CTC-driver branch table - contains addresses of the CTC processing DIE routines and exit routines.

GCV — Global resource serialization CTC-driver vector table - contains addresses of CTC-driver entry points for CTC-driver functions and information common to all CTCs used by CTC processing.

GCX — Global resource serialization CTC-driver extract table - is the parameter list required by the CTC-driver for the extraction of area lengths.

GVT Global resource serialization vector table - contains common information (global queues, pointers and entry point addresses) for all global resource serialization functions. It also has sections containing information for the various subcomponents.

GVTX Global resource serialization vector table extension - contains information specific to the global resource serialization address space.

MRB Message request block - contains information required to process message requests.

PEL Parameter element - is the input parameter list to ENQ/DEQ/RESERVE processing.

PEXB Pool extent block - maps a 4K page in the RQA for QWB, MRB, CRB, TWKA, or HWKA cell type; or a 4k page in the ERQA for QCB, QEL, QXB, or PQCB cell type.

PQCB Placeholder queue control block - contains the information necessary to resume a global resource serialization queue scanning request.

QCB Queue control block - describes a resource to global resource serialization.

QEL Queue element - describes the requester of a resource to global resource serialization.

QFPL ENQ/DEQ/FRR parameter list - is the FRR parameter list used by ENQ/DEQ/RESERVE processing.

QFPL1 Queue scanning services FRR parameter list - is the FRR parameter list used by queue scanning services.

QHT Queue hash table - contains queue hash table entries. Each queue hash table entry is a double-headed anchor of QCBs. There are two QHTs; one for global requests (GQHT), and one for local requests (LQHT).

QWA Queue work area - is a work area used by ENQ/DEQ/RESERVE processing modules.

QWB Queue work block - describes a resource request. A global resource request is described by a QWB in the private area of the global resource serialization address space. A local resource request is described by the permanent QWB in the SQA.

QXB Queue extension block - contains the data that describes an ENQ/DEQ/RESERVE request.

REPL Ring processing ESTAE parameter list - is the ESTAE parameter list used by ring processing.

RIB/RIBE Resource information block - contains the information that describes a resource and any requesters for the resource. The variable portion of the RIB (containing RIB extents) is located immediately after the RIB. Each RIB extent (RIBE) describes a requester of the resource. RIBs and RIBEs are returned to the issuer of the GQSCAN macro.

RNLE Resource name list entry - contains information about resources that are to be included or excluded from global resource serialization processing and RESERVE resources that are to be converted to global ENQs.

RPT Resource pool table - contains entries for each cell type in the RQA. There are two RPTs - one for global resources (GRPT), and one for local resources (LRPT). Each RPT points to the first and last PEXB for that pool.

 LY28-1695-0 (c) Copyright IBM Corp. 1987

RQA — Resource queue area - contains PEXBs that define QWBs, MRBs, CRBs, and the work areas.

RSA — Ring processing system authority message - is used to pass command data and ENQ/DEQ/RESERVE requests between global resource serialization systems in the main ring.

RSAIRCD — Ring processing information record - is used to pass control information between systems that are not both in the main ring.

RSC — Ring status change parameter list - is the parameter list used to call the interface module ISGBCI.

RSL — Ring processing system link block - contains information about a CTC and is used by global resource serialization ring processing functions.

RST — Ring processing status table - contains the status of global resource serialization systems and CTCs.

RSV — Ring processing system vector table - contains information used by the global resource serialization ring processing modules.

RVR — Ring processing FRR parameter list - provides input data to the ring processing functional recovery routine, ISGBFRCV.

SAHT — System/ASID hash table - contains entries that point to a chain of QELs that define global resource requesters from another system.

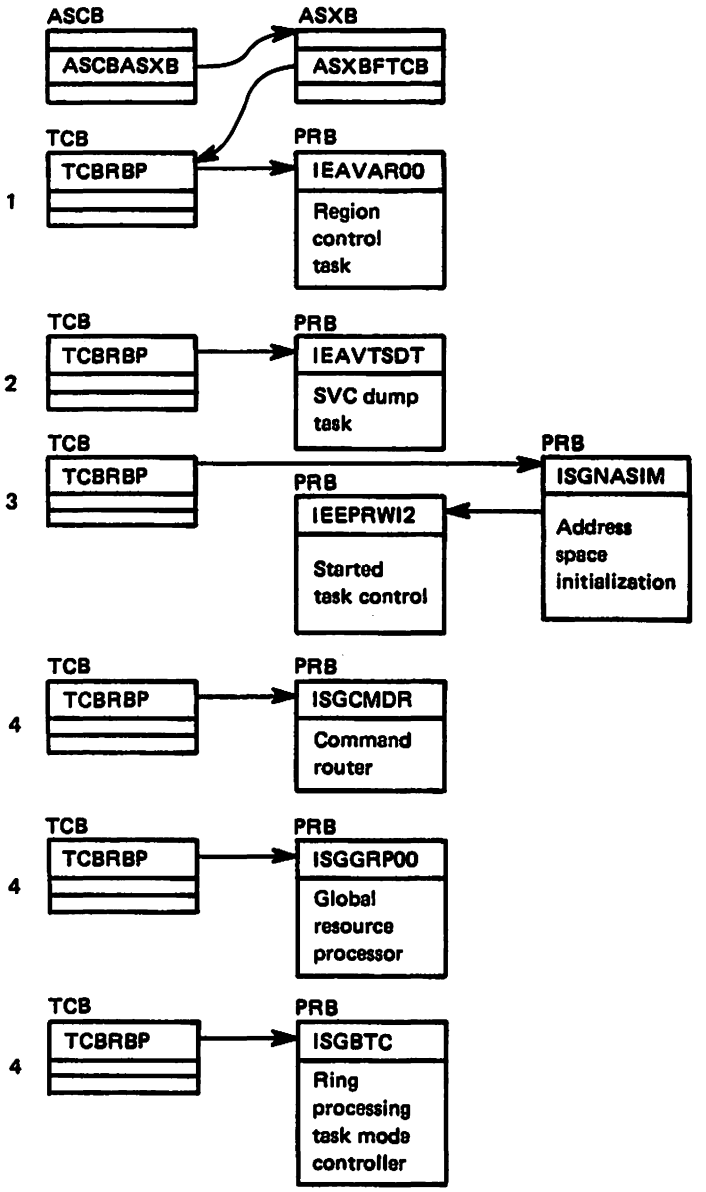
SMPL — Storage management parameter list entry - contains information for a request to global resource serialization storage management.

SNDI — Ring processing send information control block - maps the parameter list for ISGBRF (GRS Ring Processing Request Function Module).

CONTROL BLOCK STRUCTURES

The figures in this topic show the control block structures of the global resource serialization control blocks for the following:

- Permanent TCBs
- CTC processing
- Ring processing
- Command processing
- ENQ/DEQ processing
  - Local resources
  - Global resources
- Queue scanning services
  - Local resources
  - Global resources
- Storage management
- WTO/WTOR Message processing



Notes:

- The numbers show the hierarchy.
- When GRS=START or JOIN, all TCB/PRBs are permanent.
- When GRS=NONE: all TCB/PRBs are permanent except the TCB/PRB for ISGGRP00, which is temporary; and the TCB/PRB for ISGBTC, which is not present.

Figure 7. TCBs in the Global Resource Serialization Address Space

 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

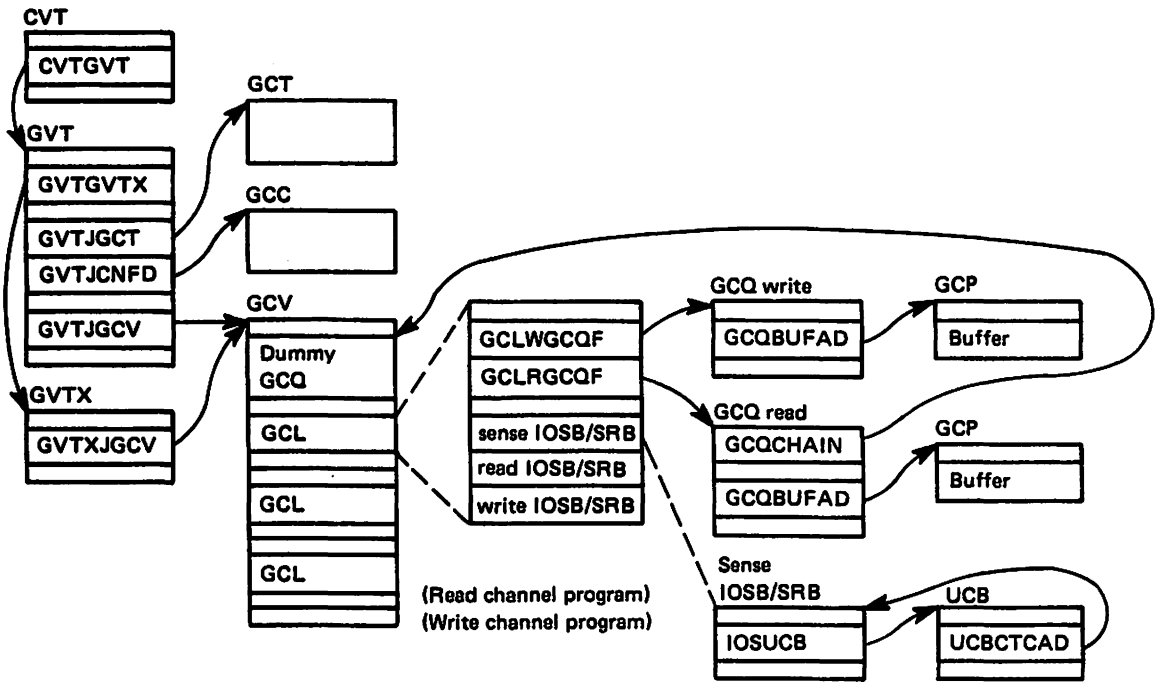


Figure 8. CTC Processing Control Block Overview

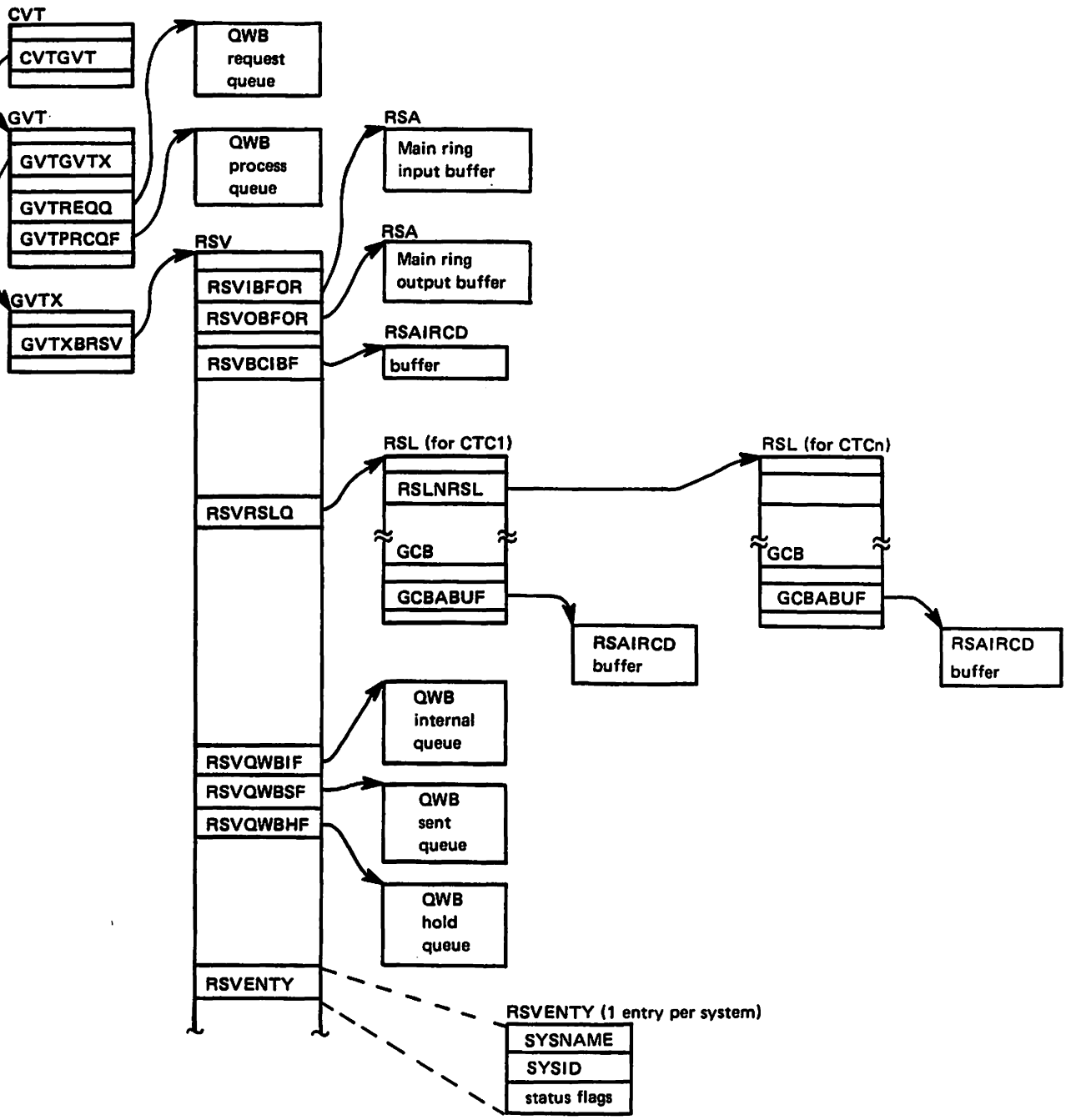


Figure 9. Ring Processing Control Block Overview

 LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

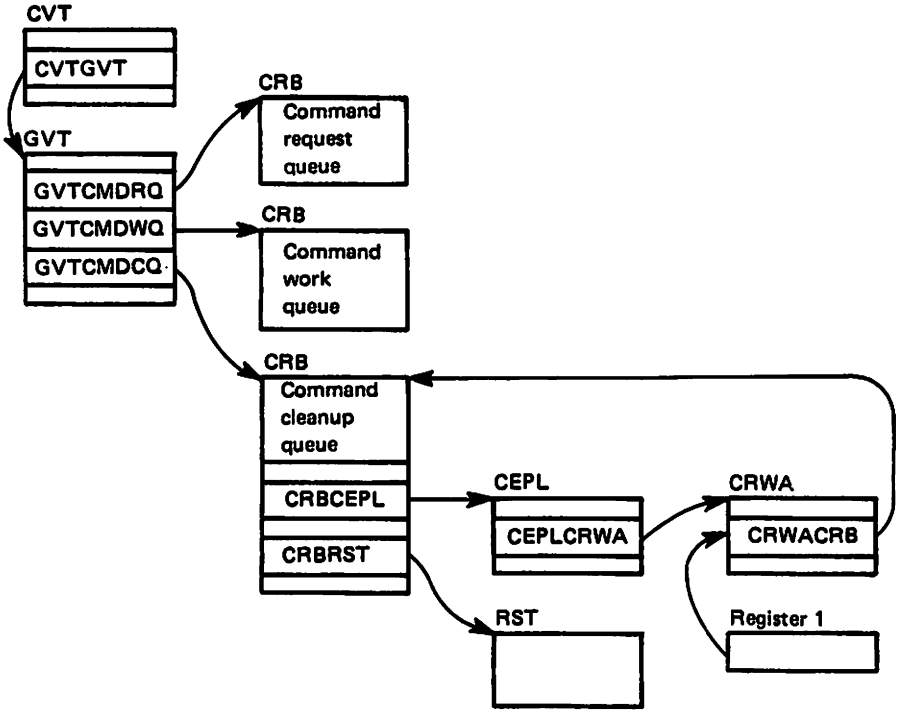


Figure 10. Command Process Control Block Overview

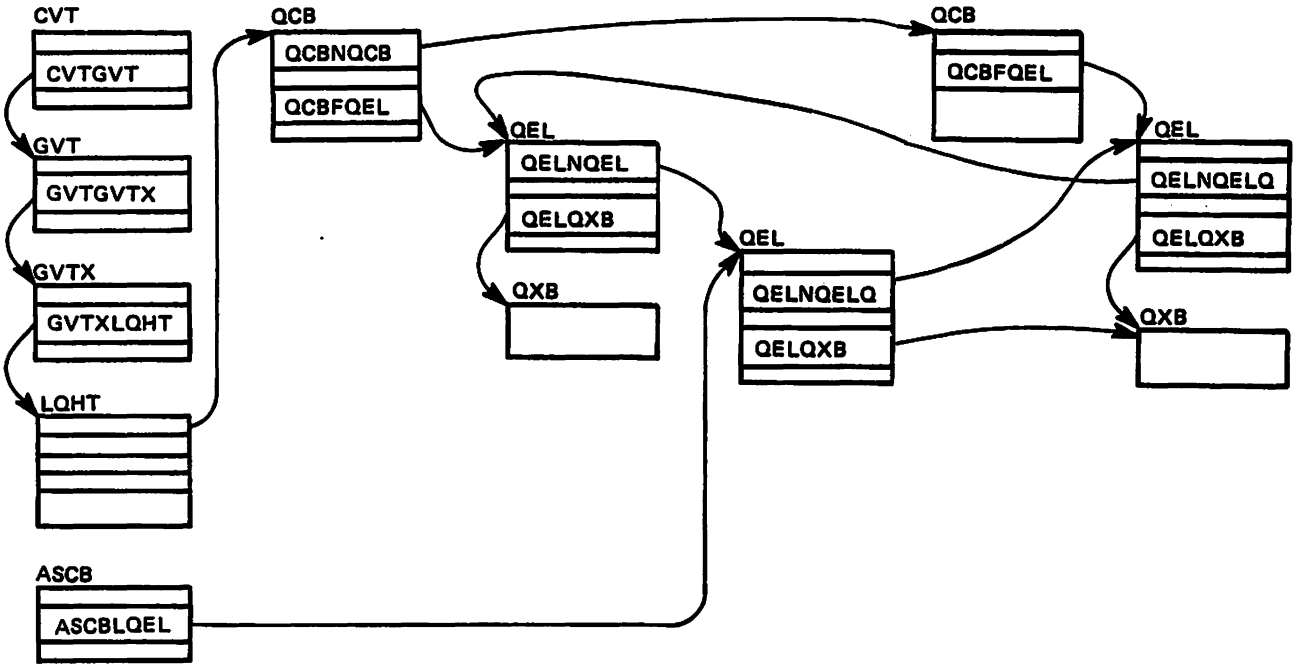


Figure 11. ENQ/DEQ Processing - Local Resources - Control Block Overview

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

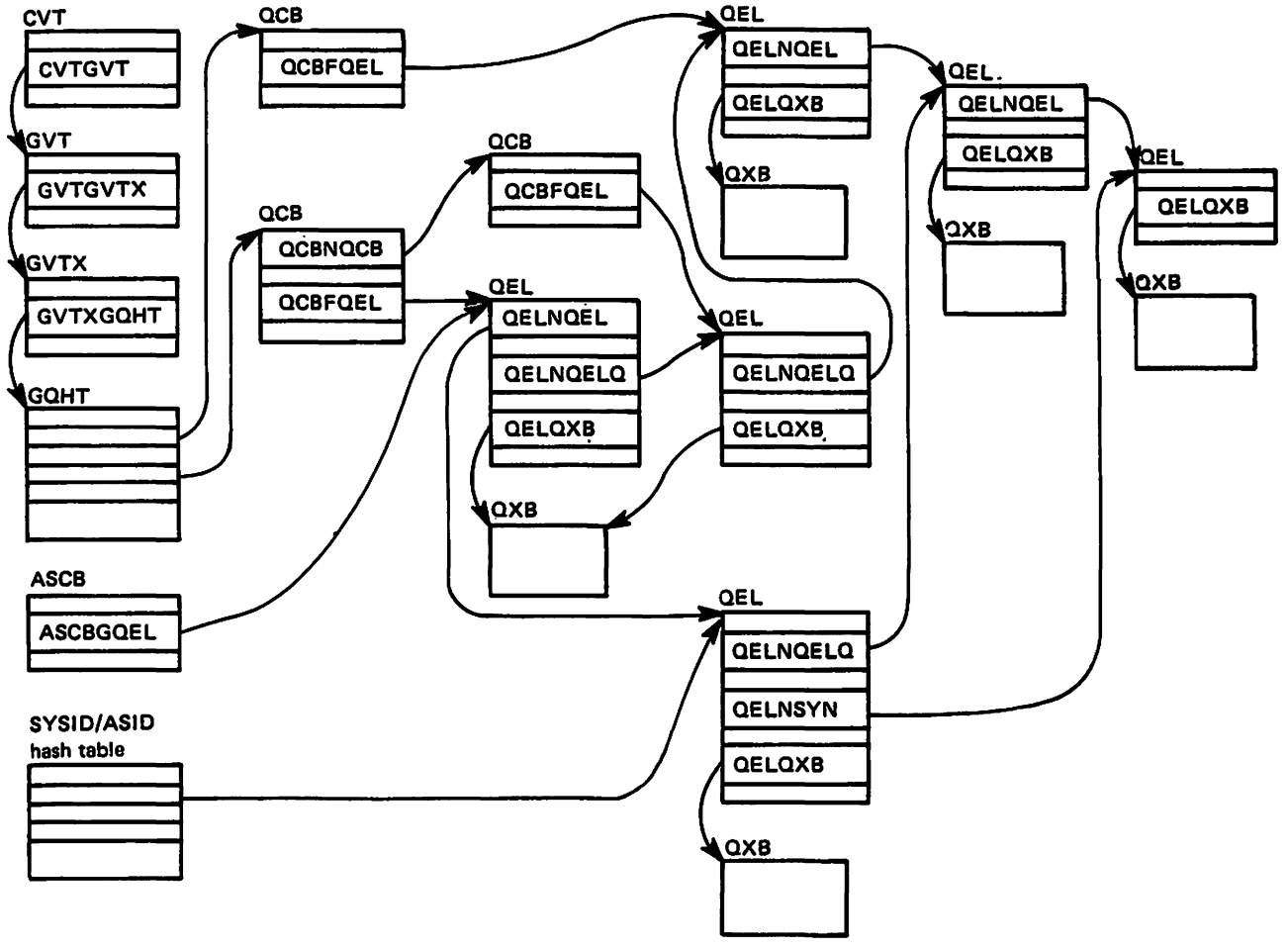


Figure 12. ENQ/DEQ Process - Global Resource - Control Block Overview

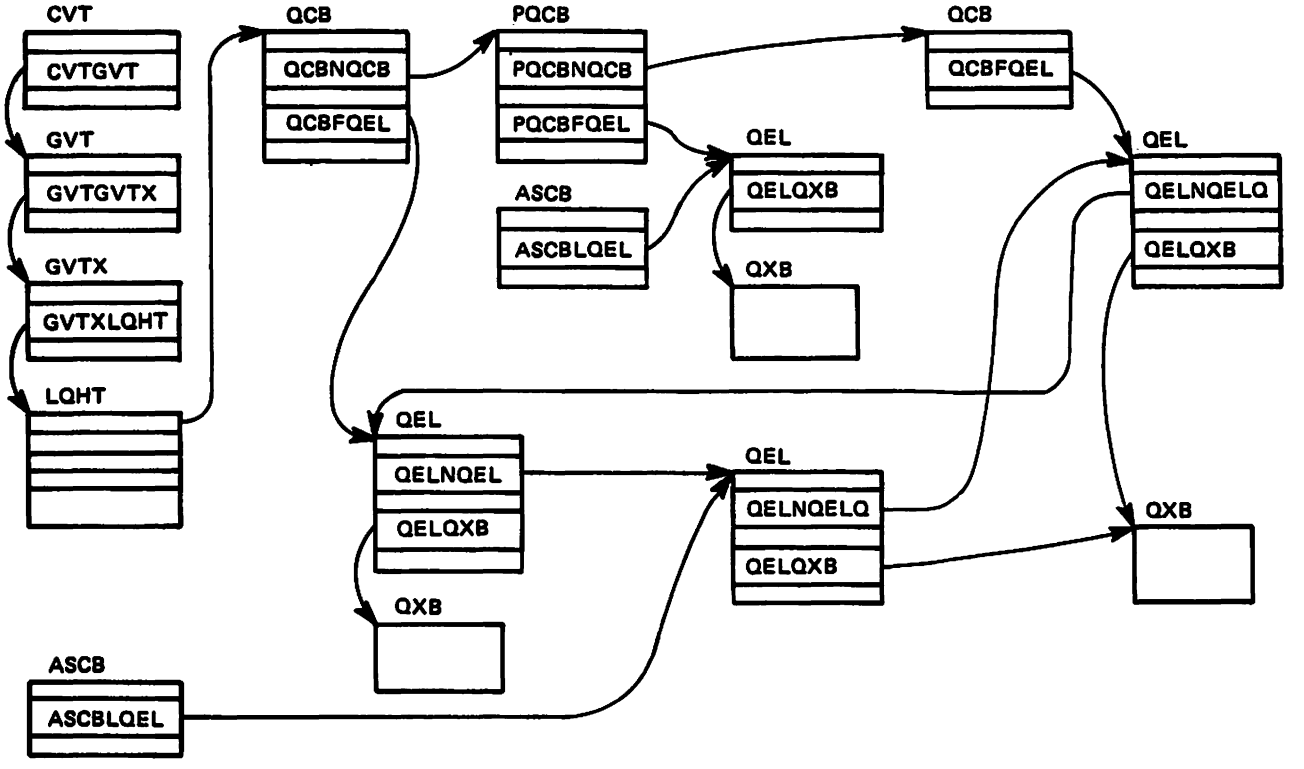


Figure 13. Queue Scanning Services Local Resources - Control Block Overview

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

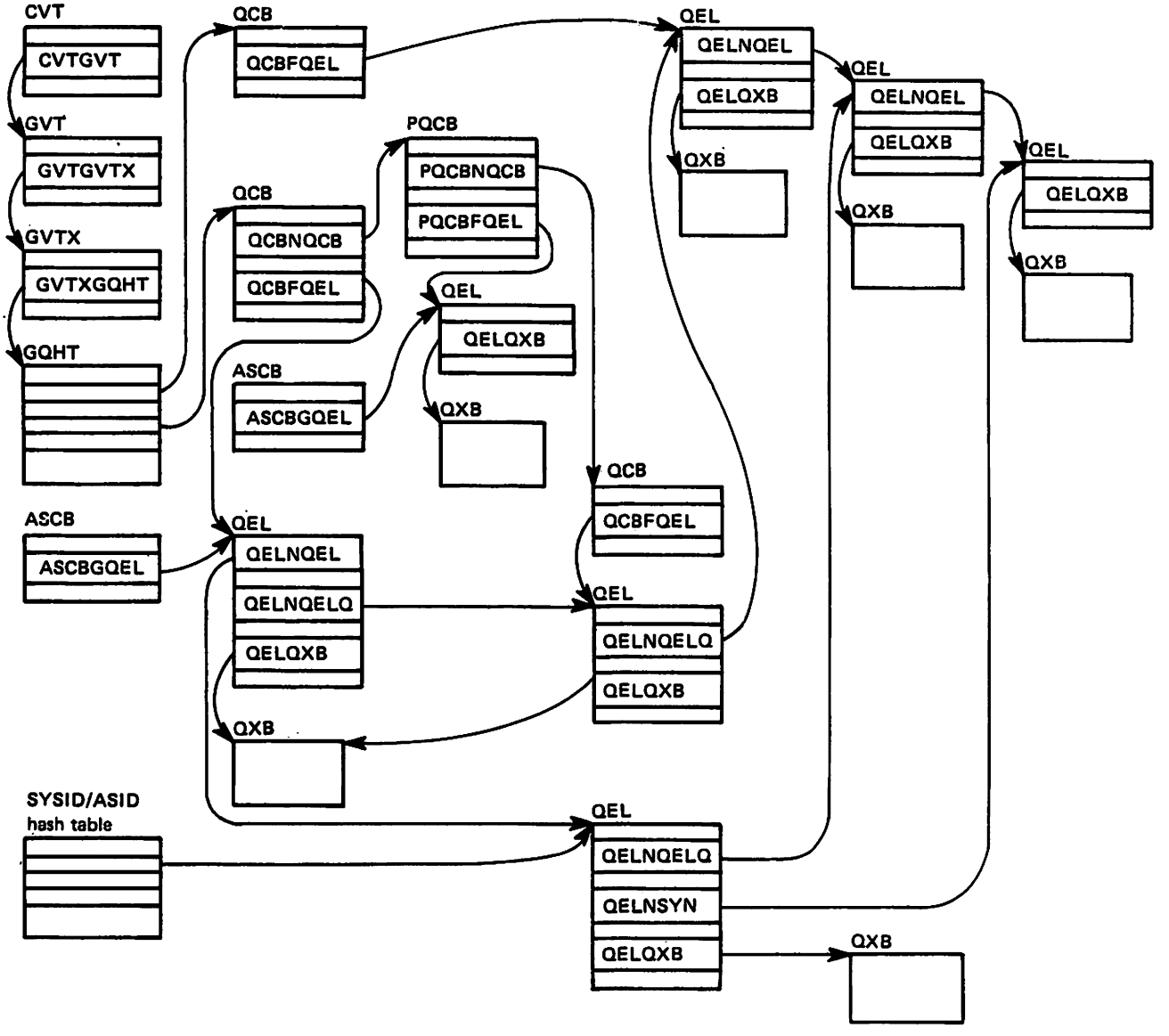


Figure 14. Queue Scanning Services Global Resources - Control block Overview

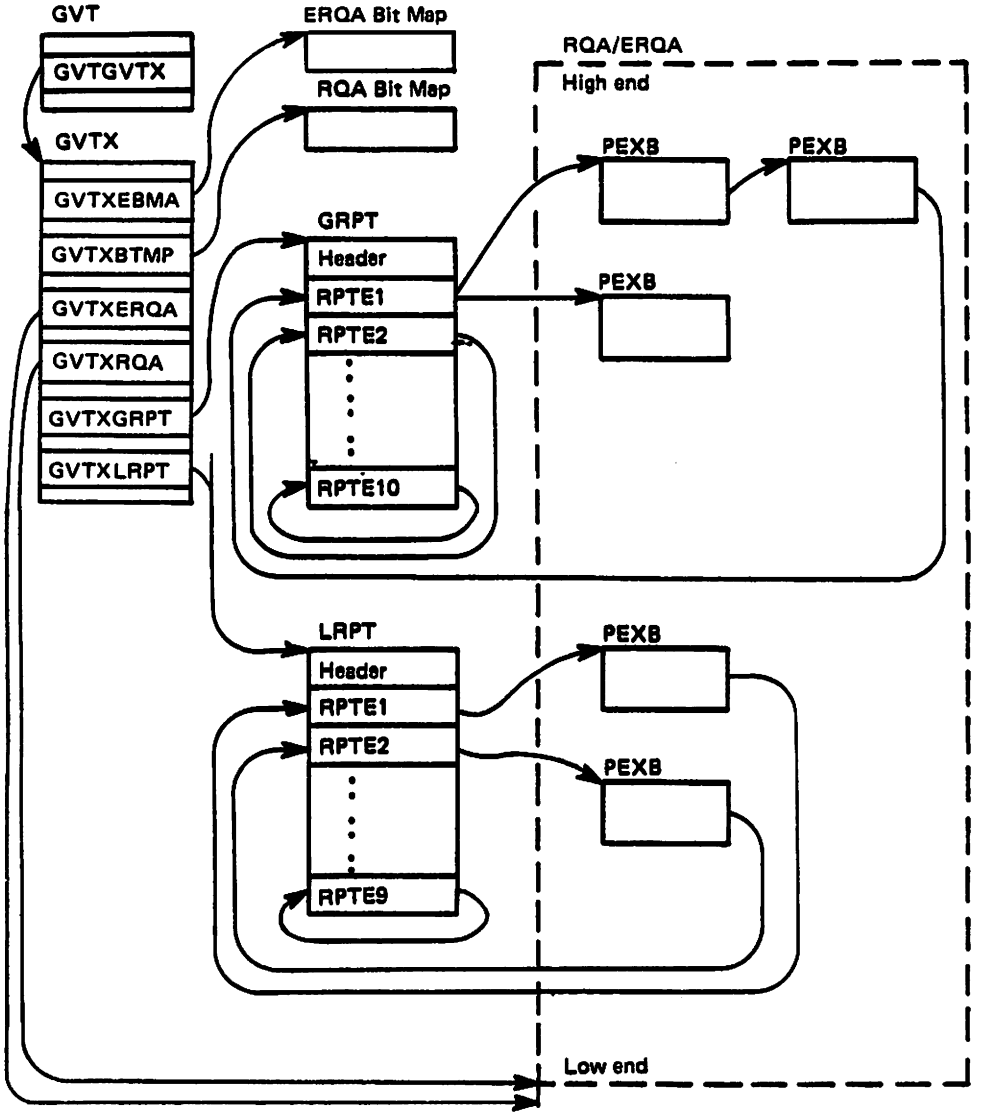
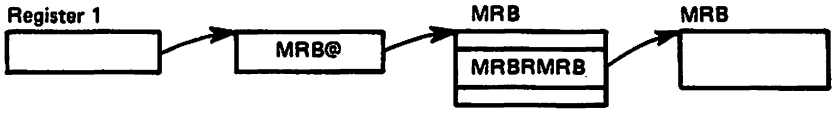


Figure 15. Storage Management Control Block Overview

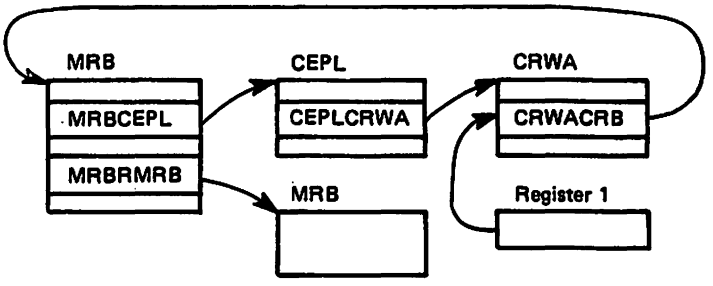
 LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Synchronous Request



Asnychronous Request



Note: Control block structure when the message processing routine (ISGMSG00) receives control.

Figure 16. WTOR/WTOR Message Processing Control Block Overview

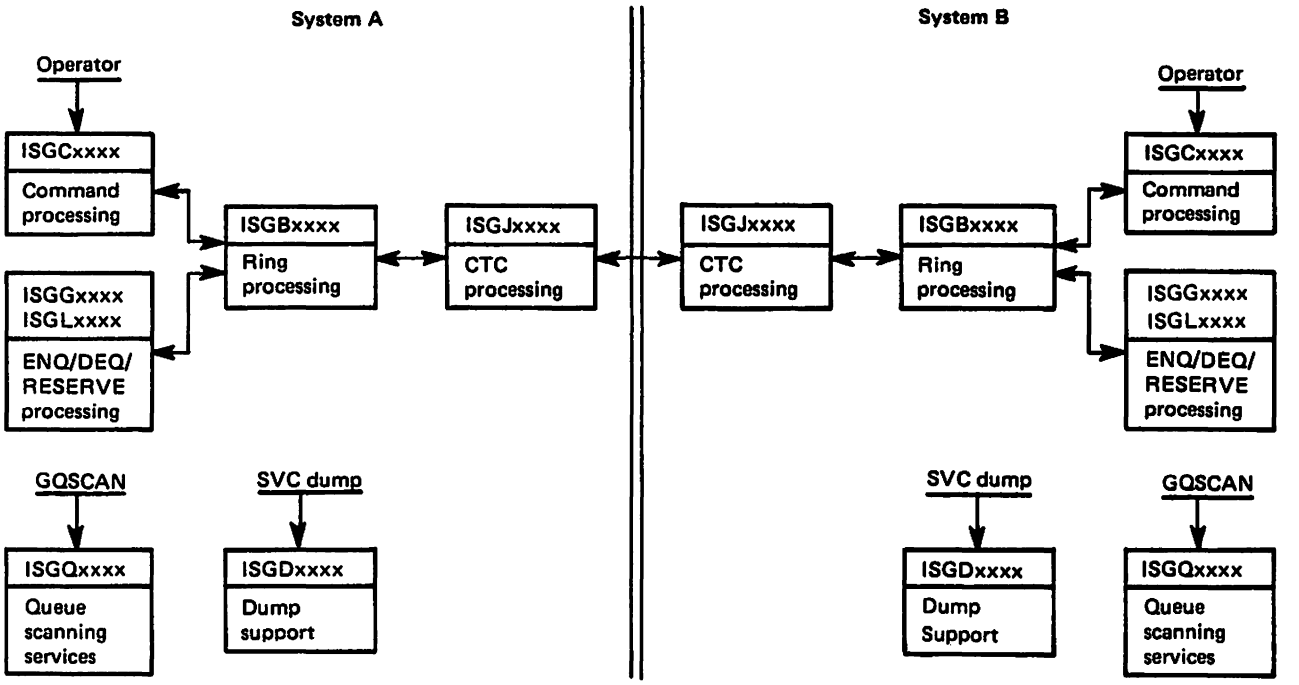


Figure Title (Module Flow for:)

CTC Processing

GRS-18 Handle Arrival of Immediate-CCW
GRS-19 Handle Arrival of RSA or RSAIRCD
GRS-20 Send a RSA or RSAIRCD

Ring Processing

GRS-21 Send/Receive a RSA
GRS-22 Send a RSAIRCD or Immediate-CCW (Requested by ISGBCI)
GRS-23 Send a RSAIRCD (Requested by ISGBTC)
GRS-24 Handle Arrival of RSAIRCD (Not Requested by This System)
GRS-25 SNAPSHOT Function
GRS-26 SENDCMD (RSCRADDS) Function
GRS-27 SENDCMD (RSCRSNAD) Function

Command Processing

GRS-28 – Command Initialization and Cleanup
GRS-29 – DISPLAY GRS
GRS-30 – VARY GRS(x), PURGE
GRS-31 – VARY GRS(x), QUIESCE to Another System
GRS-32 – VARY GRS(x), QUIESCE by a System to Quiesce Itself
GRS-33 – VARY GRS(x), RESTART to Restart Another System
GRS-34 – VARY GRS(ALL), RESTART to Restart All Systems
GRS-35 – VARY GRS(x), RESTART by a System Not in the Main Ring
GRS-36 – Join Processing at Initialization Time

ENQ/DEQ Mainline (Resource request processing)

GRS-37 – Local Resource Request
GRS-38 – Global Resource Request
GRS-39 – Termination Resource Manager

GRS-40 *Queue Scanning Services*

GRS-41 *Dump Support – SVC Dump*

Figure 17. Process Flow Overview and Directory

 LY23-1695-0 (c) Copyright IBM Corp. 1987

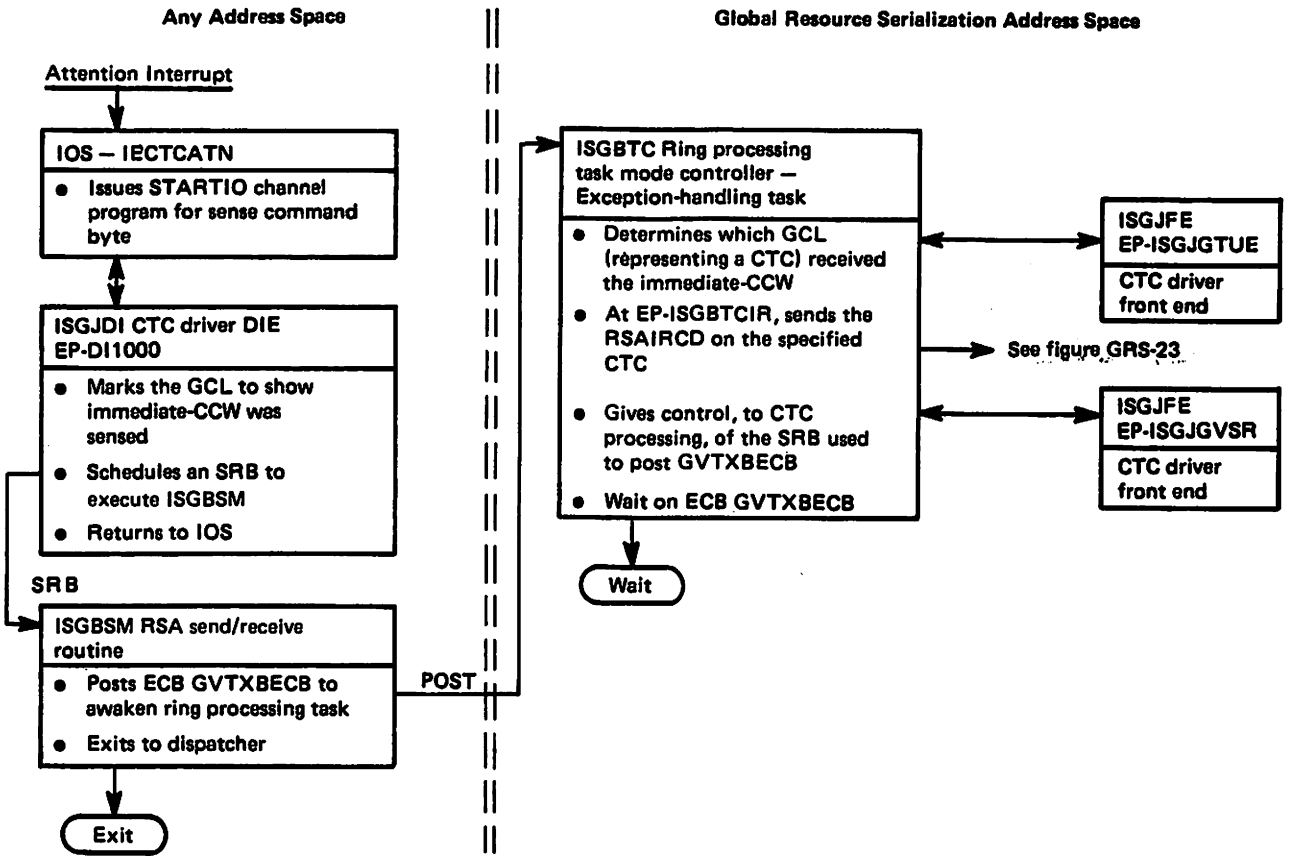


Figure 18. Process Flow for CTC Processing - Handle Arrival of Immediate CCW

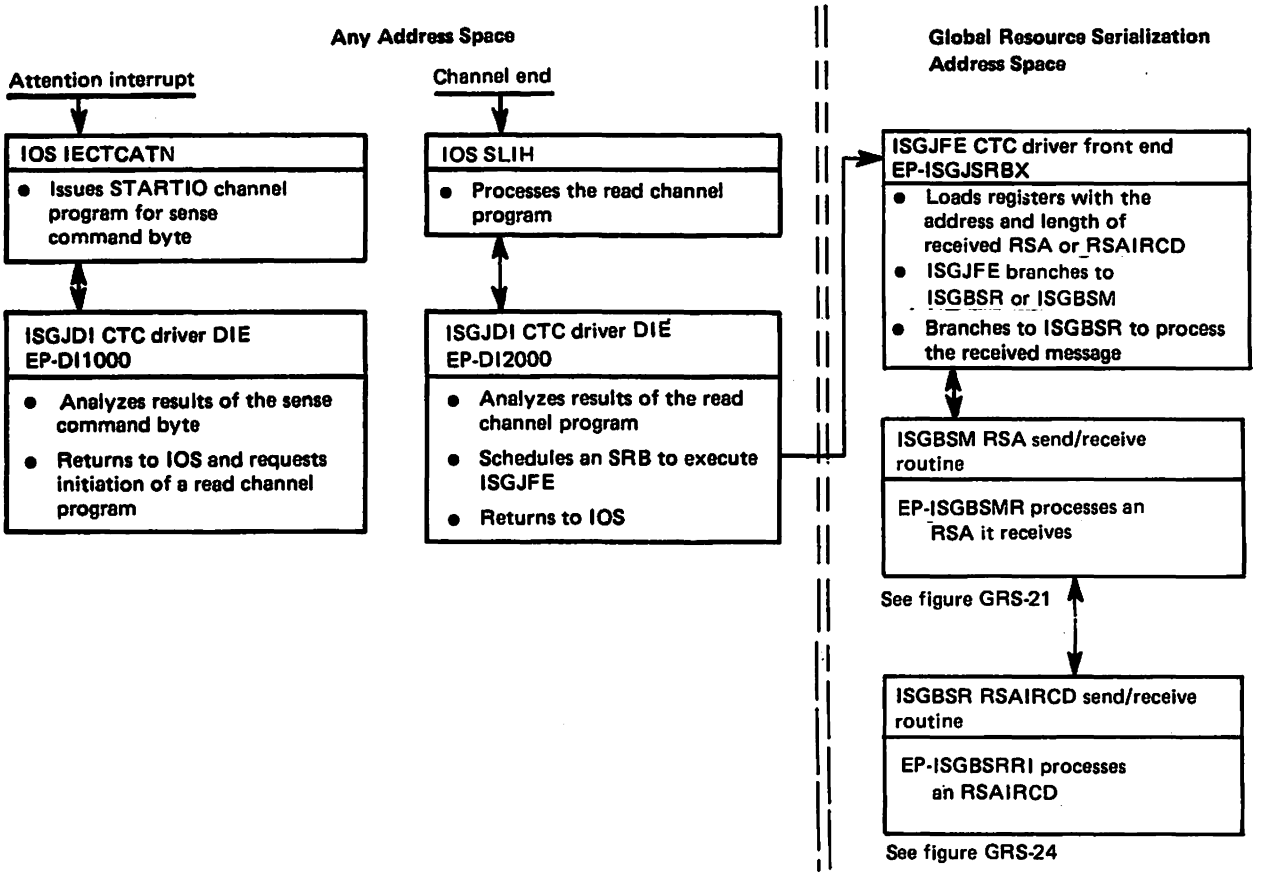


Figure 19. Process Flow for CTC Processing - Handle Arrival of RSA or RSAIRCD

LY28-1695-0 (c) Copyright IBM Corp. 1987

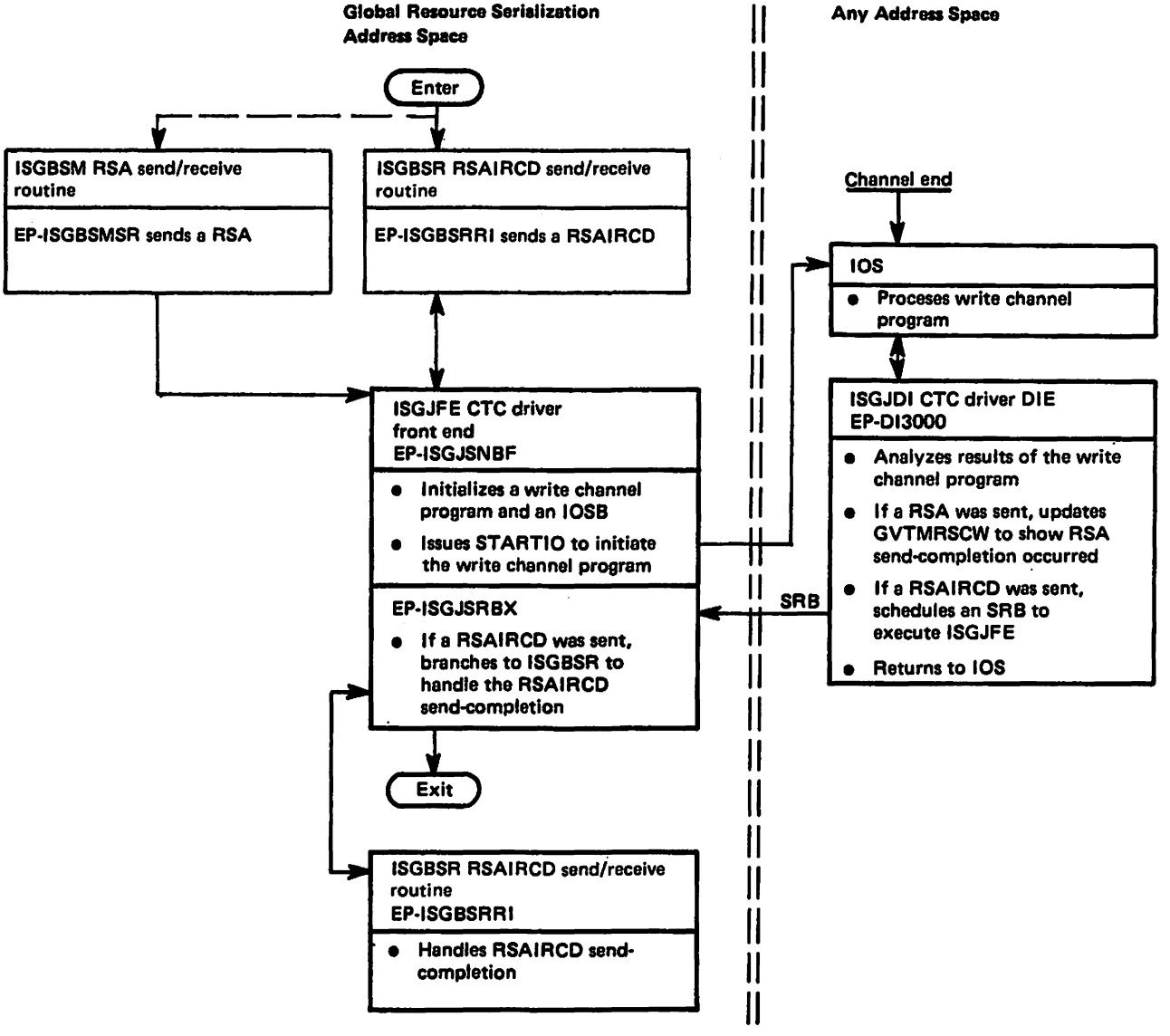


Figure 20. Process Flow for CTC Processing - Send a RSA or RSAIRCD

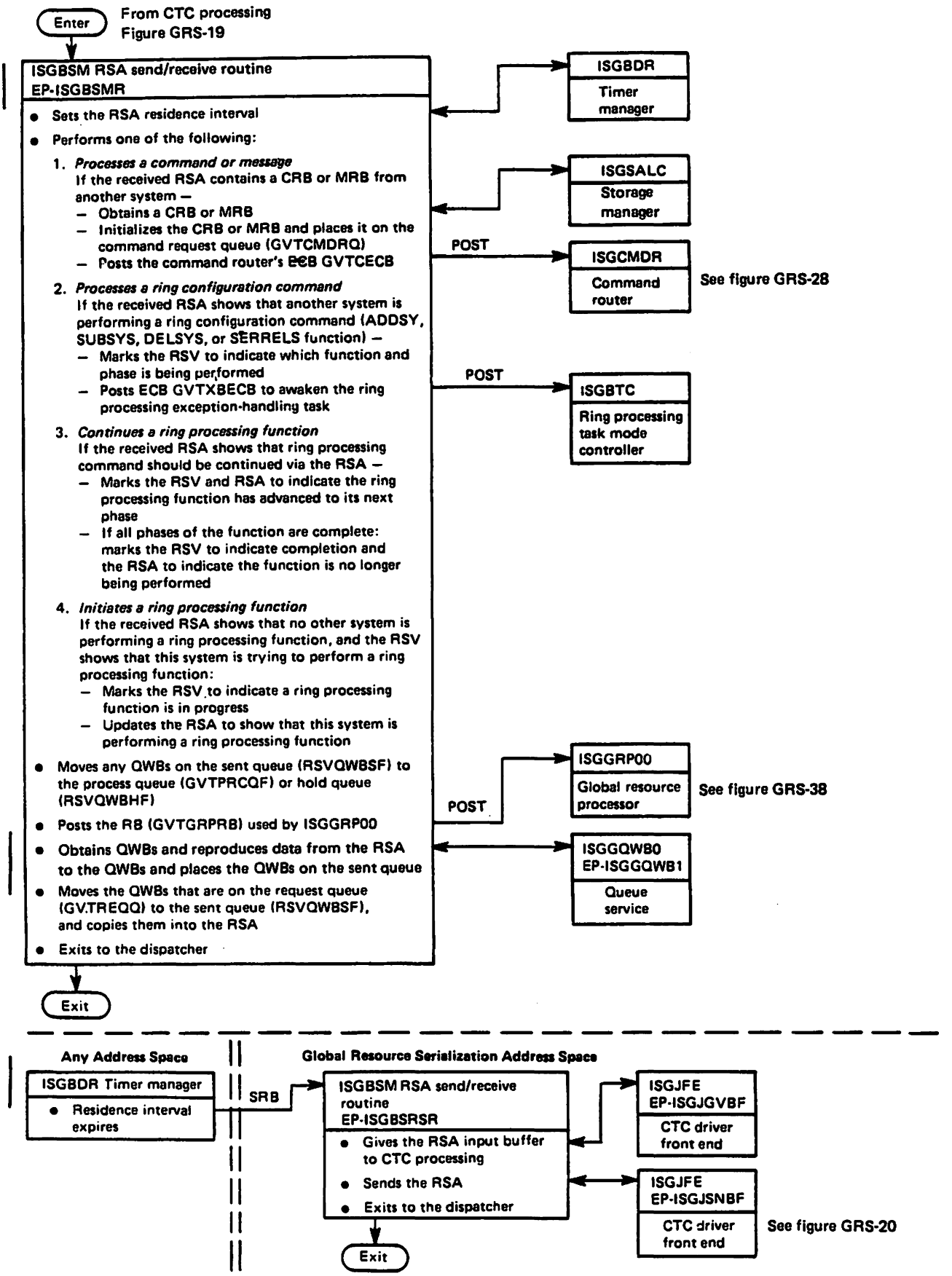


Figure 21. Process Flow for Ring Processing - Send/Receive a RSA

LY28-1695-0 (c) Copyright IBM Corp. 1987

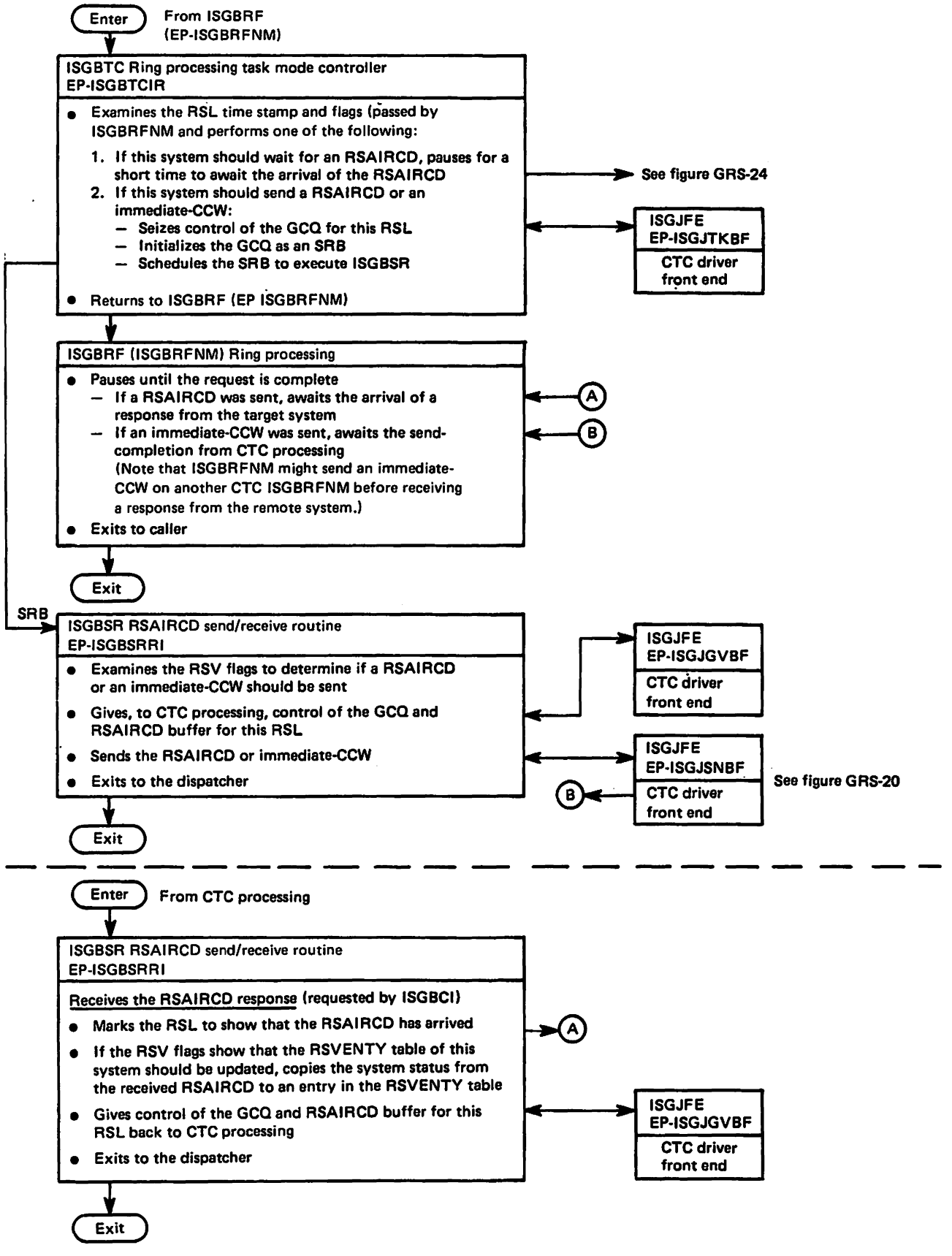


Figure 22. Process Flow for Ring Processing - Send a RSAIRCD or Immediate-CCW (Requested by ISGBCI)

 (c) Copyright IBM Corp. 1987

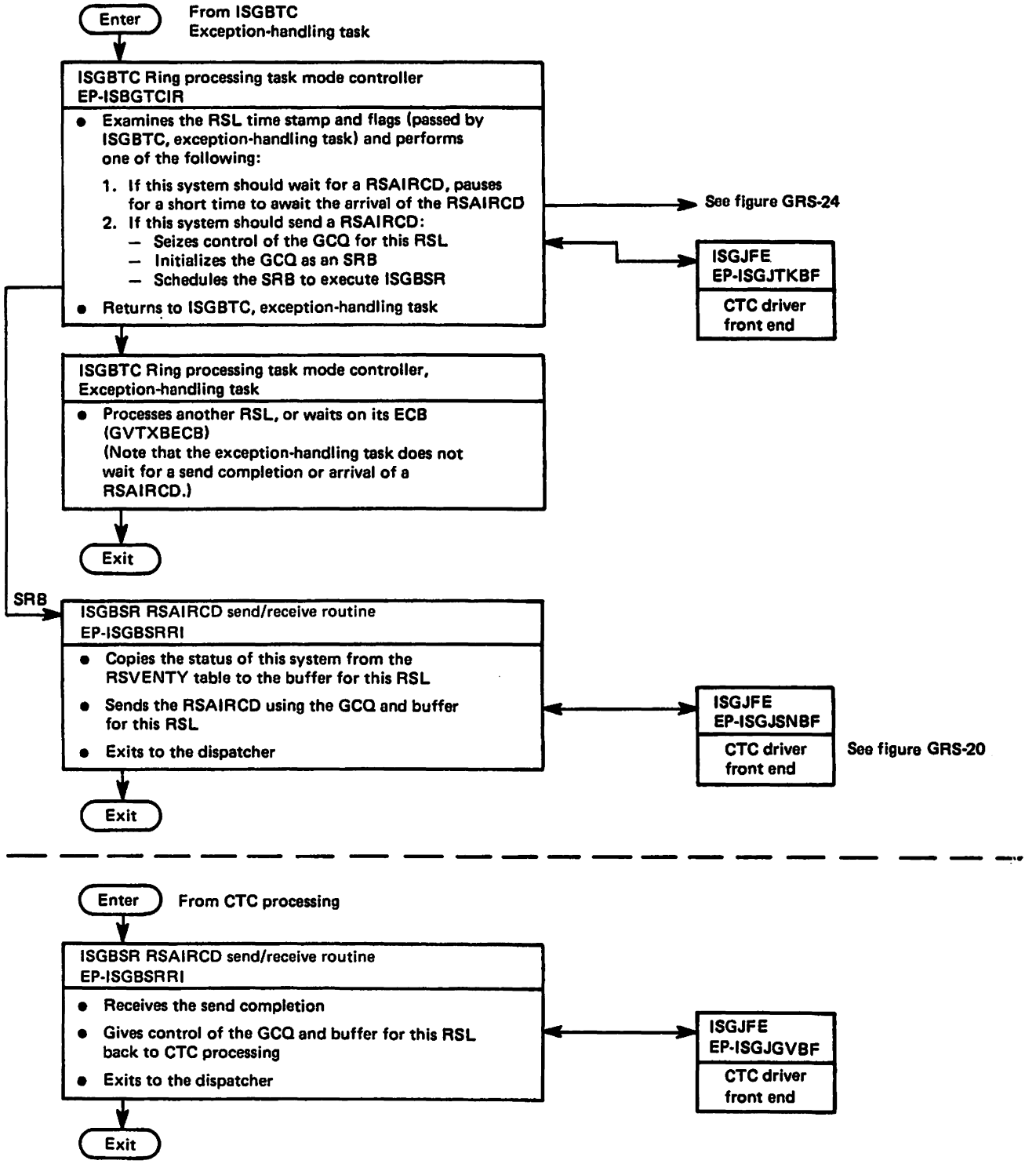


Figure 23. Process Flow for Ring Processing - Send a RSAIRCD (Requested by ISGBTC)

LY28-1695-0 (c) Copyright IBM Corp. 1987

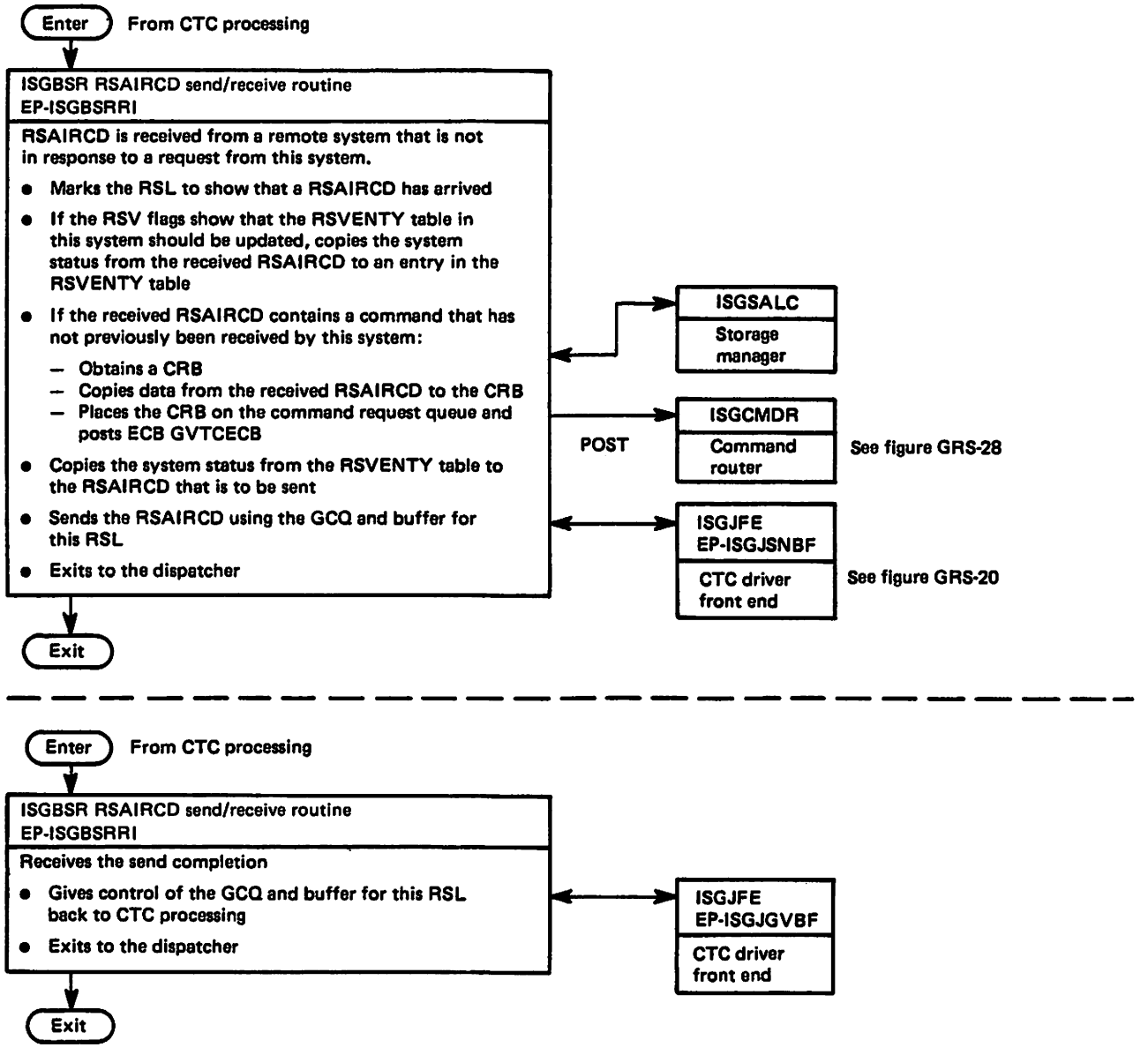


Figure 24. Process Flow for Ring Processing - Handle Arrival of RSAIRCD (Not Requested by This System

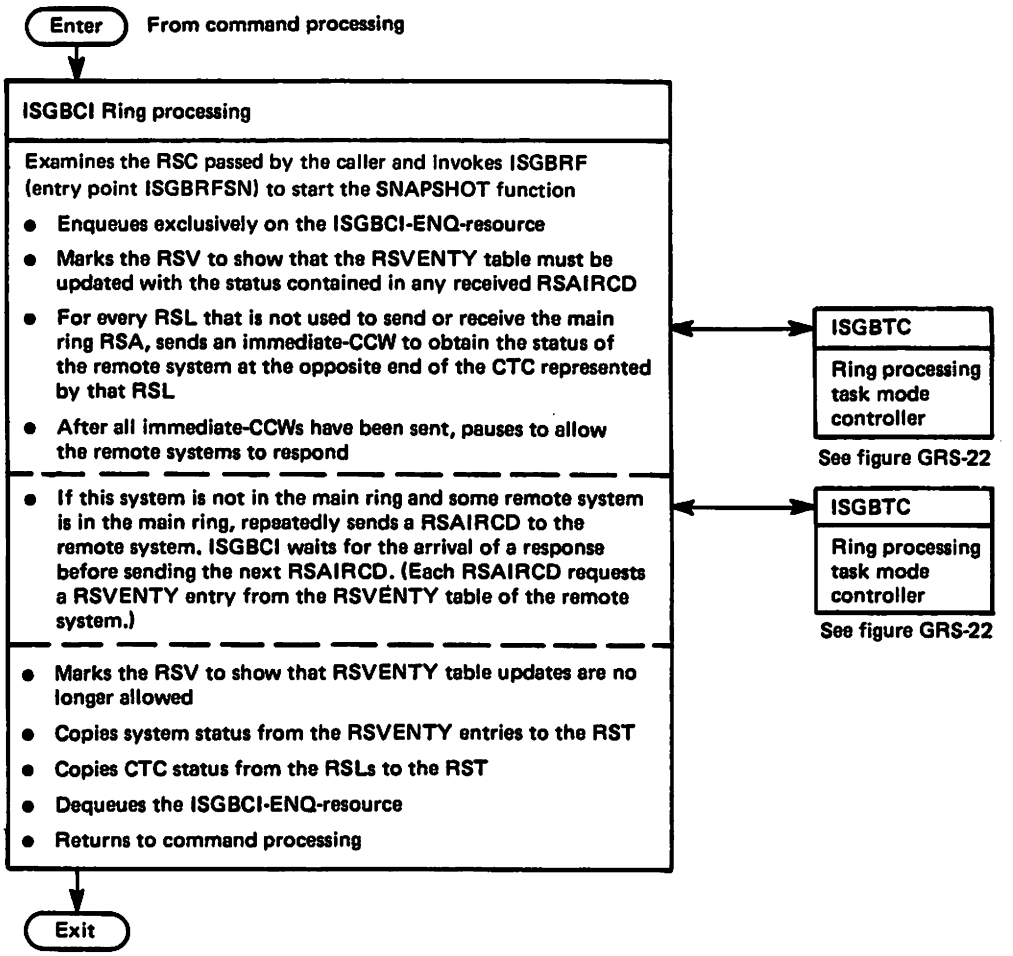


Figure 25. Process Flow for Ring Processing - SNAPSHOT Function

 LY28-1695-0 (c) Copyright IBM Corp. 1987

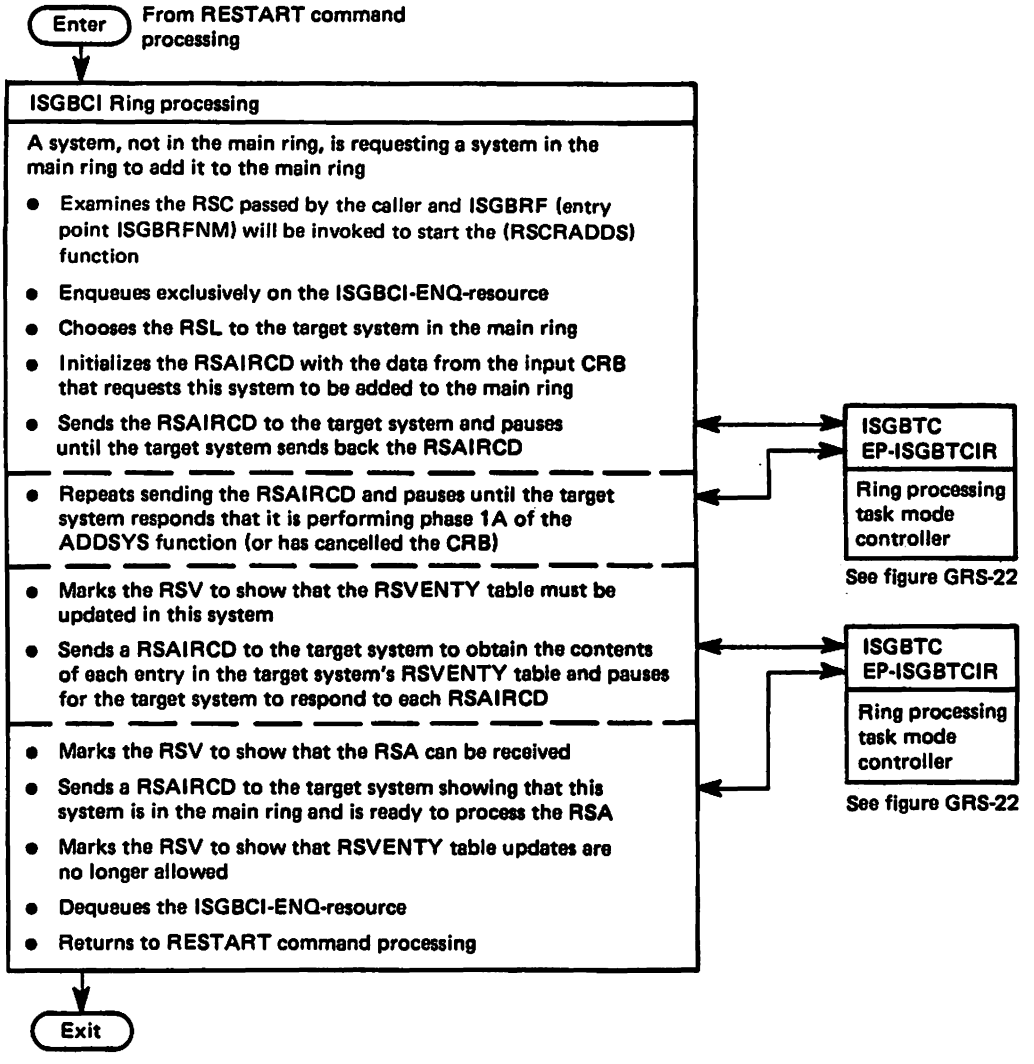


Figure 26. Process Flow for Ring Processing - SENDCMD (RSCRADDS) Function

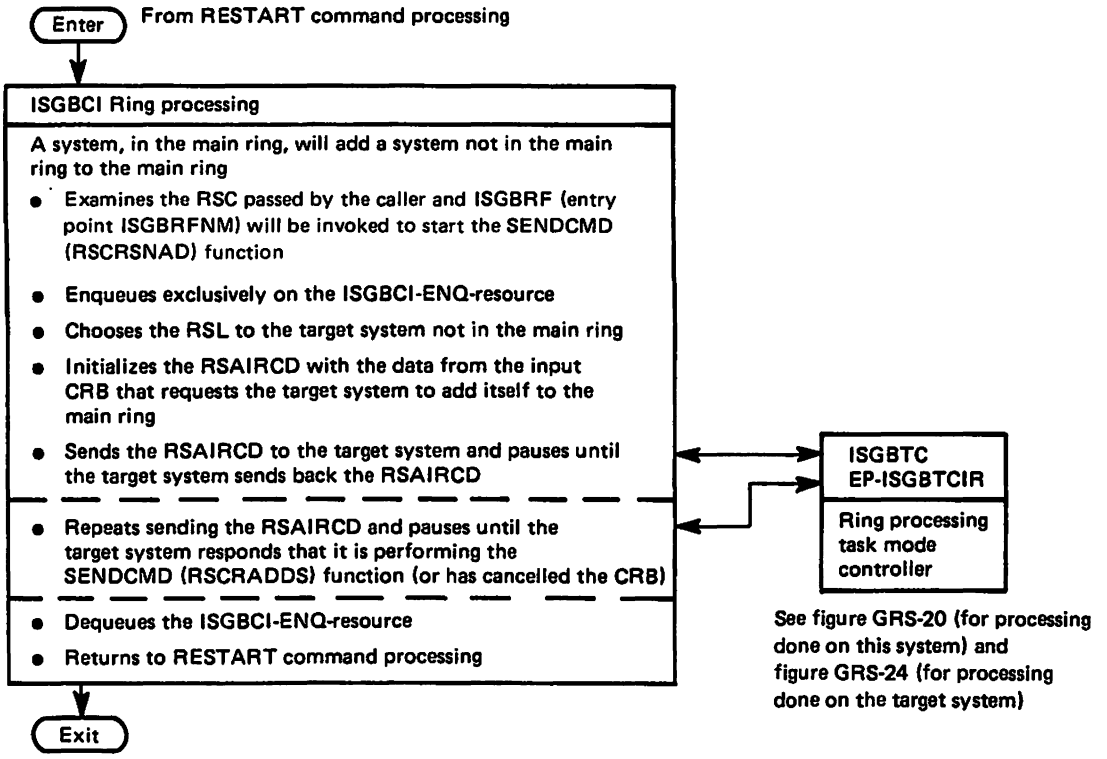
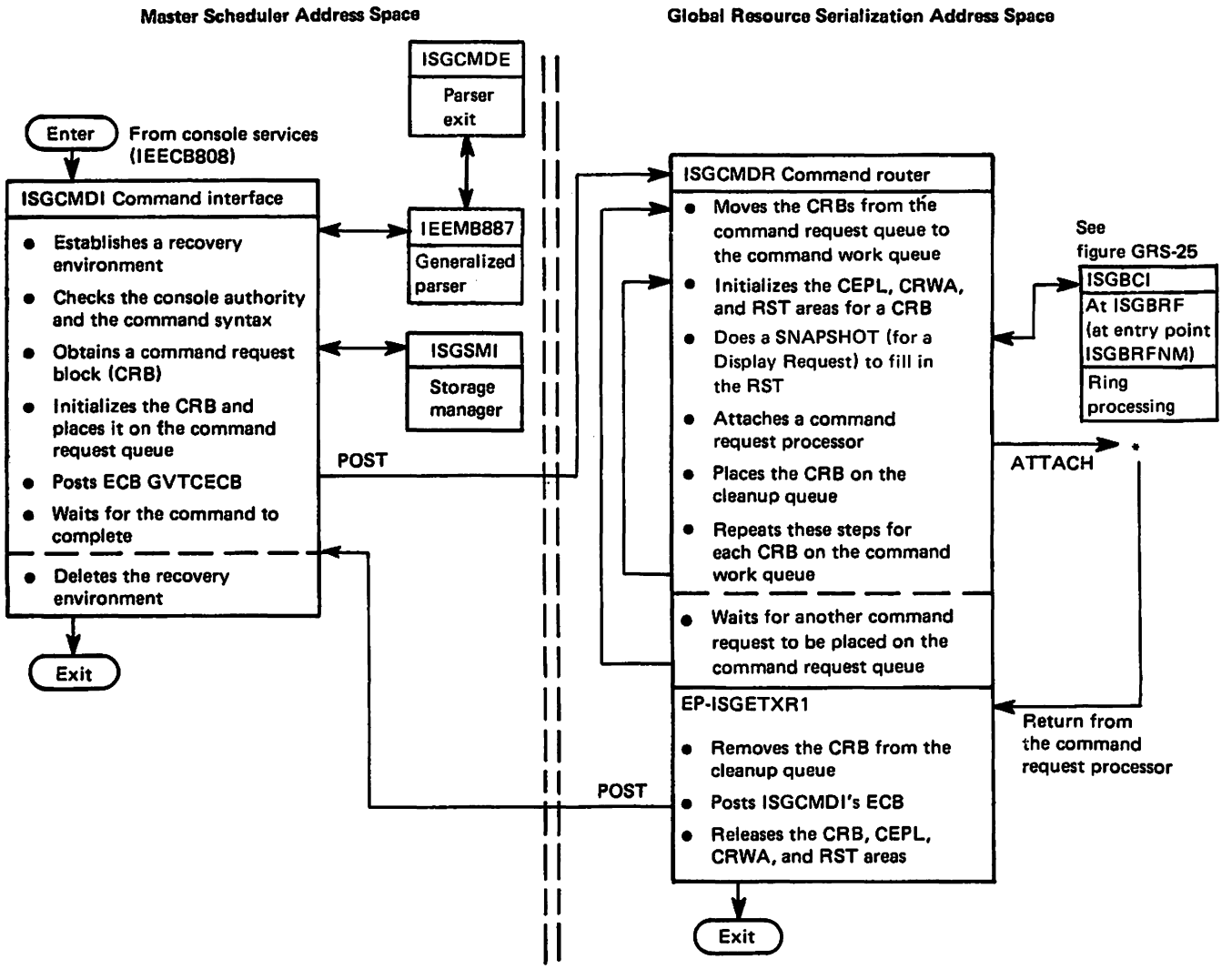


Figure 27. Process Flow for Ring Processing - SENDCMD (RSCRSNAD) Function

LY28-1695-0 (c) Copyright IBM Corp. 1987



***Command request processors**

ISGCDSP – DISPLAY GRS (figure GRS-29)
ISGCPRG – VARY GRS(x), PURGE (figure GRS-30)
ISGCQSC – VARY GRS(x), QUIESCE (figures GRS-31 and GRS-32)
ISGCRST – VARY GRS(x), RESTART (figures GRS-33, GRS-34, and GRS-35)
ISGMSG00 – Asynchronous message request

Figure 28. Process Flow for Command Initialization and Cleanup

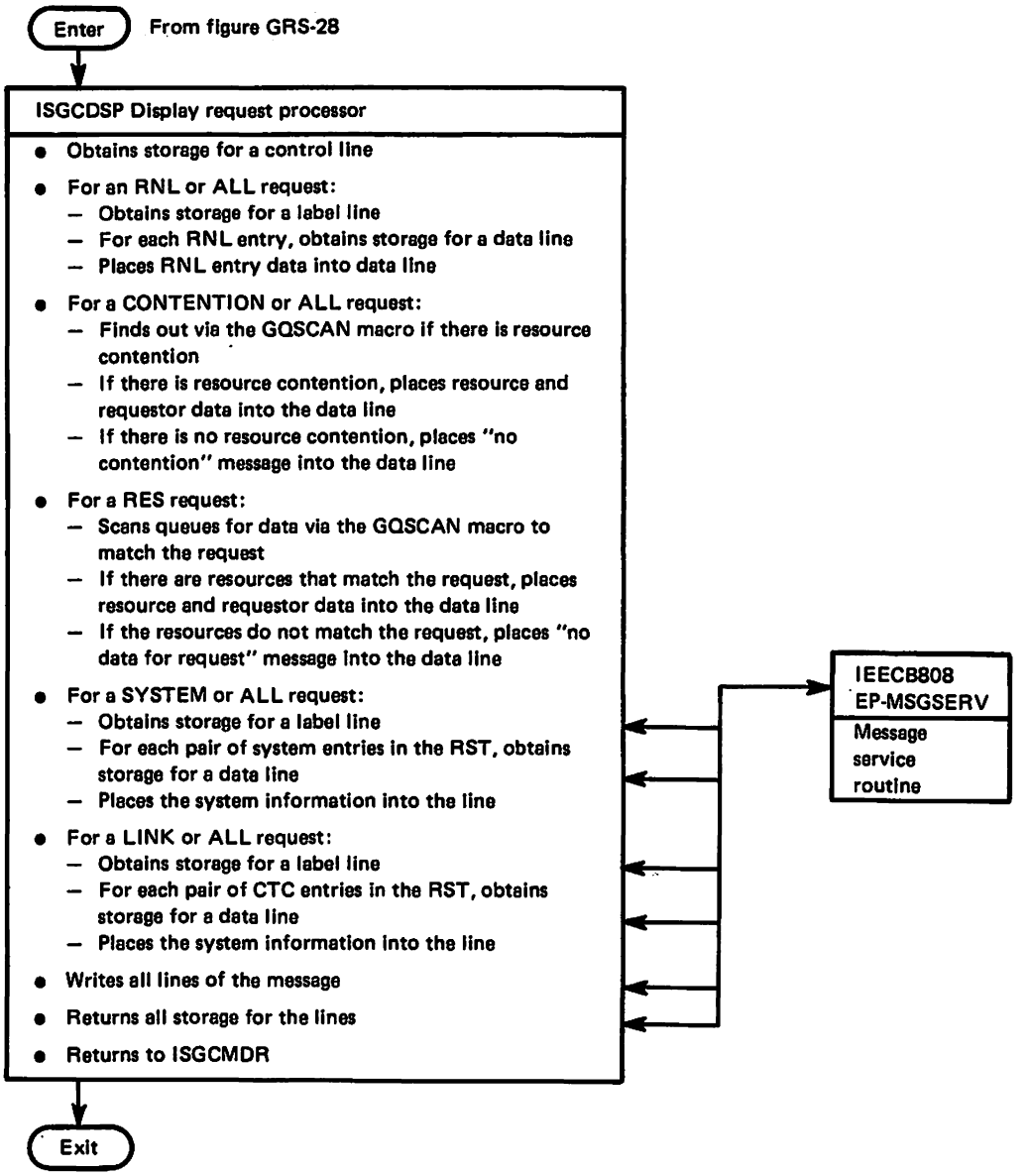


Figure 29. Process Flow for DISPLAY GRS

LY28-1395-0 (c) Copyright IBM Corp. 1987

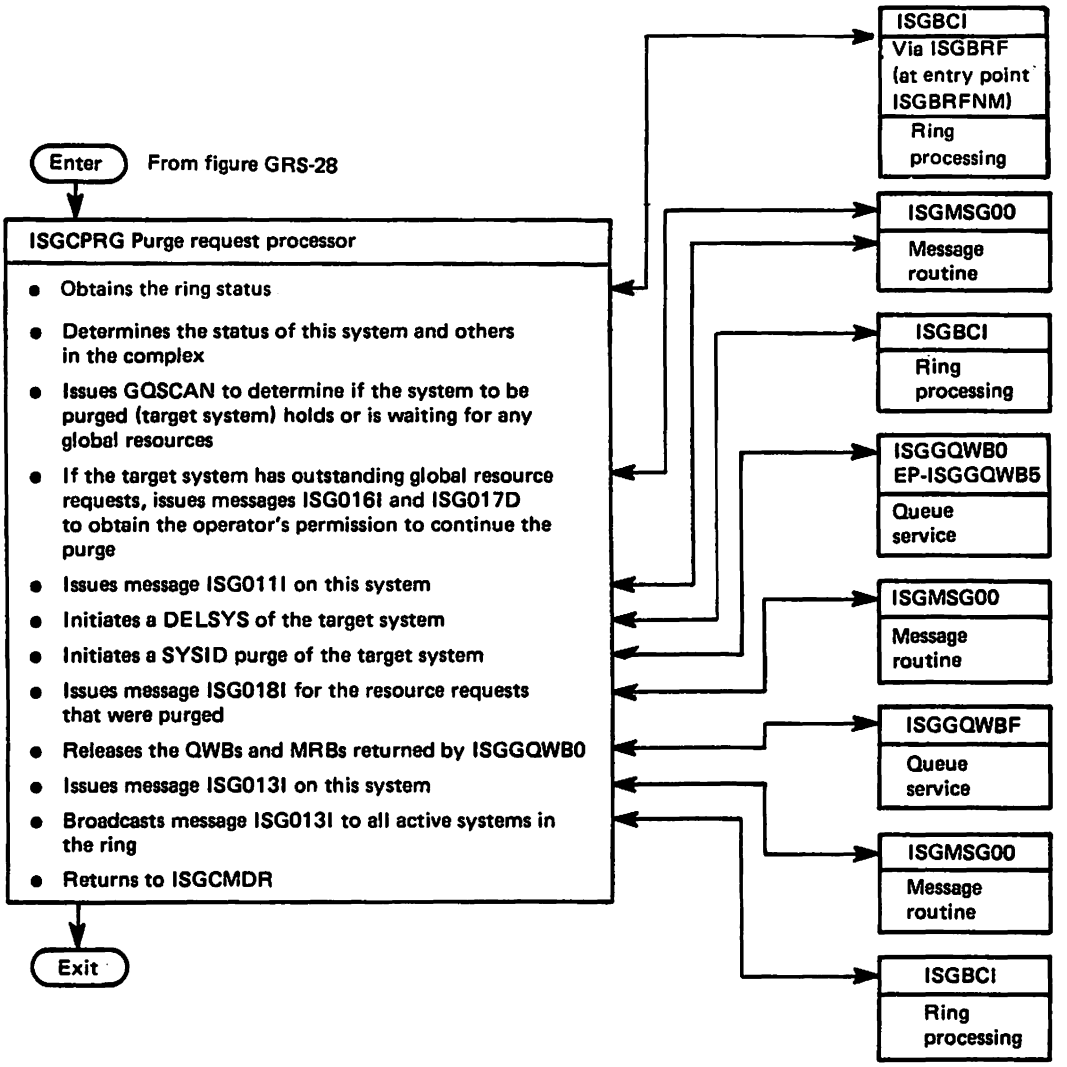


Figure 30. Process Flow for VARY GRS(x), PURGE

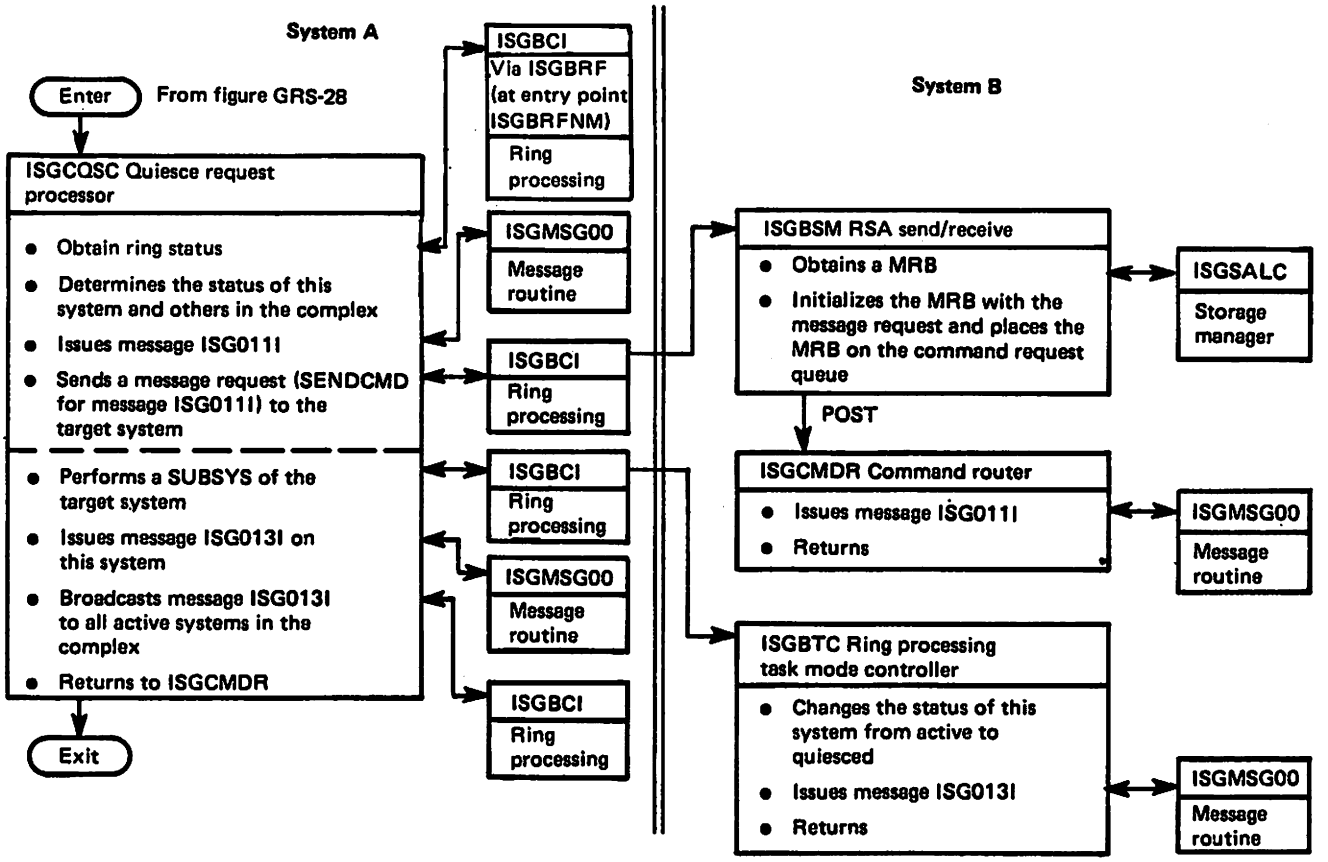


Figure 31. Process Flow for VARY GRS(x), QUIESCE to Another System

 (c) Copyright IBM Corp. 1987

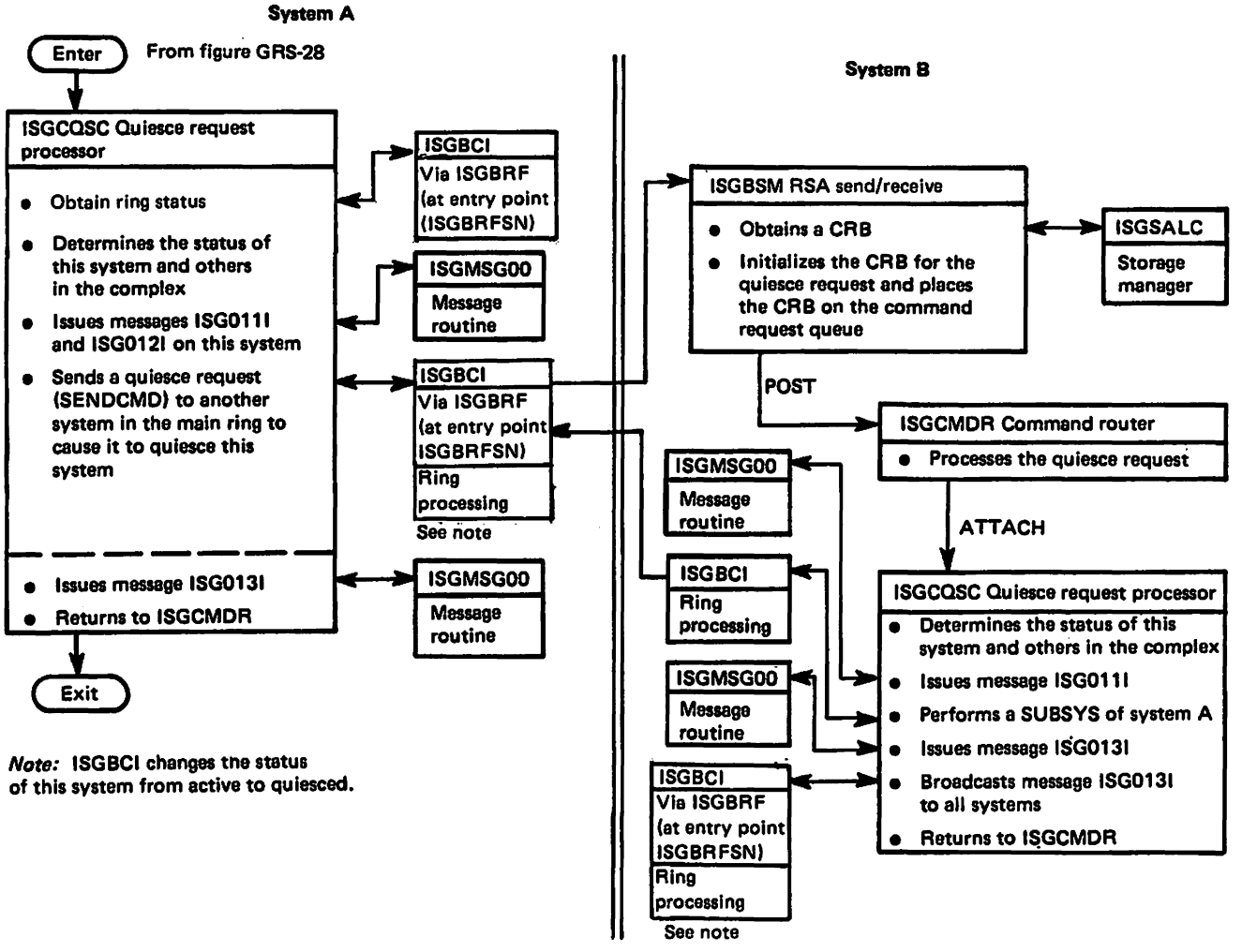


Figure 32. Process Flow for VARY GRS(x), QUIESCE by a System to Quiesce Itself

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

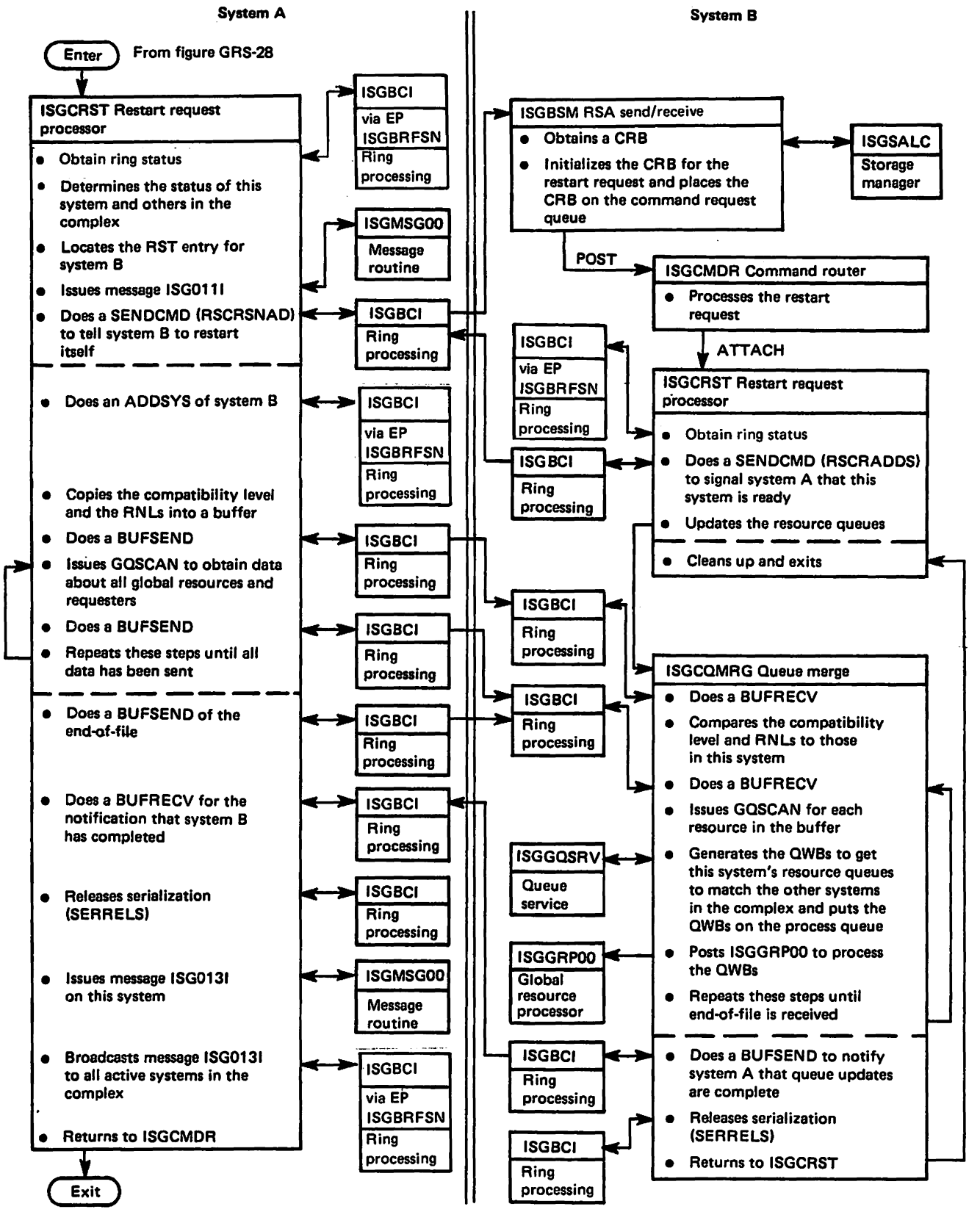


Figure 33. Process Flow for VARY GRS(x), RESTART to Restart Another System

 LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

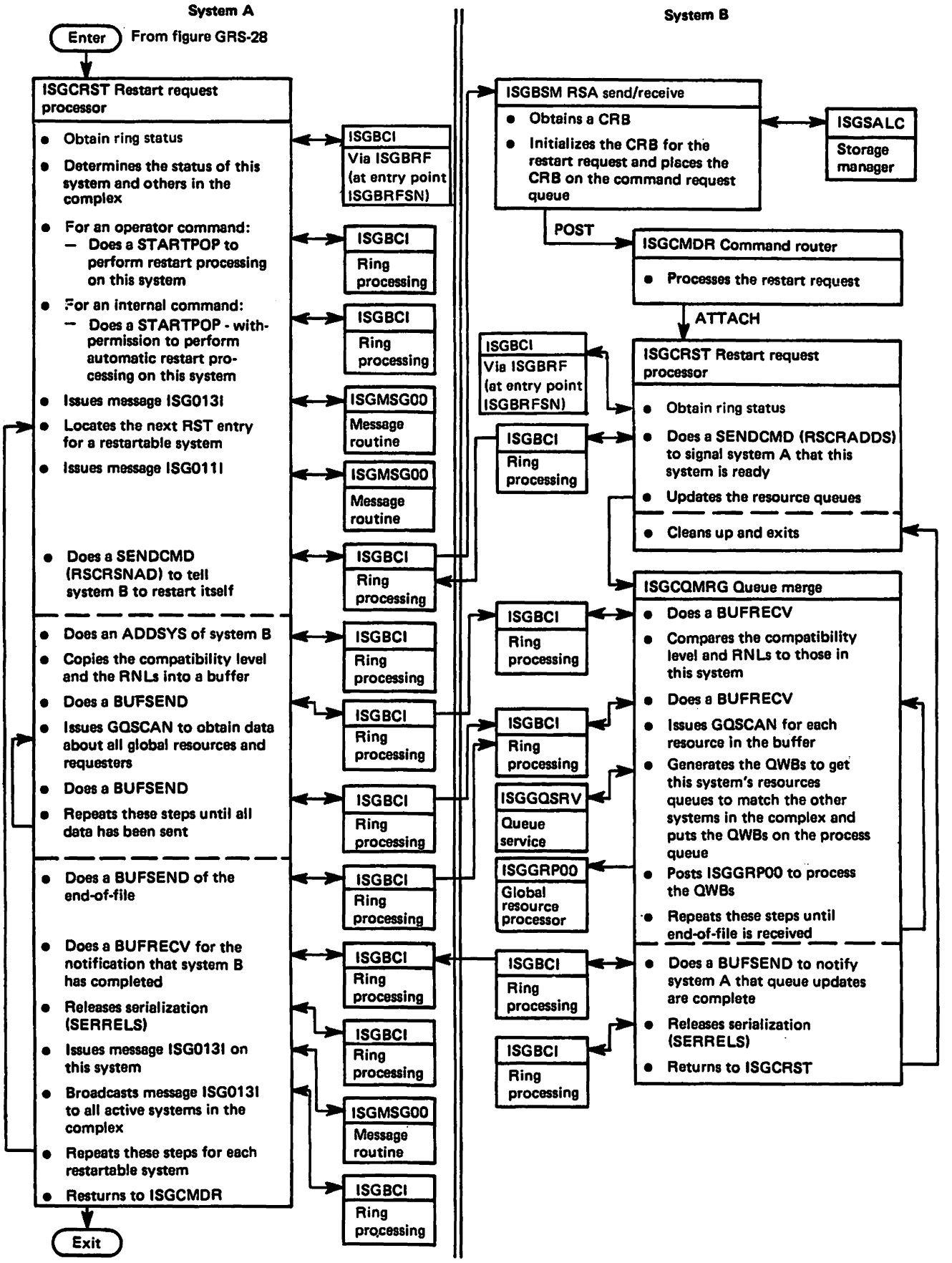


Figure 34. Process Flow for VARY GRS(ALL), RESTART to Restart All Systems

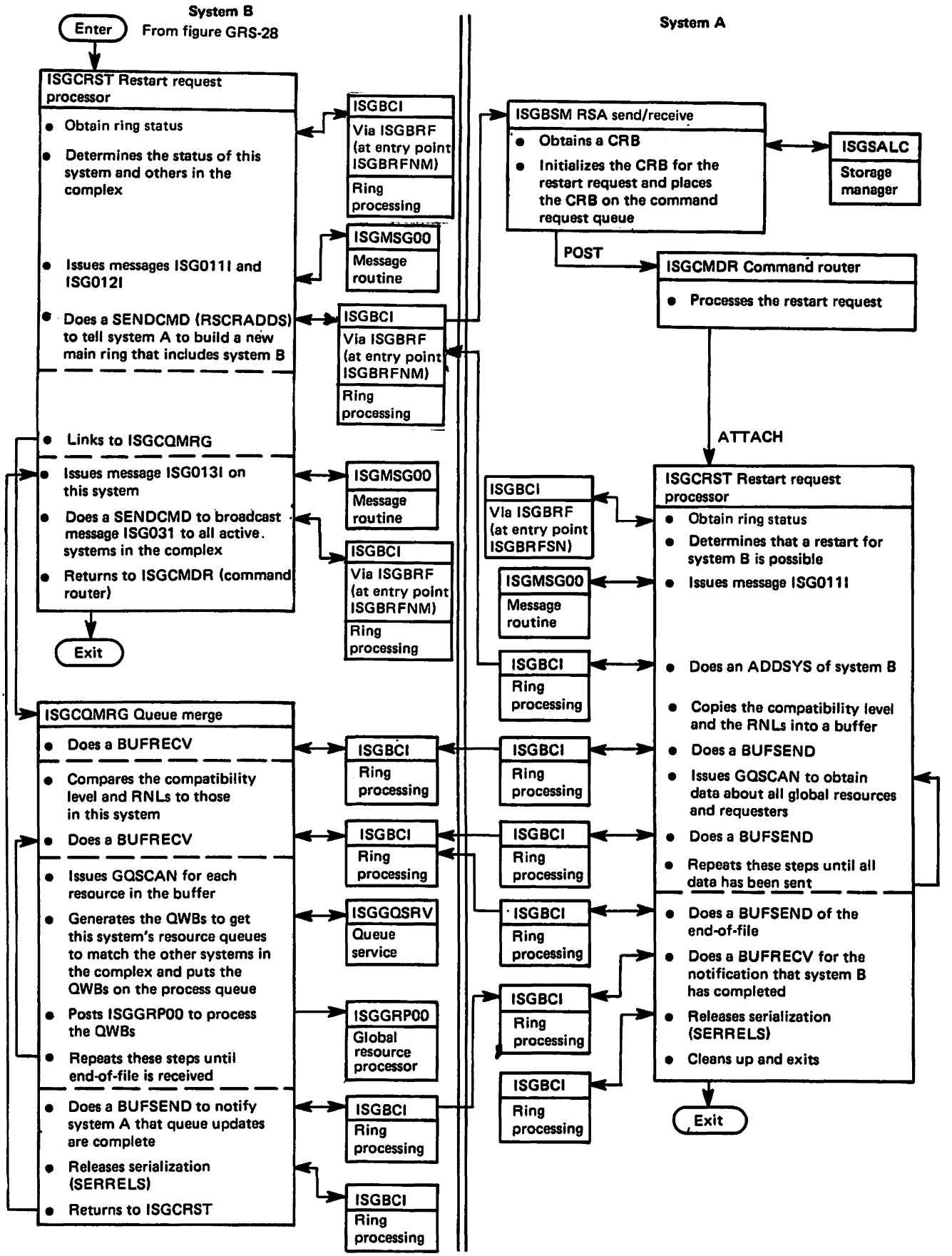


Figure 35. Process Flow for VARY GRS(x), RESTART by a System Not in the Main Ring

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

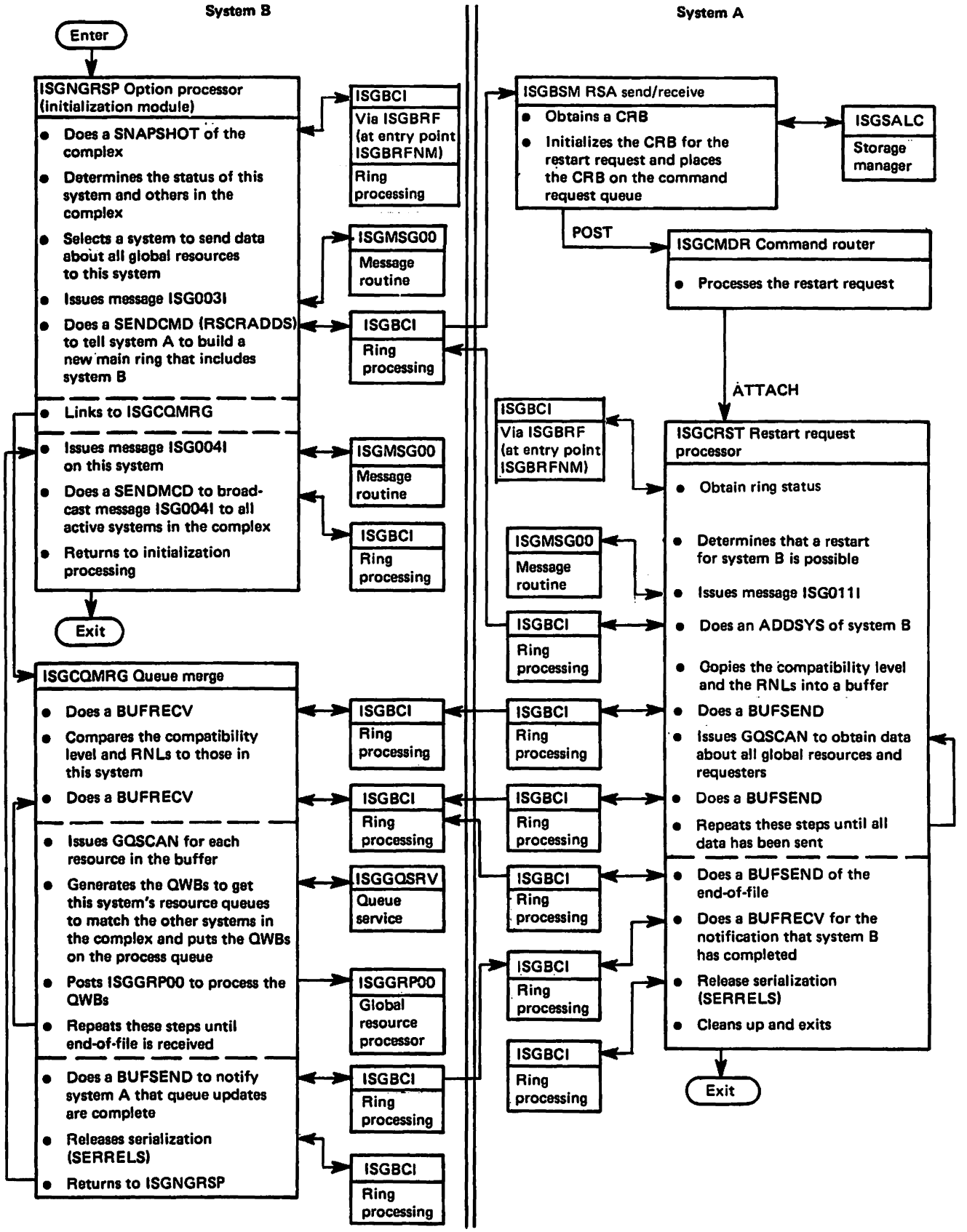


Figure 36. Process Flow for Join Processing at Initialization Time

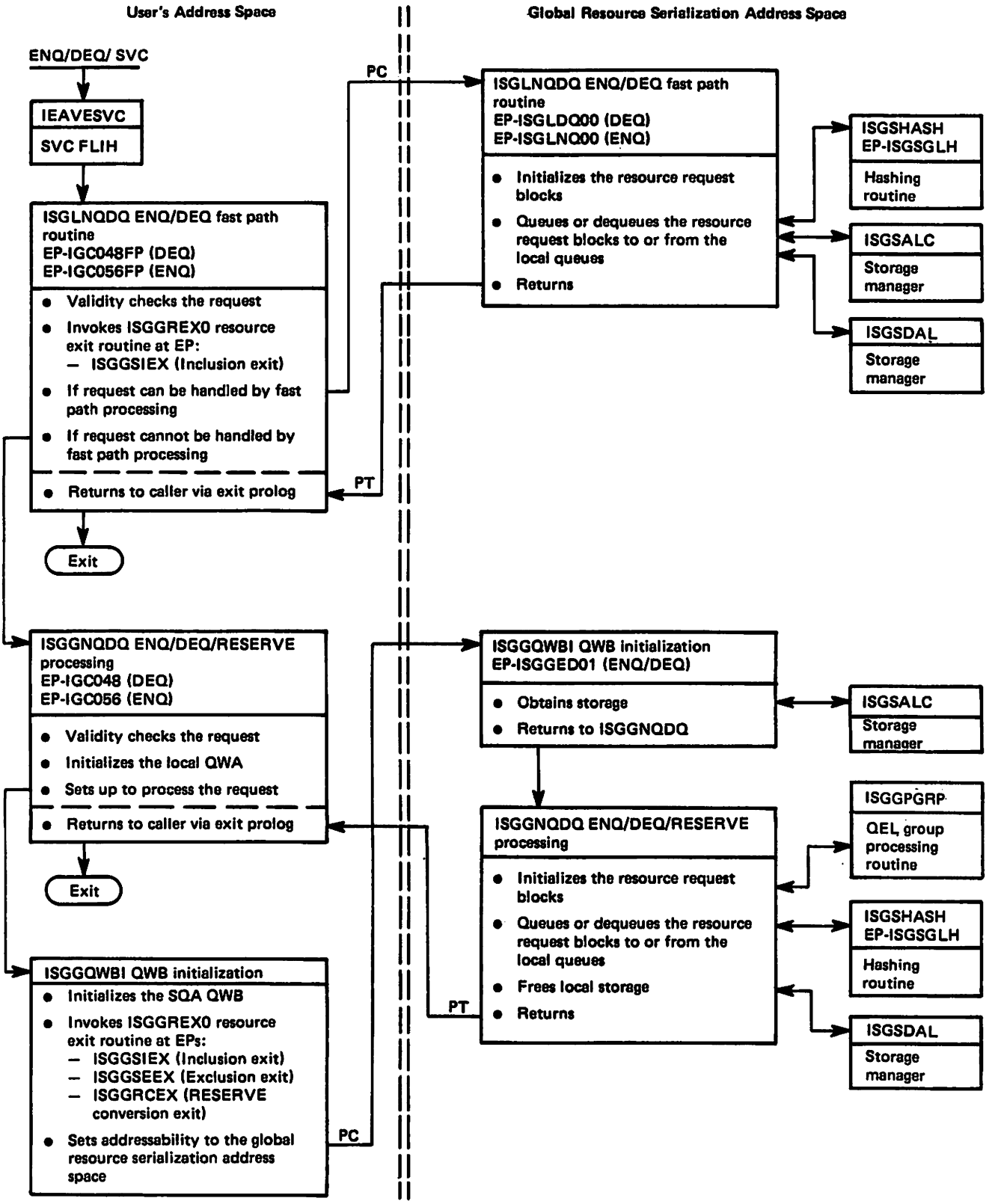


Figure 37. Process Flow for ENQ/DEQ Mainline - Local Resource Request

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

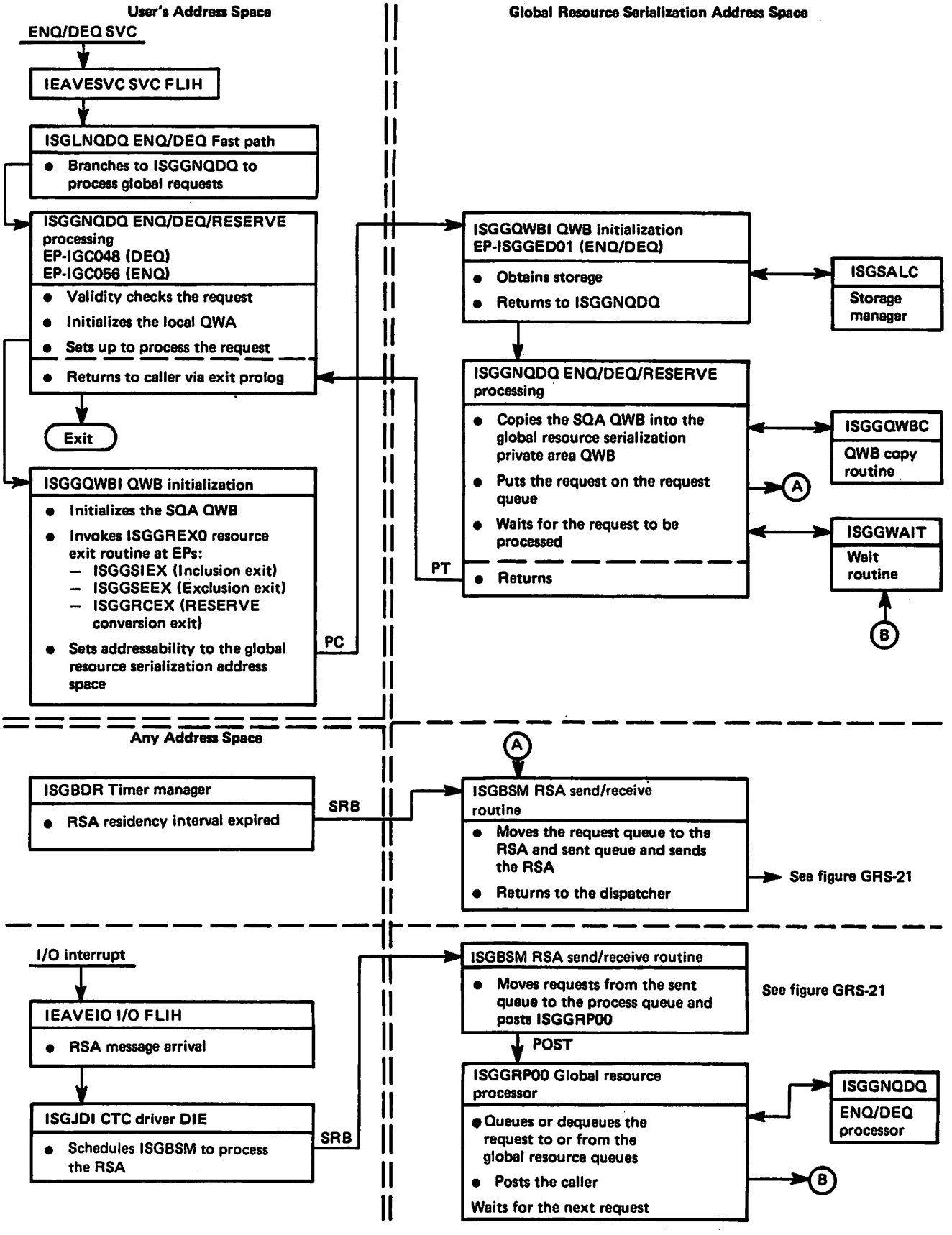


Figure 38. Process Flow for ENQ/DEQ Mainline - Global Resource Request

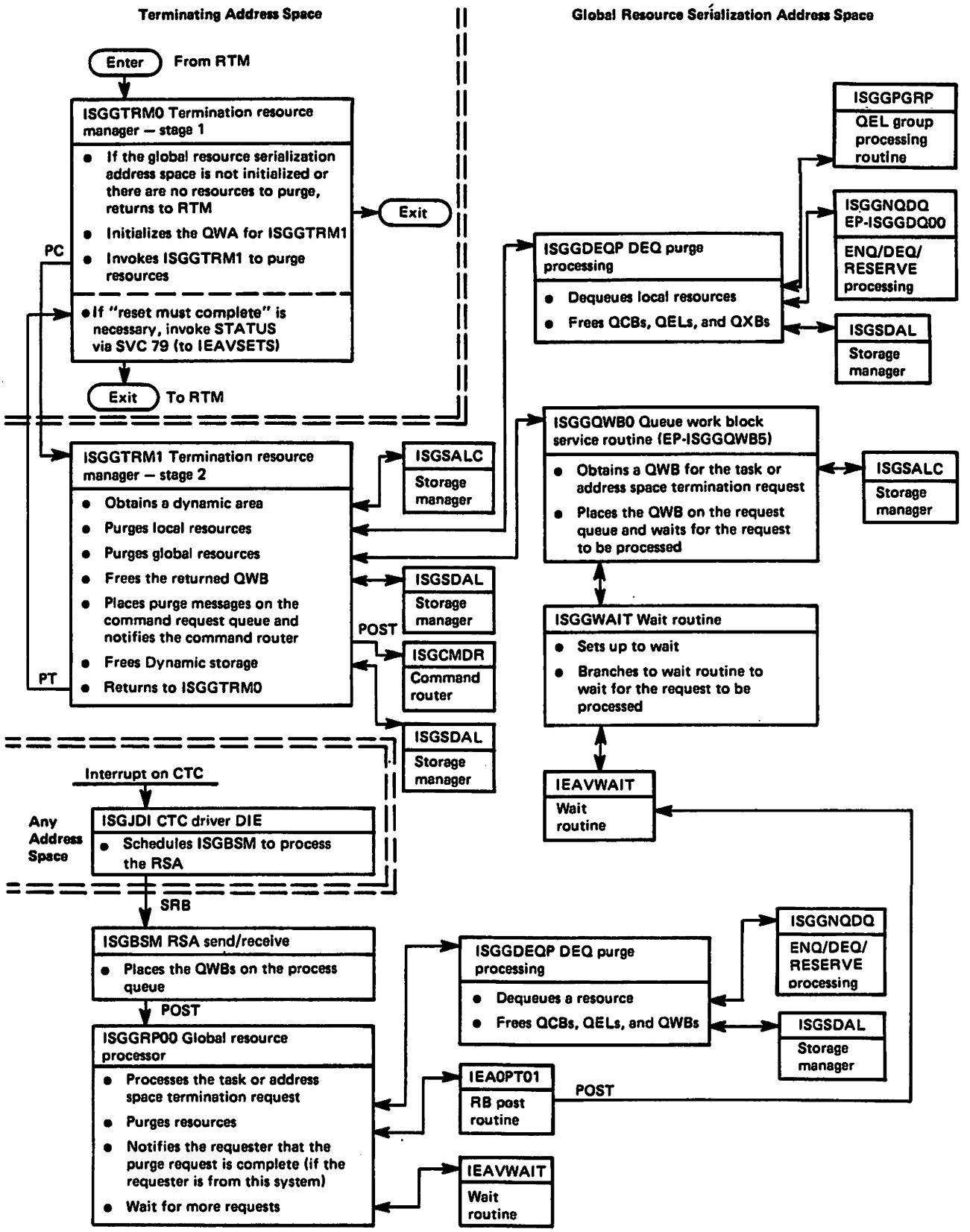


Figure 39. Process Flow for the Termination Resource Manager

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

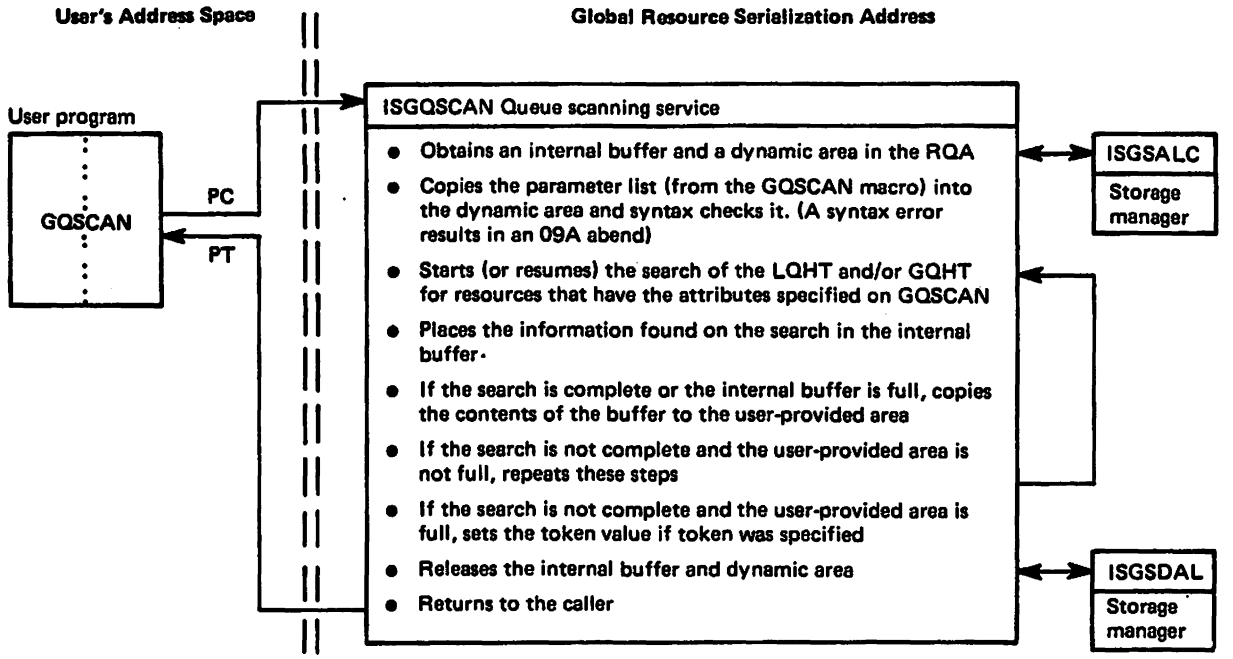


Figure 40. Process Flow for Queue Scanning Services

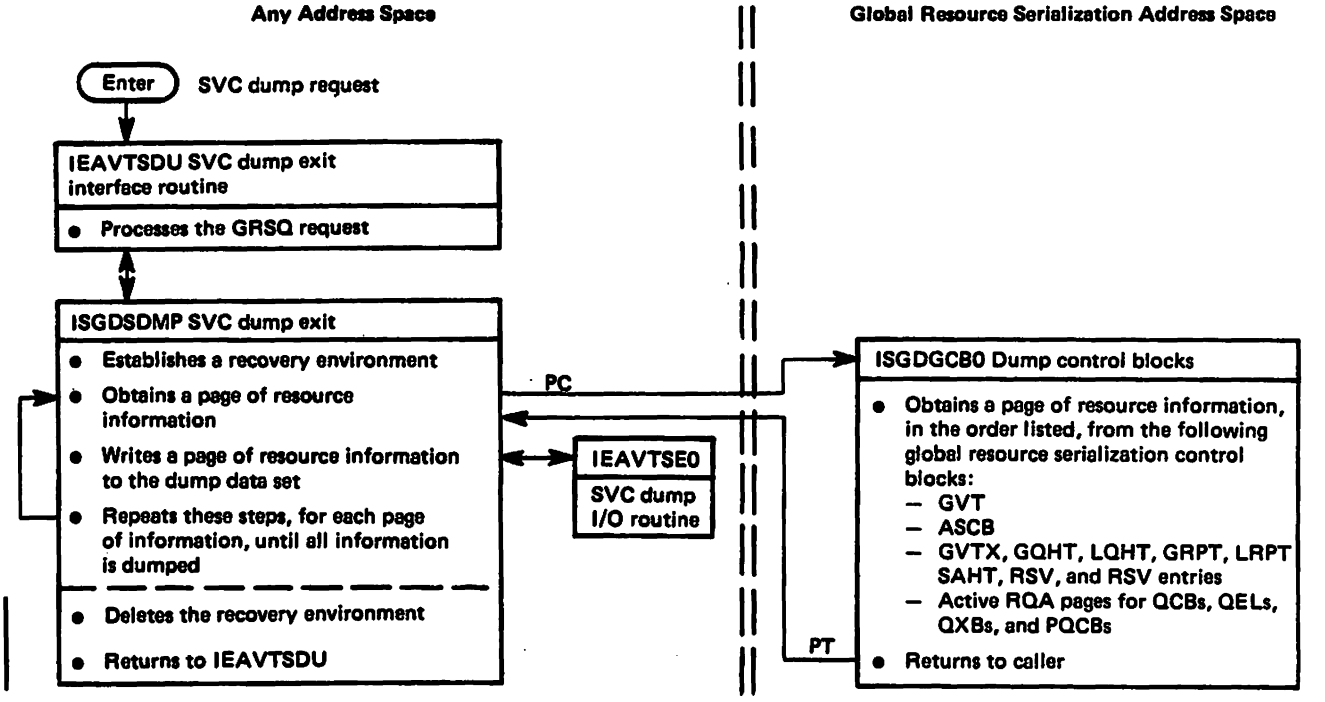


Figure 41. Process Flow for Dump Support - SVC Dump

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

The method-of-operation (m.o.) diagrams for the global resource serialization modules are named in the format "ISGxxxxx function" and are in alphabetic order, with the exception of the ring processing diagrams. Each ring processing diagram documents a separate function, not necessarily a separate module, and is named by the function documented. The ring processing diagrams are first.

The processing of modules that are not documented in separate diagrams is reflected in the diagram of the related function of the module's caller. Module descriptions of all executable global resource serialization modules except initialization modules follow the m.o. diagrams.

Note: Logic information, including m.o. diagrams, on global resource serialization initialization modules is in System Initialization Logic.

Method-of-operation 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 area. The object module name and labels in the extended description point to the code that performs the function.

Note that the relative size and the order of fields within input and output data areas do not always represent the actual size and format of the data area.

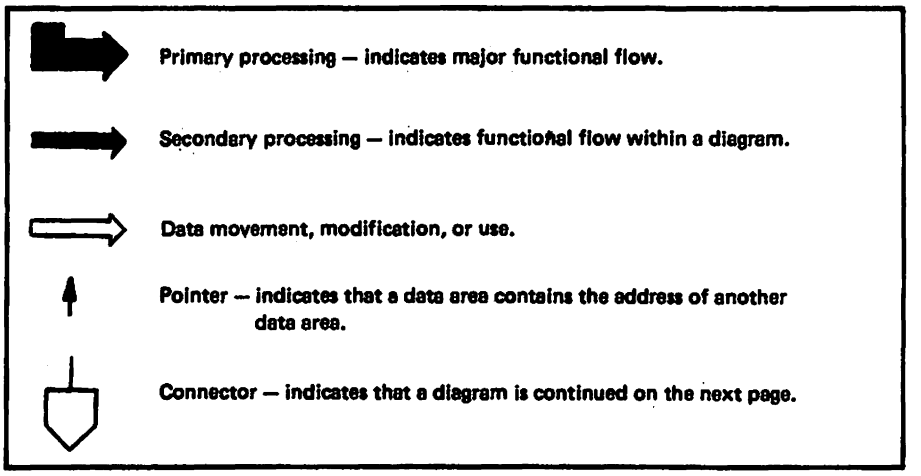


Figure 42. Key to Method-of-Operation Diagrams

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

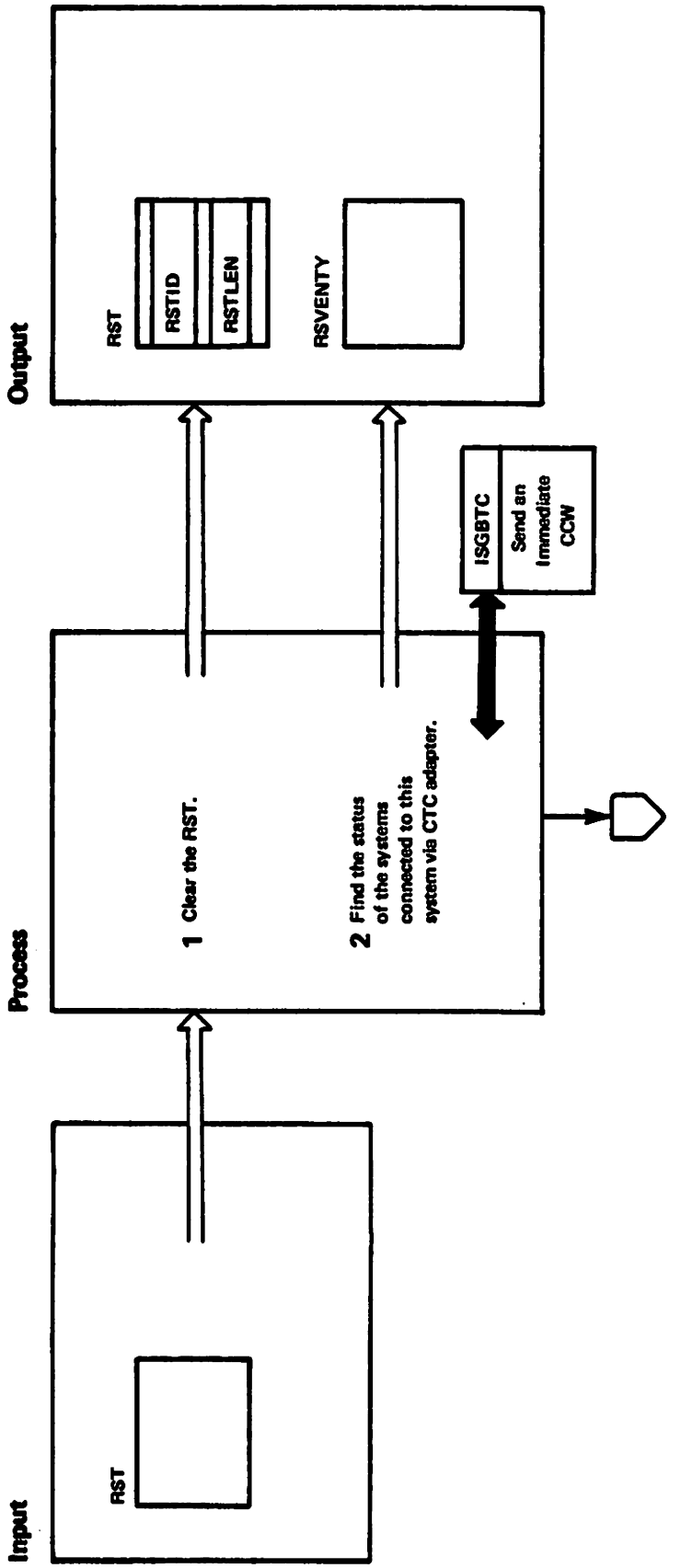


Diagram GRS-1. Provide Status Information (SNAPSHOT) (Part 1 of 4)

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-1. Provide Status Information (SNAPSHOT) (Part 2 of 4)

Extended Description	Module	Label
1 This routine is entered when the caller of ISGBCI specified the RSCFUNCT field as RSCFSNAP. This routine is referred to as the SNAPSHOT function and is called by several global resource serialization modules to get status information about the systems and CTC links in the global resource serialization complex. ISGBCI invokes ISGBRF (at entry point ISGBRFSN) to clear the ring processing status table (RST) except for the acronym (RSTID) and length (RSTLEN) fields.	ISGBRF	ISGBRFSN
2 ISGBRFSN updates the RSVENTY table to reflect the current status of the systems that are: • Immediate neighbors of this system in the main ring • Capable of responding to an immediate CCW ISGBCI invokes ISGBRF (at entry point ISGBRFRF) and initializes the entries in the RSVENTY table. It then calls ISGBTC (at entry point ISGBTCIR) once for each CTC that is not used to send or receive the main ring RSA. ISGBTC sends an immediate CCW to the system at the opposite end of the specified CTC. ISGBTC does this by scheduling the SRB to enter ISGBSR at entry point ISGBSRRI. ISGBSR calls the CTC driver (SENDBUF-IMMEDIATE function) to send an immediate CCW on the corresponding CTC.	ISGBTC ISGBSR	ISGBTCIR ISGBSRRI
After ISGBSR sends an immediate CCW on every qualifying CTC, ISGBRFRF pauses to allow asynchronous updating of the RSVENTY table. The following paragraph shows that asynchronous processing.	ISGBRF	BRFPAUSE
Asynchronous Processing		
ISGBSR is invoked (at entry point ISGBSRRI) once for each RSAIRCD received through the CTC driver. ISGBSR updates an RSVENTY entry with information taken from the received RSAIRCD.	ISGBSR	ISGBSRRI
ISGBRFRF ends its pause when all responses have been received, or when a specified time has elapsed.	ISGBRF	ISGBRFRF

LY28-1695-0 (c) Copyright IBM Corp. 1987

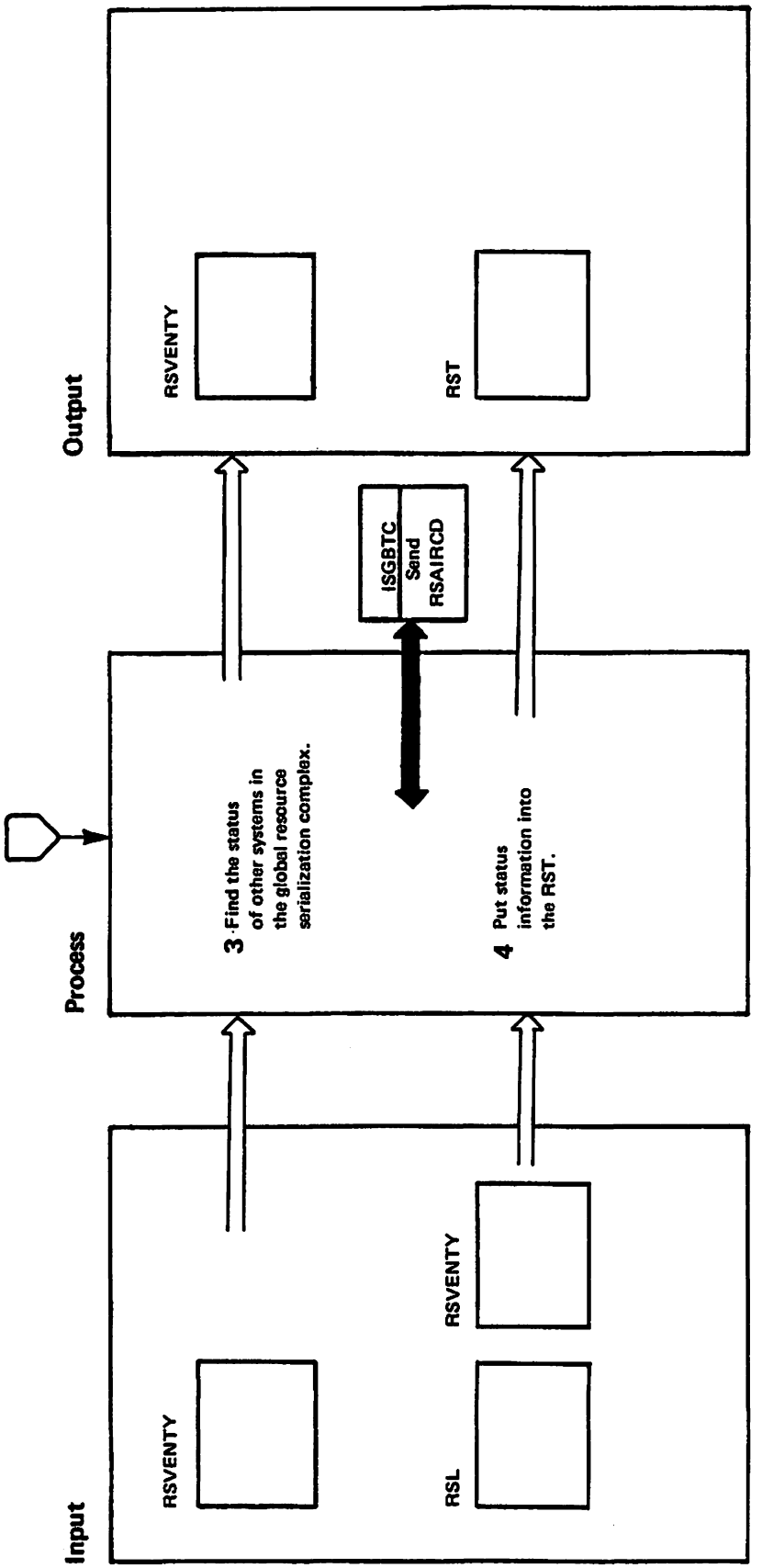


Diagram GRS-1. Provide Status Information (SNAPSHOT) (Part 3 of 4)

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-1. Provide Status Information (SNAPSHOT) (Part 4 of 4)

Extended Description	Module	Label
3 ISGBRFRF updates the RSVENTY table to reflect the current status of systems other than those covered by step 2. Those systems are:	ISGBRF	ISGBRFRF
• Systems that are in the main ring but are not neighbors of this system		
• Systems that have been removed from the main ring (via the VARY GRS, QUIESCE command) and may be incapable of responding to an immediate CCW.		
If the system executing SNAPSHOT is in the main ring, this step is a no-op. All systems in the main ring update their RSVENTY tables as systems are added to or removed from the main ring.		
If the system executing SNAPSHOT is not in the main ring but ISGBRFRF discovered a neighbor that is in the main ring, it invokes ISGBRF (at entry point ISGBRFNM) to send a series of RSAIRCDs to the neighboring system and to check for responses. Each RSAIRCD that is sent requests information from a particular RSVENTY entry in the neighboring system. Each response contains one of the following:		ISGBRFNM
• Information from the requested entry		
• Information from some other entry		
• Flag indicating that all entries have been sent in previous response RSAIRCDs. When ISGBRFRF detects this condition, the RSVENTY table on the system that requested the SNAPSHOT has been updated.		
Asynchronous Processing		
The following processing occurs at the same time that the processing in the previous paragraph is occuring.		
ISGBRF (at entry point ISGBRFNM) calls module ISGBTC (entry point ISGBTCIR) to send each RSAIRCD. ISGBTC schedules an SRB to enter module ISGBSR (at entry point ISGBSRRI). ISGBSR invokes the CTC driver function SENDBUF to send the RSAIRCD. The response RSAIRCD causes the CTC driver to schedule ISGBSR (entry point ISGBSRRI). ISGBSR then updates the RSVENTY table with information contained in the response RSAIRCD. ISGBSR sets field RSLICRF to show that the response RSAIRCD has arrived so that ISGBRFNM can send the next RSAIRCD of the series.		
4 ISGBRFSN takes the system status information from the RSVENTY table and puts it into the RST. The first entry in the RST describes this system (that is, the system executing the SNAPSHOT). Each entry in the system section describes a system known to ISGBRFSN. Then ISGBRFSN takes the CTC status information from the RSL and puts it into the CTC link entry section of the RST.	ISGBRF	
Recovery Processing		
• ISGBRF (at entry point ISGBRFRF) might set flag bits to cause its caller to re-invoke the ISGBRFRF subroutine. Conditions that cause these flags to be set are:		
• A new main ring is discovered while the ISGBRFRF is in progress. The old main ring failed while the SNAPSHOT function was in progress.		
• A table overflow occurs in the RSVENTY table of this system.		
• Return code 16 indicates that the SNAPSHOT request was unsuccessful and that communication with any other system is impossible.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

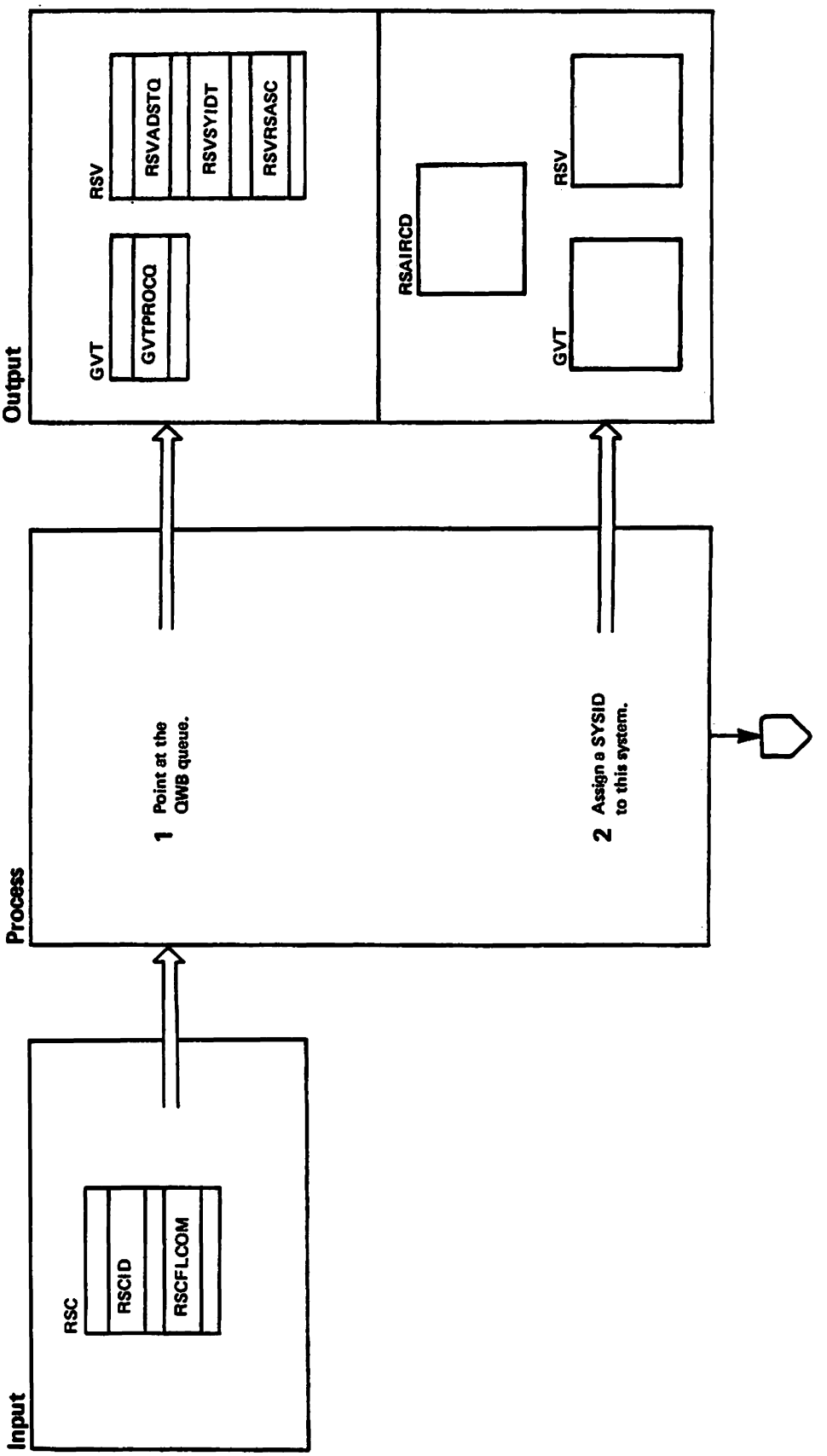


Diagram GRS-2. Initialize One-System Main Ring (STARTPOP) (Part 1 of 4)

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-2. Initialize One-System Main Ring (STARTPOP) (Part 2 of 4)

Extended Description	Module	Label
This routine is called to create a ring of one system. This is done when creating the ring for the first time (as a result of the GRS=START option) and when the operator rebuilds the main ring manually after a previous main ring failure:		
Note: The internally-issued system command that automatically rebuilds a disrupted ring invokes ISGBCI to handle the function to request permission and STARTPOP. ISGBCI invokes ISGBRF (at entry point ISGBRFSP) to handle this request. See Diagram GRS-3 for the processing.		
1 ISGBRFSP initializes field RSVADSTQ to point to the QWB process-queue.	ISGBRF	ISGBRFSP
2 The SYSID of a system is assigned when the system first enters the main ring and is used until an IPL is performed on the system again. The first system to create the main ring is assigned SYSID 1 by ISGBRF (at entry point ISGBRFSP). Other systems are assigned a SYSID as they join the main ring for the first time. ISGBRF (at entry point ISGBRFSP) places the system's SYSID into the RSAIRCD, RSVENTY, and the GVT.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

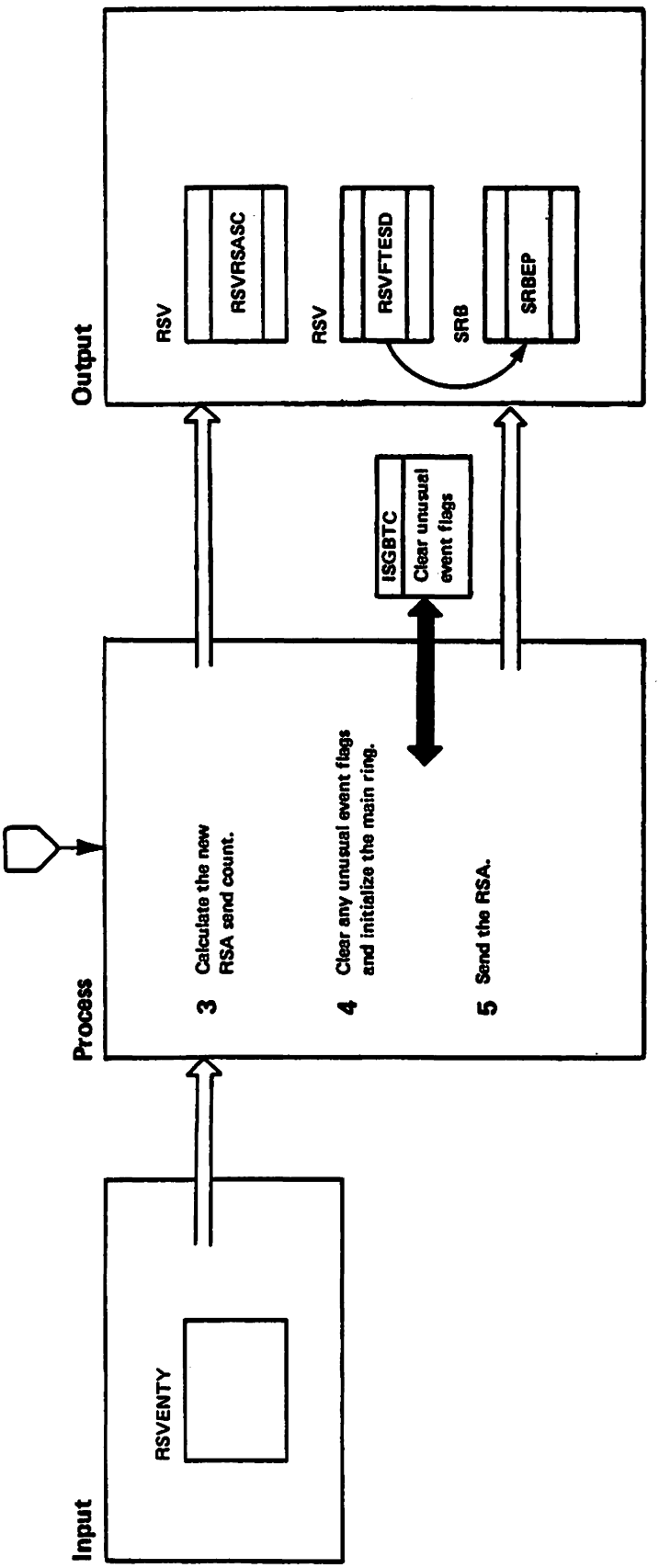


Diagram GRS-2. Initialize One-System Main Ring (STARTPOP) (Part 3 of 4)

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-2. Initialize One-System Main Ring (STARTPOP) (Part 4 of 4)

Extended Description	Module	Label
3 ISGBRF (at entry point ISGBRFSP) sets the RSA send count (RSVRSASC) to the value of RSVRSASC before the main ring failure plus the number of entries in the RSVENTY table. This ensures that the new RSA send count is unique. This is necessary to allow such systems to correctly adjust their QWB queues when they rejoin the re-created main ring.		
4 ISGBRFSP invokes ISGBTC (at entry point ISGBTCR1) to clear the unusual events. ISGBTC posts the exception handling task using the GVTXECB ECB, which is waited on by ISGBTC. The exception handling task clears unusual events and turns on flag GVTMAINR to indicate that this system is in the main ring. It then posts the RSVR1ECB to allow ISGBTC (at entry point ISGBTCR1) to proceed.	ISGBTC	ISGBTCR1
5 ISGBTC (at entry point ISGBTCR1) places the system in "one-system" mode by setting flag RSVFRNG1 and scheduling ISGBSR (entry point ISGBSRSR) to perform the first send-and-receive of the RSA. A system is in "one-system" mode when it does not send the RSA through a CTC. ISGBTC schedules ISGBSR (entry point ISGBSRSR) so that ISGBSR can simulate a send-and-receive of the RSA by copying the RSA from its output buffer to its input buffer. ISGBTC (entry point ISGBTCR1) then returns to ISGBRF (at entry point ISGBRFSP), which then returns to its caller.	ISGBTC	ISGBTCR1

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

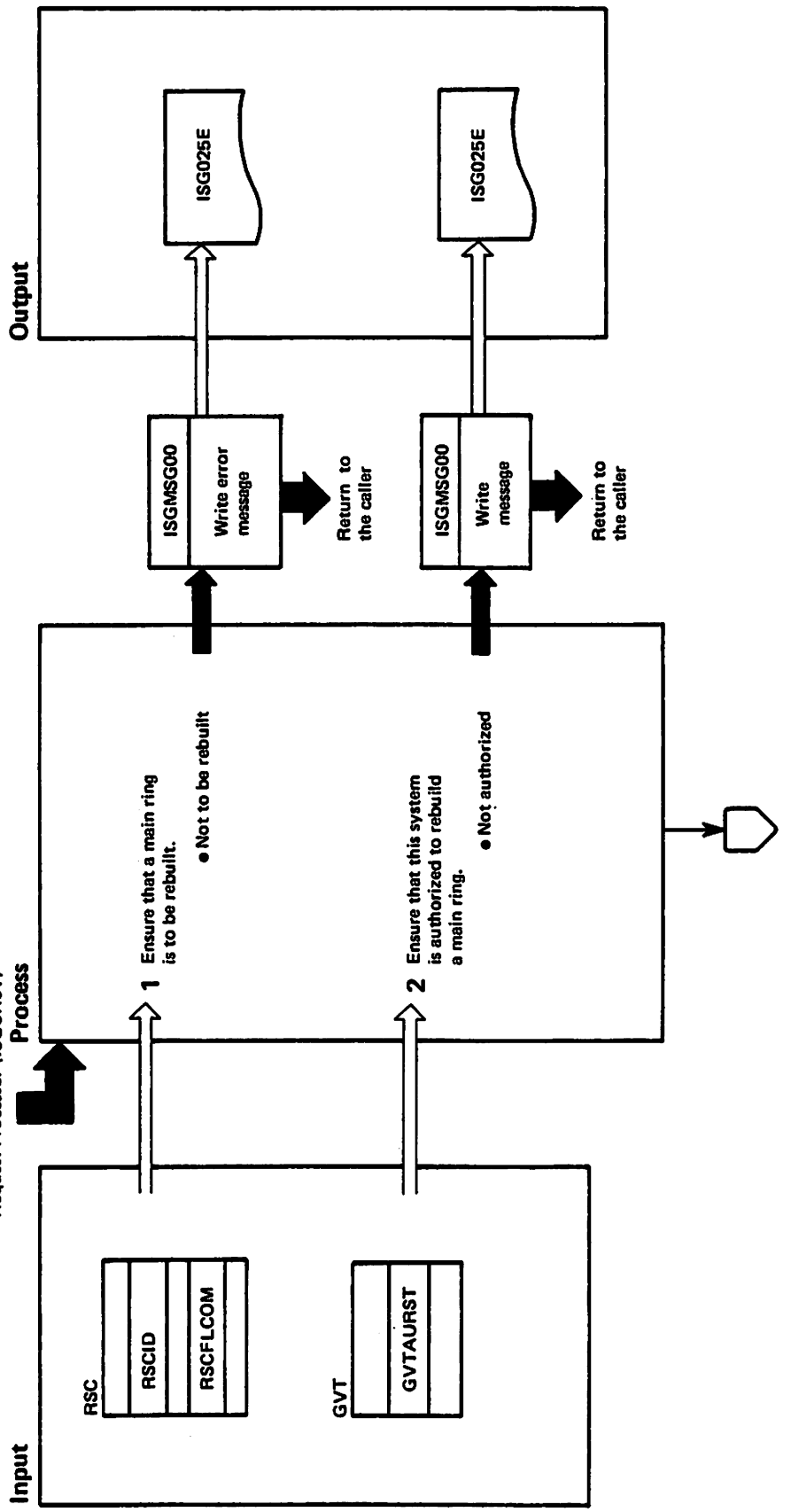


Diagram GRS-3. Request Permission to Initialize a One-System Main Ring (REQPERM) (Part 1 of 8)

LY28-1695-0 (c) Copyright IBM Corp. 1987

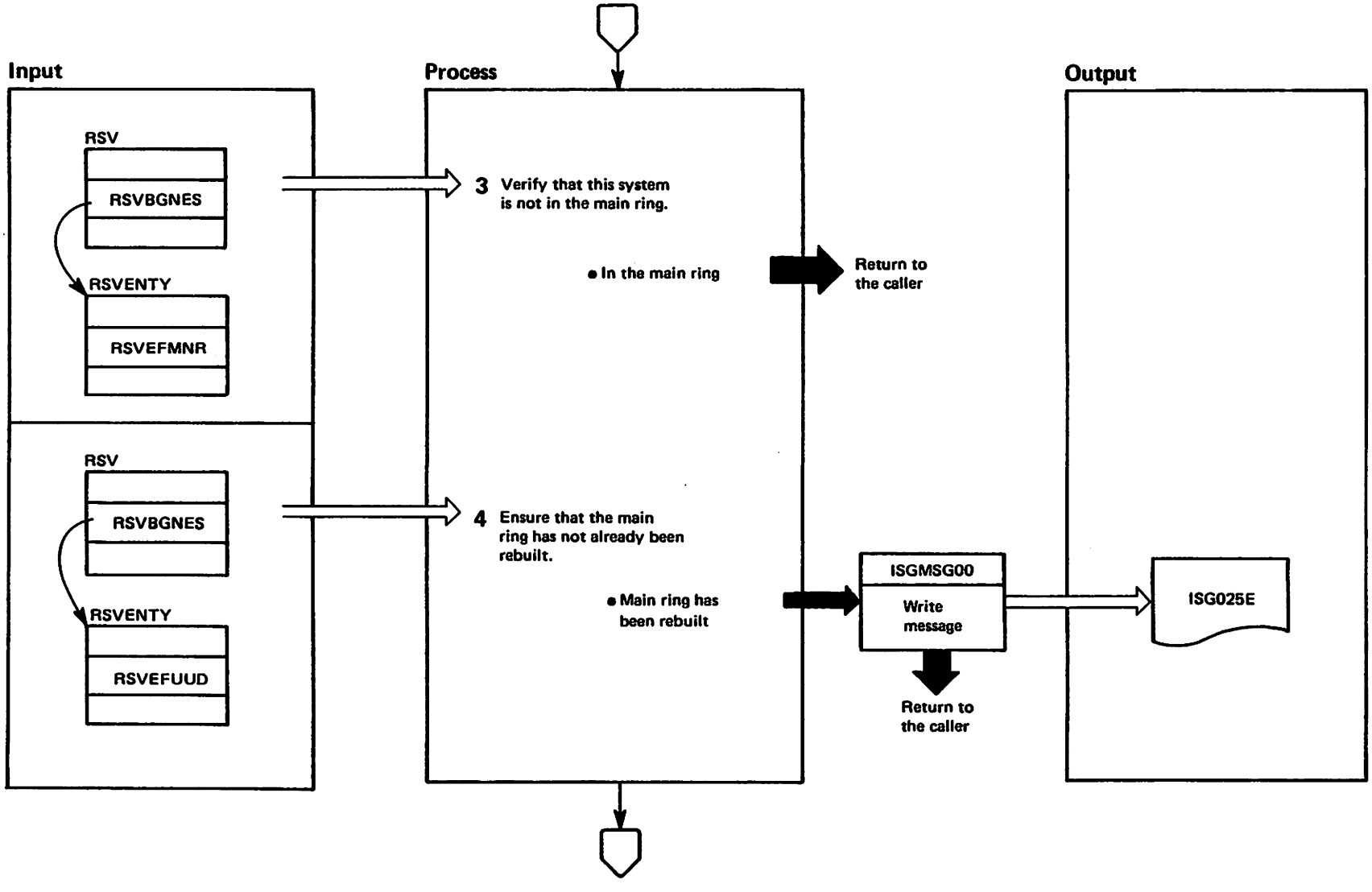
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-3. Request Permission to Initialize a One-System Main Ring (REQPERM) (Part 2 of 8)

Extended Description	Module	Label
This routine is called to build a ring of one system; it is invoked to process a system-issued VARY GRS (ALL), RESTART command that ring processing issues when it detects a main ring failure. This routine creates a ring of one system only if it can obtain permission to do so from the systems that were in the main ring when the failure occurred.		
1 If the RSCFLCOM flag is off (RSCFLCOM='0'), then ISGBCI invokes ISGBRF (at entry point ISGBRFSP) to only issue an operator message. The RSCFLCOM flag tells ISGBCI whether or not to rebuild a main ring. If ISGBRFSP is to rebuild a main ring then, processing continues with step 2; otherwise, ISGBRFSP issues message ISG025E (SYSTEM ERROR) and returns to the caller.	ISGBRF	ISGBRFSP
2 ISGBRFSP checks the GVTAURST flag in the GVT to see whether this system is authorized to rebuild a main ring. If GVTAURST indicates that RESTART (NO) was specified in the GRSCNFxx parmlib member, then this system is not authorized to rebuild the main ring. In this case, ISGBRFSP issues message ISG025E (SYSTEM NOT AUTHORIZED), sets a non-zero return code indicating that it did not rebuild a main ring, and returns to the caller. If this sytem is authorized to rebuild a main ring (RESTART(YES) was specified in the GRSCNFxx parmlib member), ISGBRFSP continues at step 3.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-3. Request Permission to Initialize a One-System Main Ring (REQPERM) (Part 3 of 8)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

Diagram GRS-3. Request Permission to Initialize a One-System Main Ring (REQPERM) (Part 4 of 8)

Extended Description	Module	Label
3 ISGBRF (at entry point ISGBRFSP) checks the RSVEFMNR field in the RSVENTY to see if this system has already been brought into the main ring that was rebuilt by some other system. If this system has already been brought into the main ring, ISGBRFSP sets a non-zero return code and returns to the caller. If this system has not already been brought into the main ring, processing continues at step 4.	ISGBRF	ISGBRFSP
4 ISGBRF (at entry point ISGBRFSP) checks the RSVEFUUD field in the RSVENTY to see if the main ring has been rebuilt by some other system with this system not being part of the rebuilt main ring. If this system is not part of this rebuilt main ring, ISGBRFSP issues message ISG025E (option ALL ACTIVE SYSTEM EXISTS), sets a non-zero return code, and returns to the caller; otherwise, processing continues at step 5.	ISGBRF	ISGBRFSP

LY28-1695-0 (c) Copyright IBM Corp. 1987

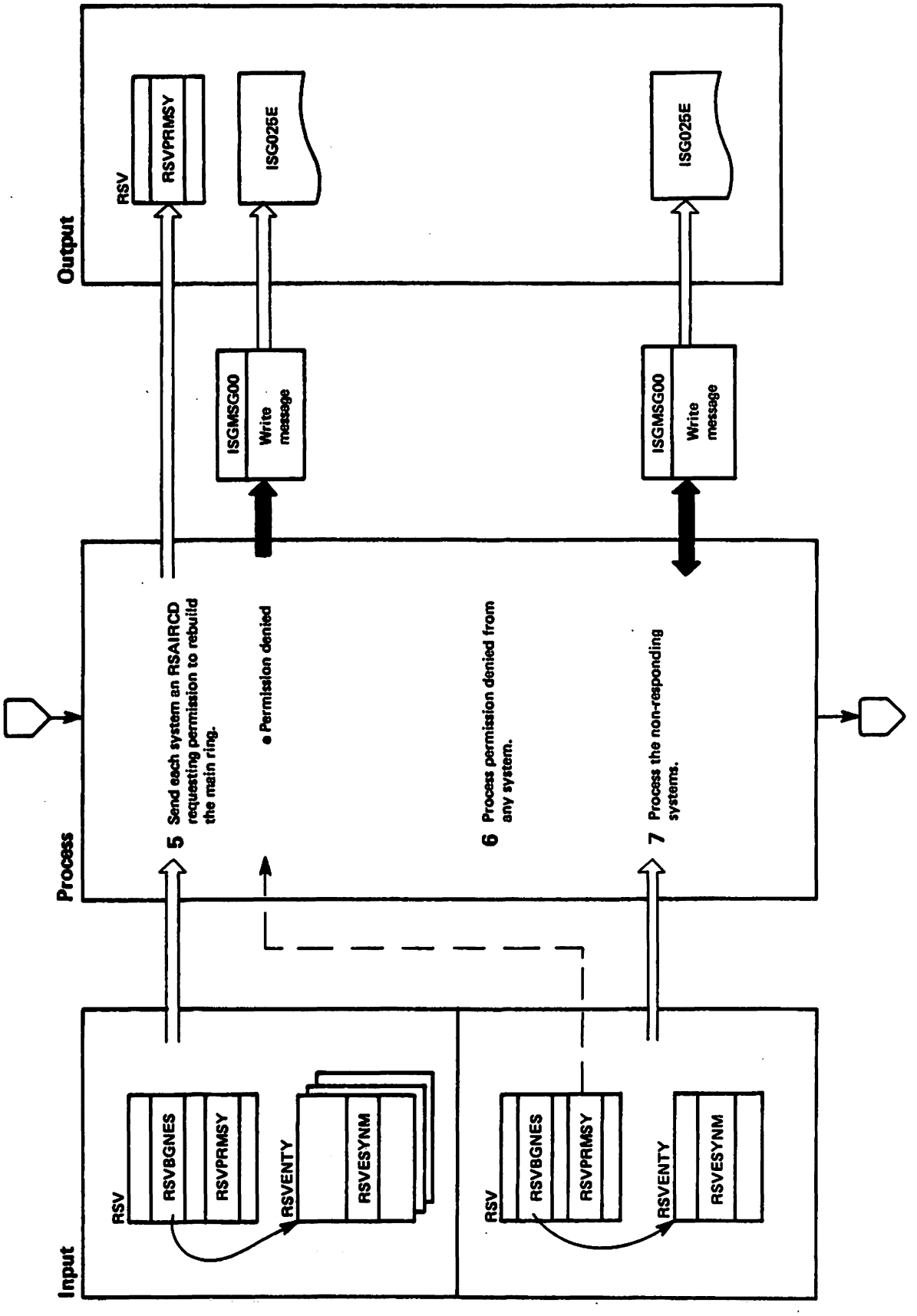


Diagram GRS-3. Request Permission to Initialize a One-System Main Ring (REQPERM) (Part 5 of 8)

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-3. Request Permission to Initialize a One-System Main Ring (REQPERM) (Part 6 of 8)

Extended Description	Module	Label
5 ISGBRF (at entry point ISGBRFSP) asks permission to rebuild the main ring of each system that was in the main ring when the main ring failure occurred. ISGBRFSP asks permission of each system one at a time, from the highest SYSNAME to lowest SYSNAME. This system (that is the one ISGBRF (at entry point ISGBRFSP) is running on) gives permission to itself by changing field RSVPRMSY of the RSV from zero to its own SYSNAME.		GETNAME GETRSL
This system calls the subroutine entry point ISGBRFNM to request permission from another system. ISGBRFNM sends a request-for-permission RSAIRCD to that system. If there is no response to the sent RSAIRCD within the required amount of time, this system calls ISGBRFNM to send the RSAIRCD again but across some other CTC. If it receives no response after trying all CTCs to a given target system, ISGBRF (at entry point ISGBRFSP) considers the system as "non-responding." ISGBRFSP then goes on to process the next system identified by the next SYSNAME.	ISGBRF	ISGBRFNM
6 ISGBRF (at entry point ISGBRFSP) can receive an indication that permission is denied to rebuild the main ring. In this case, ISGBRF (at entry point ISGBRFSP) receives the RSAIRCD that contins the name of the system that is going to automatically rebuild the main ring; ISGBRF (at entry point ISGBRFSP) then issues message ISG025E (PERMISSION GRANTED TO SYSTEM sysname). Field RSVPRMSY and the text of message ISG025E contain the SYSNAME of the system that is going to automatically rebuild the disrupted ring.		

Extended Description	Module	Label
7 ISGBRF (at entry point ISGBRFSP) processes those systems that did not respond to the RSAIRCD sent in step 5. If RESTART(YES) was specified in the GRSCNFxx parmlib member for any non-responding system that had completely entered the main ring before the failure occurred, ISGBRF (at entry point ISGBRFSP) rebuilds the main ring only if the number of responding systems exceeds the number of non-responding systems. If the non-responding systems all had RESTART(NO) in their GRSCNFxx parmlib members or had not successfully executed ISGQMRG before the main ring failure, ISGBRF (at entry point ISGBRFSP) rebuilds the main ring only if the number of responding systems exceeds or is equal to the number of non-responding systems. (A system always counts itself as a responding systems.) If the main ring is not to be rebuilt, ISGBRFSP issues message ISG025E (INSUFFICIENT NUMBER OF RESPONDING SYSTEMS) and returns to the caller with a non-zero return code indicating the results of the processing of the non-responding systems. If the main ring is to be rebuilt after processing the non-responding systems, ISGBRF (at entry point ISGBRFSP) continues with step 8.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

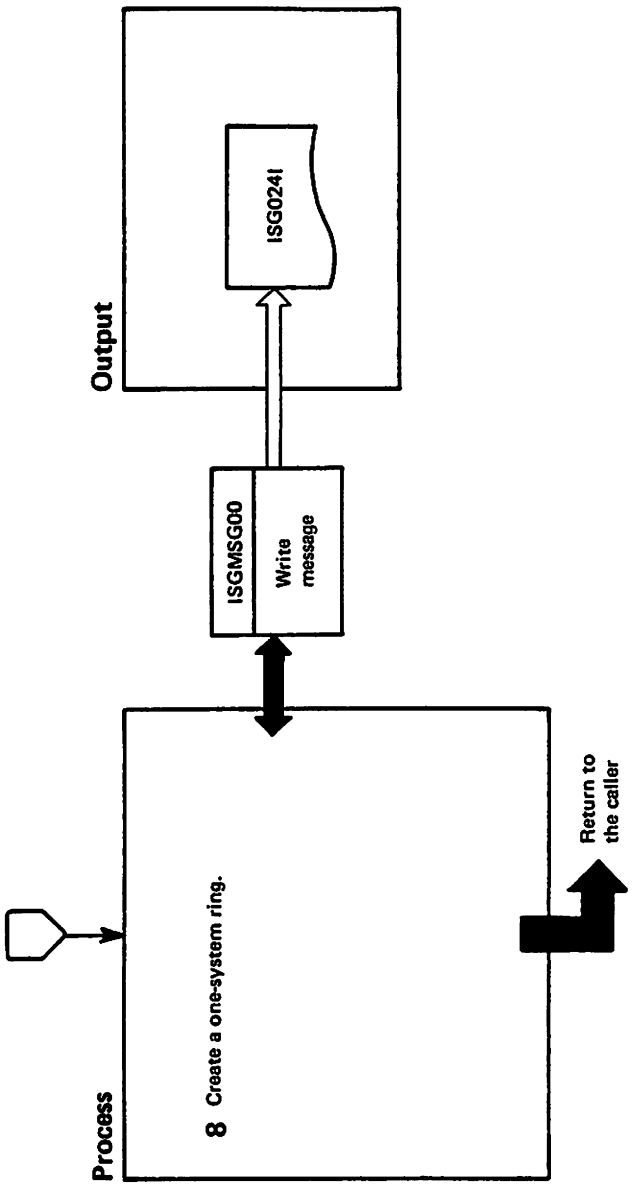


Diagram GRS-3. Request Permission to Initialize a One-System Main Ring (REQPERM) (Part 7 of 8)

LY28-1695-0 (c) Copyright IBM Corp. 1987

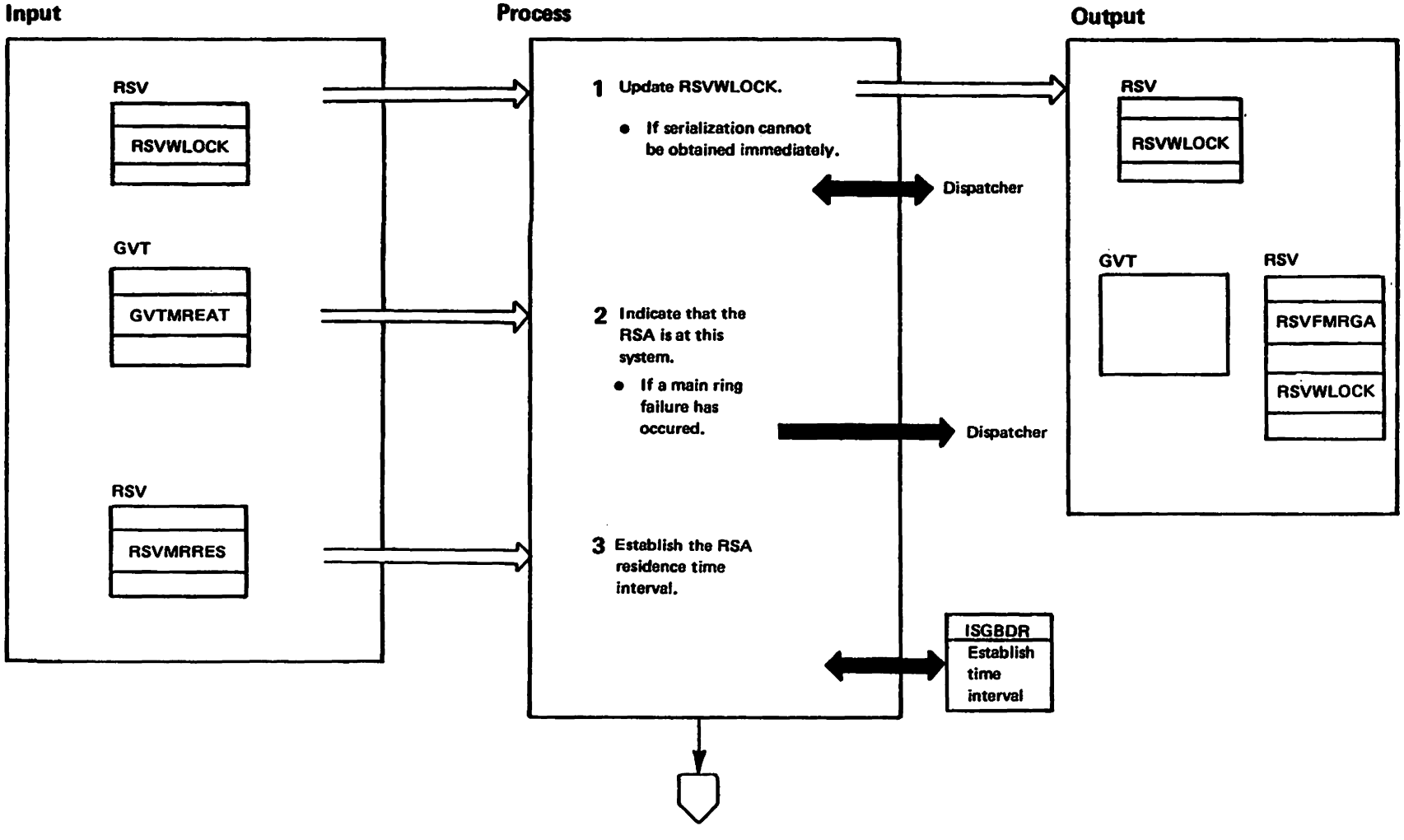
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-3. Request Permission to Initialize a One-System Main Ring (REQPERM) (Part 8 of 8)

Extended Description	Module	Label
8 ISGBRF (at entry point ISGBRFSP) builds a one-system main ring. (See the diagram "Initialize One-System Main Ring (STARTPOP)" for further information on the processing to create a one-system main ring.) ISGBRF (at entry point ISGBRFSP) then issues message ISG024I and returns to the caller with a return code of zero to indicate that a one-system main ring was rebuilt.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-4. Receive the RSA (Part 1 of 10)



"Restricted Materials of IBM"
Licensed Materials - Property of IBM

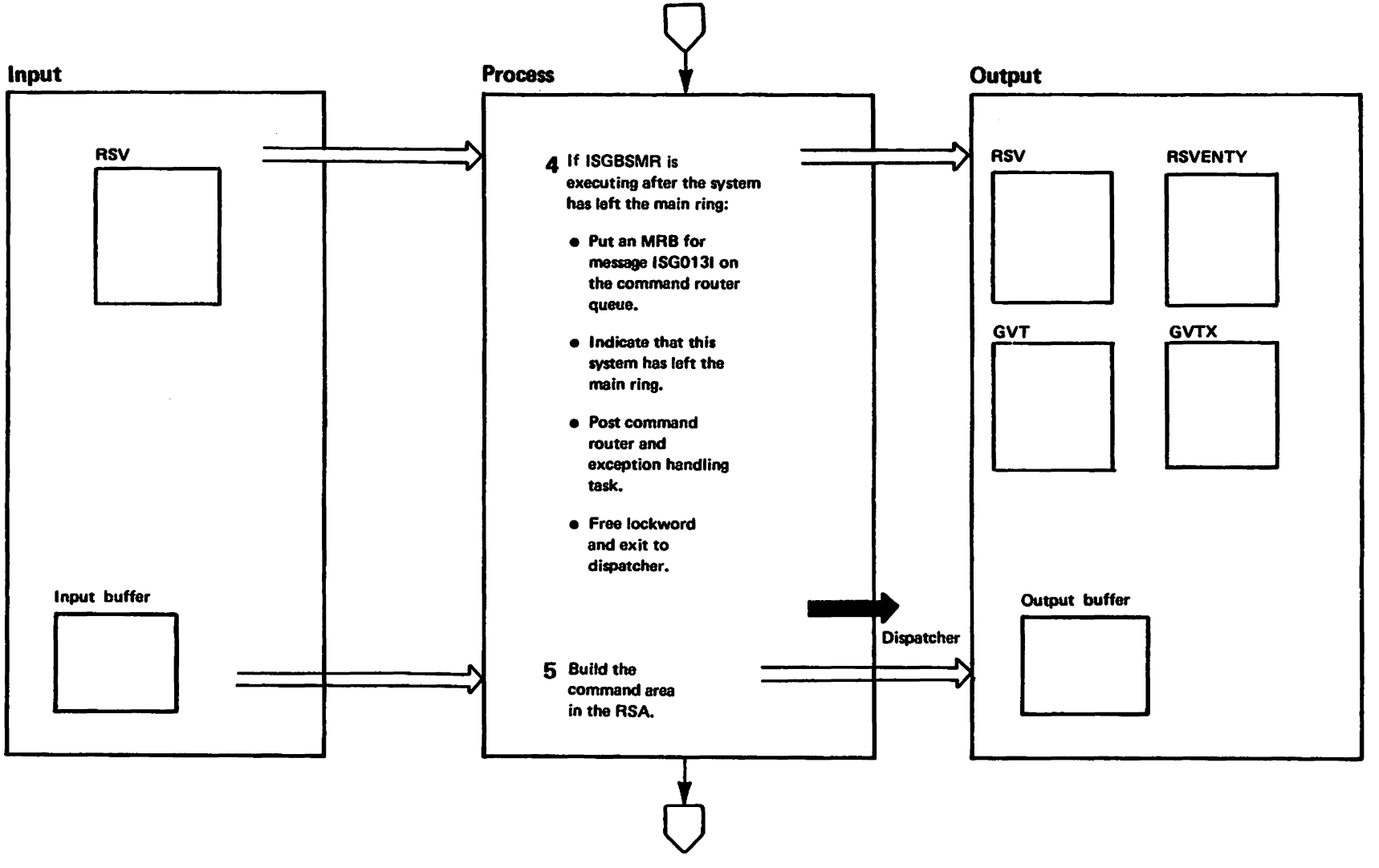
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-4. Receive the RSA (Part 2 of 10)

Extended Description	Module	Label
Entry point ISGBSM (at entry point ISGBSMR) of module ISGBSM (at entry point ISGBSMR) is scheduled when the main ring RSA is received and must be processed. It executes in SRB mode, key 0, supervisor state. Recovery is performed by module ISGBFRCV.		
1 Entry points ISGBSM (at entry point ISGBSMR) and ISGBSM (at entry point ISGBSMSR) use RSVWLOCK to serialize the RSV. If RSVWLOCK is in use by ISGBSMSR, then ISGBSM (at entry point ISGBSMR) alters RSVWLOCK and exits to the dispatcher. (ISGBSM (at entry point ISGBSMSR) will see the altered value and will branch to ISGBSM (at entry point ISGBSMR) instead of exiting to the dispatcher, when it has completed its processing.)	ISGBSM	ISGBSMR
If RSVWLOCK is not in use, ISGBSM (at entry point ISGBSMR) alters the value to indicate that it is now being used.		
2 ISGBSM (at entry point ISGBSMR) changes the low order bit of GVTMREAT from 0 to 1 to show that the RSA is at this system. If the bit is already 1, it was set by the missing event check routine in ISGBDR which determined that the RSA is overdue and scheduled entry point ISGBSRME of ISGBSR to report a main ring failure. In this case, ISGBSM (at entry point ISGBSMR) ignores the arrival of the main ring RSA, frees RSVWLOCK, and exits to the dispatcher.		
3 ISGBSM (at entry point ISGBSMR) calls ISGBDR to establish the time interval the RSA is to reside at this system. When the interval expires, entry point ISGBDRM of ISGBDR receives control and schedules ISGBSM (at entry point ISGBSMSR) to send the RSA.	ISGBDR	

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-4. Receive the RSA (Part 3 of 10)



"Restricted Materials of IBM"
Licensed Materials - Property of IBM

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-4. Receive the RSA (Part 4 of 10)

Extended Description	Module	Label
4 Flag RSVFSUB3 is on if the system executing ISGBSMR has just left the main ring. This occurs when some other system has executed a SUBSYS function to remove this system from the main ring. The issuer of the SUBSYS function may have requested that this system write a message to its operator; field RSVMENTY indicates this fact. If a message must be issued, ISGBSMR obtains an MRB, puts message ISG013I into it, and places it on the command router queue. ISGBSMR sets flag RSVFSUB5 and clears field RSVEFMNR to show that this system is no longer in the main ring. It then posts the command router task and ring processing exception handling task (in module ISGBTC) to pass on any messages and perform any needed cleanup. ISGBSMR then frees RSVWLOCK and exits to the dispatcher.		
5 The RSA command area, if present, follows the RSA header. Flag RSAFURC in the header is on if the command area is present and field RSALNCA gives the length of the command area. Field RSASYS gives the SYSID of the system that placed the command area in the RSA.		
A command can be initiated if the received RSA contains no command area and field RSVCRSAT is greater than zero; RSVCRSAT is the command type and is used to choose a command initiator routine. Command initiation routines are subroutines in ISGBSM named CMDIxxxx, where xxxx is a four-letter abbreviation of the command type. ISGBSM changes RSVCRSAT to a negative number to show that the command is in progress and updates the RSA header in the output buffer to show the command area is present. It also sets RSASYSCP in the header to show that the first command phase is in progress and RSVCACKR to point at the proper command continuation routine for the command.	ISGBSM	CMDIADDS CMDIBRCV CMDIBSEN CMDISENC
A command is continued if the received RSA contains a command area previously built by this system. The continuation routine can terminate the command (by removing the command area from the output buffer and changing RSVCRSAT to zero), advance to the next command phase of the command (by increasing phase number RSVCPHNO and field RSASYSCP in the output buffer, and modifying the command area in the output buffer), or repeat the current command phase (by leaving RSVCPHNO and RSASYSCP unchanged and placing the same command area that was sent into the output buffer). Command continuation routines are subroutines named CMDAxxxx where xxxx is a four letter abbreviation of the command type.		CMDAADDS CMDABRCV CMDASENC
A command phase is received if the input buffer contains a command area built by some other system The command area is copied from the input buffer to the output buffer and then a command receive routine is called to inspect or modify the output buffer command area. Command receive routines are subroutines named CMDRxxxx where xxxx is an abbreviation of the command type.		CMDRADDS CMDRBRCV CMDRNONE CMDRSENC

LY28-1695-0 (c) Copyright IBM Corp. 1987

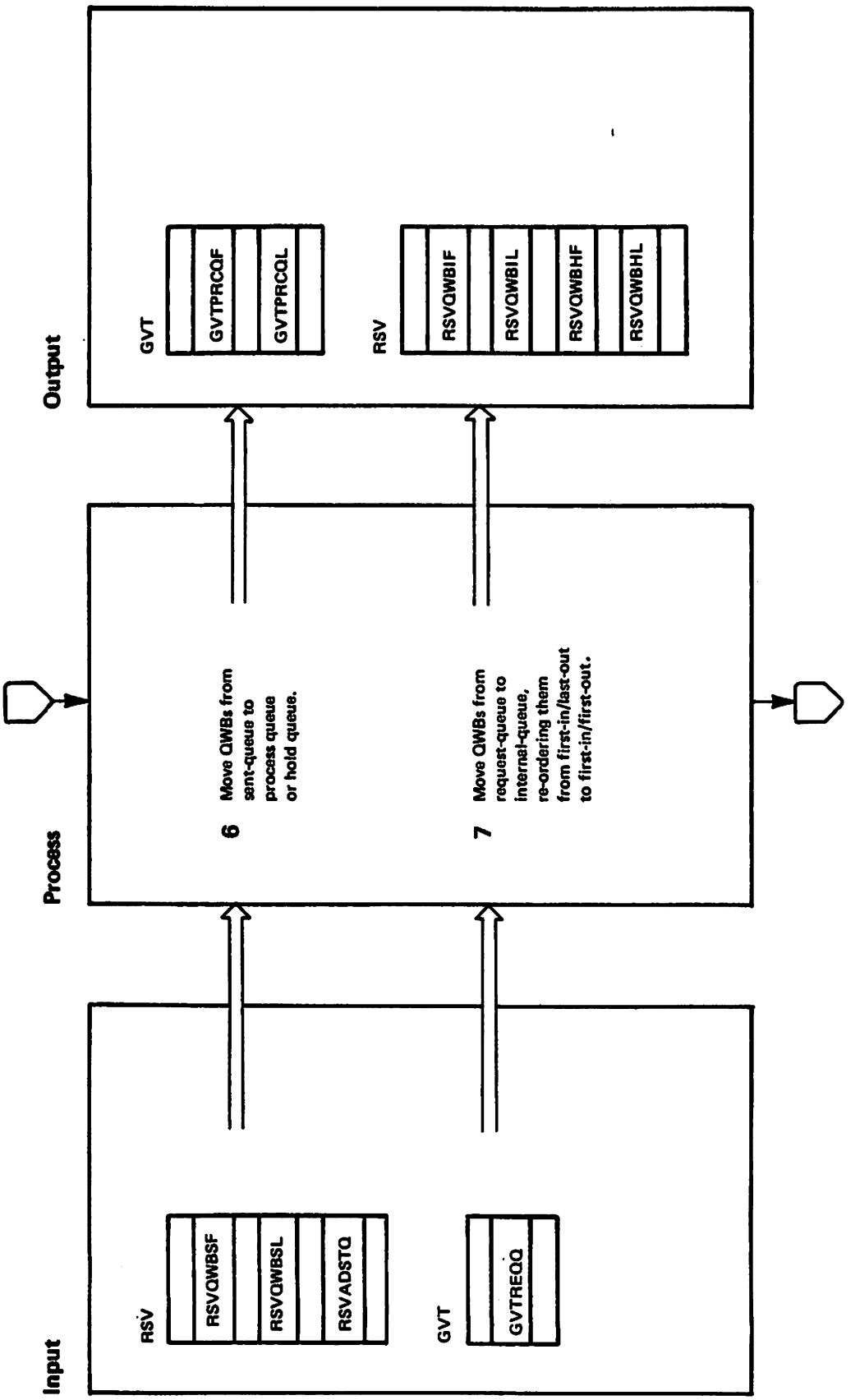


Diagram GRS-4. Receive the RSA (Part 5 of 10)

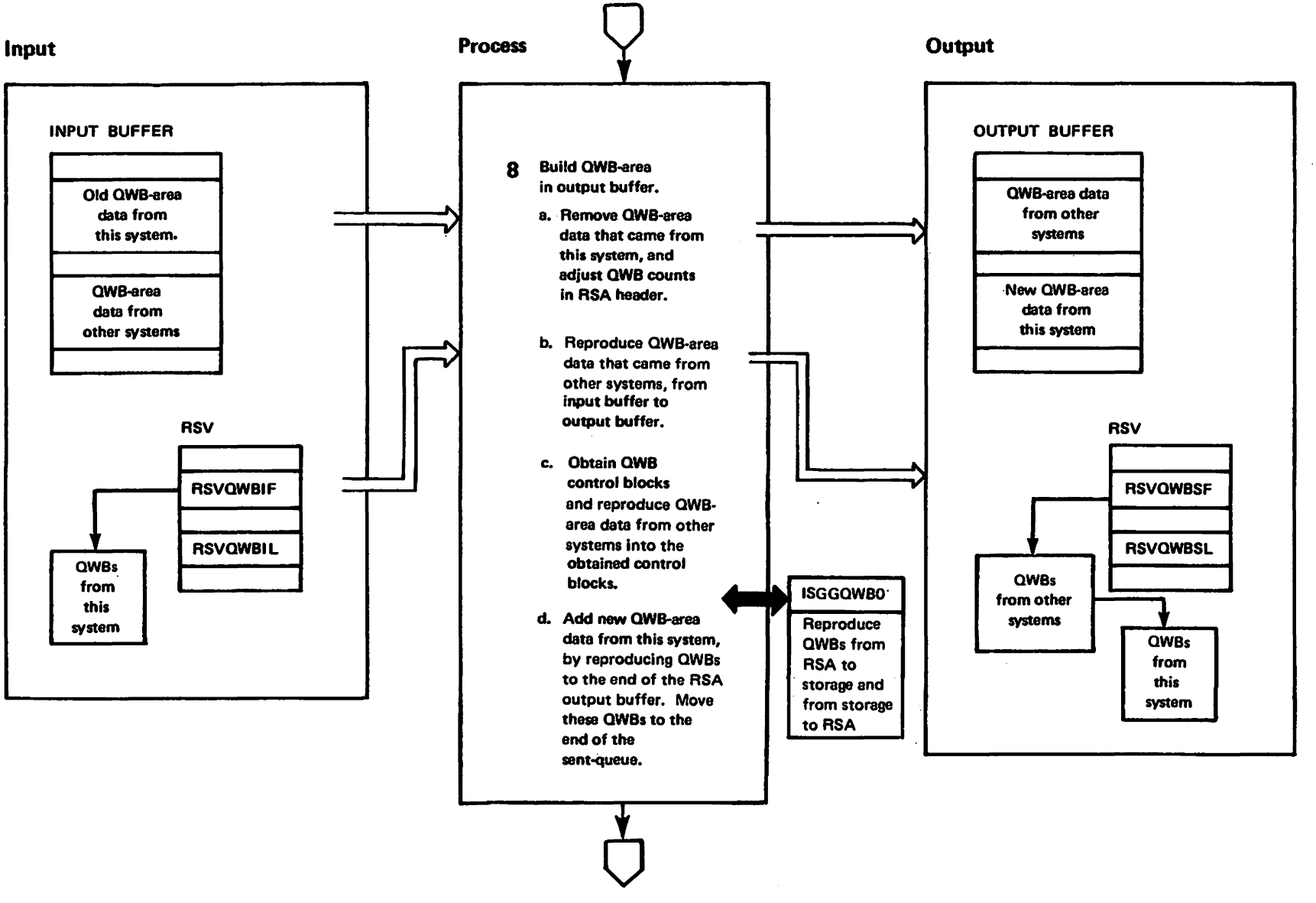
LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-4. Receive the RSA (Part 6 of 10)

Extended Description	Module	Label
6 The sent queue contains QWBs that were in the RSA when it was last sent. These QWBs have now been seen by all systems in the main ring (since the RSA has made a full circuit of the main ring), and can be placed on the process queue (anchored by fields GVTPRCQF and GVTPRCQL) or, if this system is in save QWB mode, the hold queue (anchored by fields RSVQWBHF and RSVQWBHL).		
7 The request queue (anchored by field GVTREQQ) is compare-and-swap serialized and is organized first-in-last-out. The internal queue (anchored by fields RSVQWBIF and RSVQWBIL) is serialized by RSVWLOCK and is first-in-first-out.		

Diagram GRS-4. Receive the RSA (Part 7 of 10)



 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-4. Receive the RSA (Part 8 of 10)

Extended Description	Module	Label
8 The QWB-area contains reproductions of QWB control blocks from systems in the main ring. (Entry point ISGGQWB1 of object-module ISGGQWB0 removes the QWB-area data from the RSA to the system.) (QWBs represent ENQ, DEQ, RESERVE requests.) Older QWB reproductions are at the front of the QWB-area, newer ones are at the rear.		
a. The removed data contains QWBs from this system that have been seen by all systems in the main ring. RSVBXQC has the amount by which the RSA QWB-count (field RSAQWBCT in the RSA header) is to be reduced.		
b. The reproduced data consists of copies of QWBs from other systems; these QWBs have not made a complete circuit around the ring, and have not been seen by all systems in the main ring.		
c. Entry point ISGGQWB1 obtains QWB control-blocks and reproduces QWBs by copying or (optionally) uncompressing and copying QWB-area data from the RSA to the obtained control-blocks. All complete requests are placed on the sent-queue (anchored by RSVQWBSF and RSVQWBSL). If the last request in the RSA is incomplete, it is left anchored in the parameter-list for ISGGQWB1; the incomplete request will be extended or completed when ISGGQWB1 is called after the RSA returns.		
d. QWBs from the internal-queue of this system are copied or (optionally) compressed and copied into the RSA via Ring Processing invoking entry point ISGGQWB0. If the entire request fits in the RSA, then the QWBs making up that request are moved to the sent queue. If the request does not fit in the RSA, it is left at the head of the internal queue so that subsequent QWBs of the request are sent when the RSA returns.	ISGGQWB0	ISGGQWB1

LY28-1695-0 (c) Copyright IBM Corp. 1987

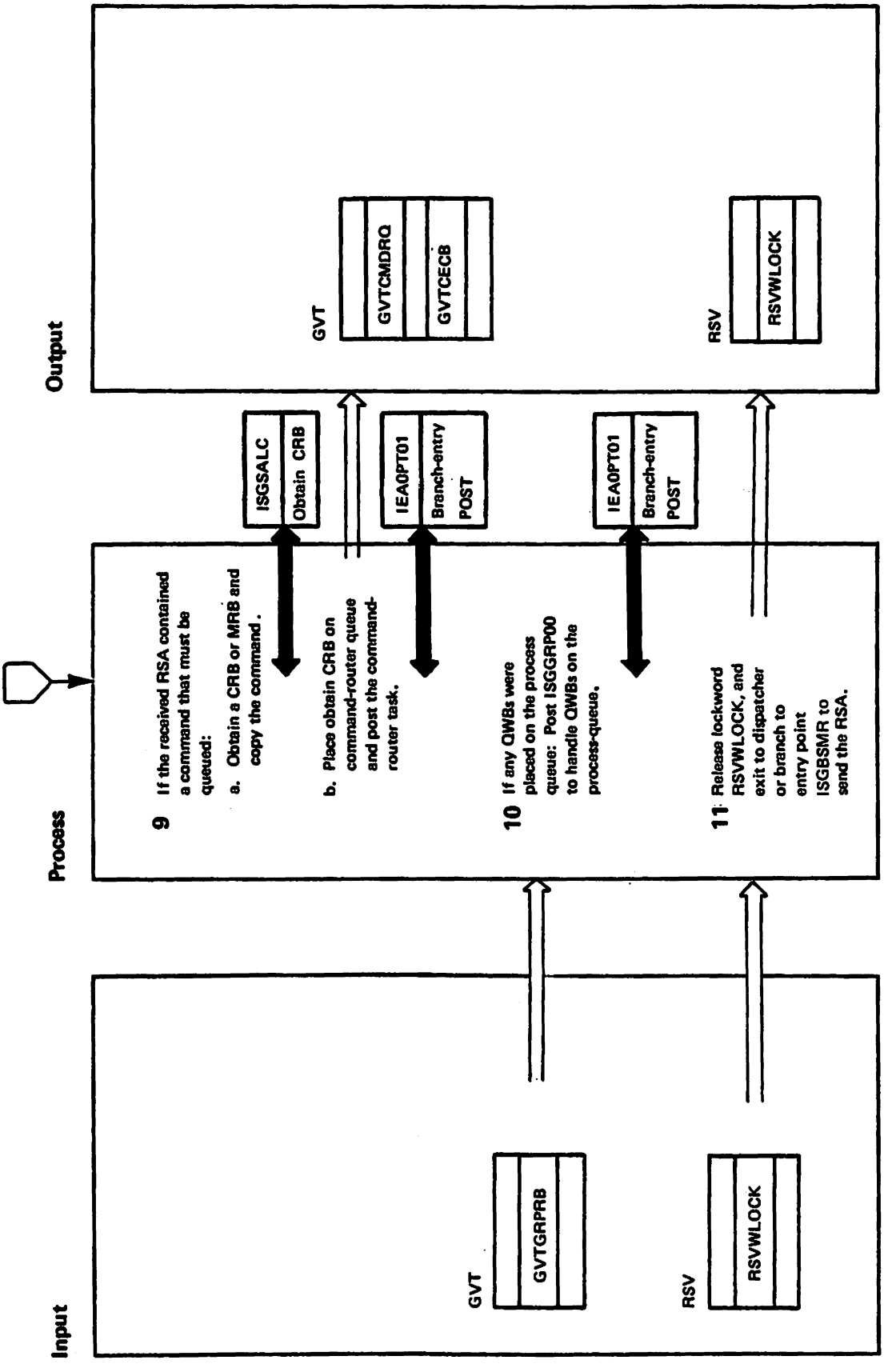


Diagram GRS-4. Receive the RSA (Part 9 of 10)

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-4. Receive the RSA (Part 10 of 10)

Extended Description	Module	Label
9 The received RSA contained a command that must be queued if some other system did a SENDCMD via the main ring to this system, or broadcast a command to all main ring systems.		
a. The RSA contains a copy of the CRB or MRB to be placed on the command-router queue.		
b. Branch-entry post entry-point 1 (pointed at by field CVTOPT01) is used to post the command-router task (via ECB GVTCECB) after the CRB has been placed on the command-router queue (anchored by field GVTCMDRQ).	IEAVSY50	IEA0PT01
10 If any QWBs are on the process-queue, they must be processed by object-module ISGGRP00. This object-module is activated by using the RB-post option of branch-entry post. The RB used for ISGGRP00 is pointed at by GVTGRPRB.		
11 Set lockword RSVWLOCK to its available state. Branch to entry point ISGBSMR if the lockword was altered by ISGBSMR while entry-point ISGBSMR was processing. Exit to dispatcher if the lockword has not been altered since ISGBSMR set it in step 1.		

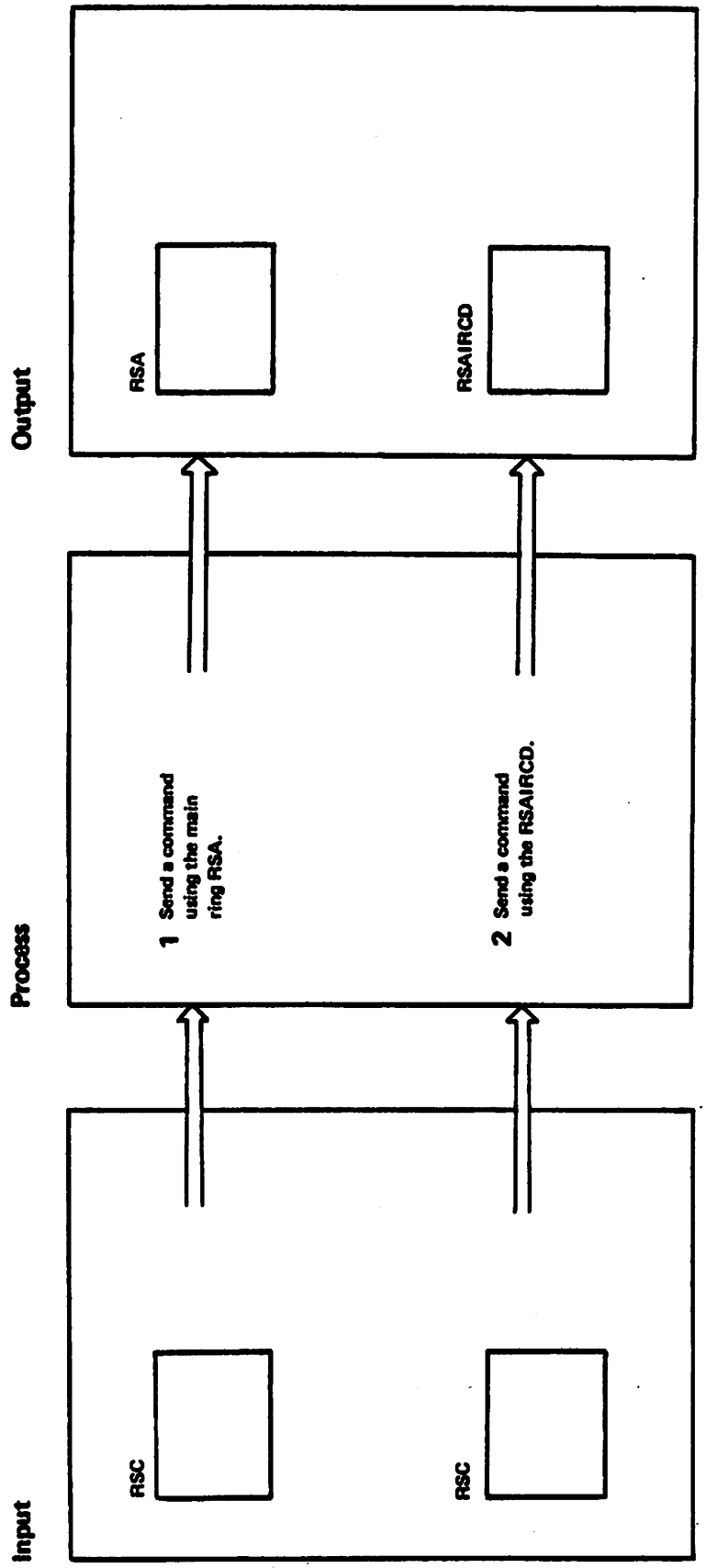


Diagram GRS-5. Send a Command to Another System (Part 1 of 2)

LY28-1695-0 (c) Copyright IBM Corp. 1987

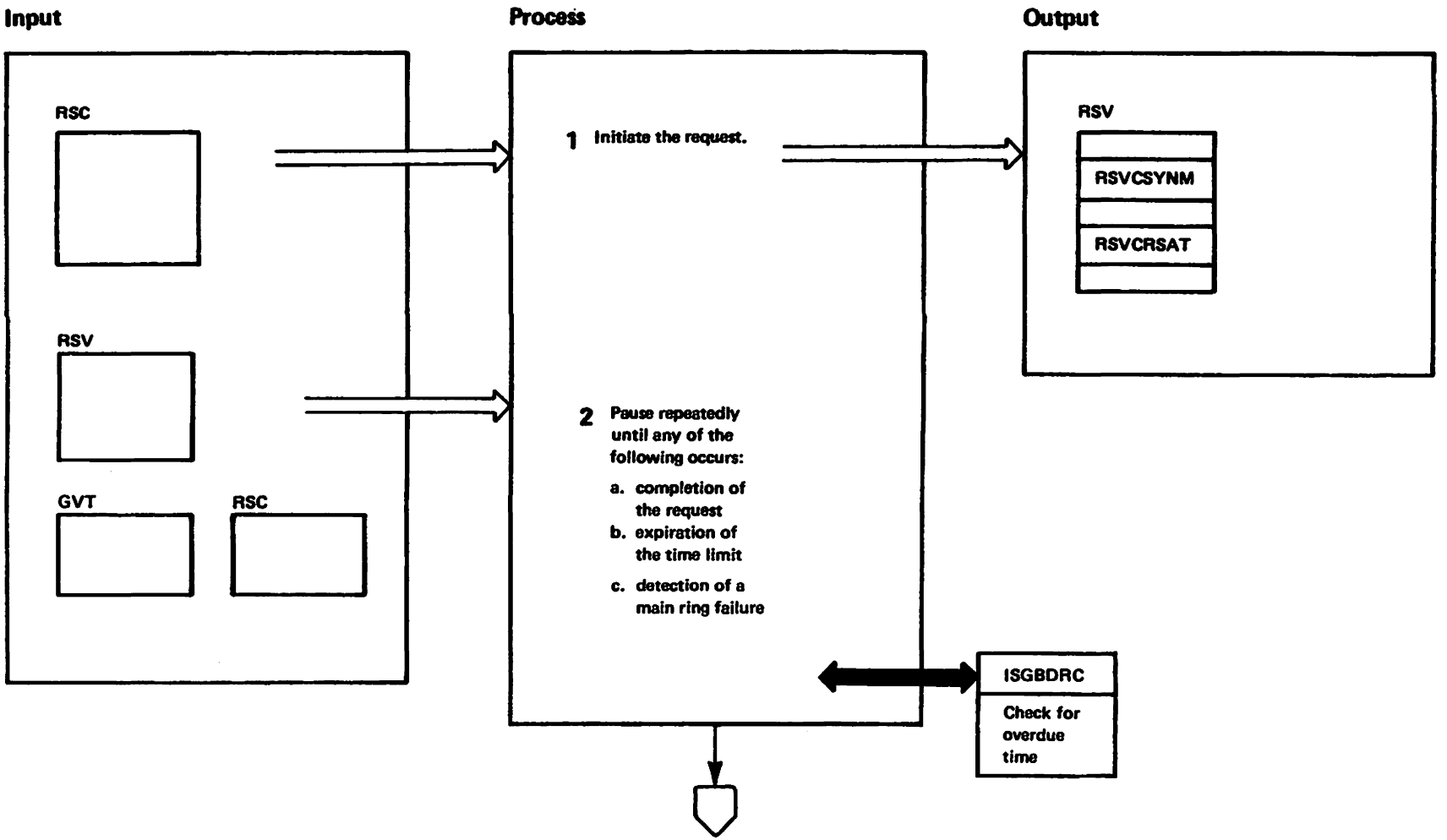
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-5. Send a Command to Another System (Part 2 of 2)

Extended Description	Module	Label
1 ISGBCI sends a command using the main ring RSA if the command sender and the command target are both in the main ring. The caller indicates this by setting bits RSCFLMRS or RSCFLBRD in the RSC that was passed to ISGBCI.	ISGBCI	MAINSEND
2 ISGBCI invokes ISGBRF (at entry point ISGBRFNM) to send a command using an RSAIRCD if either the command sender or the command target is not in the main ring. The caller indicates this by clearing bits RSCFLMRS and RSCFLBRD in the RSC that was passed to ISGBCI.	ISGBRF	ISGBRFNM

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-6. Send a Command Using the Main Ring RSA (Part 1 of 4)



"Restricted Materials of IBM"
Licensed Materials - Property of IBM

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-6. Send a Command Using the Main Ring RSA (Part 2 of 4)

Extended Description	Module	Label
1 ISGBCI puts the name of the system into the RSV if one was specified the caller of ISGBCI. If broadcast is requested, ISGBCI sets the SYSNAME to HEX zeroes so that the command will be sent to all systems in the main ring.	ISGBCI	MAINSEND
ISGBCI passes the request to ISGBSM. This is done by copying input parameters into the RSV and then changing RSVCRSAT from zero to a positive number.		
Asynchronous Processing		
Note: The processing in steps 2 and 3 occurs asynchronously.		
2 ISGBCI does a STIMER SVC to pause, then it checks exit conditions and either exits or pauses again. Each pause is approximately equal to the time needed to send the RSA around the main ring.		
a. ISGBSM sets RSVCRSAT to zero when it asynchronously completes the request.		
b. The time limit is exceeded when the sum of all pauses exceeds RSCTMLIM. ISGBCI cancels the request by changing RSVCRSAT from a positive number to zero.		
c. ISGBCI invokes ISGBSF (at entry point ISGBSFMF) to indicate a main ring failure when the main ring RSA failed to arrive in time.	ISGBSF	ISGBSFMF

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

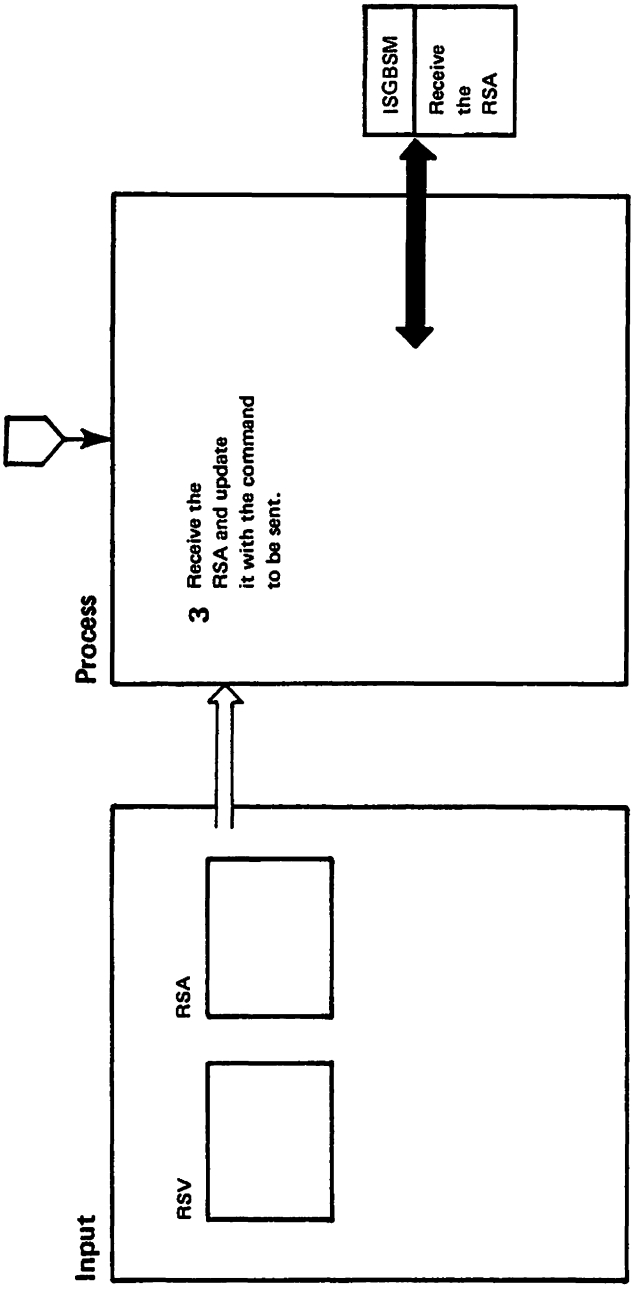


Diagram GRS-6. Send a Command Using the Main Ring RSA (Part 3 of 4)

LY28-1695-0 (c) Copyright IBM Corp. 1987

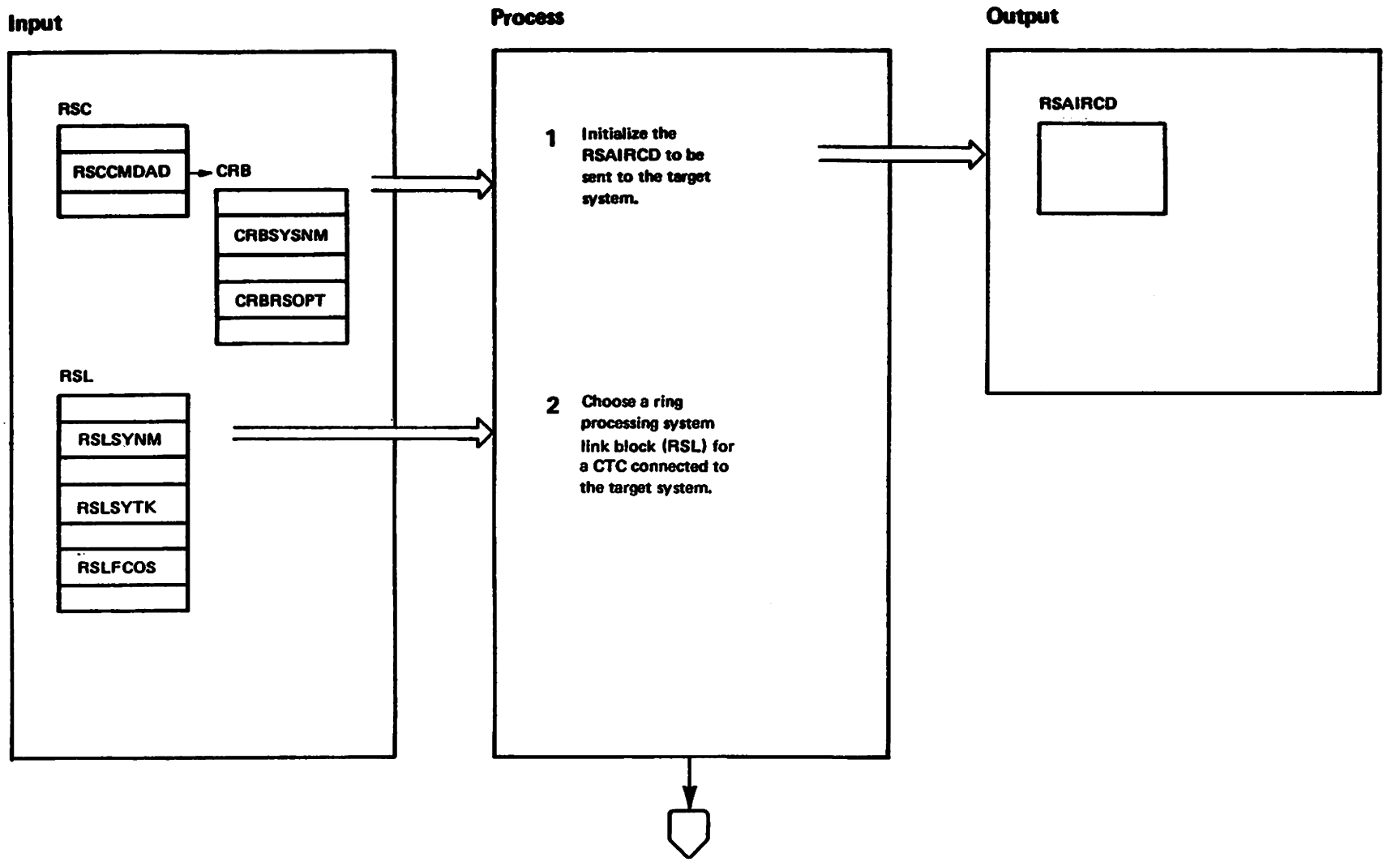
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-6. Send a Command Using the Main Ring RSA (Part 4 of 4)

Extended Description	Module	Label
3 ISGBSR executes as an SRB that is scheduled by the CTC driver whenever the RSA is received. ISGBSR places a command in the RSA and updates the RSA header to show that a command is present. ISGBSR subsequently sends the RSA. ISGBSR reports the command as complete (by setting RSVRSAT to zero) when the RSA returns after making a full circuit of the main ring.	ISGBSR	CMDISENC
Once ISGBCI exits from the loop or pauses described in step 2, it passes the return code to the caller.	ISGBCI	

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-7. Send a Command Using the RSAIRCD (Part 1 of 4)



"Restricted Materials of IBM"
Licensed Materials - Property of IBM

 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

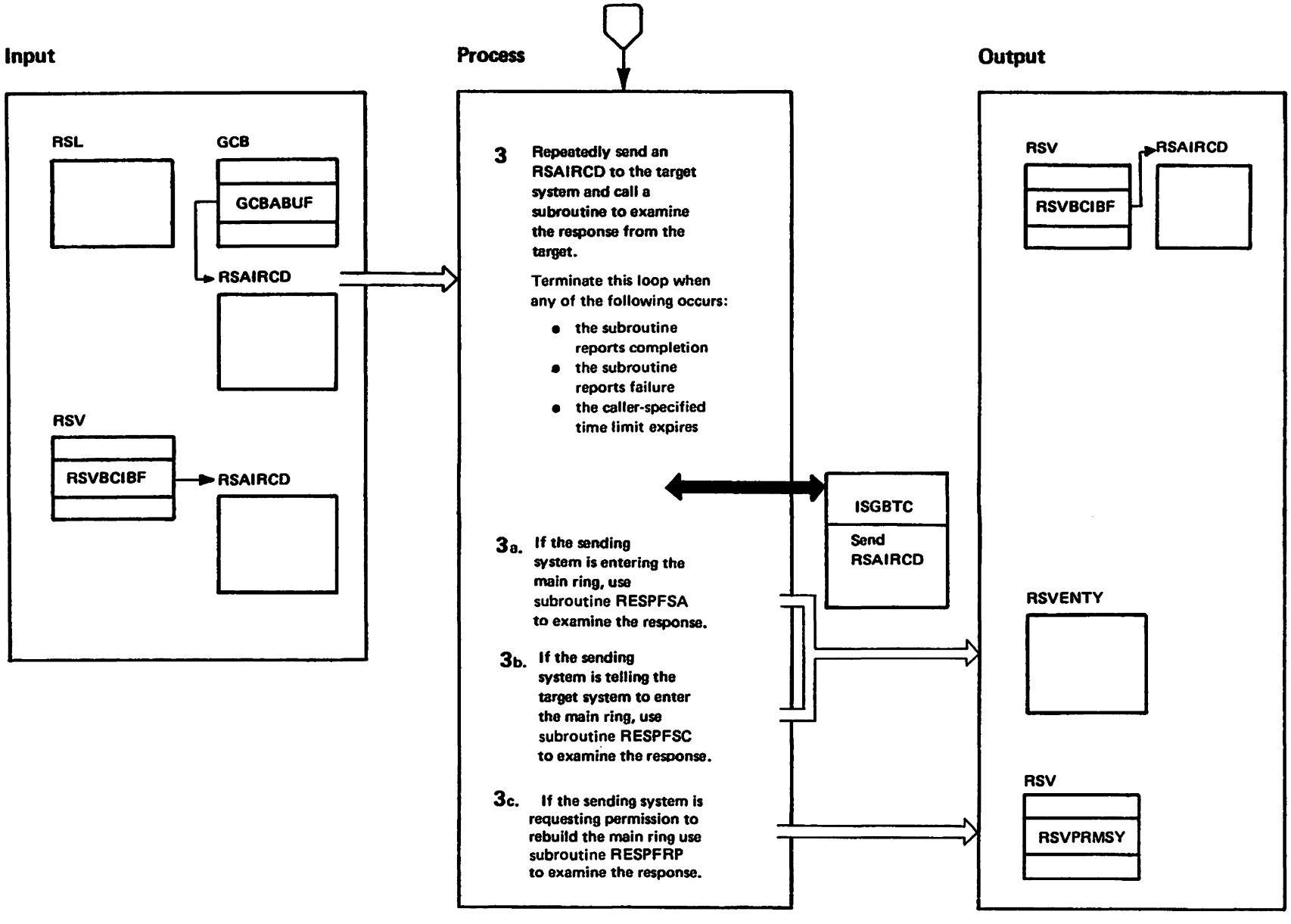
Diagram GRS-7. Send a Command Using the RSAIRCD (Part 2 of 4)

Extended Description	Module	Label
The entry point ISGBRFNM (in ISGBRF) is invoked by ISGBCI to be entered in any of three situations:		
a. The system sending the command is outside the main ring and is trying to enter the main ring. In this case, field RSCSCSFN of the 'send-command' RSC has value RSCRADDS=4. The target system (system receiving the command) must be in the main ring and it must issue the ADDSYS. The ADDSYS (on the target system) and the SENDCMD (on the sending system) complete successfully if the sending system enters the main ring.		
b. The system sending the command is in the main ring and is sending the command to a target that is outside the main ring. In this case, field RSCSCSFN of the 'send-command' RSC has value RSCRSNAD=12. The target system must issue the SENDCMD-RSCRADDS. The sending system will then complete its SENDCMD-RSCRSNAD and issue ADDSYS. ADDSYS and SENDCMD-RSCRADDS complete successfully if the target system enters the main ring.		
c. The system sending the RSAIRCD is requesting permission to rebuild the main ring. The target system denies permission to rebuild the main ring if it knows that some other system is already rebuilding the main ring; otherwise, the target system grants permission. The target system updates the RSAIRCD to indicate whether it granted or denied permission to rebuild the main ring and then sends the RSAIRCD back to the requesting system.		

Extended Description	Module	Label
1 The ISGBCI RSAIRCD buffer via entry point ISGBRFNM (pointed to by RSVBCIBF) is initialized with the identity and status of the sending system and with the system name and command options (RSACSYNM and RSACRSOP) of the command that was passed to ISGBCI.	ISGBRF	ISGBRFNM
2 The most-preferred RSL is the RSL most recently used by the target system to send a command to this system. If no such RSL exists, ISGBCI chooses any eligible RSL. An RSL is eligible if it goes to the target system, is not offline because of a previous hardware/software error, and is not used to send or receive the main ring RSA. If ISGBRFNM is requesting permission, entry point ISGBRFSP (in ISGBRF) and subroutine NMGETRSL choose the RSL.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-7. Send a Command Using the RSAIRCD (Part 3 of 4)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-7. Send a Command Using the RSAIRCD (Part 4 of 4)

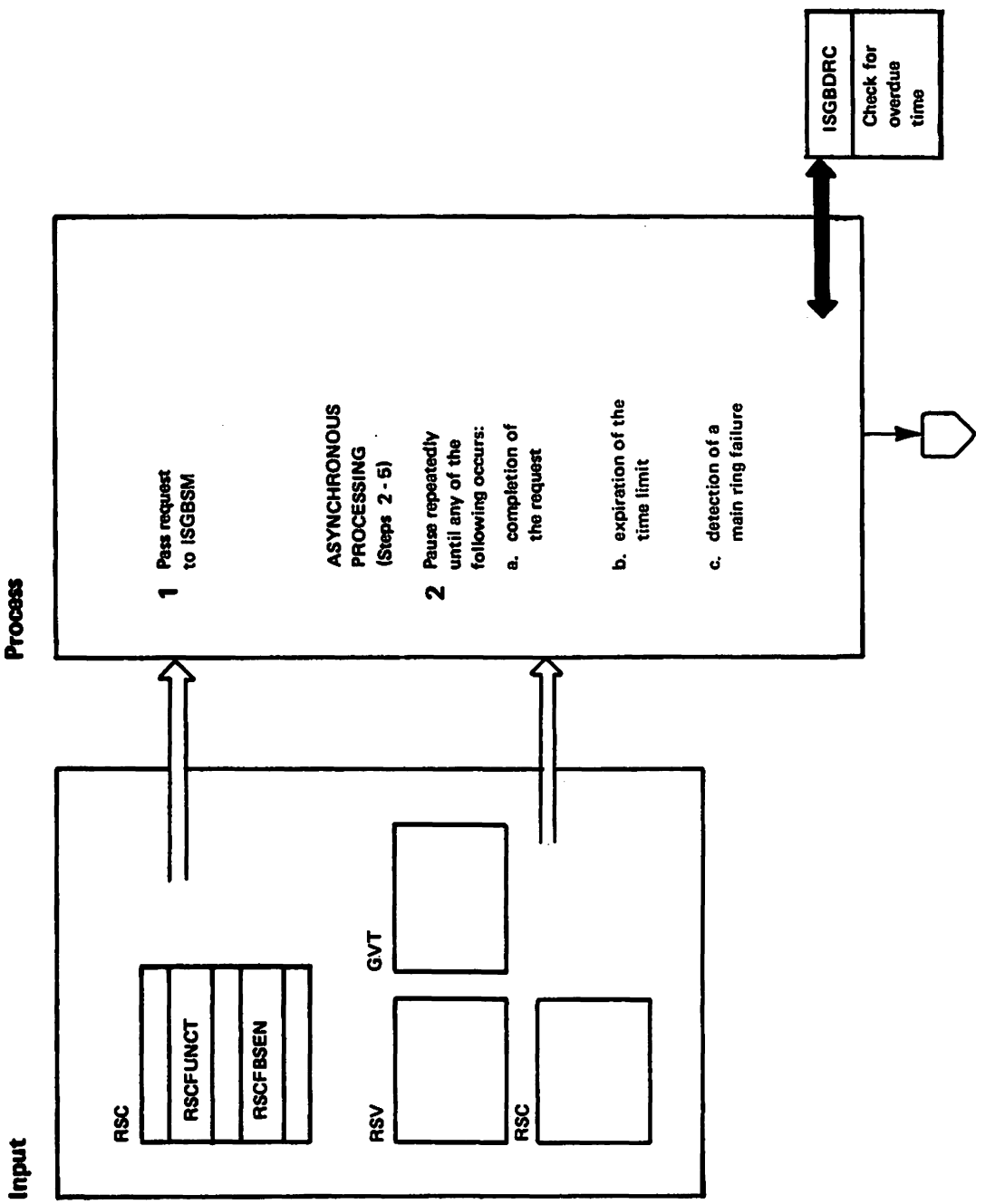
Extended Description	Module	Label
3 The RSAIRCD that is to be sent is placed in the RSAIRCD buffer pointed to by RSVBCIBF. The response RSAIRCD is read into the buffer owned by the RSL being used. The entry point (ISGBCFNM) that is called can modify the RSAIRCD that is being sent, during the subsequent loop, and can also detect a failure in the target system. The RSAIRCD is sent and the response is received asynchronously. Entry point ISGBTCIR of module ISGBTC is called to initiate this process. ISGBCI loops and pauses repeatedly (via STIMER-WAIT) until the response arrives. The called entry point can request that another RSAIRCD be sent or that the loop be terminated.	ISGBCI	
a. Subroutine RESPISA in ISGBRF repeatedly sends a SENDCMD-RSCRADDS RSAIRCD until the response indicates that the target system has issued an ADDSYS. The REPLSH subroutine then modifies each successive RSAIRCD so that the target system returns an RSVENTY entry in each RSAIRCD. This allows the RESPESA subroutine to update its RSVENTY table to match the RSVENTY table of the main ring systems. The RESPFSA subroutine also compares its saved send count (RSVRSASC) to the value saved by the main ring. This comparison tells subroutine CLNUREJN how to adjust QWB queues to match QWB queues in the main ring. After the RSVENTY table is updated, subroutine RESPFSA prepares to receive the main ring RSA and sends its last RSAIRCD to the target system indicating that it is ready to enter the main ring.	ISGBCI	RESPADDS
b. Subroutine RESPFSC repeatedly sends a SENDCMD-RSCRSNAD RSAIRCD until the response indicates that the target system has issued SENDCMD-RSCRADDS. Subroutine RESPFSC then tells ISGBFNM (entry point in ISGBRF) to stop sending the RSAIRCD and to return to the caller of ISGBCI. The target system continues to repeatedly send a SENDCMD-RSCRADDS RSAIRCD and the caller of ISGBCI in the RESPFSC system subsequently calls ISGBCI for the ADDSYS function.		

Extended Description	Module	Label
c. Subroutine RESPFRP examines the response RSAIRCD to determine whether the target system granted or denied permission and updates the RSVPRMSY field of the RSV to reflect whether permission was granted or denied. **Return codes** The return code of ISGBCI may indicate: a. success b. failure due to hardware/software error in communicating to target system c. failure due to some condition in the target system (e.g. the target of the SENDCMD-RSCRADDS was unable to build a main ring containing the sending system) d. failure due to expiration of the time-limit		

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-8. Send Data to Another System (Part 1 of 4)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-8. Send Data to Another System (Part 2 of 4)

Extended Description	Module	Label
1 ISGBCI places the address and length of the buffer in the RSV and then passes the request to ISGBSM by setting RSVCRSAT to a positive number.	ISGBCI	MAINSEND
Asynchronous Processing		
Step 2 and steps 3, 4, and 5 occur asynchronously.		
2 ISGBCI does an STIMER SVC to pause then it checks exit conditions and either exits or pauses again. Each pause is approximately equal to the time needed to send the RSA around the main ring.		
a. ISGBSM sets RSVCRSAT to zero when it asynchronously completes the request.		
b. The time limit is exceeded when the sum of all pauses exceeds RSCTMLIM. ISGBCI cancels the request by changing RSVCRSAT from a positive number to zero.		
c. ISGBCI invokes entry point ISGBSFMF to indicate a main ring failure when the main ring RSA failed to arrive in time.	ISGBSF	ISGBSFMF

LY28-1695-0 (c) Copyright IBM Corp. 1987

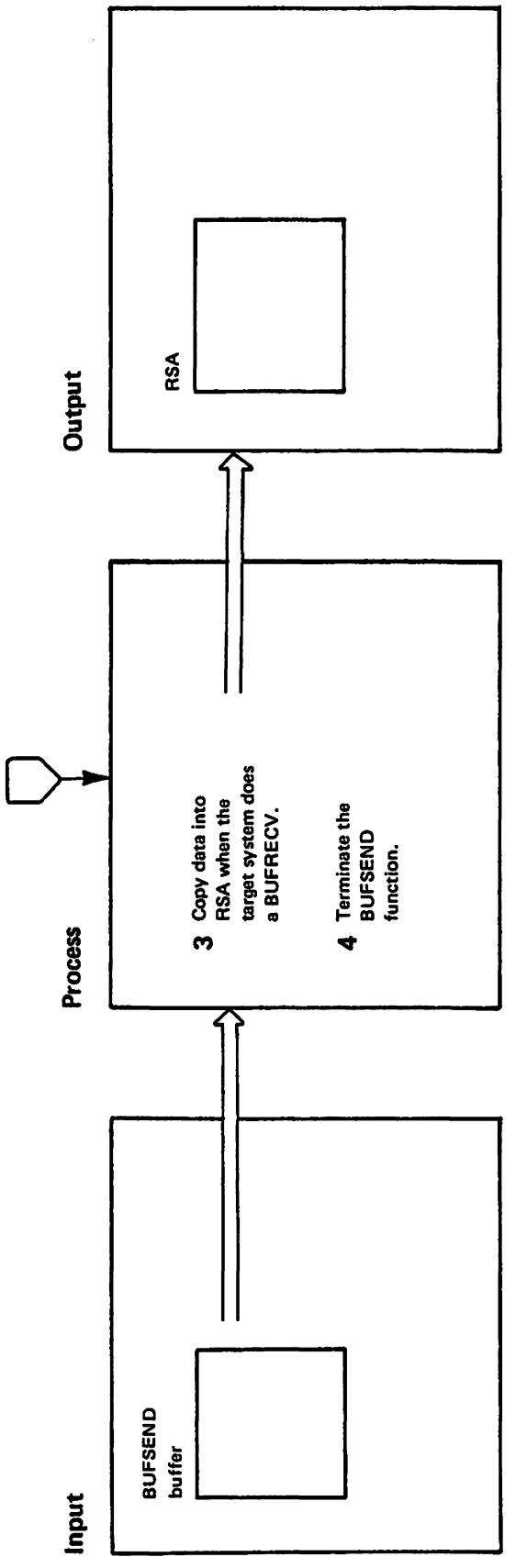


Diagram GRS-8. Send Data to Another System (Part 3 of 4)

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-8. Send Data to Another System (Part 4 of 4)

Extended Description	Module	Label
3 The received RSA may indicate that the target of the BUFSEND function is currently executing a BUFRECV function. If so, ISGBSM updates the RSA to contain data to be sent to the target system. If not, the BUFSEND request remains outstanding until the target performs a BUFRECV, the time limit expires, or the main ring fails.		
4 If the system receiving the data has indicated that all of the data has been received, ISGBSM then terminates the BUFSEND function.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

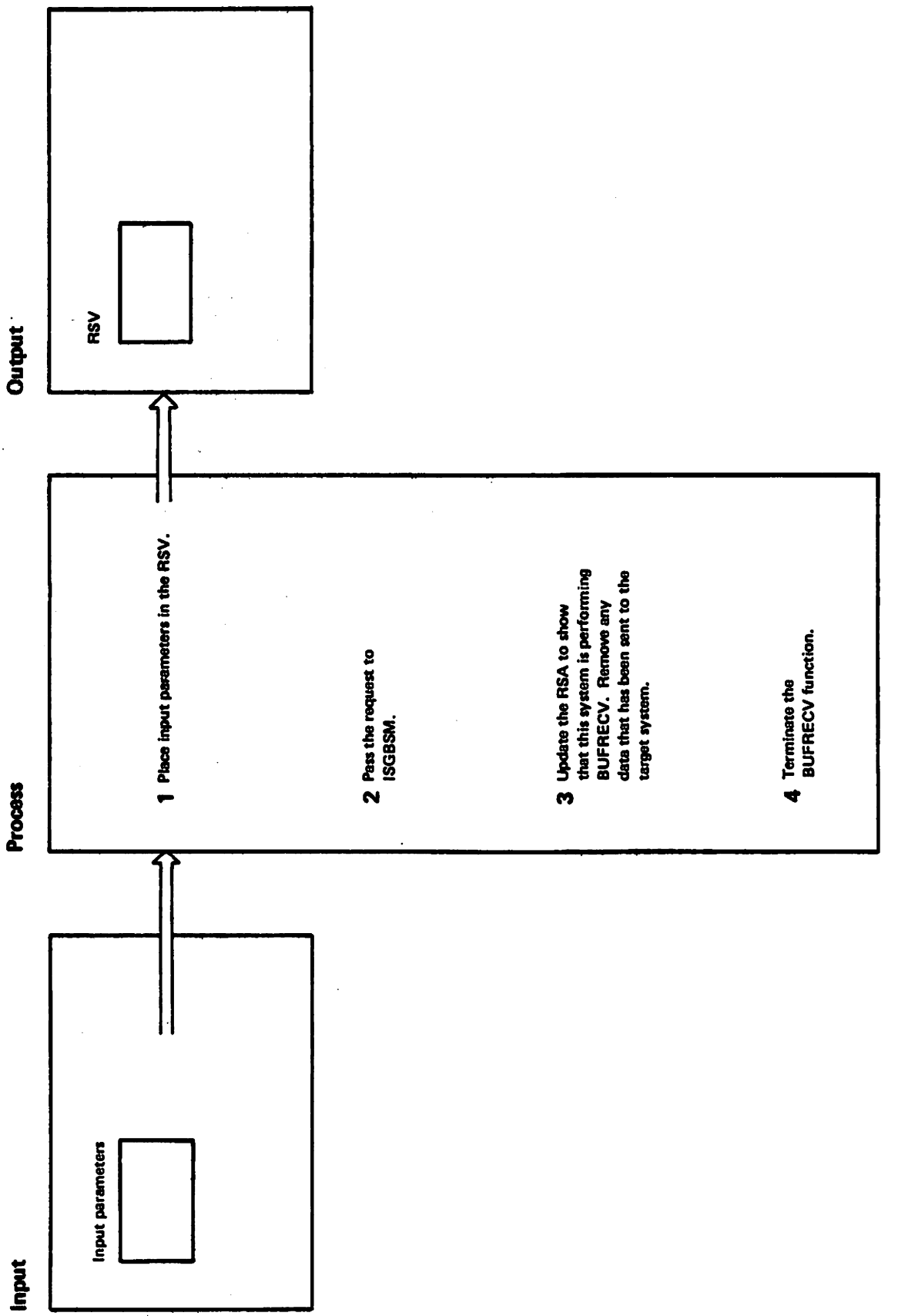


Diagram GRS-9. Receive Data from a System (Part 1 of 2)

LY28-1695-0 (c) Copyright IBM Corp. 1987

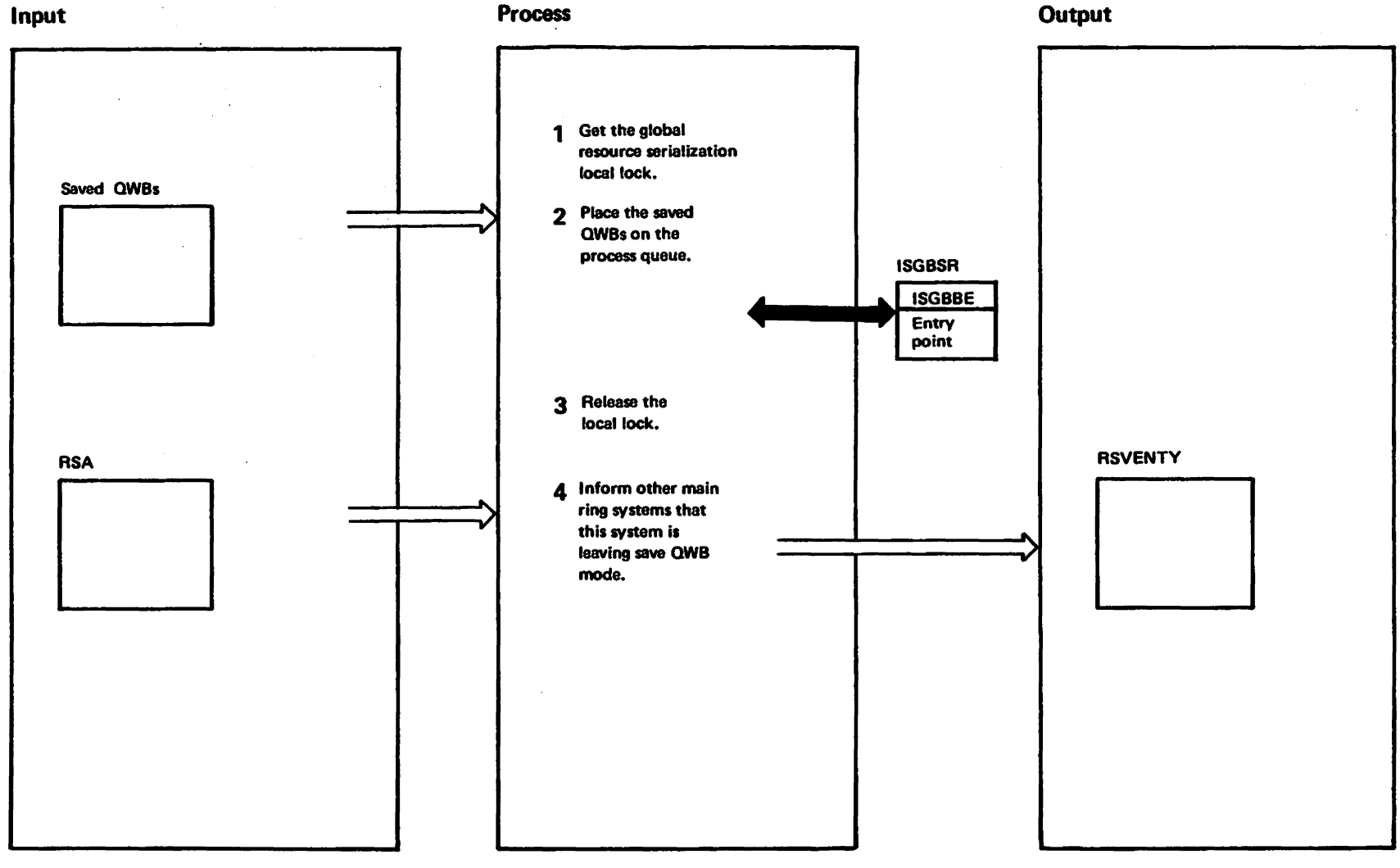
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-9. Receive Data from a System (Part 2 of 2)

Extended Description	Module	Label
1 ISGBCI places, into the RSV, the address and length of a BUFSEND buffer that is to receive data and the SYSNAME of the system that must send the data.	ISGBCI	MAINSEND
2 ISGBCI sets the address and length of the buffer it expects to receive. ISGBCI passes the request to ISGBSM by changing RSVCRSAT to a positive number.		
3 When the RSA is received, ISGBSM updates the RSA with a BUFRECV marker to show that this system is performing a BUFRECV and then sends it around the main ring. When the RSA returns, it contains either the data from the target or an indication that the target has not performed a BUFSEND. If the RSA contains data, the data is removed from the RSA and copied into the BUFSEND buffer. If the RSA contains no data, ISGBSM updates the RSA to remove the BUFRECV marker and sends the RSA. When the RSA returns, the BUFRECV marker is put in the RSA again and the process is repeated. This process is repeated until ISGBCI detects that a time limit has expired and cancels the request by changing RSVCRSAT back to zero.		
4 ISGBSM terminates the BUFRECV function by setting a return code (and placing the length of the received data) in the RSV and changing RSVCRSAT to zero.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-10. Leave Save QWB Mode (Part 1 of 2)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-10. Leave Save QWB Mode (Part 2 of 2)

Extended Description	Module	Label
1 ISGBSF (at entry point ISGBSFDP) obtains the local lock of the global resource seralization address space.	ISGBSF	ISGBSFDP

The ISGBCI SERRELS function is used to cause a system to leave save QWB mode.

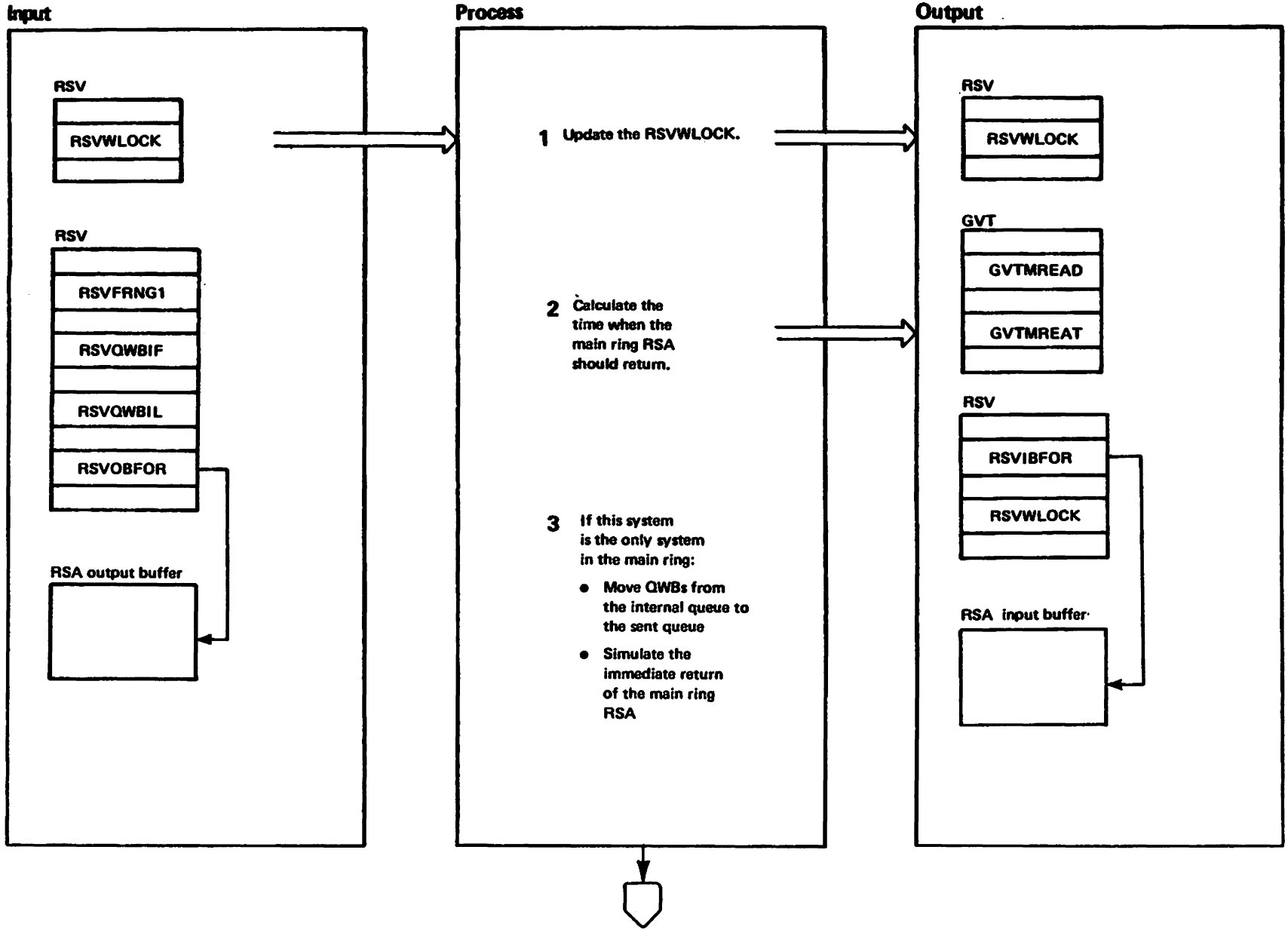
2 ISGBSF (at entry point ISGBSFDP) puts the saved QWBs on the process queue via a call to the ISGBBE entry point of module ISGBSR.

If the process queue is empty, ISGBBE moves the QWB string into the process queue. If the process queue is not empty, add to the QWB string to the end of the process queue.

3 ISGBSF (at entry point ISGBSFDP) takes the system out of 'save-QWB' mode and releases the local lock.

4 The RSA is used to inform all other main ring systems that this system has left save QWB mode. Each system updates its RSVENTY table to reflect this fact.

Diagram GRS-11. Send the RSA (Part 1 of 4)



"Restricted Materials of IBM"
Licensed Materials - Property of IBM

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-11. Send the RSA (Part 2 of 4)

Extended Description	Module	Label
The main ring RSA is sent by entry point ISGBSMSR of module ISGBSM. This module executes in SRB mode, key 0, supervisor state. Recovery for this function is provided by ISGBFRCV.		
ISGBSMSR is scheduled by entry point ISGBDRS of module ISGBDR when a time interval expires. This time interval is called the main ring RSA residence time.		
1 RSVWLOCK is used to serialize between entry points ISGBSMSR and ISGBSMR. If RSVWLOCK is still in use by ISGBSMR, then ISGBSMSR alters RSVWLOCK and exits to the dispatcher. (Entry point ISGBSMR will see the altered value and will branch to ISGBSMSR instead of exiting to the dispatcher, when it has completed its processing.	ISGBSM	ISGBSMSR
If RSVWLOCK is not in use, ISGBSMSR alters it to show that it is now in use.		
2 If this system is no longer in the main ring, ISGBSM frees the resources (including the RSV lockword RSVWLOCK), and exits to the dispatcher.		
Assuming this system is still in the main ring, ISGBSM field GVTMREAD with the number of miliseconds needed for the main ring RSA to make a full circuit of the main ring. ISGBSM also sets field GVTMREAT with the time when the RSA is being sent from this system and clears the low order bit of GVTMREAD to indicate that the main ring RSA is no longer at this system.		
3 Flag RSVFRNG1 is on when this system is the only system in the main ring.		
• ISGBSM moves the QWBs from the internal queue to the sent queue so that the QWBs will be moved from the sent queue to the process queue when ISGBSMR is subsequently entered.		
• ISGBSM simulates the immediate return of the main ring RSA by copying the RSA from the output buffer into the input buffer and altering RSVWLOCK to show that the RSA has been received. This simulates the system sending the RSA to itself and the RSA returning to this system as soon as it is sent.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

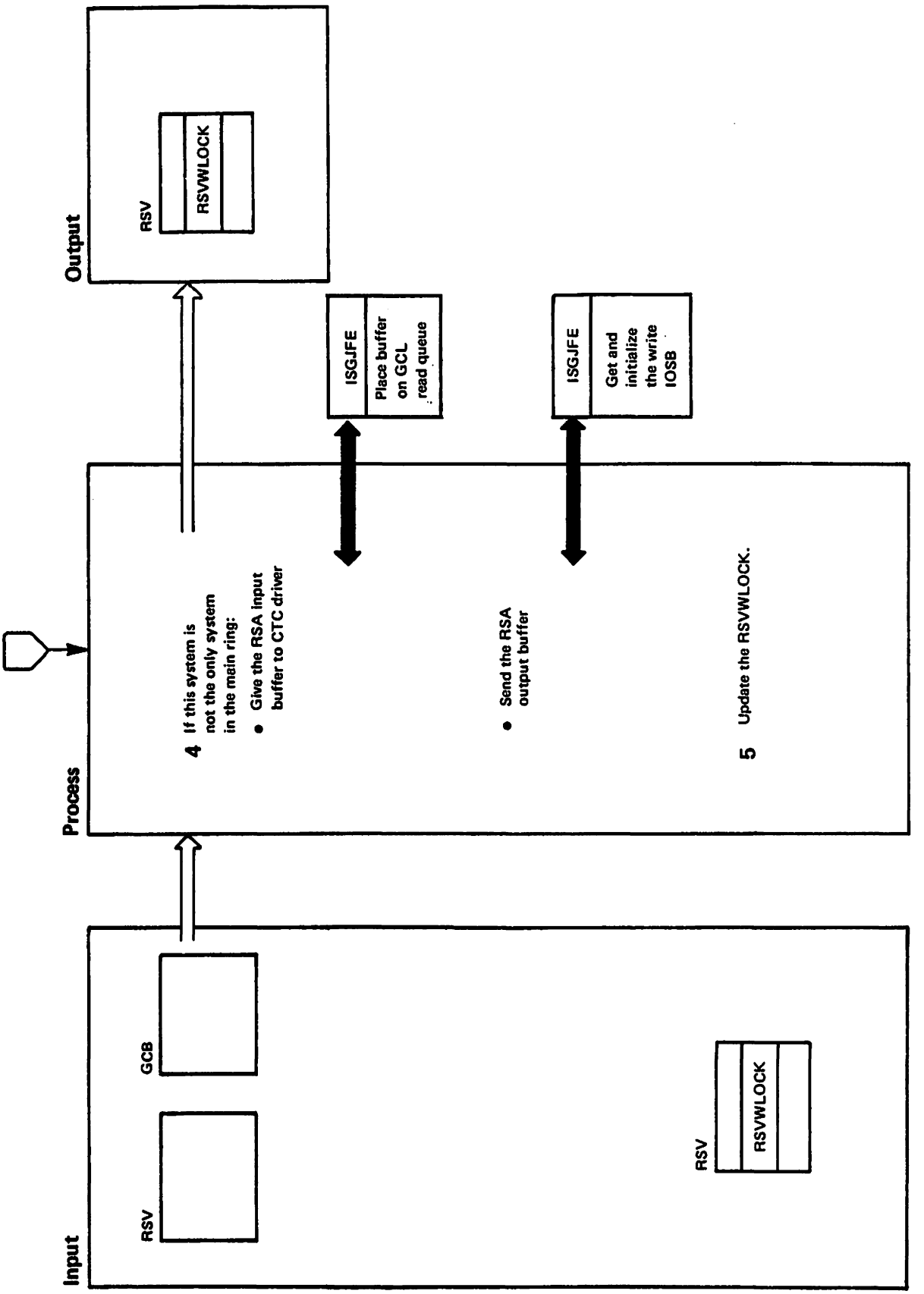


Diagram GRS-11. Send the RSA (Part 3 of 4)

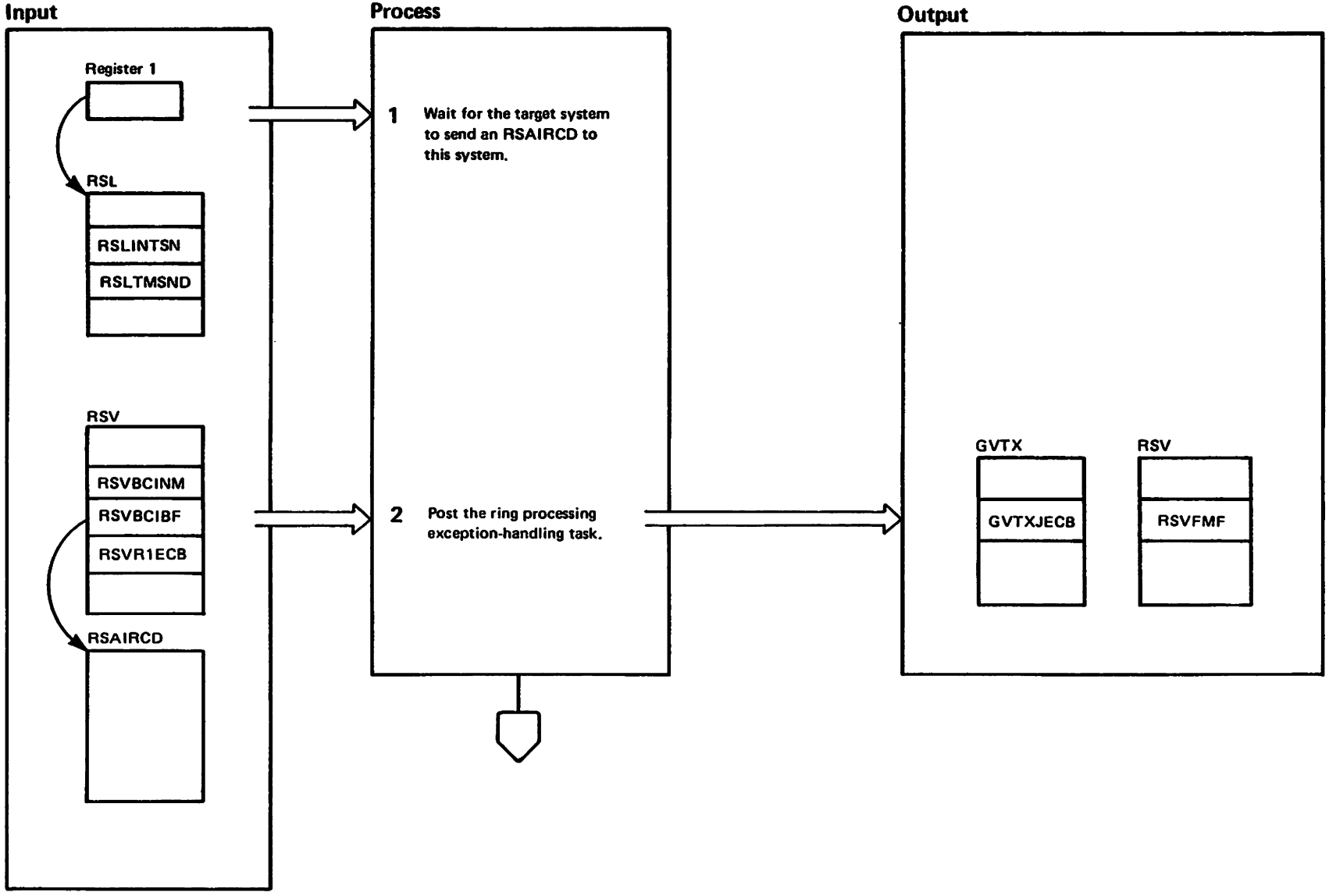
LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-11. Send the RSA (Part 4 of 4)

Extended Description	Module	Label
4 If the main ring contains two or more systems, the RSA must be sent using the CTC driver. a. RSVGCBIP points at the RSA input GCB. The GCB is pre-initialized so the CTC driver will schedule entry point ISGBSMR when the RSA has been read into the RSA input buffer. b. RSVGCBOP points at the RSA output GCB. The GCB is pre-initialized so the CTC driver will send the RSA from the RSA output buffer.		
5 ISGBSM branches to ISGBSMR if RSVWLOCK indicates that ISGBSMR was dispatched while ISGBSMR held RSVWLOCK. (This can occur if the RSA returns before ISGBSMR exits or if this system is the only system in the main ring.) ISGBSMR exits to the dispatcher if RSVWLOCK indicates that ISGBSMR has not been dispatched yet.		

Diagram GRS-12. Send the RSAIRCD (Part 1 of 6)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-12. Send the RSAIRCD (Part 2 of 6)

Extended Description	Module	Label
Entry point ISGBTCIR of ISGBTC is called to send an RSAIRCD. ISGBTC determines whether to schedule entry point ISGBSRRI of ISGBSR, or to do nothing and allow the CTC driver to schedule ISGBSRRI. ISGBTC also performs special processing when it is sending an RSAIRCD in order to enter the main ring.		
1 If the target system is expected to send an RSAIRCD, pause to wait for it. The target system is expected to send an RSAIRCD to this system when all of the following conditions are met: • RSLINTSN is non-zero (that is, the arrival of the RSAIRCD is not overdue) • The current time is not later than the time when the RSAIRCD was sent plus RSLINTSN milliseconds plus GVTOLINT milliseconds. When this system receives the RSAIRCD, ISGBSR (entry point ISGBSRRI) updates the RSAIRCD and sends it back to the target system. ISGBTC does not schedule ISGBSR to send the RSAIRCD. ISGBTC determines whether the RSAIRCD has been sent by checking whether RSLTMSND (the time when the RSAIRCD was sent) has changed since control was passed to ISGBTC.	ISGBTC	ISGBTCIR
2 If this system is sending the last RSAIRCD before entering the main ring, ISGBTC (entry point ISGBTCIR) is called under a task other than the ring processing exception-handling task. The exception-handling task must clear unusual events and set status flags to indicate that the sending system is in the main ring. This is done by setting flag RSVFMF and posting ECB GVTXJECB to awaken the exception-handling task. The exception-handling task clears RSVFMF and posts RSVR1ECB when it has performed the request.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

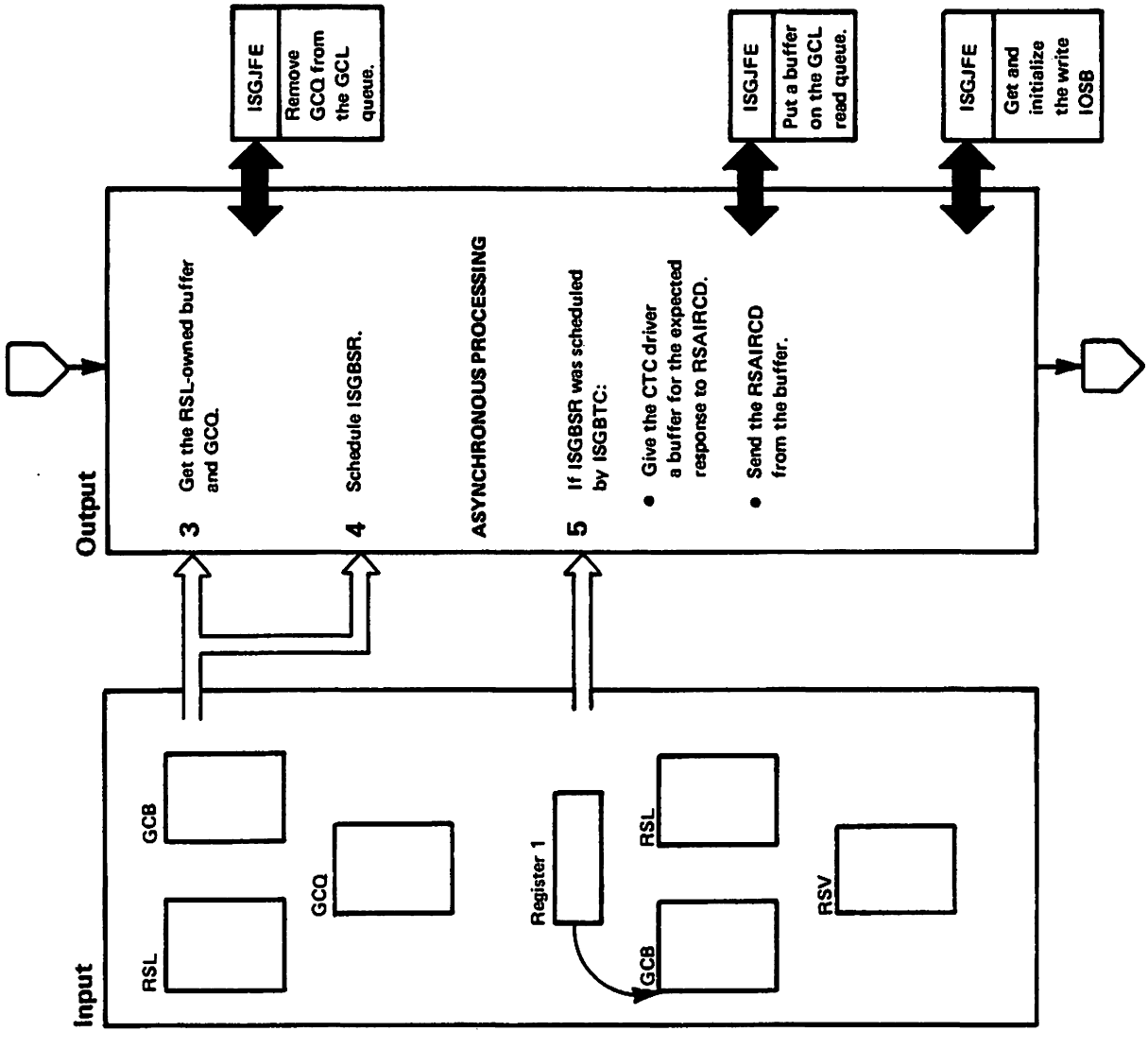


Diagram GRS-12. Send the RSAIRCD (Part 3 of 6)

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-12. Send the RSAIRCD (Part 4 of 6)

Extended Description	Module	Label
3 Each RSL owns a GCQ (pointed to by field GCBAGCQ of the GCB that follows the RSL). Field RSLBFCTC indicates whether this GCQ (and the associated buffers and GCB) have been given to the CTC driver. If the CTC driver has the GCQ, ISGBTC calls ISGJTK to get the GCQ.	ISGJFE	ISGJTKBF
4 Field RSLWLOCK is set to 31 to show that ISGBTC scheduled ISGBSR to send the RSAIRCD. An SRB is built in the RSL-owned GCB and is scheduled to cause ISGBSR to execute asynchronously. Return code 0 indicates ISGBTC has scheduled ISGBSR.		
The following processing occurs asynchronously to the task that called ISGBTC (entry point ISGBTCIR).		
5 ISGBSR at entry point ISGBSRRI is dispatched as an SRB routine. The SRB may have been scheduled by ISGBTC (indicated by an RSLWLOCK value of 31), by the global resource serialization CTC driver when a send completion occurs (indicated by flag RSLFSIP being on when ISGBSR is entered), or when the CTC driver receives a message. Register 1 points to the GCB that immediately follows the RSL used by ISGHSR.	ISGBSR	ISGBSRRI
• The RSAIRCD is being sent for ISGBCI when the RSV field RSVBCIMM points at the RSL used by ISGBSR. In this case, ISGBSR gives the CTC driver the RSL-owned buffer and GCB for future use in reading the response RSAIRCD from the target system.		
• If the send is being done for ISGBCI and it is sending the last RSAIRCD before entering the main ring, ISGBSR gives the CTC driver the main ring RSA input buffer (pointed to by RSVIBFOR) using the main ring input GCB (pointed to by RSVGCBIP) and GCQ.	ISGJFE	ISGJSNBF
If the send is being done for ISGBCI, the RSAIRCD is sent using the ISGBCI-owned GCB. When ISGBCI is sending an immediate CCW, it will have set flag GCBFSNIM in the ISGBCI-owned buffer. RSLFSIP is set before calling the CTC driver. The CTC driver will schedule ISGBSR when the send is complete.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

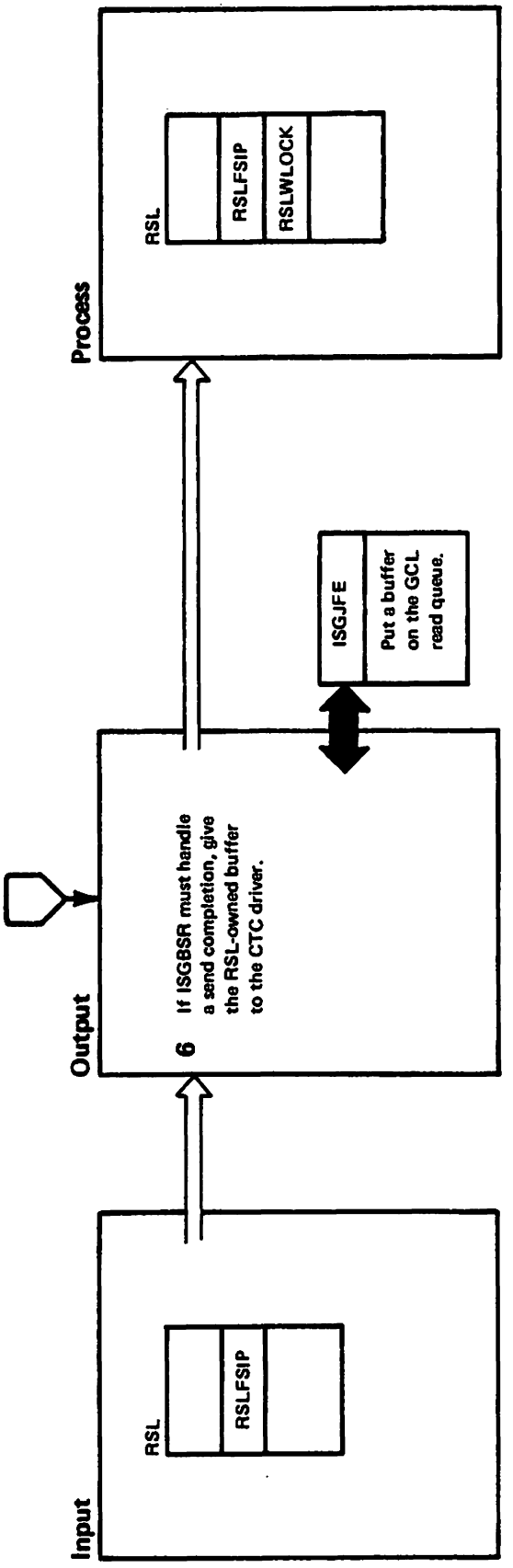


Diagram GRS-12. Send the RSAIRCD (Part 5 of 6)

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-12. Send the RSAIRCD (Part 6 of 6)

Extended Description	Module	Label
6 A send completion occurs if the RSAIRCD was previously sent with the RSL-owned buffer. The RSL-owned buffer is given to the CTC driver so it can be used to read any RSAIRCD sent by the remote system.	ISGJFE	ISGJGVBF

LY28-1695-0 (c) Copyright IBM Corp. 1987

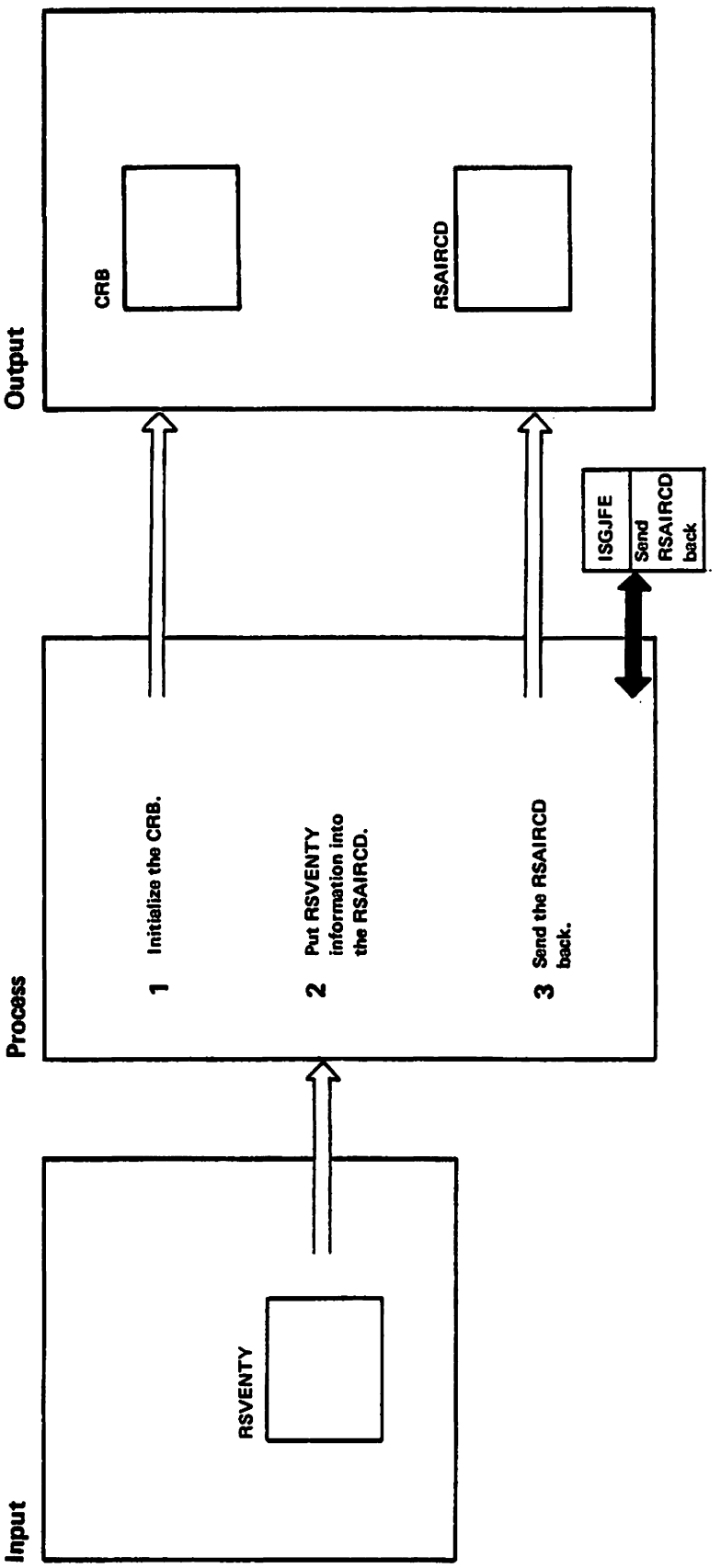


Diagram GRS-13. Receive the RSAIRCD (Part 1 of 2)

LY28-1695-0 (c) Copyright IBM Corp. 1987

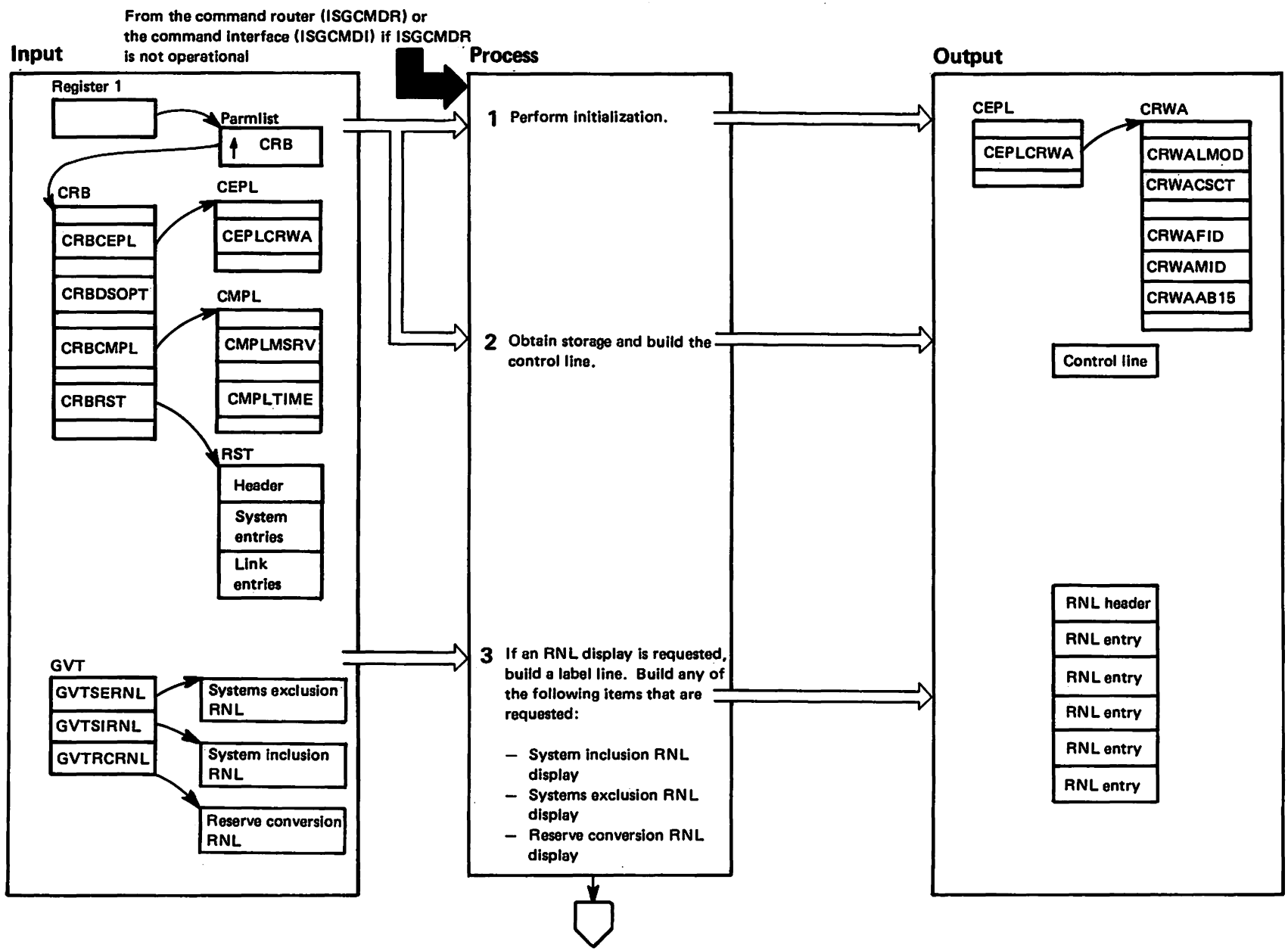
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-13. Receive the RSAIRCD (Part 2 of 2)

Extended Description	Module	Label
1 If the RSAIRCD contains a command (field RSACCMDCB is non-zero), ISGBSR gets a CRB from the GRS storage manager (entry point ISGSALC) and copies data from the RSAIRCD into the CRB. ISGBSR places the CRB on the command router queue and posts the command router.	ISGBSR	
2 If flag RSAIFIDR is on, ISGBSR copies the RSVENTY information into the RSAIRCD. Field RSACTBIX indicates which RSVENTY is to be copied		
3 ISGBSR sends the RSAIRCD using the RSL-owned buffer via a call to ISGJSNBF. Then exits to the dispatcher.	ISGJSNBF	ISGJSNBF

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-14. ISGCDSP – Global Resource Serialization DISPLAY GRS Request Processor (Part 1 of 6)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

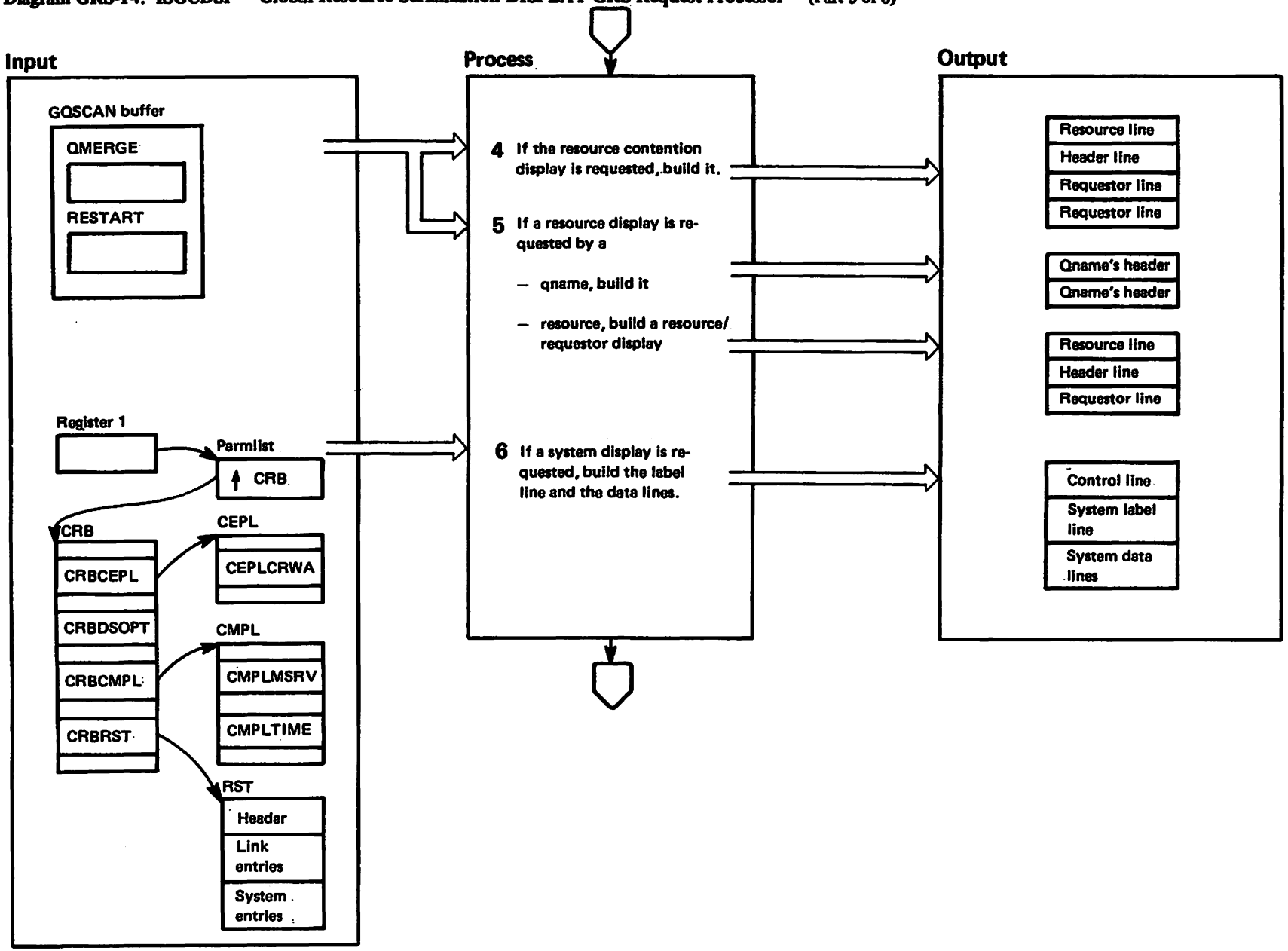
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-14. ISGCDSP – Global Resource Serialization DISPLAY GRS Request Processor (Part 2 of 6)

Extended Description	Module	Label
ISGCDSP processes global resource serialization DISPLAY GRS status requests and produces the ISG020I message. The global resource serialization command router (ISGCMDR) attaches ISGCDSP when it finds a command request block (CRB) for a DISPLAY GRS request on the global resource serialization command request queue. If communication with ISGCMDR is not possible ISGCMDI attaches ISGCDSP.		
1 ISGCDSP initializes a command request workarea (CRWA) with recovery information and places it on the CRWA queue.		
2 ISGCDSP calls IEECB808 at entry point MSGSERV to obtain storage to build a control line containing a time stamp and the message text.	IEECB808	MSGSERV
3 For an RNL or an ALL request, ISGCDSP builds a display for the requested RNLs. ISGCDSP obtains the RNL contents from the SQA and invokes IEECB808 to get storage for a line. The display consists of a label line for each entry in the RNLs that are to be displayed.	IEECB808	MSGSERV

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-14. ISGCDSP – Global Resource Serialization DISPLAY GRS Request Processor (Part 3 of 6)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

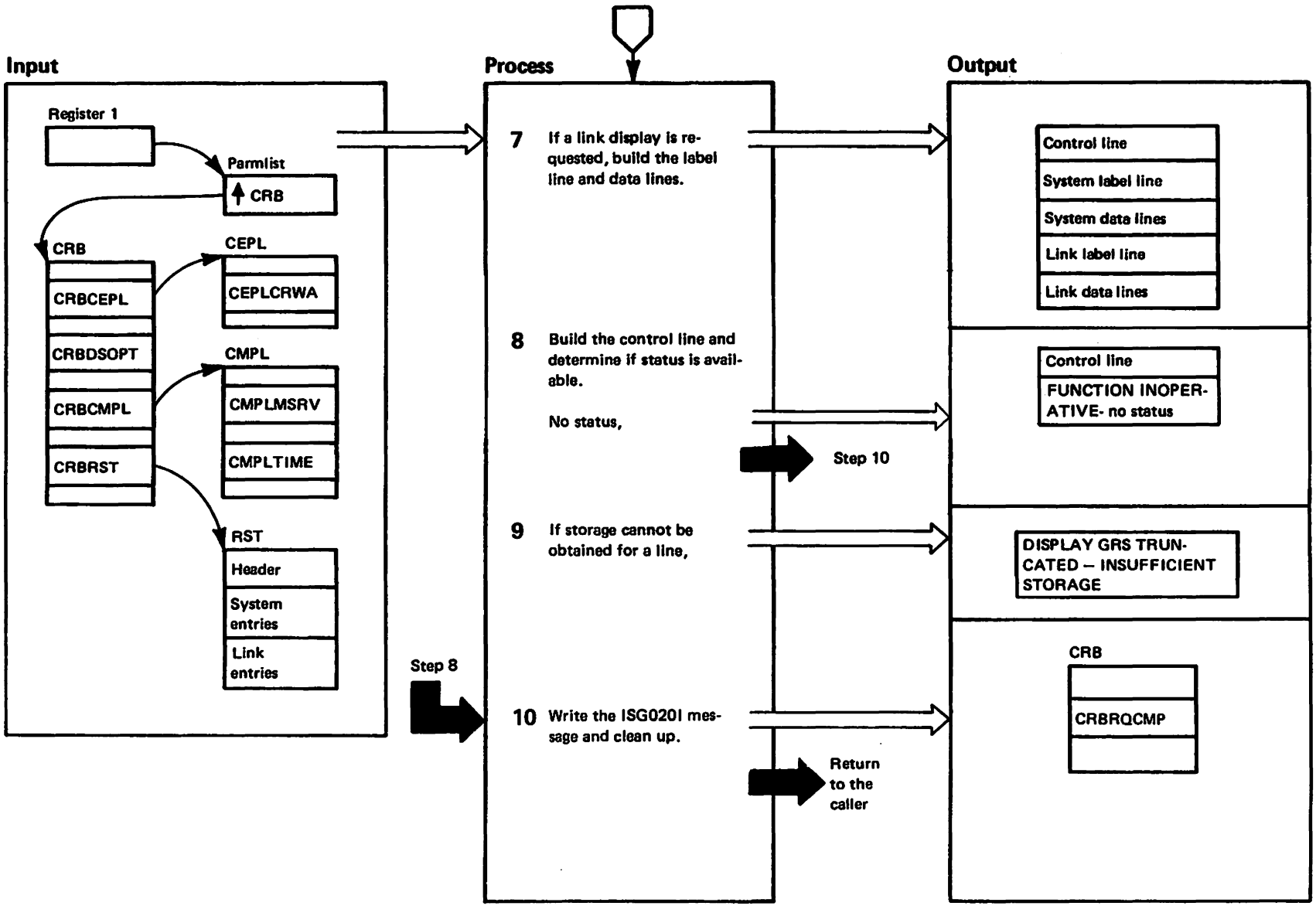
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-14. ISGCDSP – Global Resource Serialization DISPLAY GRS Request Processor (Part 4 of 6)

Extended Description	Module	Label
4 For a CONTENTION or an ALL request, ISGCDSP builds a display of all resources that have requestors that are waiting for the resource. ISGCDSP issues a GQSCAN macro to obtain resource contention information and invokes IEECB808 to get storage for a line. The display consists of two label lines for the resource name, one label for a requestor header line, and one data line for each requestor of the resource. This is repeated for each resource that has requestors waiting for the resource.	IEECB808	MSGSERV
5 For a RES request that is a qname's only request, ISGCDSP builds a display that contains all the qnames that match the request. ISGCDSP issues a GQSCAN macro to obtain resource information for the request. The display consists of a header (label) line followed by enough data lines to contain all the qnames that match the request, at eight qnames per data line.		
For a RES request that is a resource request, a resource display is built for all resources that match the request. ISGCDSP issues a GQSCAN macro to obtain resource information for the request and invokes IEECB808 to get storage for a line. The display format is the same as that for a CONTENTION request.	IEECB808	MSGSERV
6 For SYSTEM or ALL requests, ISGCDSP builds a label line and then a data line, describing two systems, for each pair (or single) of SYSTEM entries in the ring status table RST. While building the message, ISGCDSP calls IEECB808 at entry point MSGSERV to obtain storage for each line prior to building the line.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-14. ISGCDSP – Global Resource Serialization DISPLAY GRS Request Processor (Part 5 of 6)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-14. ISGCDSP – Global Resource Serialization DISPLAY GRS Request Processor (Part 6 of 6)

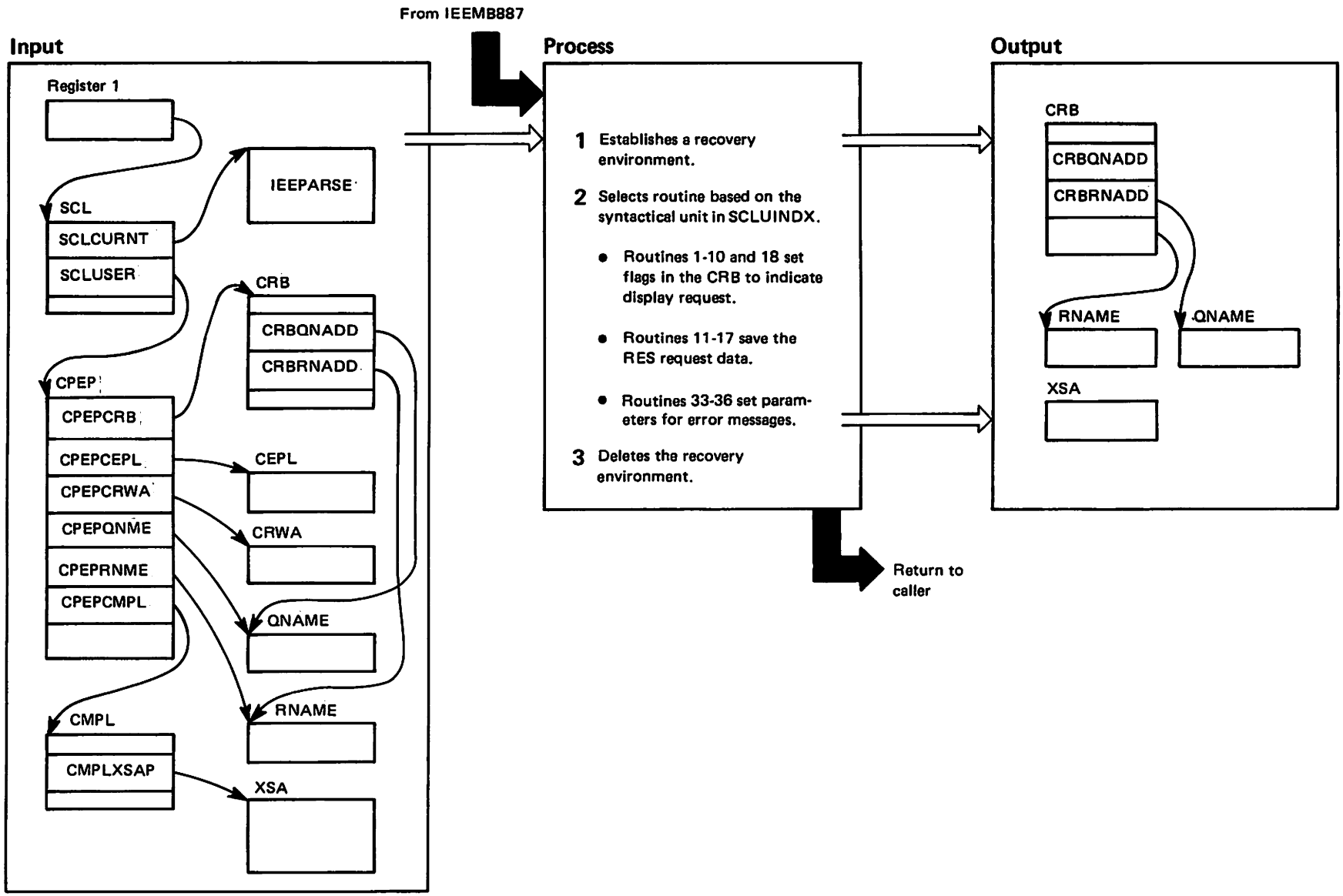
Extended Description	Module	Label
7 For LINK or ALL requests, ISGCDSP builds a label line and then a data line, describing 2 CTCs, for each pair (or single) LINK entries in the RST. If no link status is available, meaning there are no LINK entries in the RST, ISGCDSP builds a "NO LINKS" data line. While building the message, ISGCDSP calls IEECB808 at entry point MSGSERV to obtain storage for each line prior to building the line.	IEECB808	MSGSERV
8 ISGCDSP then determines if status information is available (CRBRST ≠0). If no status is available or if RNLs do not exist, ISGCDSP obtains storage and builds a "FUNCTION INOPERATIVE – NO STATUS" data line. Processing continues at step 10.		
9 If there is insufficient storage at any point in ISGCDSP, the following message is issued: "DISPLAY GRS TRUNCATED – INSUFFICIENT STORAGE".		
10 ISGCDSP calls MSGSERV to write the ISG020I message and to perform clean-up processing. ISGCDSP sets CRBRQCMP=1 to indicate that the request has been processed and returns to the caller.		

Recovery Processing

The global resource serialization command processing recovery routine (ISGCRCV) gives ISGCDSP control at entry point ISGCDS02 to do recovery processing. ISGCDSP at this entry point performs clean-up and returns to the caller.

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-15. ISGCMDE – DISPLAY GRS Command Parser Exit Routine (Part 1 of 2)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

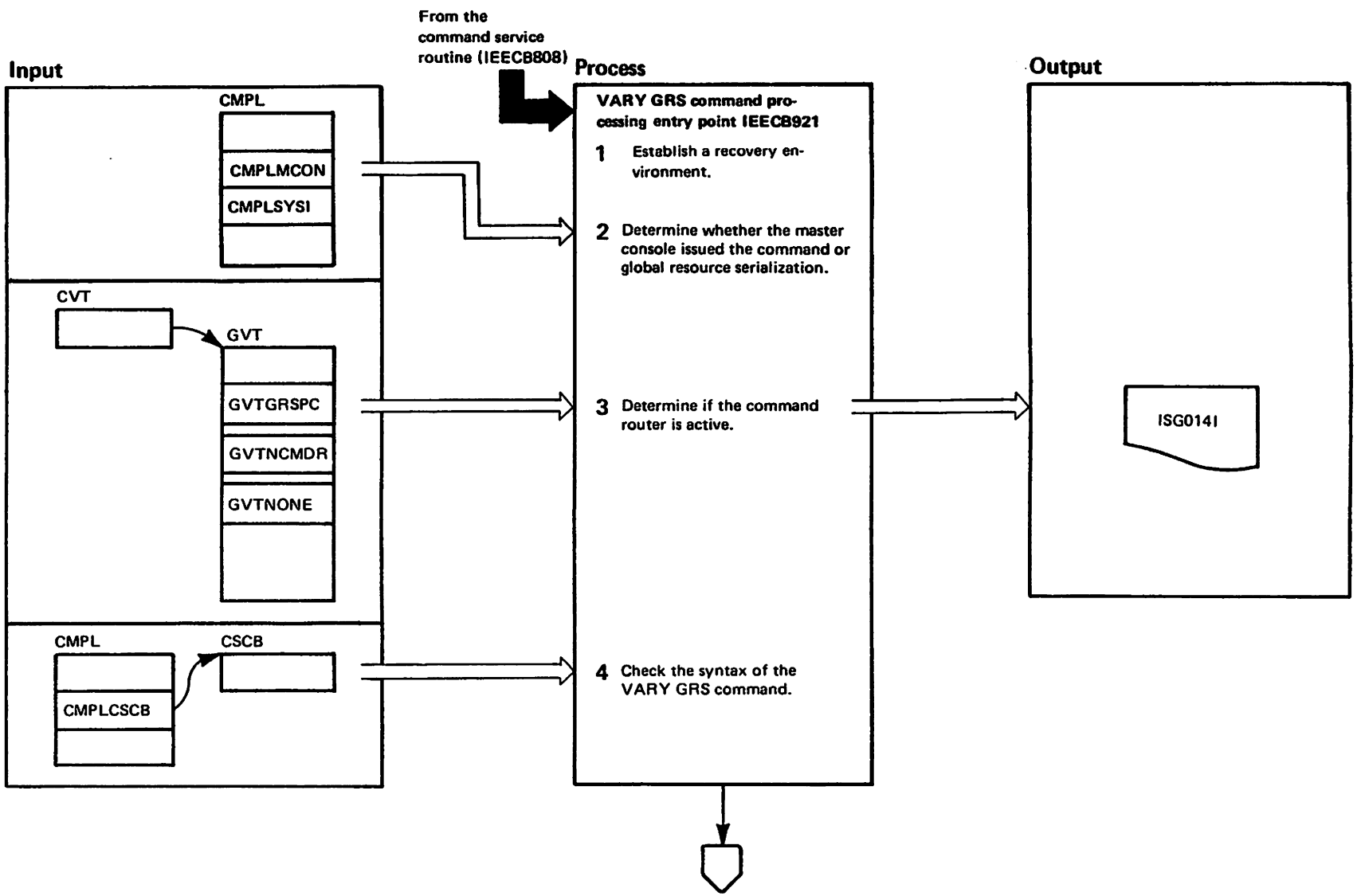
Diagram GRS-15. ISGCMDE – DISPLAY GRS Command Parser Exit Routine (Part 2 of 2)

Extended Description	Module	Label
ISGCMDE is an exit routine from the generalized parser (IEEMB887). ISGCMDI supplies the parse table and other parameters to the generalized parser to parse a DISPLAY GRS command. As each element of the DISPLAY GRS command is identified or as specific error conditions are found, IEEMB887 invokes ISGCMDE to record the finding in either the command request block (CRB) for correct syntax or the extended savearea (XSA) for incorrect command syntax.	ISGCMDE	
1 ISGCMDE establishes a recovery routine by putting the command recovery workarea (CRWA) on the command ESTAE parameter list (CEPL) stack and indicating why ISGCMDE was called.		
2 ISGCMDE selects the routine based on the particular syntactical unit being used. SCLUINDX is a parameter passed by IEEMB887 that identifies the syntactical unit that IEEMB887 found.		
• If a keyword is found, ISGCMDE indicates that in the CRB.		
• If the RES keyword is being parsed and a particular unit such as the qname or rname is being used, ISGCMDE saves them and converts them to EBCDIC.		
• If a syntax error is found, ISGCMDE indicates the error message in the XSA.		
3 ISGCMDE removes the CRWA from the CEPL stack to delete the recovery environment.		
Recovery Processing:		
The generalized parser's recovery environment and the GRS command recovery environment protect ISGCMDE. ISGCMDI establishes the GRS command recovery.		

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-16. ISGCMDI – Global Resource Serialization Command Interface (Part 1 of 6)



 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

"Restricted Materials of IBM"
Licensed Materials -- Property of IBM

Diagram GRS-16. ISGCMDI – Global Resource Serialization Command Interface (Part 2 of 6)

Extended Description

Module **Label**

The global resource serialization command interface performs authority checking for the VARY GRS command and syntax checking for the VARY GRS and DISPLAY GRS commands. Entry point IEECB921 processes the VARY command and entry point IEECB922 processes the DISPLAY command.

1 ISGCMDI issues an ESTAE to establish ISGCRCV as its recovery routine.

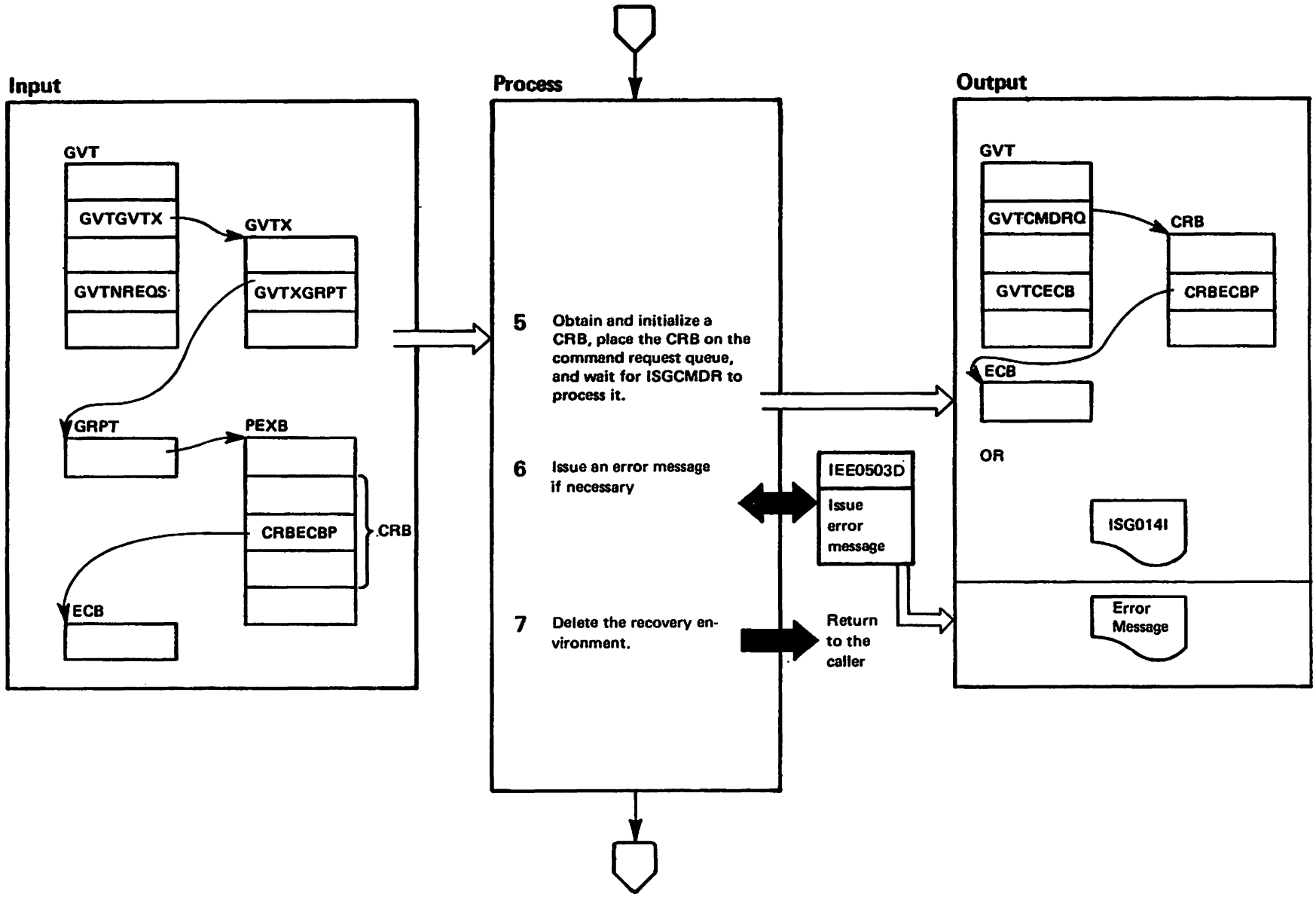
2 If the command parameter list master console bit is on (CMPLMCON=1) indicating that the master console issued the command, processing continues. Processing also continues when the system-issued bit is on (CMPLSYSI='1') in the command parameter list, indicating that the system issued the command. Otherwise, ISGCMDI issues error message IEE345I at step 6, indicating invalid VARY authority.

3 If the command router (ISGCMDR) is active (GVTNCMDR=0), global resource serialization option processing is complete (GVTGRSPC=1), and GRS=NONE was not specified at IPL (GVTNONE=0), then ISGCMDI continues processing. If one of the above is not true, ISGCMDI issues an error message (ISG014I) at step 6 indicating that global resource serialization or the command processor is inoperative.

4 This module checks the VARY GRS command syntax for the proper placement of delimiters and operands. If the syntax is not correct, ISGCMDI issues the appropriate error message at step 6.

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-16. ISGCMDI – Global Resource Serialization Command Interface (Part 3 of 6)



 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

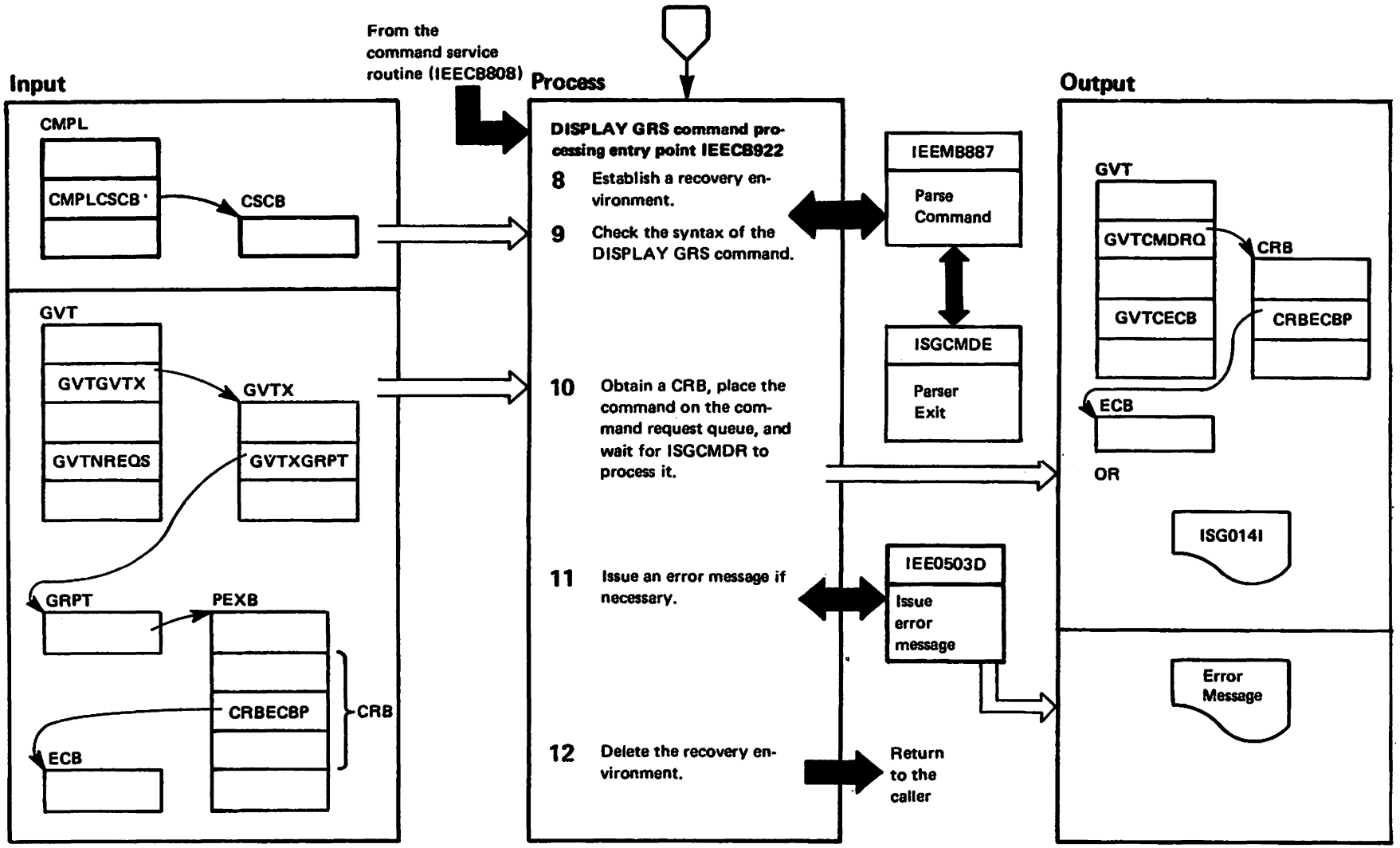
"Restricted Materials of IBM"
Licensed Materials – Property of IBM

Diagram GRS-16. ISGCMDI – Global Resource Serialization Command Interface (Part 4 of 6)

Extended Description	Module	Label
5 If requests are allowed on the command request queue (GVTNREQS=0), ISGCMDI invokes ISGSMI to obtain a command request block (CRB) from the global resource serialization address space, initializes the CRB, places it on the command request queue, and notifies the command request router (ISGCMDR) of work. This module then waits for ISGCMDR to process the VARY, if ISGCMDR returns with an error post code, ISGCMDI issues the appropriate error message at step 6. If requests are not allowed on the command request queue, ISGCMDI issues message ISG014I.	ISGSMI ISGCMDR	
6 ISGCMDI calls the appropriate module to issue any error message required for an error that occurred while processing steps 2-5. ISGMSG00 issues message ISG014I and IEE0503D issues the rest.	ISGMSG00 IEE0503D	
If an error occurs while processing a VARY GRS (ALL), RESTART command that global resource serialization issued, ISGCMDI calls ISGMSG00 to issue message ISG025E indicating that this system was unable to automatically rebuild the disrupted ring.		
7 ISGCMDI issues an ESTAE to delete the recovery environment.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-16. ISGCMDI – Global Resource Serialization Command Interface (Part 5 of 6)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

Diagram GRS-16. ISGCMDI – Global Resource Serialization Command Interface (Part 6 of 6)

Extended Description	Module	Label
8 ISGCMDI issues an ESTAE to establish ISGCRCV as its recovery routine.		
9 ISGCMDI invokes the generalized parser (IEEMB887) to check the DISPLAY GRS command syntax for the proper placement of delimiters and operands. IEEMB887 uses ISGCMDE during the syntax check. If the syntax is not correct, ISGCMDI issues the appropriate error message at step 11.	IEEMB887	
10 ISGCMDI invokes ISGSMI to obtain a command request block (CRB) from the global resource serialization address space, places the CRB on the command request queue, and notifies the command router (ISGCMDR) of this work provided that the following conditions exist: • The command router (ISGCMDR) is active (GVTNCMDR=0). • Requests are allowed on the command request queue (GVTNREQS=0) • Global resource serialization option processing is complete (GVTGRSPC=1). • GRS=NONE was not specified during the IPL (GVTNONE=0).	ISGSMI ISGCMDR	
ISGCMDI then waits for ISGCMDR to process the DISPLAY. If ISGCMDR returns with a post code indicating an error, ISGCMDI issues the appropriate error message at step 11. If one of the above conditions does not exist, ISGCMDI attaches ISGCDSP to do one of the following: a) If no RNLs exist, issues a "FUNCTION INOPERATIVE–NO STATUS" message b) If the contention display, the RNL display, or the resource displays are requested, builds the requested display		

Extended Description	Module	Label
11 ISGCMDI calls IEE0503D to issue any error message required resulting from an error that occurred while processing steps 9 and 10.	IEE0503D	
12 ISGCMDI issues an ESTAE to delete the recovery environment.		
Recovery Processing The command recovery routine (ISGCRCV) gives ISGCMDI control at entry point ISGCDIRV to do recovery processing. When entered at ISGCDIRV, ISGCMDI checks the CRWA for a CRB address. If one is found, ISGCMDI verifies the CRB, invokes ISGSMI to release the CRB, and returns to the caller to continue with termination. If the CRB found in the CRWA is on the command request queue and a wait has not been issued, ISGCMDI retries at the wait. If the CRWA does not contain the address of a CRB, ISGCMDI returns to caller to continue with termination.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

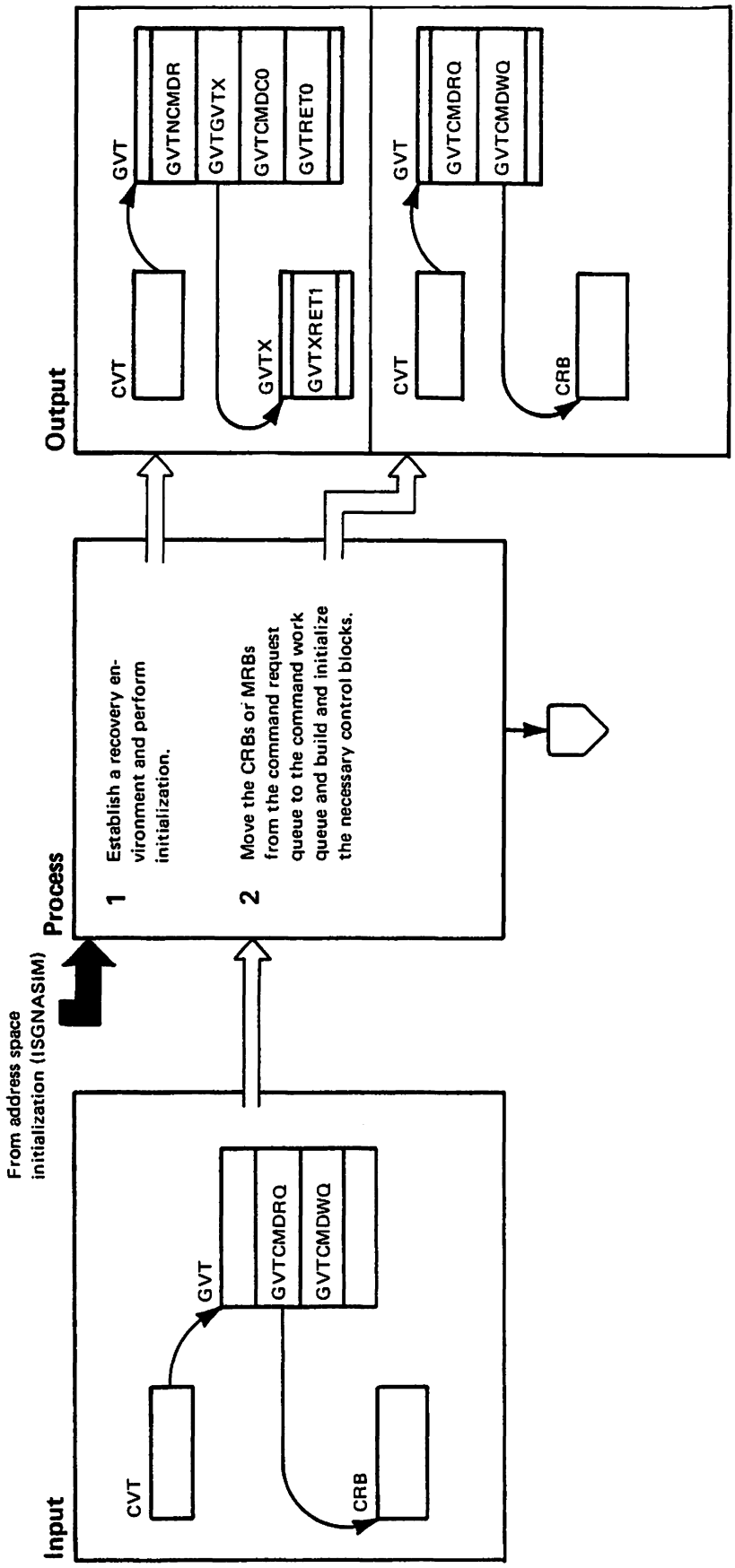


Diagram GRS-17. ISGCMDR – Global Resource Serialization Command Router (Part 1 of 8)

LY28-1695-0 (c) Copyright IBM Corp. 1987

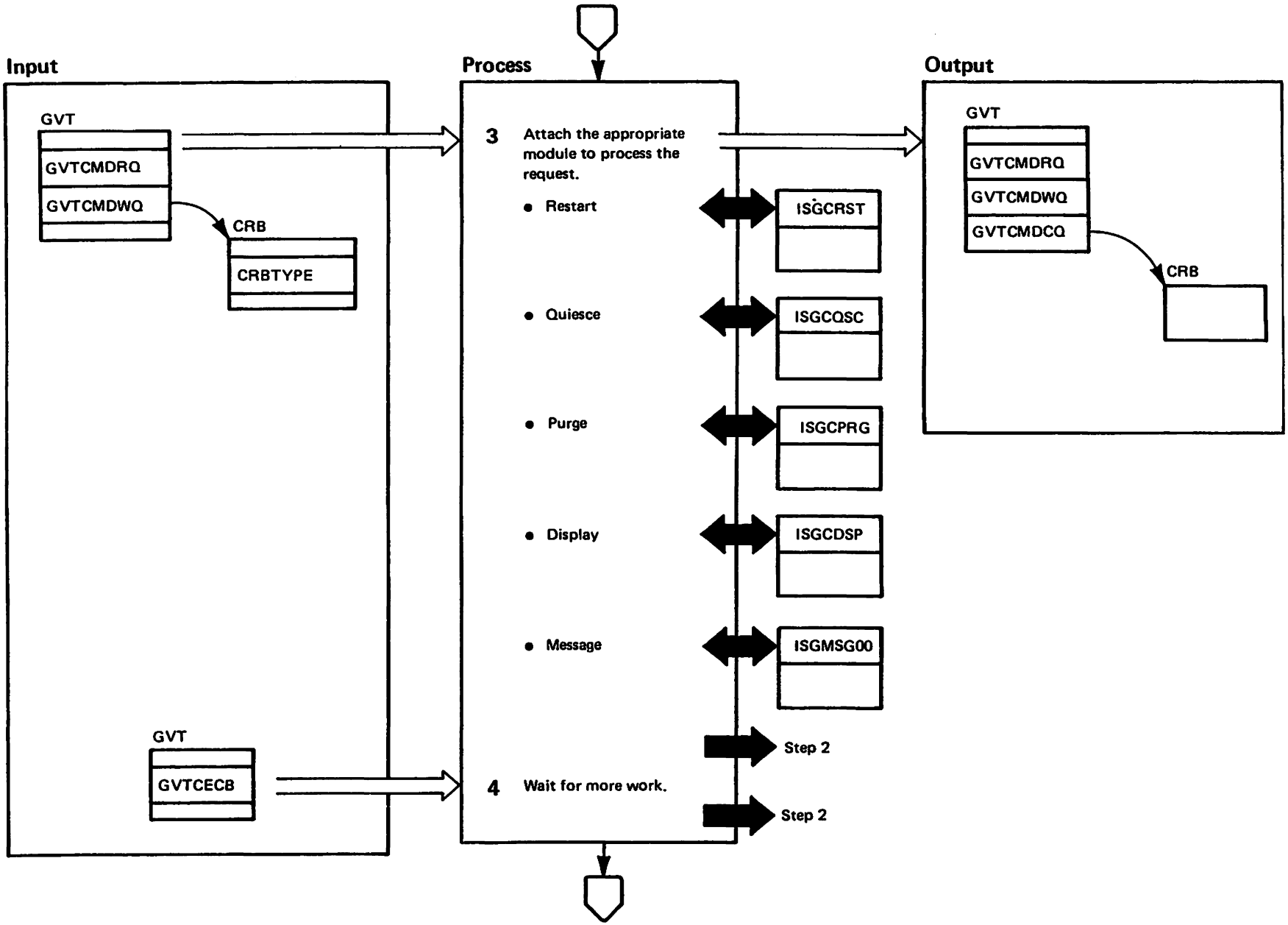
"Restricted Materials of IBM"
Licensed Materials - property of IBM

Diagram GRS-17. ISGCMDR -- Global Resource Serialization Command Router (Part 2 of 8)

Extended Description	Module	Label
The global resource serialization command router attaches the message module to process message requests and the restart, quiesce, purge, and display processors to process the VARY GRS and DISPLAY GRS commands. This module is also called at entry point ISGCTXR1 to detach the command processor and release any storage it obtained for the command processor.		
1 ISGCMDR issues an ESTAE to establish ISGCRCV as its recovery routine. This module then loads ISGCRET0 and ISGCRET1 and sets the GVTNCMDR bit off to indicate that the command router is active. ISGCMDR verifies the command cleanup queue by ensuring that each element on the queue is in a page with no storage checks and that each element is either a CRB or MRB, otherwise, ISGCMDR truncates the queue.		
2 The command router uses compare and double swap to move a command request block (CRB) and/or message request block (MRB) from the command request queue to the command work queue. If the request is restart, quiesce, purge, or display, ISGCMDR obtains storage for the command ESTAE parameter list (CEPL), command recovery workarea (CRWA), and a full ring status table (RST), initializes them, and saves their addresses in the CRB. ISGCMDR (for a display request) then calls ISGBCI which invokes entry point ISGBRFSN (in ISGBRF) to get the status of each system in the CTCs for each system. If the request is for a message, ISGCMDR obtains storage for the CEPL and CRWA and saves their addresses in the MRB.	ISGBCI	ISGBRFSN

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-17. ISGCMDR – Global Resource Serialization Command Router (Part 3 of 8)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

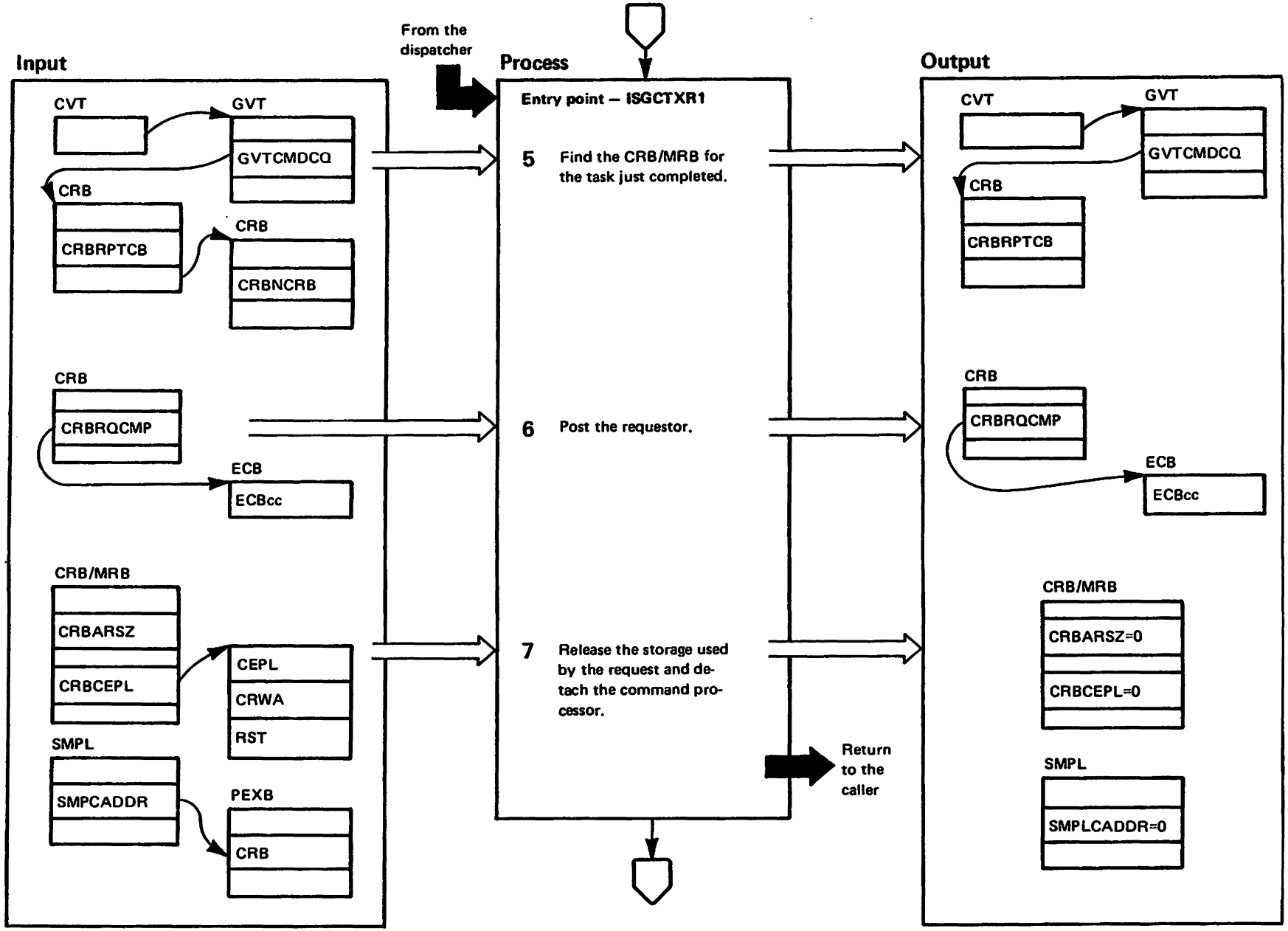
"Restricted Materials of IBM"
Licensed Materials – Property of IBM

Diagram GRS-17. ISGCMDR – Global Resource Serialization Command Router (Part 4 of 8)

Extended Description	Module	Label
3 ISGCMDR attaches the appropriate request processor. If the attach is successful, ISGCMDR saves the TCB address in the CRB or MRB and uses compare and swap to place the CRB or MRB onto the clean-up queue. If the attach fails, this module returns an error post code and frees any unneeded storage. Steps 2 and 3 are repeated until the command work queue is empty.		
4 When both the command request queue and command work queue are empty, ISGCMDR issues a wait on GVTCECB. This ECB is posted by either the command interface routine (ISGCMDI), the RSA SEND/RECEIVE routime (ISGBSM), the termination resource manager (ISGGTRM0), or the mainline recovery routine (ISGGFRR0).		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-17. ISGCMDR – Global Resource Serialization Command Router (Part 5 of 8)



 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – property of IBM

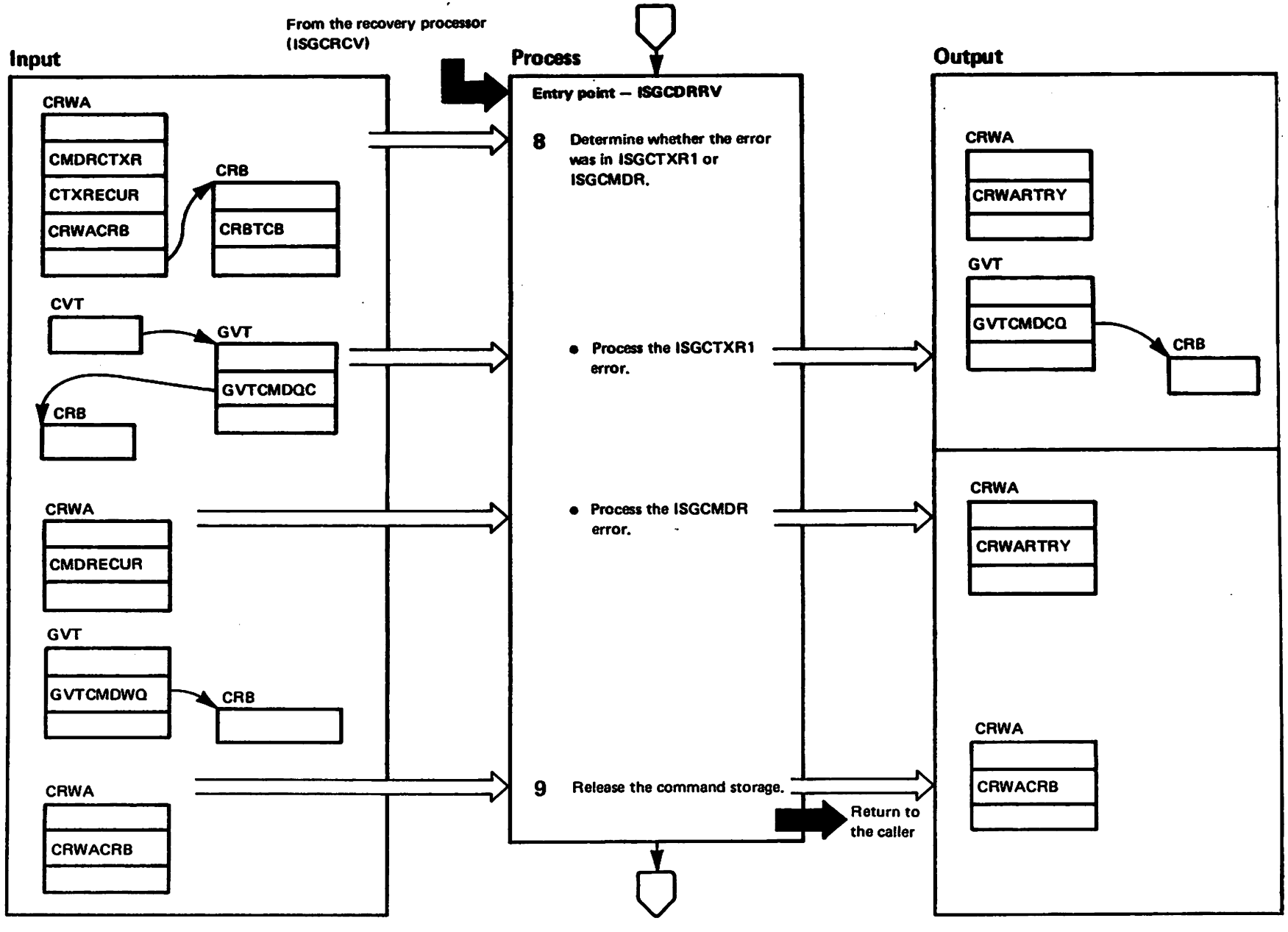
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-17. ISGCMDR – Global Resource Serialization Command Router (Part 6 of 8)

Extended Description	Module	Label
Entry point ISGCTXR1 – The dispatcher gives control to ISGCMDR at entry point ISGCTXR1 after a global resource serialization command processor has completed. At this entry point ISGCMDR releases command related storage and detaches the command processor.		
5 ISGCMDR finds the CRB/MRB for the completed command cleanup queue by matching the TCB for the completed task to a TCB in the control blocks on the command cleanup queue.		
6 If there is an ECB address in the CRB/MRB, ISGCMDR posts the command requestor with the results of the command (0 for success and 8 for failure).		
7 This module issues a FREEMAIN macro to release the storage occupied by the CRWA, the CEPL, and the RST. ISGCMDR calls ISGSDAL to return the cell used by the CRB/MRB to the pool extent block (PEXB), detaches the completed command processor, and returns to the caller.	ISGSDAL	

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-17. ISGCMDR – Global Resource Serialization Command Router (Part 7 of 8)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

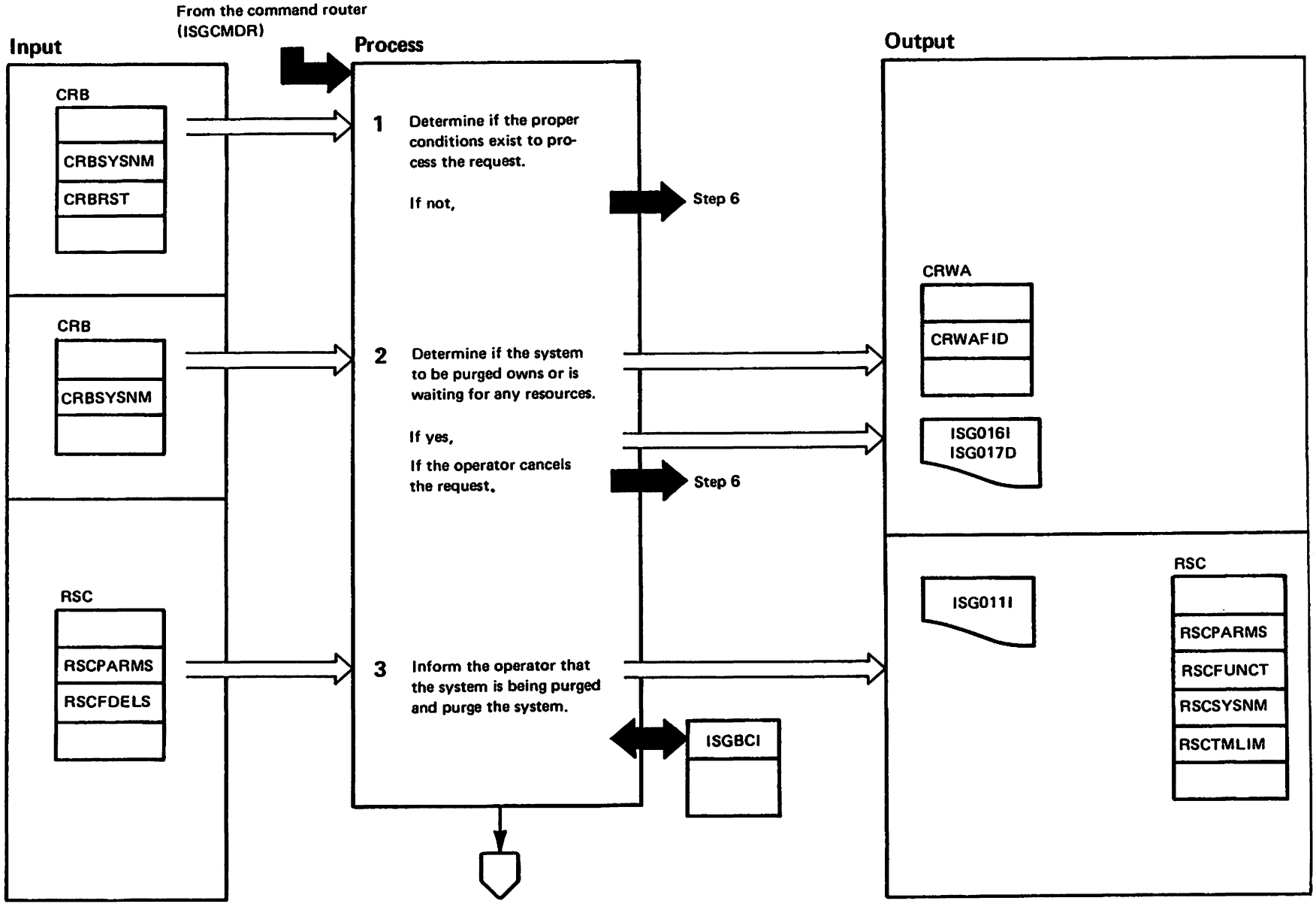
"Restricted Materials of IBM"
Licensed Materials – Property of IBM

Diagram GRS-17. ISGCMDR – Global Resource Serialization Command Router (Part 8 of 8)

Extended Description	Module	Label
Entry point ISGCDRRV – The command recovery routine (ISGCRCV) gives ISGCMDR control at entry point ISGCDRRV to do recovery processing.		
8 ISGCMDR determines whether the error occurred in ISGCMDR or ISGCTXTR1 by checking CMDRCTXR. If CMDRCTXR is set to one, the error occurred in ISGCTXR1, if not, the error was in ISGCMDR. If the error was in ISGCTXR1, ISGCMDR determines if this is a recursion (CTXRECUR=1), and if so, continues with termination. If this is not a recursion, ISGCMDR sets up for a retry at entry point ISGCTXR1, detaches the command processor, and cleans up the command-related resources. ISGCMDR then verifies the command cleanup queue by ensuring that each element on the queue is in a page with no storage checks and that each element is either a CRB or MRB, otherwise, ISGCMDR truncates the queue. If the error was in ISGCMDR, the processing is the same except the retry is set for the appropriate entry point in ISGCMDR.		
9 If the CRWA points to a CRB (CRWACRB contains an address), ISGCMDR calls ISGSDAL to return the ROA control block cells back to the pool extent block (PEXB). This module then returns to ISGCRCV indicating whether to retry or continue with termination.	ISGSDAL	

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-18. ISGCPRG – Global Resource Serialization VARY GRS PURGE Request Processor (Part 1 of 4)



LY28-1595-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-18. ISGCPRG – Global Resource Serialization VARY GRS PURGE Request Processor (Part 2 of 4)

Extended Description

ISGCPRG processes the PURGE parameter of the VARY GRS command. The PURGE parameter removes a system from the global resource serialization complex. ISGCPRG receives control from the command router (ISGCMDR) when a command request block (CRB) for a purge request is found on the global resource serialization command work queue. ISGCPRG obtains ring status by invoking ISGBCI which invokes ISGBRF (at entry point ISGBRFSN).

1 The following conditions must be met to process the purge request:

- The system issuing the purge request must be an active system in the global resource serialization ring.
- The system being purged must be known to the global resource serialization complex, and must not be an active, joining, or restarting system.

If these conditions are not met, ISGCPRG rejects the request and processing continues at step 6.

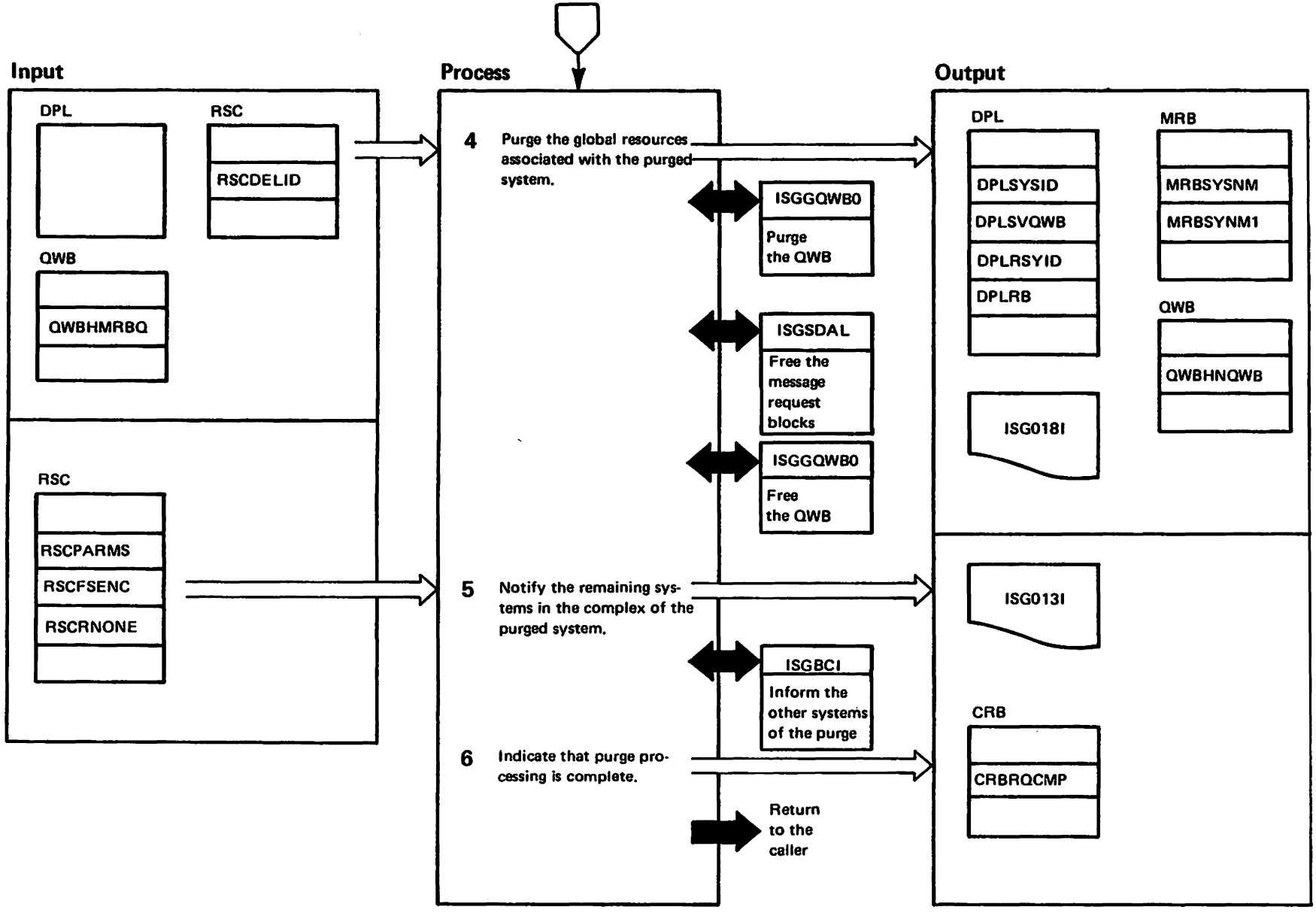
2 ISGCPRG issues the GQSCAN macro to determine if the system being purged owns or is waiting for any resources. If there are resources associated with the system to be purged, ISGCPRG issues message ISG016I informing the operator of that fact, then issues message ISG017D to give the operator the chance to cancel the purge request. If the operator replys "NO" ISGCPRG cancels the request and continues processing at step 6.

3 ISGCPRG issues message ISG011I informing the operator that the system named in the request is being purged. ISGPRG then calls ISGBCI to remove the requested system from the global resource serialization complex.

Module	Label
ISGBCI	

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-18. ISGCPRG – Global Resource Serialization VARY GRS PURGE Request Processor (Part 3 of 4)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-18. ISGCPRG – Global Resource Serialization VARY GRS PURGE Request Processor (Part 4 of 4)

Extended Description

Module **Label**

4 ISGCPRG sets up a dequeue purge list (DPL) and calls ISGGQWB0 at entry point ISGGQWB5 to perform a SYSID purge of the resources held or requested by the system being purged. ISGGQWB5 passes back the address of a queue of messages to be issued regarding the resources that it purged. ISGCPRG builds a header message to go on top of those messages and calls ISGMSG00 to issue the messages. ISGCPRG sets up a storage manager parameter list (SMPL) describing the MRBs and calls ISGSDAL to free them. This module then calls ISGGQWB0 at entry point ISGGQWBF to free the QWB returned by ISGGQWB5.

ISGGQWB0 ISGGQWB5

ISGMSG00

ISGSDAL

ISGGQWB0 ISGGQWBF

5 ISGCPRG calls ISGBCI which invokes ISGBRF (at entry point ISGBRFNM) at SENDCMD to inform the remaining systems in the complex of the purged system and calls ISGMSG00 to inform the operator on this system of the purged system.

ISGBCI ISGBRFNM (SENDCMD)

ISGMSG00

6 ISGCPRG sets CRBRQCMP=1 indicating that purge request processing is complete and returns to the command router (ISGCMDR).

Recovery Processing

The command recovery routine (ISGCRCV) gives ISGCPRG control at entry point ISGCPG02 to do recovery processing. ISGCPRG issues message ISG015I to indicate which function caused the error and the reason for the error. ISGCPRG indicates in the CRB that purge processing is complete and, if the failure was caused by an error in ISGBCI, records the ring status changed parameter list (RSC) in the SDWA. ISGCPRG sets a recovery processing return code (0=recovery processing successful and 4=unsuccessful) and returns to the caller.

LY28-1695-0 (c) Copyright IBM Corp. 1987

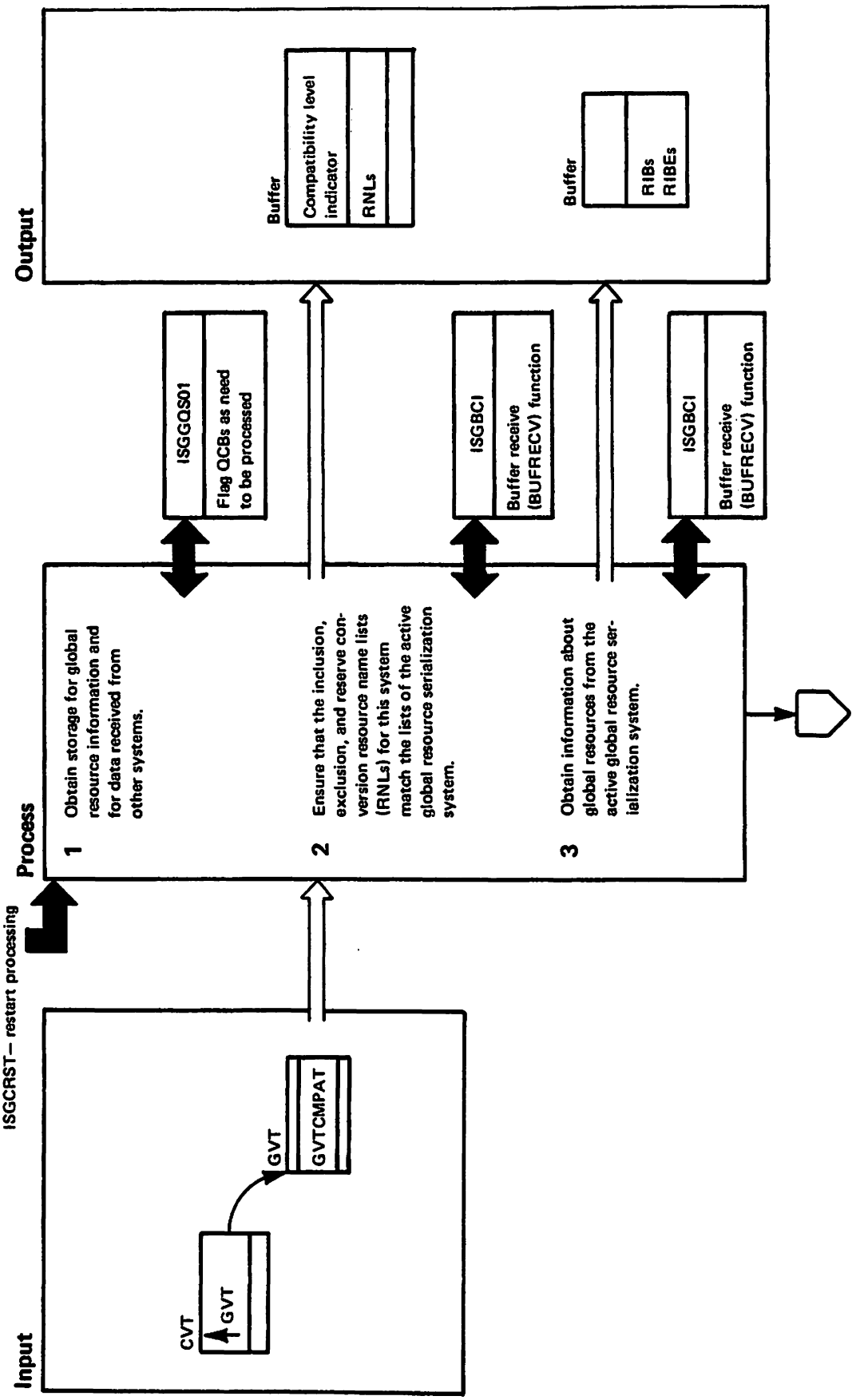


Diagram GRS-19. ISGCQMRG – Global Resource Serialization Queue Merge (Part 1 of 6)

LY28-1695-0 (c) Copyright IBM Corp. 1987

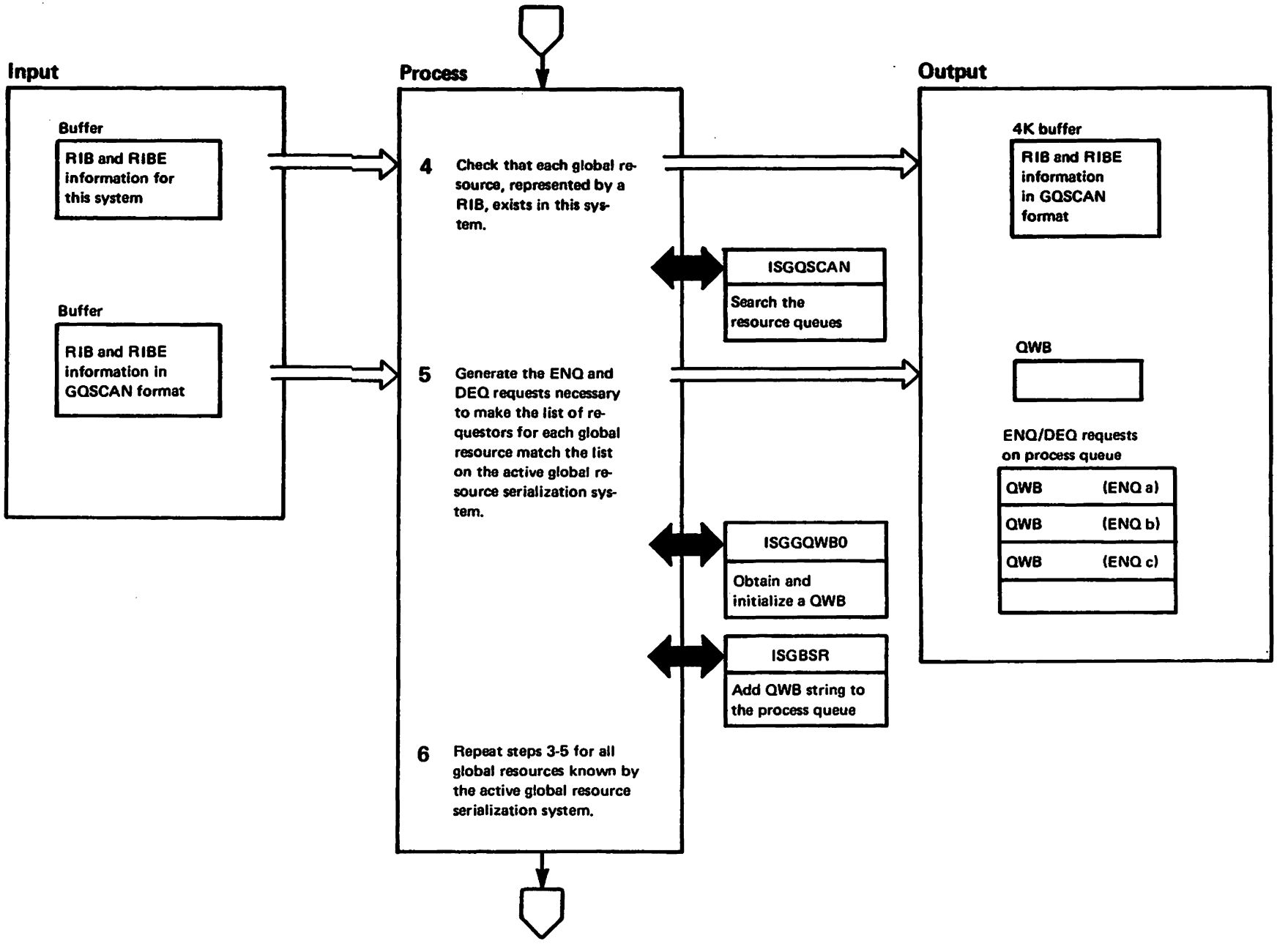
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-19. ISGCQMRG – Global Resource Serialization Queue Merge (Part 2 of 6)

Extended Description	Module	Label
Whenever a system is joining the global resource serialization ring or restarting global resource serialization, join processing (ISGNGRSP) or restart processing (ISGCRST) calls queue merge (ISGCQMRG) to perform the following: • Verify that the inclusion, exclusion, and reserve conversion resource name lists (RNLs) of the system joining or restarting in the global resource serialization ring match the inclusion, exclusion, and reserve conversion lists of the active global resource serialization system. • Generate the ENQ or DEQ requests necessary to make this system's global resource queues match those of the active global resource serialization system. ISGCQMRG loads module ISGGQSRV to use the various global resource serialization service routines provided by ISGGQSRV.		
1 ISGCQMRG performs some initialization for subsequent processing and initializes and queues the recovery workarea for the ESTAE/I recovery routine (ISGCRCV). ISGCQMRG issues a GETMAIN macro to obtain 64K bytes of storage from subpool 229. 60K bytes of this storage is used as a buffer to hold the data sent from an active global resource serialization system. The remaining 4K is used to contain information about the global resource queues of this system. ISGCQMRG invokes ISGGQSRV at entry point ISGGQS01 to set a flag in all the global QCBs. The flag indicates that the QCB has not yet been processed by ISGCQMRG.	ISGCQMRG ISGGQSRV	 ISGGQS01
2 ISGCQMRG invokes the buffer receive function of ISGBCI. The first buffer sent by the active global resource serialization system contains the global resource serialization compatibility level indicator followed by the inclusion, exclusion, and reserve conversion RNLs. To preserve data integrity, these lists must match the ones specified for this system. If the compatibility level indicator does not match that of the active global resource serialization system, or if the resource name lists do not match, ISGCQMRG issues an X'09A' ABEND with the appropriate reason code. If the compatibility levels are the same and the lists match, processing continues.	ISGBCI	BUFRECV
3 ISGCQMRG invokes the buffer receive function of ISGBCI again to cause the active global resource serialization system to send information about global resources in the form of resource information blocks (RIBs) and resource information block extensions (RIBEs) to this system. ISGBCI does not return control until it copies the data into the buffer area obtained in step 1. ISGCQMRG ensures that the RIBs and RIBEs are constructed properly. If they are not, ISGCQMRG issues an X'09A' ABEND with an appropriate reason code; otherwise, processing continues.	ISGBCI	BUFRECV

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-19. ISGCQMRG – Global Resource Serialization Queue Merge (Part 3 of 6)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

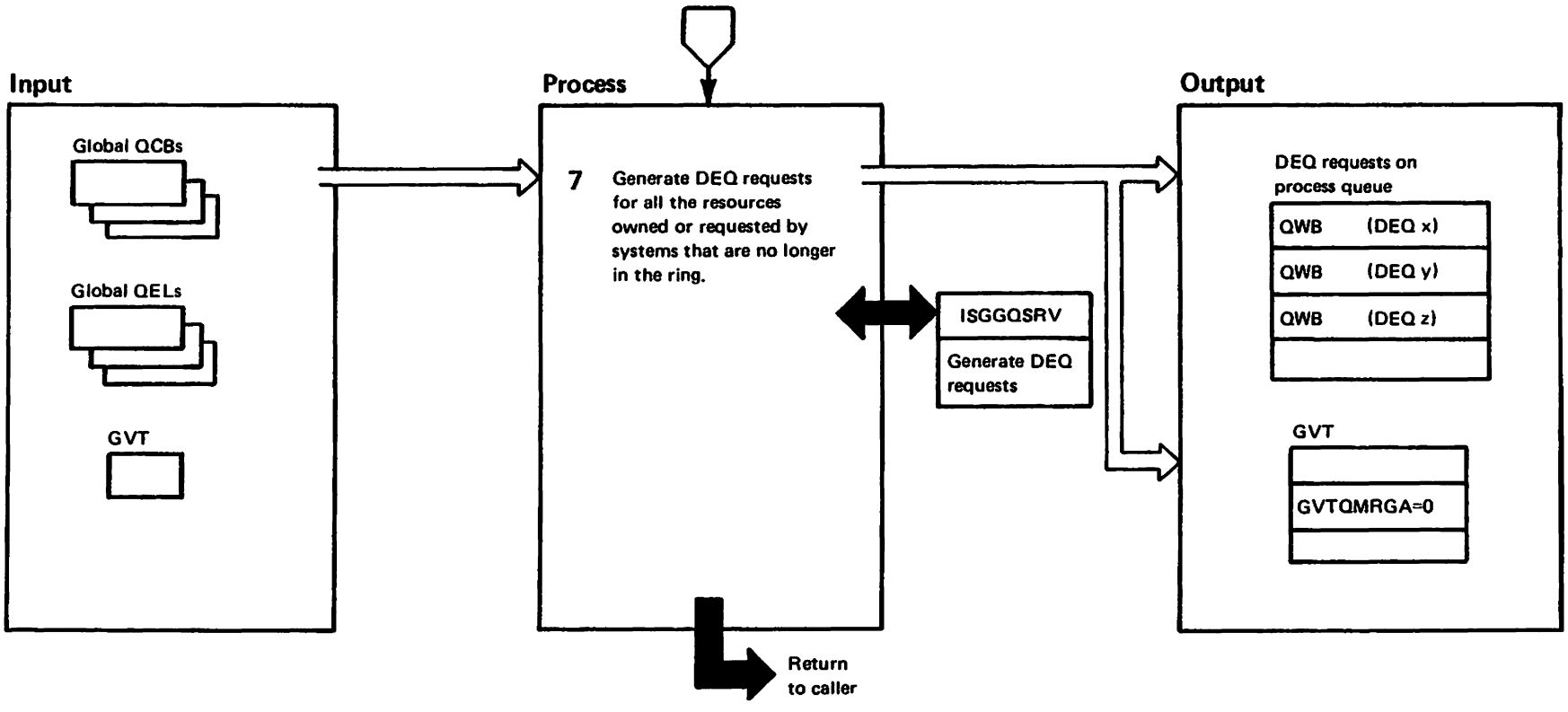
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-19. ISGCQMRG – Global Resource Serialization Queue Merge (Part 4 of 6)

Extended Description	Module	Label
4 ISGCQMRG copies the qname and rname from the resource information block (RIB) for each global resource and passes them to ISGQSCAN via the GQSCAN macro to obtain any information this system has describing requestors of the global resource. If ISGQSCAN returns with a return code indicating that no data was found, ISGCQMRG issues an X'09A' ABEND with an appropriate reason code.	ISGQSCAN	QSCAN
5 ISGCQMRG generates and queues the ENQ and DEQ requests necessary to make this system's list of requestors for the global resource match the list on the active global resource serialization system. To do this, ISGCQMRG:		
• Invokes ISGGQWB0 at entry point ISGGQWB2 to copy the information in the GQSCAN buffers into a queue work block (QWB) that is suitable for the process queue	ISGGQWB0	ISGGQWB2
• Invokes ISGBSR at entry point ISGBBE to place this QWB on the process queue	ISGBSR	ISGBBE
6 ISGCQMRG repeats steps 3 thru 5 until the active global resource serialization system has sent all the information about each global resource on that system.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-19. ISGCQMRG – Global Resource Serialization Queue Merge (Part 5 of 6)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-19. ISGCQMRG – Global Resource Serialization Queue Merge (Part 6 of 6)

Extended Description

Module **Label**

ISGCQMRG

7 If this system had been quiesced and is now restarting global resource serialization, this system can indicate some global resources being owned by other systems that have actually released those resources. These resources were not in the list that was sent by the active global resource serialization system and must be removed from this system's global resource queues. ISGCQMRG invokes ISGGQSRV at entry point ISGGQS03 to scan the QCB/QEL chains and generate DEQ requests for all requestors of global resources not known to the other systems in the global resource serialization ring. The global resource queues of this system now match the queues of the active global resource serialization system. If ISGGQS03 is unsuccessful, ISGCQMRG issues an X'09A' ABEND with a reason code identifying the error. ISGCQMRG frees the storage used to contain information about the global resource queues of this system and the data sent from the active global resource serialization system. ISGCQMRG returns to the caller with an indication in the GVT that the queue merge process was successful (GVTQMRGA=0).

ISGGQSRV ISGGQS03

Recovery Processing

When an error occurs while ISGCQMRG is executing, RTM calls ISGCRCV. ISGCRCV passes control to a special error exit routine in ISGCQMRG to perform the following:

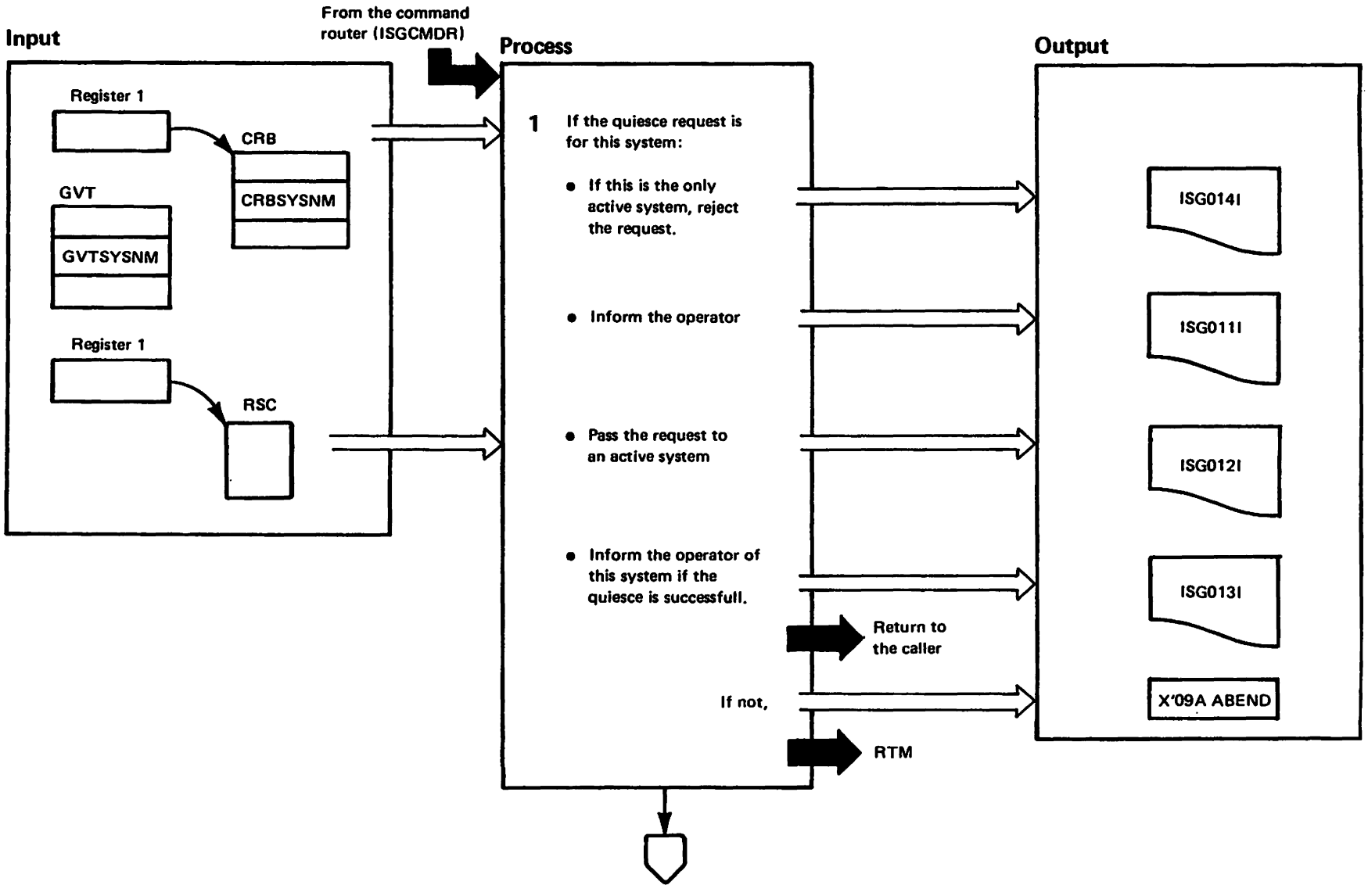
- Release any storage obtained for QWBs
- Delete module ISGGQSRV
- Specify storage to be released by ISGCRCV

ISGCQMRG returns control to ISGCRCV to process the following recovery options:

- Retry if allowed
- Take a dump using default options
- Release dynamic area and buffer area obtained for GQSCAN and BUFRECV.
- If retry is not allowed, ISGCRCV returns control to RTM to continue with termination.

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-20. ISGCQSC – Global Resource Serialization VARY GRS QUIESCE Request Processor (Part 1 of 4)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

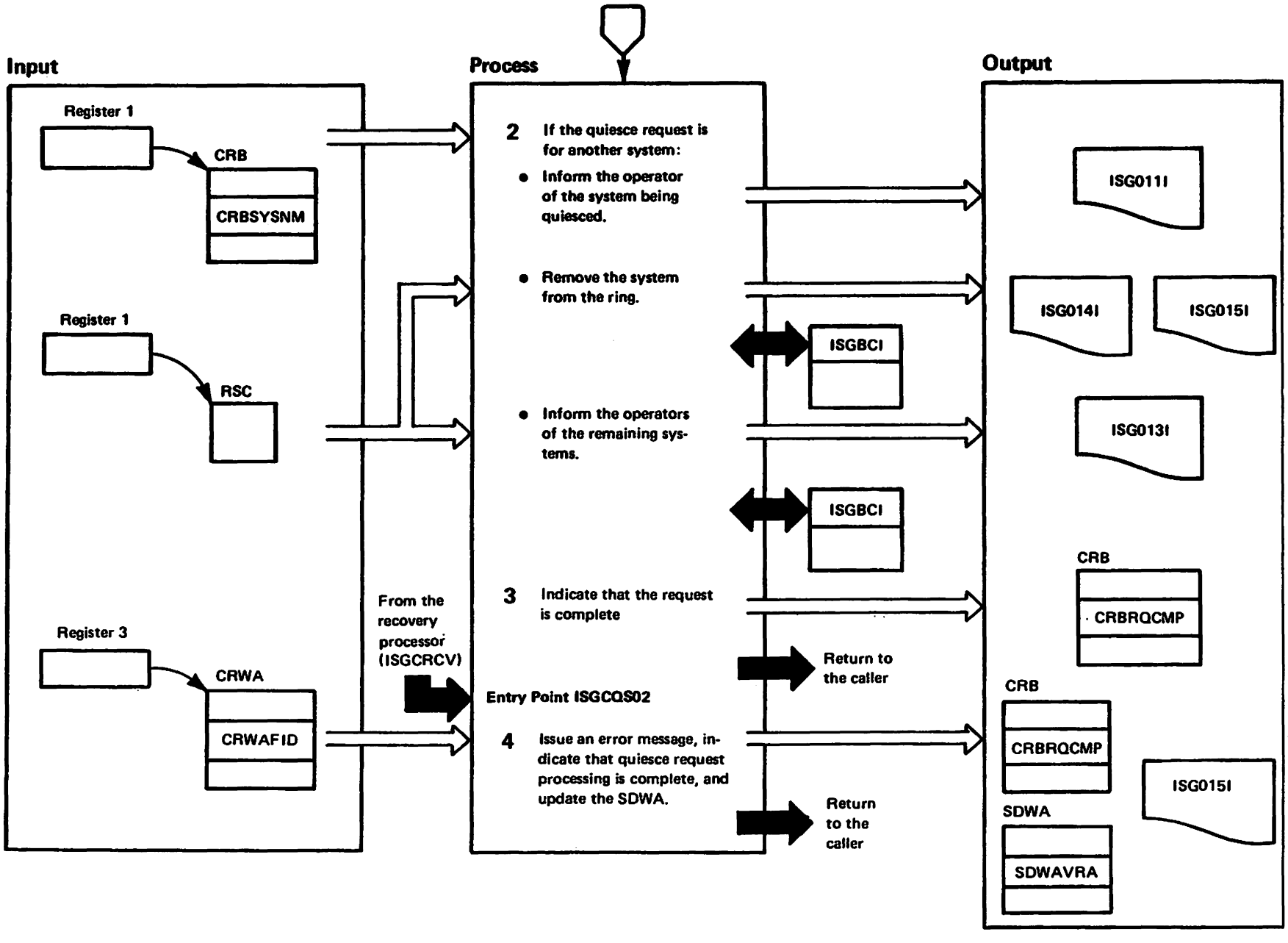
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-20. ISGCQSC – Global Resource Serialization VARY GRS QUIESCE Request Processor (Part 2 of 4)

Extended Description	Module	Label
ISGCQSC processes the QUIESCE parameter of the VARY GRS command. The QUIESCE parameter removes a system from a global resource serialization ring. ISGCQSC receives control from the command router (ISGCMDR) when a command request block (CRB) for quiesce processing is found on the global resource serialization command work queue. ISGCQSC obtained the ring status by invoking ISGBCI which invokes ISGBRF (at entry point ISGBRFSN).		
1 If the operator requests a quiesce of his own system, ISGCQSC determines if the system is the only active system. If true, this module rejects the request and issues message ISG014I. Otherwise, ISGCQSC issues message ISG011 to inform the operator that this system is quiescing global resource serialization. Since the system being quiesced cannot process the quiesce request itself, ISGCQSC calls ISGBCI which invokes ISGBRF (at entry point ISGBRFNM) to pass the request to another active system in the ring. When ISGBCI returns, ISGCQSC checks for successful completion of the request. If the request was successful, ISGCQSC issues message ISG013I, if not, it issues an X'09A' abend.	ISGBCI	ISGBRFNM

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-20. ISGCQSC – Global Resource Serialization VARY GRS QUIESCE Request Processor (Part 3 of 4)



"Restricted Materials of IBM"
Licensed Materials – Property of IBM

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-20. ISGCQSC – Global Resource Serialization VARY GRS QUIESCE Request Processor (Part 4 of 4)

Extended Description

Module **Label**

2 If the quiesce request is for a system other than the one issuing the command or is a request sent from another system, ISGCQSC issues message ISG011I to the system being quiesced and the system that issued the command, informing the operator that his system is being quiesced. ISGCQSC calls ISGBCI which invokes ISGBRF (at entry point ISGBRFNM) to remove the requested system from the global serialization complex. If the requested system is not active, ISGCQSC issues message ISG014I and ISG015I to inform the operator that the command was rejected because the target system was not active. If the system was successfully quiesced, this module calls ISGBCI to issue message ISG013I to the remaining active systems in the complex informing them that the quiesced system has been removed from the complex.

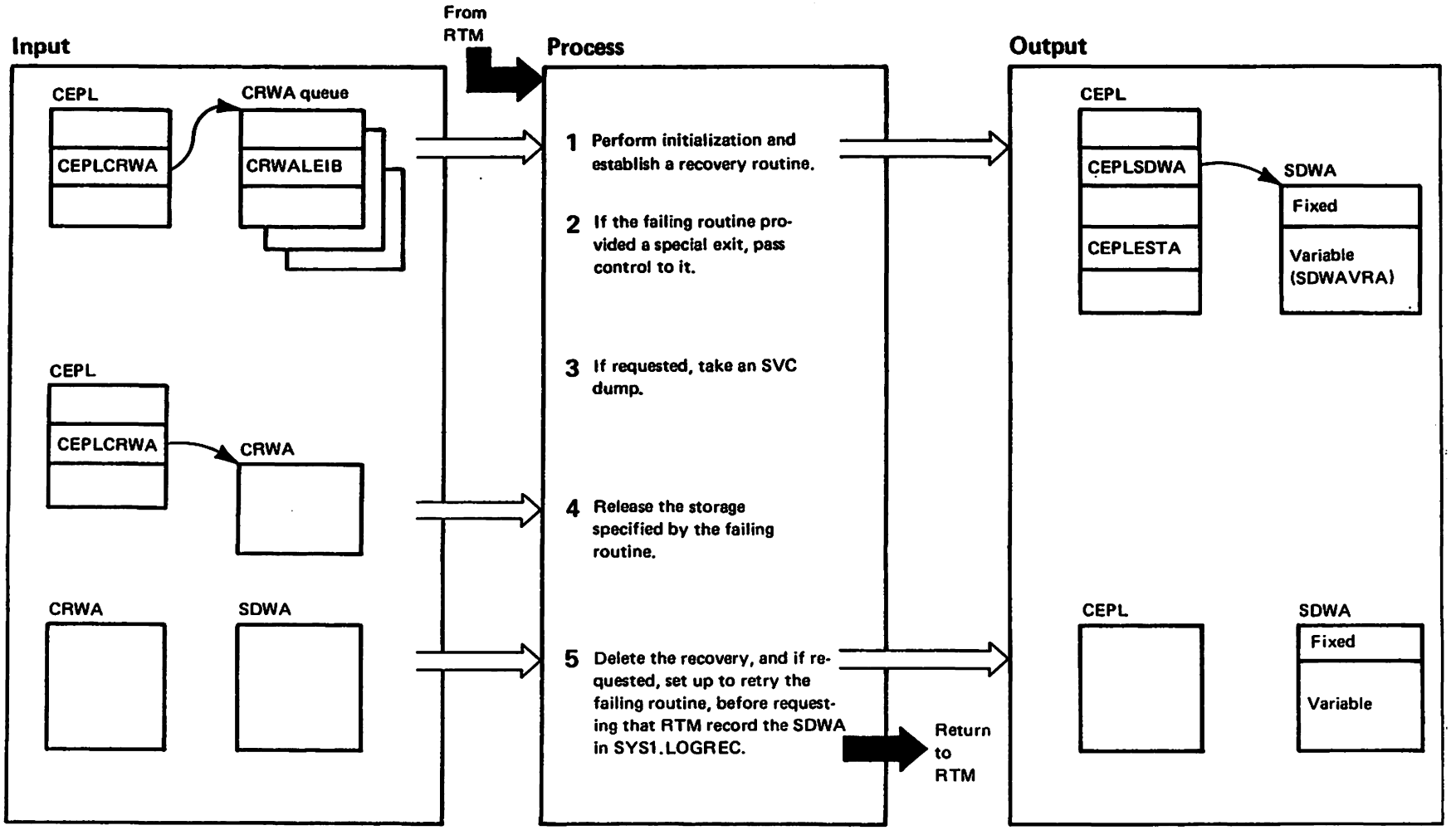
3 ISGCQSC indicates in the request's CRB that the quiesce request is complete and returns to ISGCMDR.

Entry Point ISGCQS02

4 The recovery routine (ISGCRCV) calls ISGCQSC at entry point ISGCQS02 to do recovery processing. When entered here, ISGCQSC issues message ISG015I to indicate the function that caused the error and the reason for the error. ISGCQSC indicates in the CRB that quiesce processing is complete and if the failure was caused by an error in ISGBCI, this module records the RSC in the SDWA. ISGCQSC sets a recovery processing return code (0=recovery processing successful and 4=unsuccessful) and returns to the caller.

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-21. ISGCRCV – Global Resource Serialization Command Recovery (Part 1 of 2)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-21. ISGCRCV – Global Resource Serialization Command Recovery (Part 2 of 2)

Extended Description | **Module** | **Label**

ISGCRCV is the ESTAE/I routine used by the following global resource serialization command processing and initialization routines:

- ISGCDSP – DISPLAY GRS
- ISGCMDI – Command interface
- ISGCMDR – Command router
- ISGCPRG – VARY GRS PURGE
- ISGCQMRG – Queue merge
- ISGCQSC – VARY GRS QUIESCE
- ISGCRST – VARY GRS RESTART
- ISGMSG00 – Message processing
- ISGNASIM – Address space initialization
- ISGNGRSP – GRS=Option processing

ISGCRCV performs SYS1.LOGREC recording, takes SVC dumps, routes control to special exit routines, and releases storage for the failing module. This module then indicates to RTM whether a retry should be attempted or termination continued.

1 If an SDWA is available, ISGCRCV copies information from the command recovery work area (CRWA) into the variable area of the SDWA (SDWAVRA). ISGCRCV obtains storage from subpool 229 for ESTAE and dump parameter lists, and for information about storage to be released. This module establishes a recovery routine to protect against an error occurring during special exit processing or within itself. If a recovery routine cannot be established, ISGCRCV indicates in the CEPL (CEPLESTA=0) that special exit processing should not be invoked. ISGCRCV then copies the CRWALEIB subfield of each CRWA processed into SDWAVRA. When recorded in SYS1.LOGREC, SDWAVRA will contain a trace of this recovery processing.

2 If the failing routine has a special recovery exit, before passing the exit control, ISGCRCV determines if a dump is also requested and if so invokes an SVC dump. If an SDWA is available (CEPLSDWA=1), then ISGCRCV validates the GVT, GVTX, and CRB/MRB addresses. If an SDWA is not available this module assumes these addresses to be invalid and then passes control to the special recovery exit of the failing routine. Upon return ISGCRCV indicates whether the exit was successful or not in CRWASERR.

Extended Description | **Module** | **Label**

3 If a dump was requested (CRWADMP=1) and one has not already been taken, as in step 2, ISGCRCV invokes SVC dump.

4 ISGCRCV releases all storage specified in the CRWA. CRWASTRG contains descriptions of storage ranges. If the starting address and length fields for a range are not zero, ISGCRCV issus a FREEMAIN for that range.

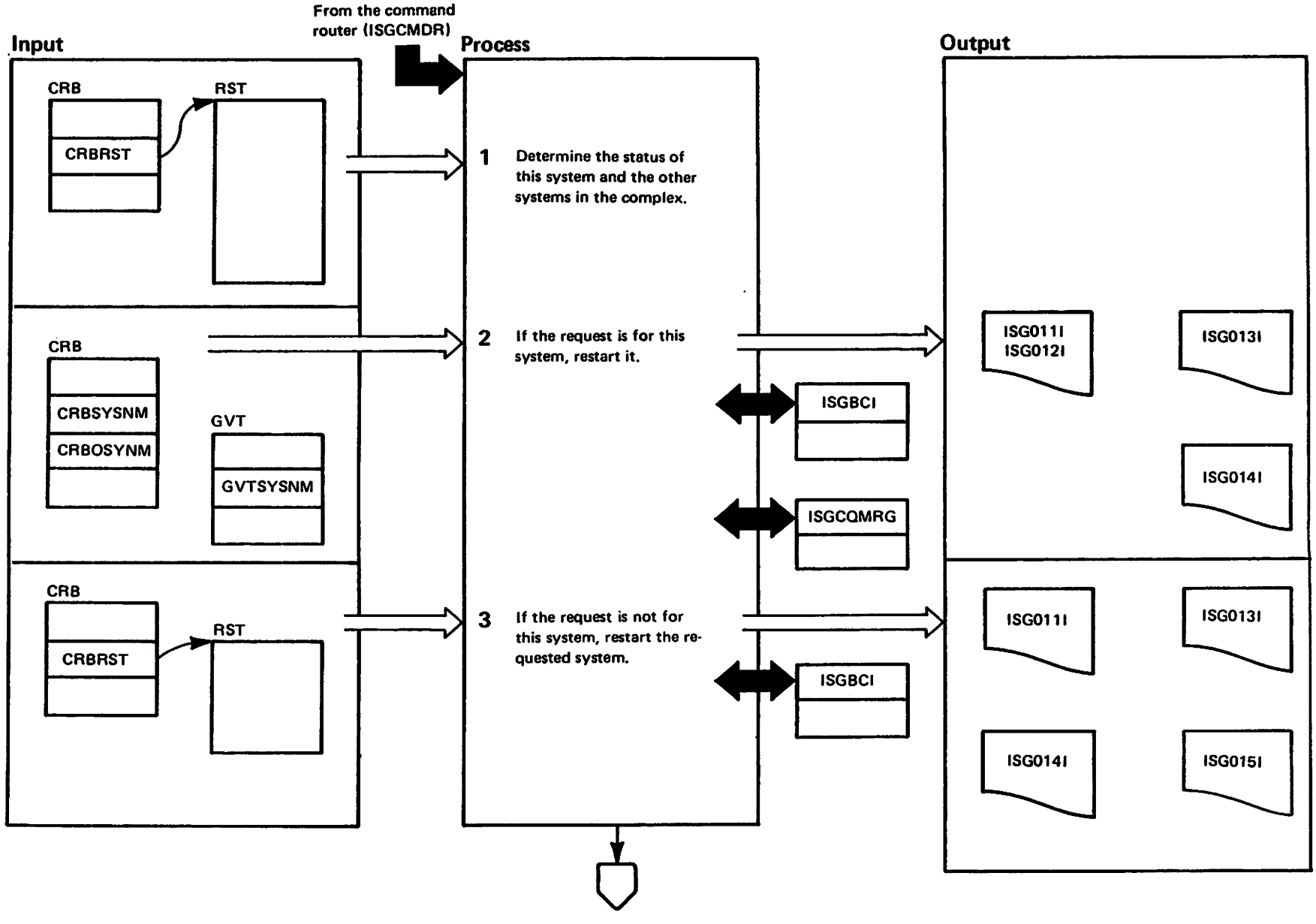
5 ISGCRCV deletes its recovery environment and sets up to retry the failing routine if requested (CEPLRTRY=1). If this is a recursion (CRWART2=1), then this module does not allow a retry. ISGCRCV issues a SETRP to request that RTM record the SDWA in SYS1.LOGREC and retry if either is requested.

Recovery Processing

ISGCRCV establishes recovery to provide re-entry into itself if a failure occurs in a called routine.

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-22. ISGCRST – Global Resource Serialization VARY GRS RESTART Request Processor (Part 1 of 4)



"Restricted Materials of IBM"
Licensed Materials – Property of IBM

LY28-1595-0 (c) Copyright IBM Corp. 1987

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-22. ISGCRST – Global Resource Serialization VARY GRS RESTART Request Processor (Part 2 of 4)

Extended Description	Module	Label
ISGCRST processes the RESTART parameter of the VARY GRS command. The RESTART parameter performs the following functions: • Bring a new system into the global resource serialization complex. • Restart global resource serialization on a quiesced system. • Restart global resource serialization processing on one or more systems after a disruption in the complex. ISGCRST receives control from the command router (ISGCMDR) when a command request block (CRB) for a restart request is found on the global resource serialization command work queue. ISGCRST obtains the ring status by invoking ISGBCI which invokes ISGBRF (at entry point ISGBRFSN). **1** ISGCRST first determines if this system is an active global resource serialization system. If it is not active ISGCRST determines if the resource queues are up to date on this system, whether another system in the complex has the same name as this system, and whether this system is connected to more than one global resource serialization complex. ISGCRST checks the other systems in the complex for the same conditions and sets the appropriate internal indicators for all the tests just made, for use in later processing. **2** If the restart request is for this system, ISGCRST determines if the request was issued by this system or sent from another system in the complex. If the request originated on this system, an active global resource serialization system exists, and this system is quiesed or inactive, ISGCRST issues messages ISG011I and ISG012I indicating that this system is restarting global resource serialization and the restart request is being passed to another system. If the restart request did not originate on this system, an active global resource serialization system exists and this system is quiesced or inactive. ISGCRST issues message ISG011I indicating that this system is restarting global resource serialization. This module then calls ISGBCI which calls ISGBRF (at entry point ISGBRFNM) to pass the request to an active global resource serialization system to restart this system. ISGCRST calls the queue merge routine (ISGCQMRG) to merge the restarting system's global resource serialization queues with the global resource serialization queues of the other active systems. ISGCRST then issues the restart completion message (ISG031I) on this system and broadcasts the same message to the other active systems in the complex.	ISGBRF ISGCQMRG	ISGBRFNM

Extended Description	Module	Label
If the request is for this system and this system is already active, ISGCRST issues message ISG014I indicating that an active system cannot be restarted and rejects the request. If this system is inactive and no active global resource serialization system exists, ISGCRST determines if the global resource queues of this system are accurate and calls ISGBCI to restart this system and on return issues message ISG013I on this system. If the global resource queues are not accurate, ISGCRST issues message ISG014I indicating that the queues are damaged and rejects the request. **3** If the restart request is for another system, ISGCRST determines if this system is an active global resource serialization system, and if so looks for the system specified on the request in the global resource serialization complex. If the specified system is not part of the complex, ISGCRST issues message ISG014I indicating that the specified system could not be found. If the specified system is found, ISGCRST checks the indicators set in step 1 to determine if the specified system is restartable. If not, this module issues message ISG014I indicating why the specified system cannot be restarted, and rejects the request. If it is restartable and inactive, ISGCRST issues message ISG011I indicating that the specified system is being restarted. If the specified system did not request the restart, ISGCRST calls ISGBCI to request the specified system to send back a restart request. ISGCRST calls ISGBCI again to add the specified system to the global resource serialization ring. This module then issues a GQSCAN macro to get information about this system's global resources and then calls ISGBCI which calls ISGBRF (at entry point ISGBRFNM) to send the information to the specified system. When the specified system completes restart processing, ISGCRST issues message ISG013I on this system and sends it to all other systems in the complex indicating that the specified system has restarted global resource serialization. If this system is inactive, it cannot process a restart request for another system and ISGCRST issues message ISG014I indicating such.	ISGBRF	ISGBRFNM

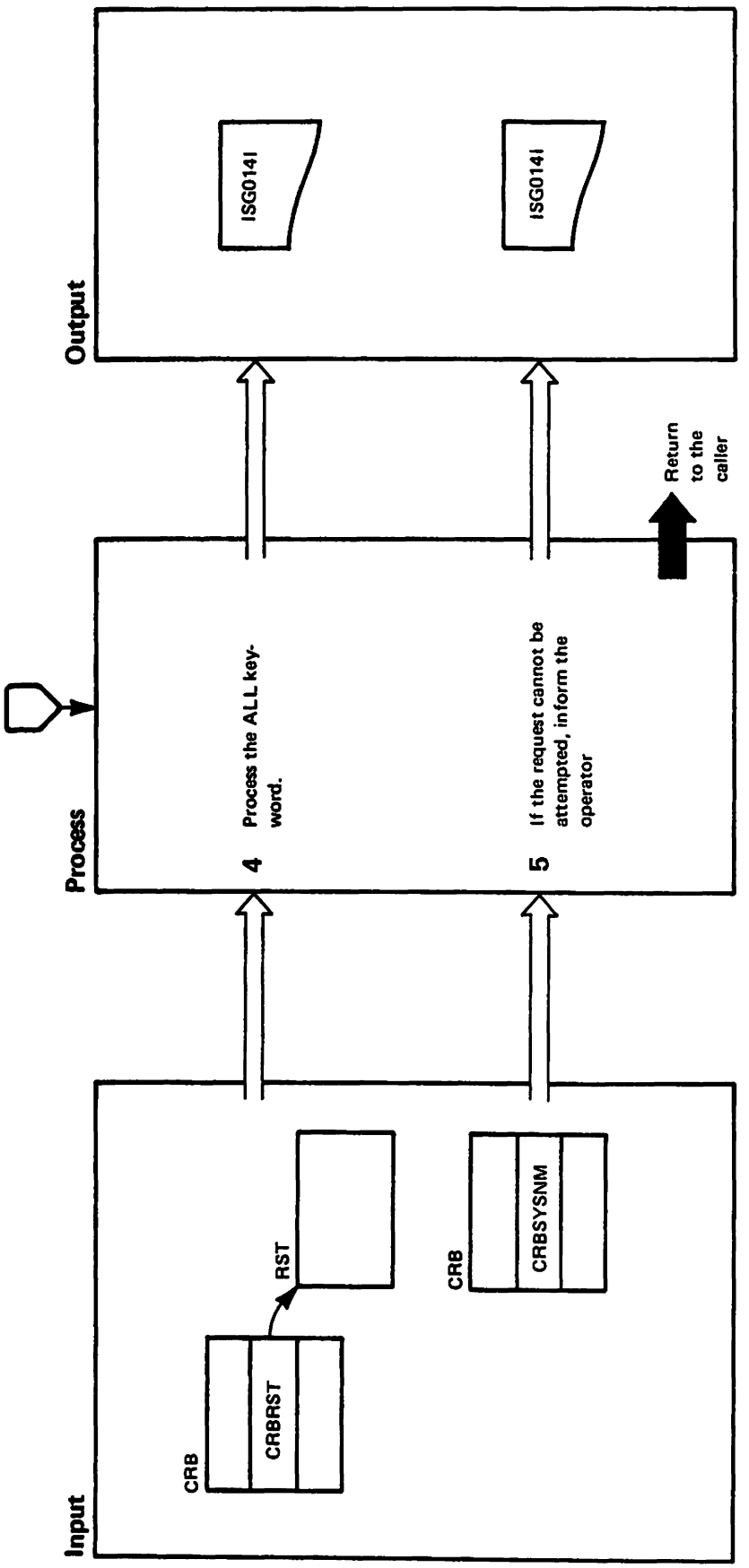


Diagram GRS-22. ISGCRST – Global Resource Serialization VARY GRS RESTART Request Processor (Part 3 of 4)

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-22. ISGCRST – Global Resource Serialization VARY GRS RESTART Request Processor (Part 4 of 4)

Extended Description **Module** **Label**

4 If VARY GRS (ALL) RESTART is specified by an active system and there are no restartable systems, ISGCRST issues message ISG014I indicating such and rejects the command. If VARY GRS (ALL) RESTART is specified from an inactive system and an active global resource serialization system exists, ISGCRST issues message ISG014I indicating such, and rejects the command. If VARY GRS (ALL) RESTART is specified on an inactive global resource serialization system and other restartable inactive systems exist, ISGCRST restarts this system using the same processing described in step 2.

After this system has restarted global resource serialization successfully, ISGCRST performs the processing described in step 3 for each restartable system.

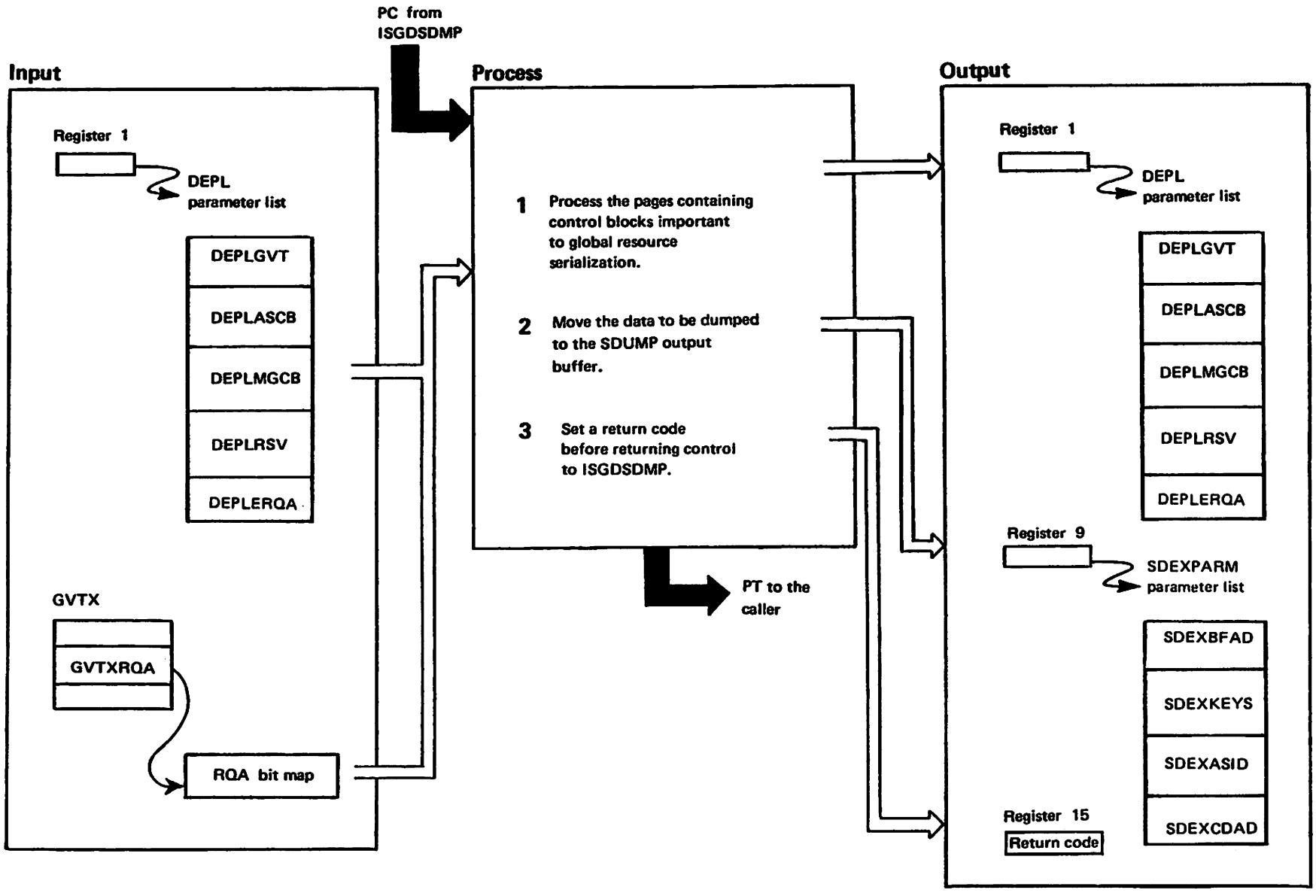
5 If multiple global resource serialization complexes exist, two or more systems share the same system name, or the global resource serialization queues are damaged, ISGCRST issues message ISG014I indicating one of the above and rejects the command.

Recovery Processing

ISGCRCV handles recovery processing for this module. ISGCRCV calls ISGCRST at entry point ISGCRS02 if an error occurs while restarting another system. At this entry point ISGCRST removes partially restarted systems from the global resource serialization ring and releases serialization.

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-23. ISGDGCB0 – Global Resource Serialization Dump Control Blocks Exit Routine (Part 1 of 2)



"Restricted Materials of IBM"
Licensed Materials – Property of IBM

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-23. ISGDGCB0 -- Global Resource Serialization Dump Control Blocks Exit Routine (Part 2 of 2)

Extended Description	Module	Label

ISGDGCB0 receives control via a program call instruction from ISGDSDMP. Its purpose is to move from the global resource serialization address space to the SDUMP output buffer, the pages containing the following:

- GVT (global vector table)
- ASCB (global resource serialization ASCB)
- GVTX (global vector table extension)
- GQHT (global queue hash table)
- LQHT (global queue hash table)
- GRPT (global resource pool table)
- LRPT (local resource pool table)
- SAHT (system/ASID hash table)
- RSV (ring-processing system vector table)
- RSV entries
- The active global resource serialization ERQA pages for PQCBs, QCBs, QELs, and QXBs

Extended Description	Module	Label

1 ISGDGCB0 processes only a page of data at one time. Control blocks which occupy more than one page of storage, or span pages, are processed in multiple invocations of ISGDGCB0. ISGDGCB0 turns on flags in the SDUMP ESTAE parameter list (DEPL) to indicate that processing is to be initiated for each of the items listed in the introduction. When ISGDGCB0 is processing a page in the resource queue area (RQA) or extended resource queue area (ERQA), ISGDGCB0 checks the corresponding bit in the bit map to determine if the page is allocated (the bit is on). If so, then ISGDGCB0 determines if the page is from the ERQA containing QCBs, QELs, QXBs, or PQCBs and dumps only those pages.

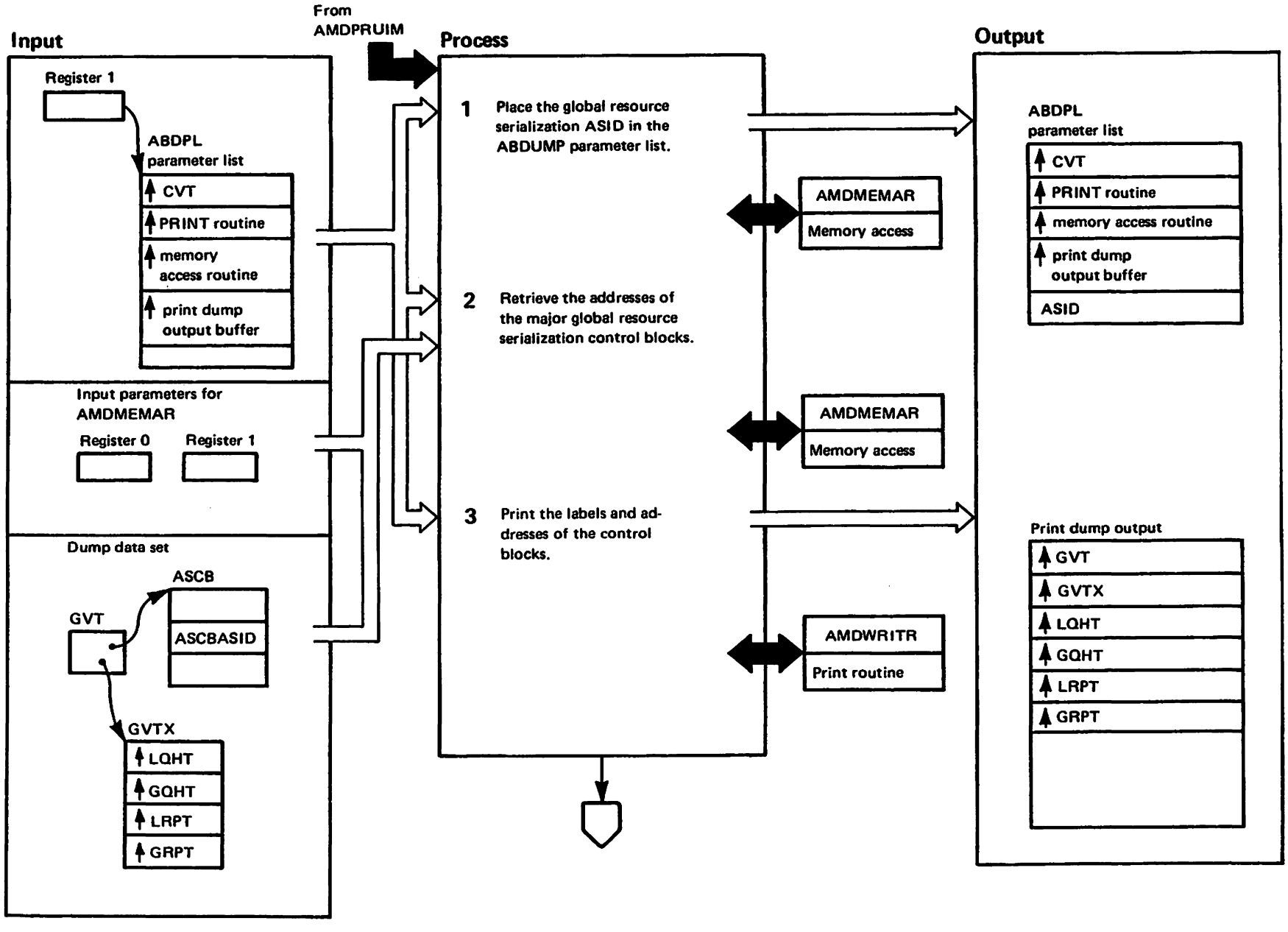
2 ISGDGCB0 moves one page of data to the SDUMP output buffer.

3 ISGDGCB0 returns control to ISGDSDMP after setting one of the following return codes:

Return Code	Reason
0	Dump is complete, return to the caller
4	Write data to the dump data set and return to the caller
8	Write data to the dump data set and return to ISGDGCB0 to dump more data.

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-24. ISGDPDMP – Global Resource Serialization Print Dump Exit Routine (Part 1 of 4)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

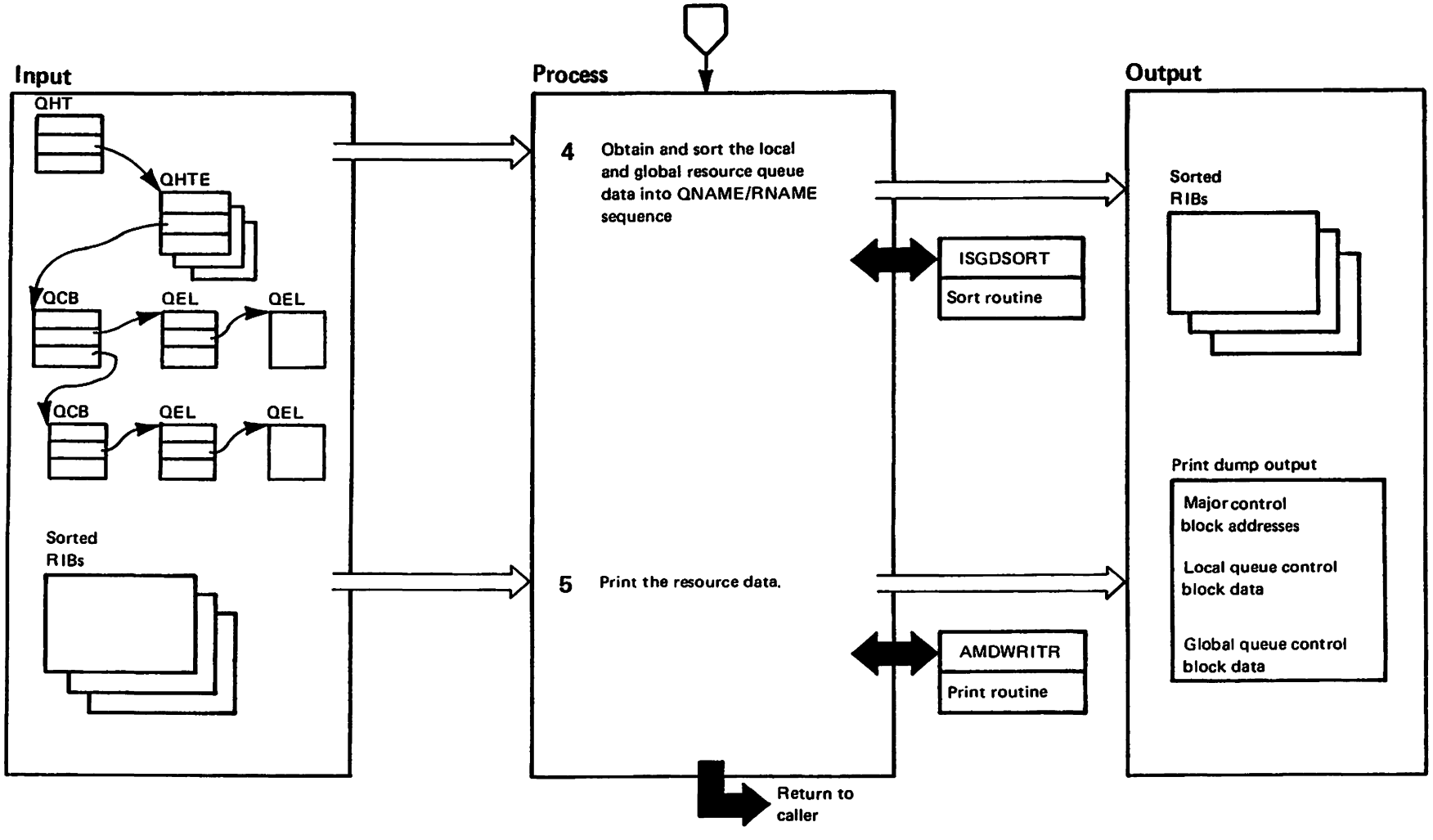
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-24. ISGDPDMP – Global Resource Serialization Print Dump Exit Routine (Part 2 of 4)

Extended Description	Module	Label
When global resource serialization control block information located in the dump data set needs to be formatted and printed, AMPRUIM branches to ISGDPDMP to do this.		
1 ISGDPDMP calls the memory access routine (AMDMEMAR) to search the global resource serialization address space contained in the dump data set to obtain and return the address of the areas where the required information can be found. AMDMEMAR requires two parameters: register 0 contains the virtual address to be referenced and register 1 contains the address of the ABDUMP parameter list passed to ISGDPDMP from AMDPRUIM.	AMDMEMAR	
ISGDPDMP obtains the virtual address of the gloabl resource serialization vector table (GVT) from the CVT and passes the address to AMDMEMAR in register 0. If the GVT address is accessible in the dump data set, then AMDMEMAR returns to ISGDPDMP an address in register 0 where it can be located. ISGDPDMP obtains the GVT address, converts it to printable hex, and stores it in a buffer to be printed. ISGDPDMP updates the ASID field in the ABDUMP parameter list to the global resource serialization ASID, found in the global resource serialization ASCB pointed to by the GVT, to notify AMDMEMAR from which address space to access data in the dump data set.	AMDMEMAR	
2 ISGDPDMP then obtains the virtual address of the global resource serialization vector table extension control block (GVTX) located in the GVT, and passes it to AMDMEMAR. If the GVTX address is accessible, then ISGDPDMP converts the address to printable hex and stores it into a buffer to be printed. Next the virtual addresses of the resource serialization storage management control blocks located in the GVTX are passed to AMDMEMAR. If they are accessible, then ISGDPDMP also converts these addresses to printable hex and stores them into a buffer to be printed.	AMDMEMAR	
3 ISGDPDMP calls the print service routine (AMDWRITR) with the address of the ABDUMP parameter list (ABDPL) in register 1. The ABDPL contains the address of the buffer needed to print the following control block labels and the corresponding addresses in the beginning of the dump. • GVT • GVTX • LQHT (local queue hash table) • GQHT (global queue hash table) • LRPT (local resource pool table) • GRPT (global resource pool table)	AMDWRITR	

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-24. ISGDPDMP – Global Resource Serialization Print Dump Exit Routine (Part 3 of 4)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

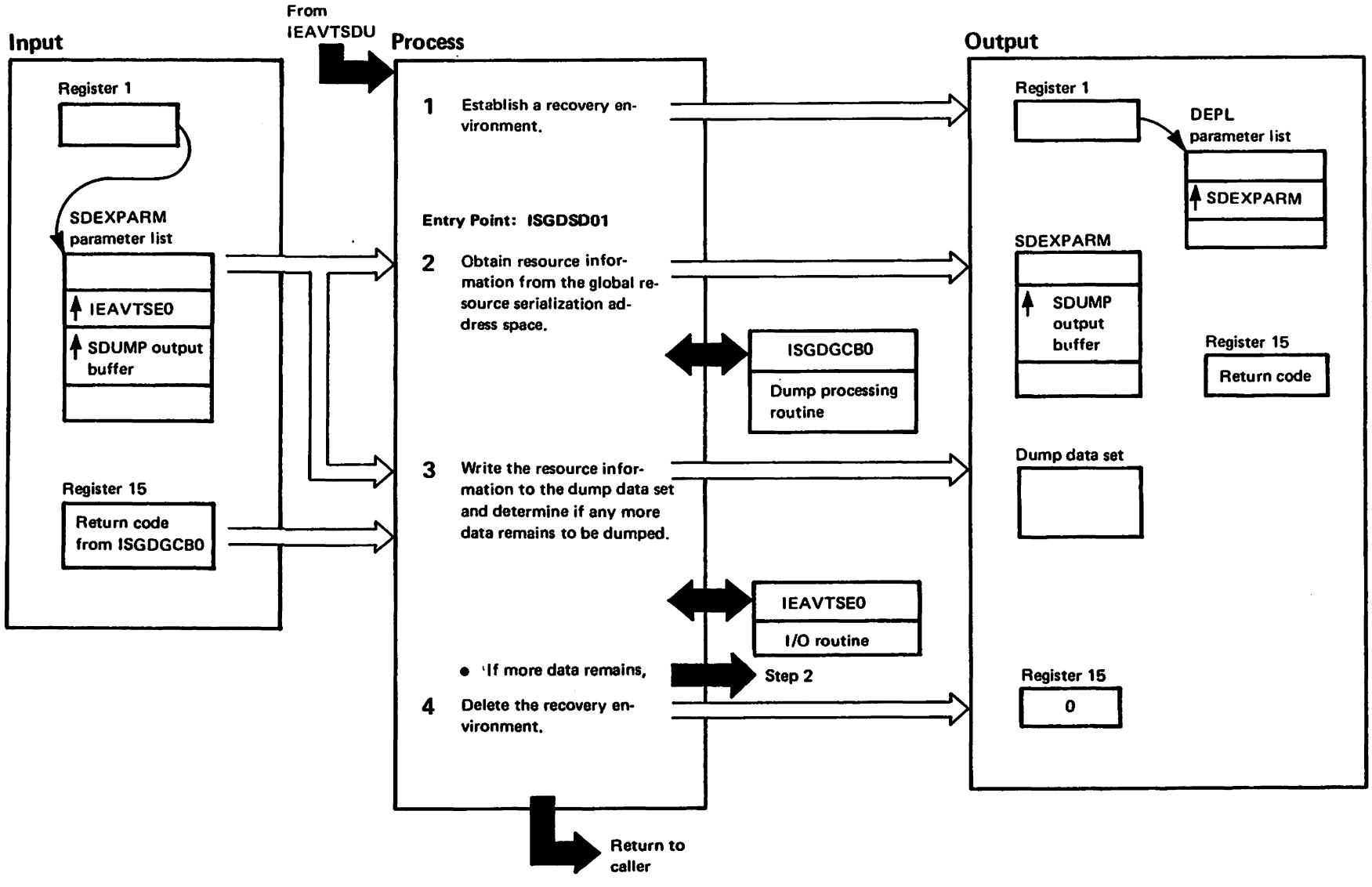
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-24. ISGDPDMP – Global Resource Serialization Print Dump Exit Routine (Part 4 of 4)

Extended Description	Module	Label
4 ISGDPDMP obtains and sorts the local and then the global resource queue data in the following manner. ISGDPDMP obtains information about each resource described by a queue control block (QCB) and stores it into alphabetical order by resource name. For each QCB on the QCB synonym chains pointed to by the local and global queue hash table entries, ISGDPDMP performs the following: • Builds a resource information block (RIB) • Scans the QEL chain pointed to by a QCB for data about the requestors of the resource • Stores some of the information found in the QCB and QEL into the RIB	ISGDPDMP	
When all the synonym chains have been processed, ISGDPDMP calls the global resource serialization dump sort routine (ISGDSORT) to sort the RIBs into alphabetical order using the resource name (QNAME/RNAME) as the sort argument.	ISGDSORT	
5 ISGDPDMP calls the print service routine (AMDWRITR) to print the information about each resource following the control block labels and addresses. For each resource, ISGDPDMP scans the QEL chain saved in the RIB and prints information for each requestor.	AMDWRITR	

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-25. ISGDSDMP – Global Resource Serialization SVC Dump Exit Routine (Part 1 of 2)



 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-25. ISGDSDMP – Global Resource Serialization SVC Dump Exit Routine (Part 2 of 2)

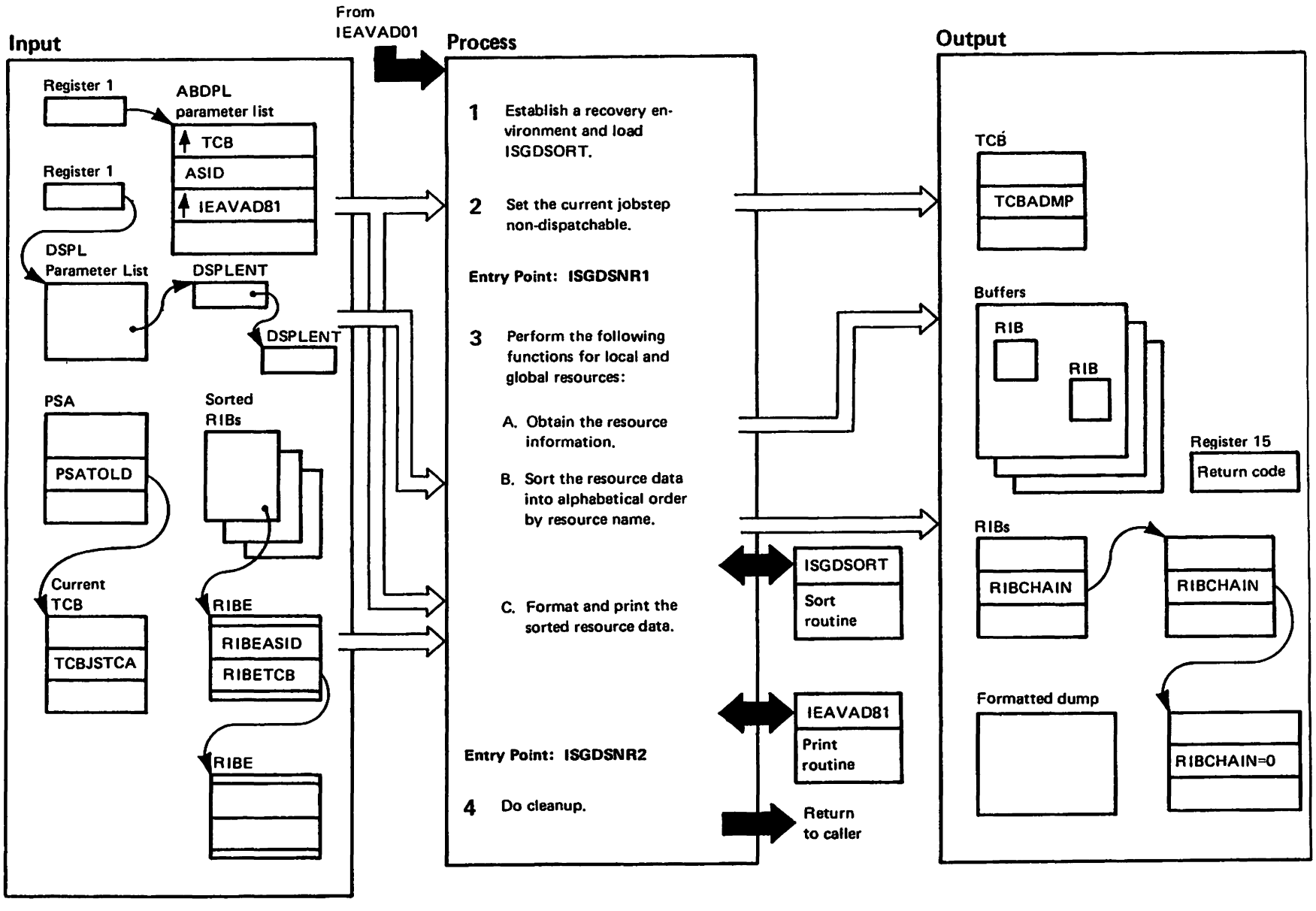
Extended Description	Module	Label
IEAVTSDU passes control to ISGDSDMP to write those pages containing important global resource serialization control blocks to the dump data set. ISGDSDMP is called by IEAVTSDU in any address space. When ISGDSDMP is called, it is enabled, but the system is set nondispatchable. At entry, register 1 contains the address of the SVC dump exit parameter list (SDEXPARM). SDEXPARM contains a 200-byte workarea for ISGDSDMP to use.		
1 ISGDSDMP puts the address of SDEXPARM into the SDUMP ESTAE parameter list (DEPL) and issues an ESTAE macro establishing ISGDSDRV as the recovery routine for ISGDSDMP and ISGDGCB0. (See "Recovery Processing" for a description of ISGDSDRV).	ISGDSDMP	
2 ISGDSDMP issues a program call to ISGDGCB0 which resides in the global resource serialization address space. ISGDSDMP passes ISGDGCB0 the address of the SDUMP ESTAE parameter list in register 1. Only one page of data can be processed at a time. ISGDGCB0 updates SDEXPARM with the required data concerning the page to be dumped, then moves the page to the SDUMP output buffer area. ISGDGCB0 returns control to ISGDSDMP via a program transfer instruction after setting one of the following return codes.	ISGDGCB0	

Return code	*Action to be taken*
0	Dump complete, return to the caller.
4	Write a page to the dump data set, then return to the caller.
8	Write a page to the dump data set, then return to ISGDGCB0 to process more data.

Extended Description	Module	Label
3 If ISGDGCB0 returned a nonzero return code indicating that there is data to be written, ISGDSDMP calls IEAVTSE0 to write a page of data to the dump data set. If the return code is eight, there is more data to be dumped. ISGDSDMP repeats the process beginning at step 2; otherwise, all the data has been dumped and processing continues at the next step.	IEAVTSE0	
4 ISGDSDMP issues an ESTAE macro to delete the recovery environment and sets a zero return code to indicate successful processing.		
Recovery Processing:		
When an error occurs while either ISGDSDMP or ISGDGCB0 is executing, RTM calls ISGDSDRV to record the recovery information in the SDWA and record the SDUMP ESTAE parameter list in the SDWA's variable recording area (VRA). ISGDSDRV issues a SETRP macro to indicate to RTM to retry at entry point ISGDSD01 (step 2). Twenty-two retries are allowed: • One retry for a failure while processing the major global resource serialization control blocks (GVTX, LQHT, LPRT, GQHT, GRPT, SAHT) • One retry for a failure while processing the ring status vector table • Twenty retries for failures while processing the resource queue area.	ISGDSDRV	
No retries are allowed if a failure occurs while processing the GVT or the global resource serialization ASCB. If a retry is not allowed, RTM is notified to continue with termination.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-26. ISGDSNAP – Global Resource Serialization SNAP Dump Exit Routine (Part 1 of 2)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

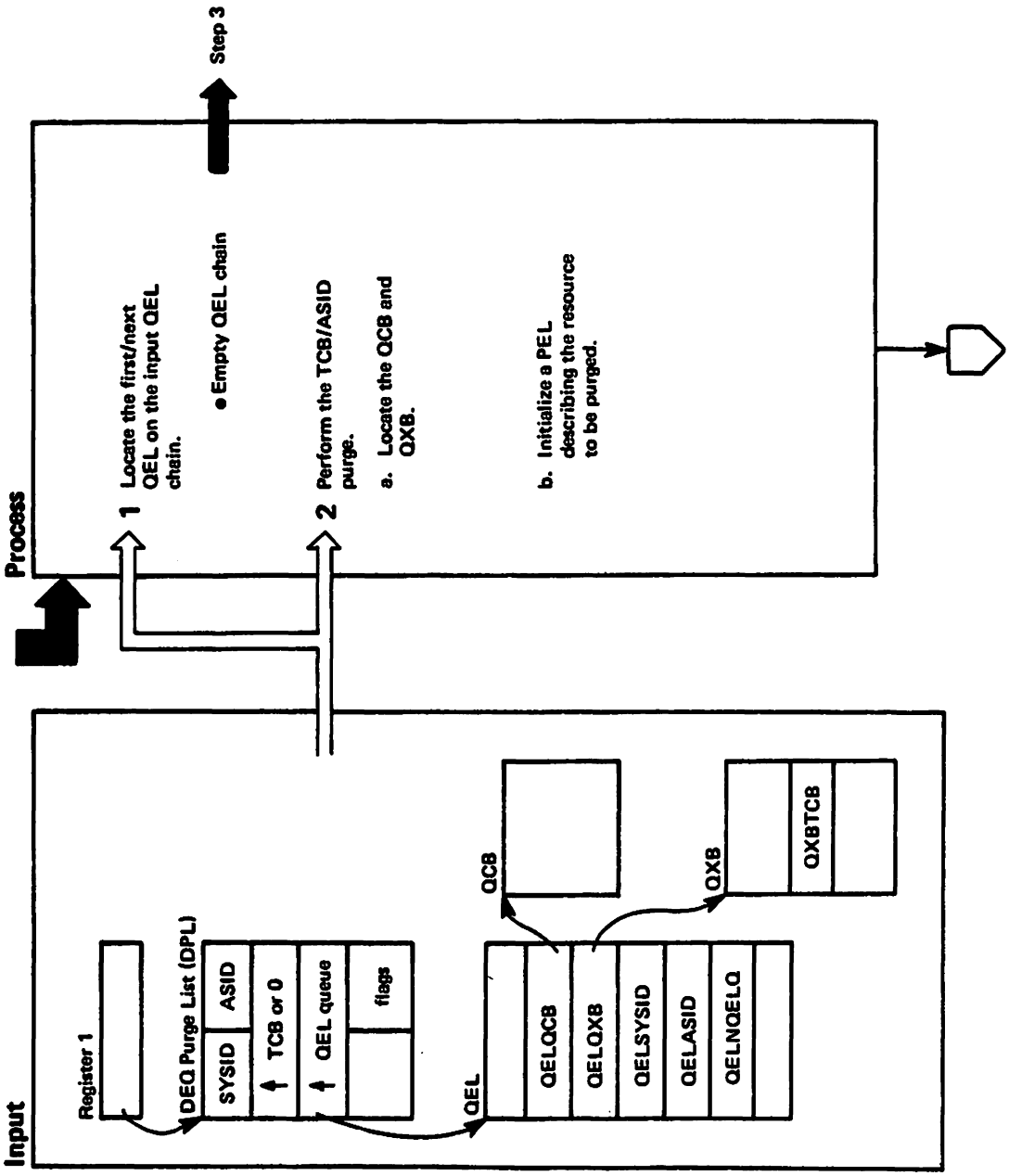
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-26. ISGDSNAP – Global Resource Serialization SNAP Dump Exit Routine (Part 2 of 2)

Extended Description	Module	Label
When it is necessary to format and print information about the resources associated with all the tasks in the current jobstep, IEAVAD01 branches to ISGDSNAP.		
1 ISGDSNAP issues an ESTAE macro to establish ISGDSNRV as the recovery routine. (See "Recovery Processing" for a description of ISGDSNRV.)	ISGDSNAP	
2 If the dump is for the current task, ISGDSNAP sets the TCBADMP bit in the TCB to indicate that the current task is making the jobstep non-dispatchable. ISGDSNAP then issues the STATUS macro to set the jobstep non-dispatchable so that the resources owned by the jobstep will not be released during the dumping process.		
3 ISGDSNAP performs the following functions for local and global resources:		
A. ISGNSNAP issues a GQSCAN macro to obtain from the global resource serialization address space resource and requestor information associated with the ASID specified in the ABDUMP parameter list. If no buffer exists for the GQSCAN output, ISGDSNAP obtains one. The GQSCAN service routine (ISGQSCAN) stores the data in resource information blocks (RIBs) and resource information extension blocks (RIBEs) and moves the requested information into the buffer in ISGDSNAP's address space. When ISGQSCAN returns control, ISGDSNAP checks the return code. If the return code is eight, there is more data to be accessed. ISGDSNAP obtains another buffer and invokes ISGQSCAN again. If the return code is zero, all the information has been obtained, and processing continues at 3B.	ISGQSCAN	
B. ISGDSNAP calls ISGDSORT to sort the resource information contained in the buffers. Before calling ISGDSORT, ISGDSNAP obtains and initializes the ISGDSORT parameter list. It initializes the entry section of the parameter list with buffer information such as the address of the first buffer to be sorted, the number of RIBs contained in the buffer, and a pointer to the next buffer to be processed. ISGDSNAP then calls ISGDSORT to sort the resources into alphabetical order by resource name (QNAME/RNAME).	ISGDSORT	
C. ISGDSNAP formats and prints the resource information for the tasks in the current jobstep. The RIBs/RIBEs contain information about the resources associated with the ASID in the ABDUMP parameter list. To format and print the resource information associated with the tasks in the current jobstep, ISGDSNAP searches for the requestor by comparing the RIBEASID with the ASID specified in the ABDUMP parameter list and comparing the TCB jobstep (TCBJSTCA) pointed to by the RIBETCB with the current TCB jobstep. If a match is found, then ISGDSNAP calls IEAVAD81 to print the resource information.	IEAVAD81	
4 ISGDSNAP performs the following cleanup functions: • Resets the TCB bit (TCBADMP) and issues the STATUS macro to reset the jobstep dispatchable • Releases any previously obtained storage • Deletes the sort routine ISGDSORT to remove the CDE entry and issues the ESTAE macro to delete the recovery routine (ISGDSNRV) • Returns to the caller with a zero return code		
Recovery Processing		
When an error occurs while ISGDSNAP is executing, RTM calls ISGDSNRV to record the recovery diagnostic information in the SDWA and to issue an SDUMP macro for the LSQA, which contains the buffers used to contain the resource information returned by the GQSCAN service routine, and the ISGDSORT parameter list. Unless a recursive error has occurred, ISGDSNRV attempts a retry. If global resources have not been processed, it retries at entry point ISGDSNR1 (step 3); otherwise, it retries at entry point ISGDSNR2 (step 4) to perform cleanup processing. If a retry is not allowed, ISGDSNRV resets the jobstep dispatchable and returns control to RTM.	ISGDSNRV	

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-27. ISGGDEQP – TCB/ASID Purge (Part 1 of 6)



LY28-1695-0 (c) Copyright IBM Corp. 1987

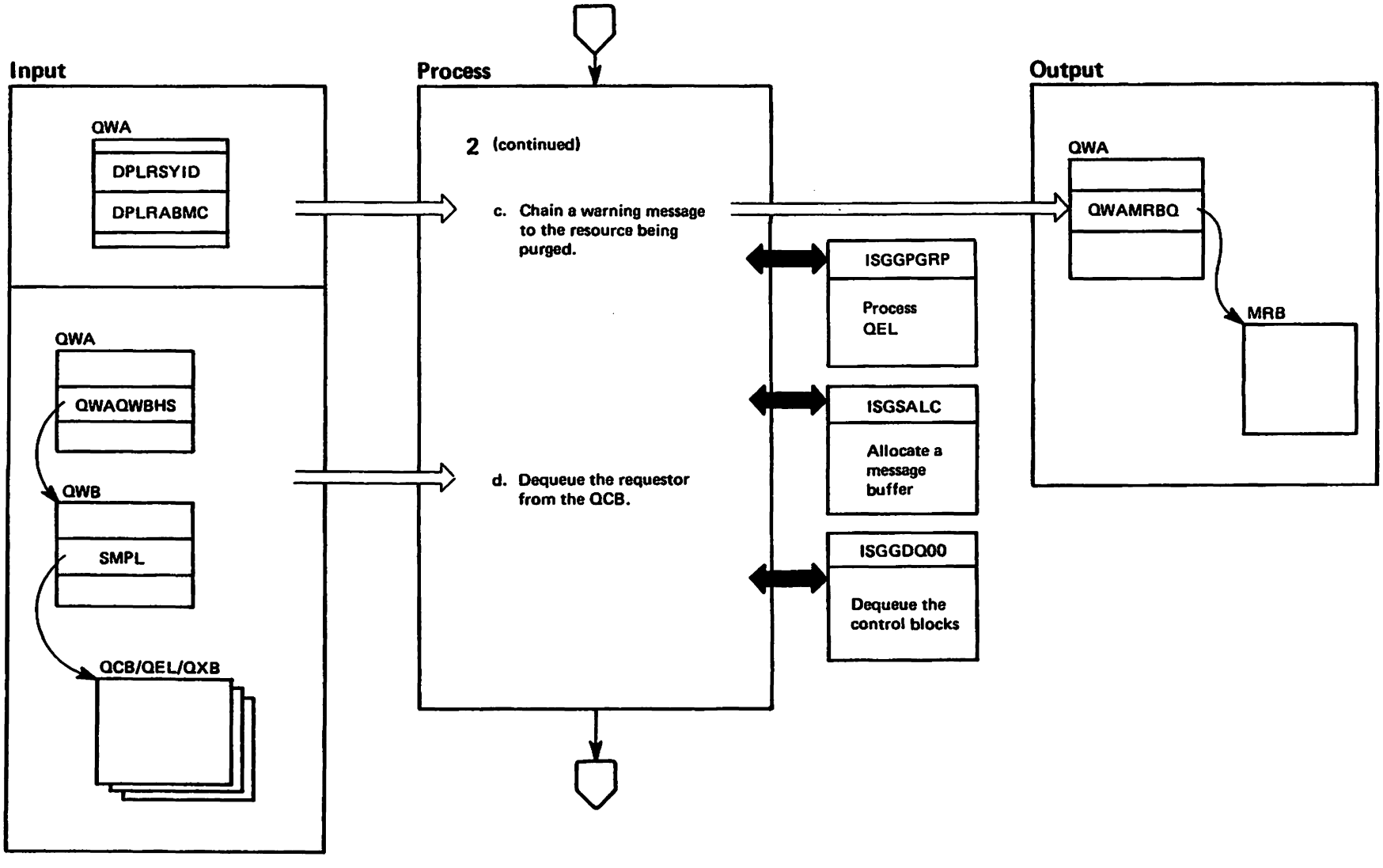
"Restricted Materials of IBM"
Licensed Materials – Property of IBM

Diagram GRS-27. ISGGDEQP – TCB/ASID Purge (Part 2 of 6)

Extended Description	Module	Label
ISGGDEQP purges the resources associated with a task or address space. These resources are defined on any of the following QEL queues. • ASCB global QEL queue • ASCB local QEL queue • SYSID/ASID QEL queue The input parameter list contains the type of purge requested (either TCB or ASID), the SYSID, TCB, and/or ASID to be purged, and one of the above QEL queues to be scanned.		
1 Search the QEL queue pointed to by the input parameter list in order to find the element to be purged. If the QEL queue is empty, continue at step 3.	ISGGDEQP	
2 If this is a TCB purge request, then purge only those QELs associated with the input TCB; otherwise purge all the QELs defined on the QEL queue pointed to by the input parameter list. a. Use the QEL to get addressability to the QCB and the QXB. b. Extract information from the QCB, QEL, and QXB. Initialize the PEL section of the queue work block (QWB) supplied as input. Use the SQA QWB PEL when purging a local queue. Use the input QWB PEL when purging a global queue.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-27. ISGGDEQP – TCB/ASID Purge (Part 3 of 6)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-27. ISGGDEQP – TCB/ASID Purge (Part 4 of 6)

Extended Description	Module	Label
2 (continued)		
c. Determine if purge messages should be issued. Invoke ISGSALC to allocate a message buffer (MRB) if purge messages should be issued. These messages reside in the MRB and are queued from the QWA and later processed by ISGGTRM1 for task or address space termination or processed by ISGCPRG for a SYSID purge command. If the caller has indicated that the requester was in "must complete" mode, owns the resource, and the resource request represents an exclusive request with scope of SYSTEM or SYSTEMS, then build an MRB for message ISG032E. For a SYSID purge request, build an MRB for each resource to be purged; each MRB is for message ISG018I. If the QEL is a MASID target (as determined by calling ISGGPGRP), ISGGDEQP builds an MRB for message ISG035E. Otherwise, continue with step 2d.	ISGSALC ISGGPGRP	
d. Invoke ISGGDQ00 to scan the QCB DEL chain for the SYSID, ASID, or TCB passed as input. When the input SYSID, ASID, or TCB is found, chain the control blocks to be freed from the storage manager parameter list (SMPL) and continue at step 1 until the QEL chain is empty.	ISGGNQDQ	DCURQEL

LY28-1695-0 (c) Copyright IBM Corp. 1987

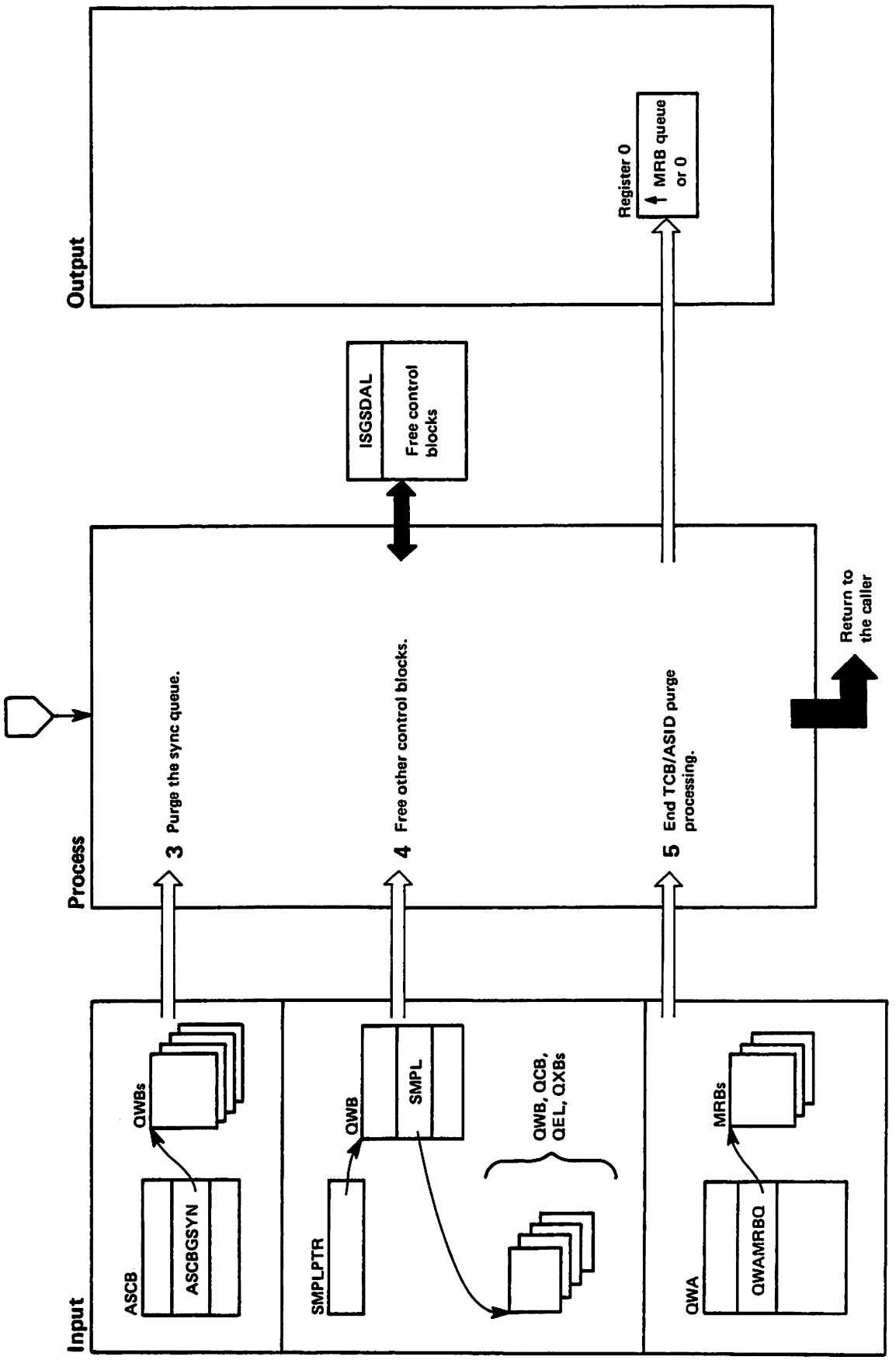


Diagram GRS-27. ISGGDEQP – TCB/ASID Purge (Part 5 of 6)

LY28-1695-0 (c) Copyright IBM Corp. 1937

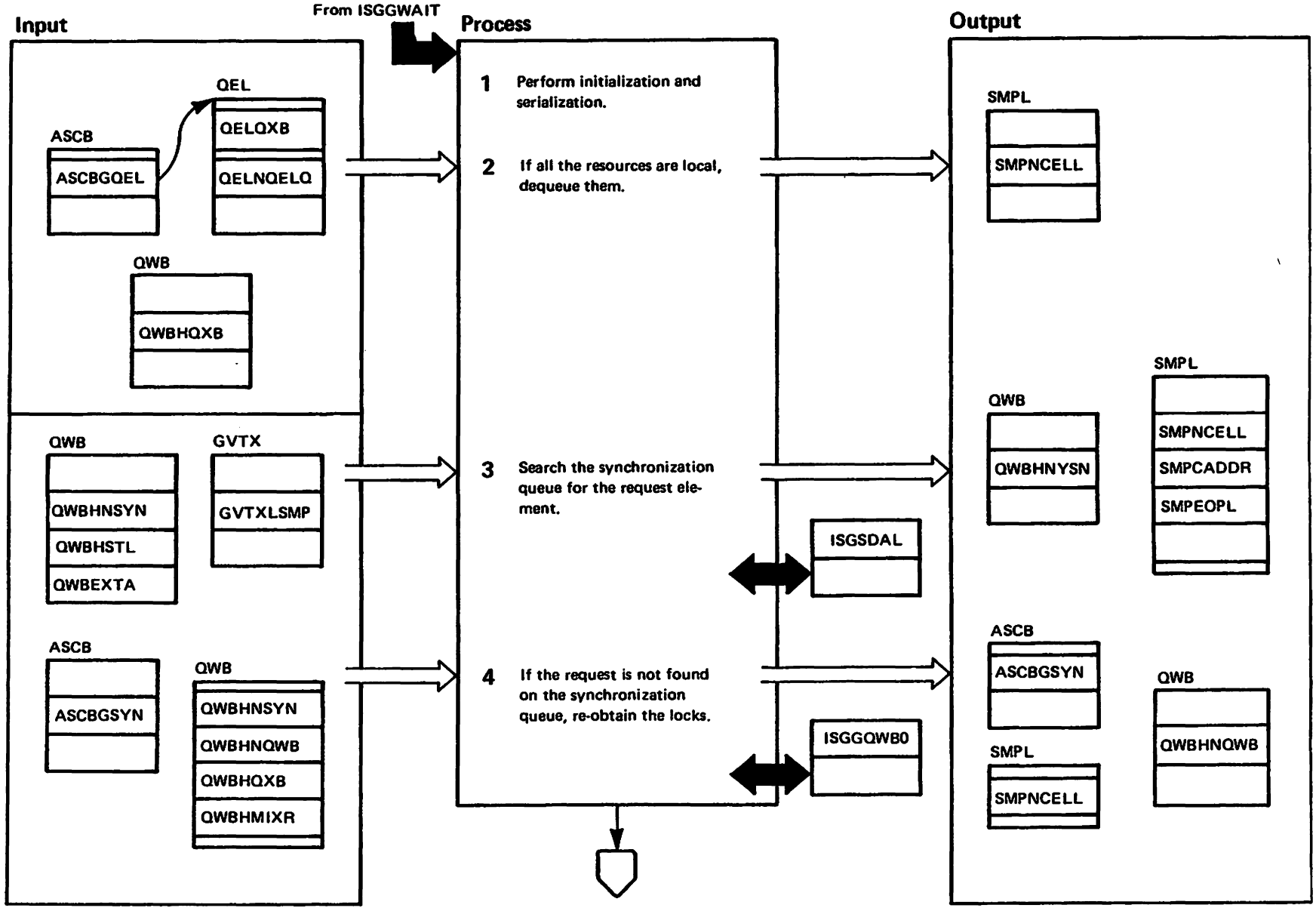
"Restricted Materials of IBM"
Licensed Materials – Property of IBM

Diagram GRS-27. ISGGDEQP – TCB/ASID Purge (Part 6 of 6)

Extended Description	Module	Label
3 The sync queue represents steal requesters awaiting sync ownership. The sync queue prevents ENQ requests "directed" to the failing task from running before the first request is processed. Entries normally exist on the sync queue only when the address space is abnormally terminating. When the address space is normally terminating or if a TCB purge is requested and the task is normally terminating, no entries exist for the task or address space. If the task is abnormally terminating, the ISGGNQDQ ESTAE routine, ISGGEST0, cleans up the QWB.	ISGGDEQP	
4 If control blocks have been placed in the SMPL, call the storage manager deallocation routine to free them, using the CMS ENQ/DEQ class lock for serialization. The caller has previously obtained the global resource serialization local lock if global resources were to be purged.	ISGSDAL	
5 Return to the caller with register 0 containing the address of the MRB queue or zero. The purge is complete.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-28. ISGGEST0 – Global Resource Serialization ENQ/DEQ/RESERVE Mainline ESTAE Routine (Part 1 of 4)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-28. ISGGEST0 – Global Resource Serialization ENQ/DEQ/RESERVE Mainline ESTAE Routine (Part 2 of 4)

Extended Description	Module	Label
ISGGEST0 recovers from errors that occur while the ENQ/DEQ/RESERVE mainline processing is waiting for a request to be processed. This module does not establish a recovery environment; if an error occurs, the task termination manager cleans up.		
1 ISGGEST0 issues a PCLINK macro to save linkage information, issues the SETLOCK macro to obtain the requestor's local and CMS locks and calls ISGSALC to obtaing a work area.	ISGSALC	
2 ISGGEST0 determines if the resources are local and if so dequeues them. To do this ISGGEST0 scans the ASCB local QEL queue and dequeues any resource request identified with a QXB in the RB extended save area. ISGGEST0 calls ISGGQWB0 at ISGGQWB4 to build a DEQ request and the calls ISGGNQDQ at ISGGDQ00 to perform the DEQ.	ISGGQWB0 ISGGNQDQ	ISGGQWB4 ISGGDQ00
3 ISGGEST0 searches the synchronization queue to determine if the request has been processed yet. If found and it is not the top request on the synchronization queue, this module calls ISGSDAL to free the QWB associated with the synchronization QWB.	ISGSDAL	
4 If the request is not found or is found at the top of the synchronization queue, ISGGEST0 calls ISGGQWB0 at ISGGQWB5 to build a SYNC QWB to make sure that all outstanding requests issued by this task have been processed. Upon return ISGGEST0 obtains the local and CMS locks. If the request is at the top of the synchronization queue, ISGGEST0 frees the synchronization QWB and all the QWBs related to the request. If the request was not found on the synchronization queue, ISGGEST0 dequeues all outstanding global and local resources. ISGGEST0 decreases the task global resource count (TCBGRES) by either the number of global ENQ requests or DEQ requests that ISGGRP00 processed.	ISGGQWB0	ISGGQWB5

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

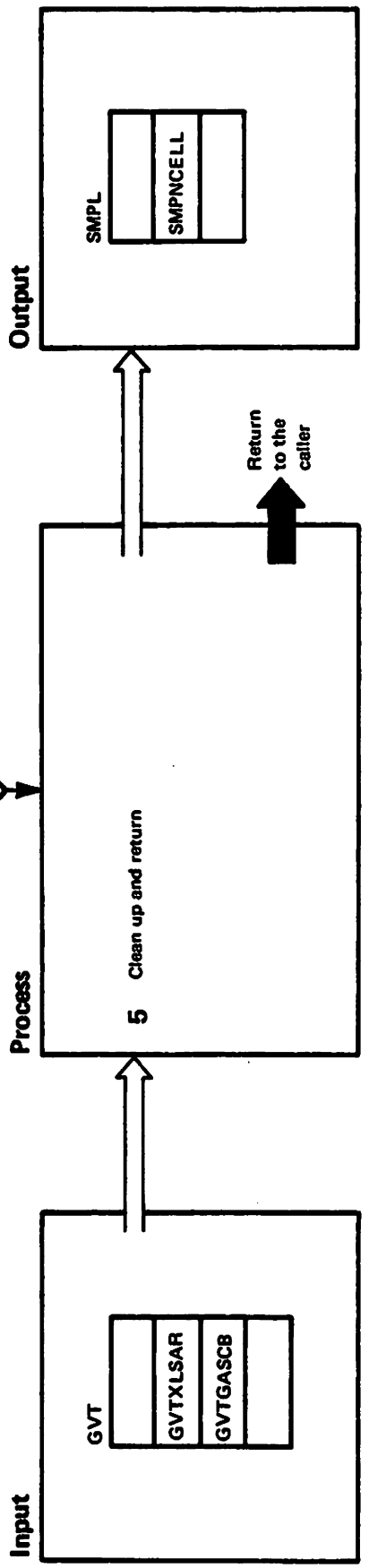


Diagram GRS-28. ISGGEST0 – Global Resource Serialization ENQ/DEQ/RESERVE Mainline ESTAE Routine (Part 3 of 4)

LY28-1695-0 (c) Copyright IBM Corp. 1987

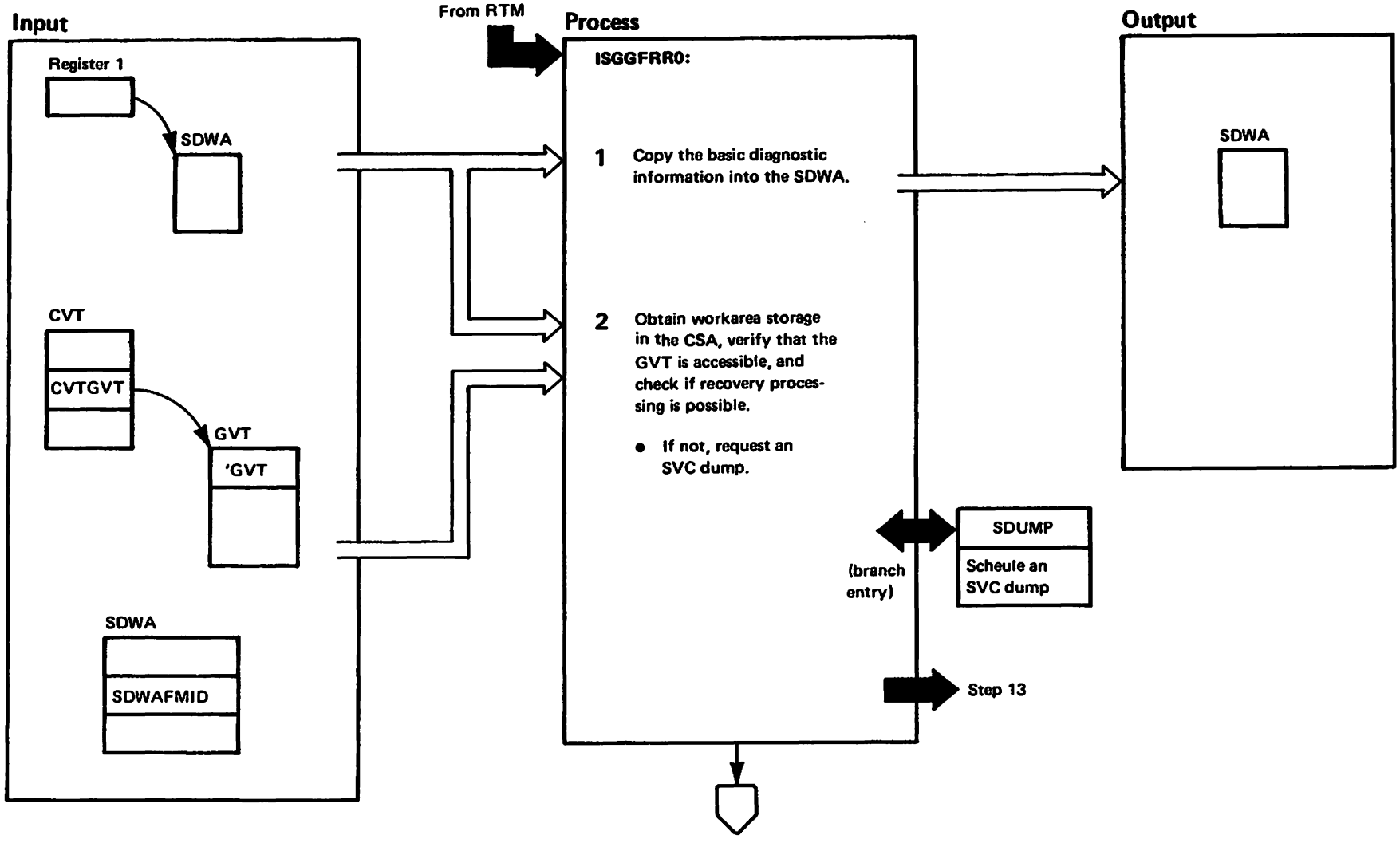
"Restricted Materials of IBM"
Licensed Materials -- Property of IBM

Diagram GRS-28. ISGGEST0 – Global Resource Serialization ENQ/DEQ/RESERVE Mainline ESTAE Routine (Part 4 of 4)

Extended Description	Module	Label
5 ISGGEST0 restores the linkage information, releases the locks it obtained and returns to the caller.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-29. ISGGFRR0 – ENQ/DEQ/RESERVE Recovery Routine (Part 1 of 12)



"Restricted Materials of IBM"
Licensed Materials - Property of IBM

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-29. ISGGFRR0 – ENQ/DEQ/RESERVE Recovery Routine (Part 2 of 12)

Extended Description	Module	Label

ISGGFRR0 is the FRR routine used to protect the global resource serialization modules shown in the table below. In some cases, a module must issue the SETFRR macro to establish ISGGFRR0 as its recovery routine. In other cases, the module's caller has already established ISGGFRR0 as the recovery routine.

Entry Point Name	*Issues SETFRR*
IGC048	Y
IGC048FP	Y
IGC056	Y
IGC056FP	Y
ISGGDEQP	N
ISGGQWBC	N
ISGGQWBI	N
ISGGREX0	N
ISGGRP00	Y
ISGGTRM0	Y
ISGGTRM1	Y
ISGSALC	N
ISGSDAL	N
ISGSHASH	N

This routine fills in the SDWA for LOGREC recording, takes a dump, and performs resource validation and repair.

1 ISGGFRR0 adds the recovery routine name and failing subcomponent information to the SDWA. The SDWA already contains the following default options:

- Record the SDWA in LOGREC
- Do not take a dump
- Continue with termination

2 ISGGFRR0 uses a branch entry GETMAIN to conditionally request storage for a workarea. Storage is requested from subpool 239 (an SQA subpool allocated from the CSA). The workarea must be in common storage because it will be used after primary addressability has been switched to the global resource serialization address space.

Extended Description	Module	Label

ISGGFRR0 ensures that there are no storage errors associated with the GVT and that the acronym is correct. The global resource serialization vector table (GVT) must be accessible in order to attempt resource validation and repair.

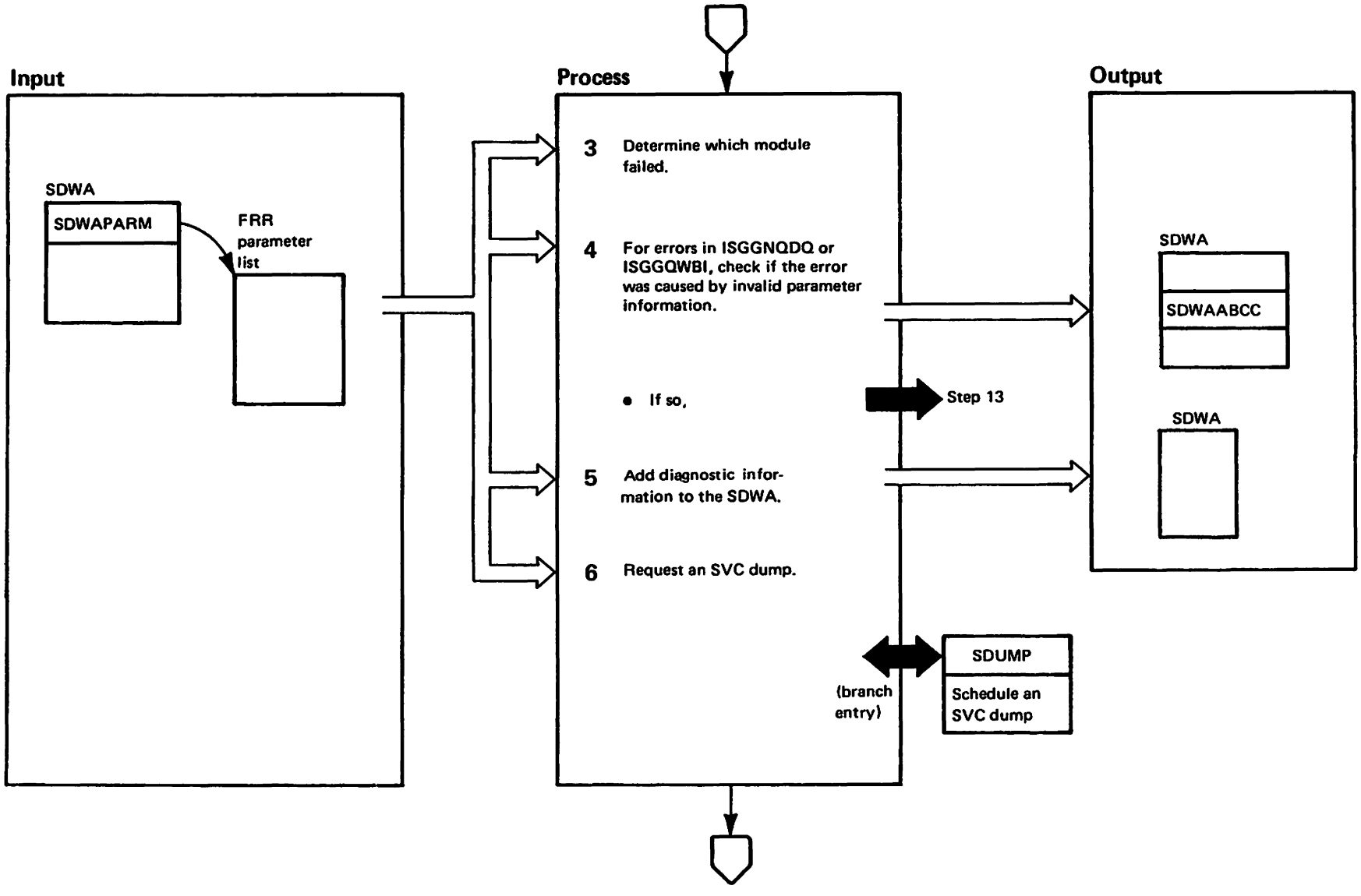
Recovery is not possible in any of the following situations:

- The private area of the failing address space is not accessible.
- The GVT failed the accessibility tests.
- A workarea could not be obtained.

When recovery is not possible, ISGGFRR0 requests an SVC dump and processing continues at step 13. SDUMP

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-29. ISGGFRR0 – ENQ/DEQ/RESERVE Recovery Routine (Part 3 of 12)



"Restricted Materials of IBM"
Licensed Materials – Property of IBM

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-29. ISGGFRR0 – ENQ/DEQ/RESERVE Recovery Routine (Part 4 of 12)

Extended Description	Module	Label
3 For nucleus resident routines, ISGGFRR0 uses the location of the error to determine which module failed. For routines that are not nucleus resident, the 24-byte FRR parameter list contains information that is used to identify the failing module.		
4 ISGGFRR0 checks the type and location of the error to determine if it was an access exception caused by an invalid parameter on the ENQ, DEQ or RESERVE macro. If so, ISGGFRR0 converts the completion code to an ABEND 430 (for DEQ) or ABEND 438 (for ENQ/RESERVE) and bypasses recovery processing. If the error was not caused by an invalid parameter, ISGGFRR0 converts the completion code to an ABEND 730 (for DEQ) or ABEND 738 (for ENQ/RESERVE and continues recovery processing.		
5 ISGGFRR0 copies the following information into the SDWA: • Failing module name • Failing CSECT name • Compile date of the failing CSECT • PTF/product number of the failing CSECT For nucleus resident routines, ISGGFRR0 contains a table of addresses of the CSECT name, the compile date, and the failing CSECT's PTF/product number. For routines that are not nucleus resident, the FRR parameter list contains the address of an area that contains the information noted above. In either case, ISGGFRR0 uses the CSECT name to determine the load module name.		
6 ISGGFRR0 requests an SVC dump except in the following cases: • A previous recovery routine has already provided diagnostic information. • ISGGFRR0 was entered for cleanup only.	SDUMP	

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

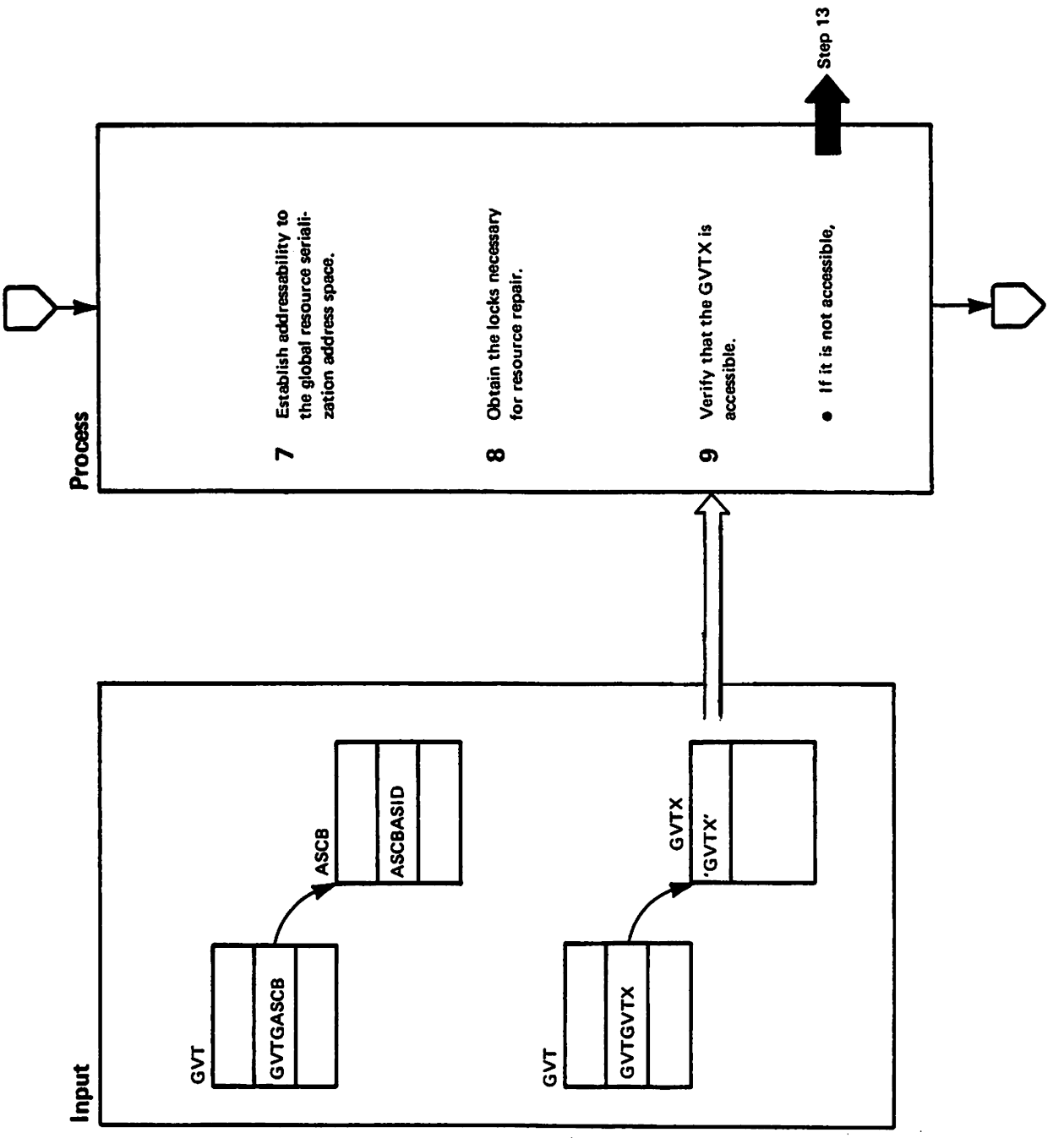


Diagram GRS-29. ISGGFRR0 – ENQ/DEQ/RESERVE Recovery Routine (Part 5 of 12)

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-29. ISGGFRR0 – ENQ/DEQ/RESERVE Recovery Routine (Part 6 of 12)

Extended Description	Module	Label
7 ISGGFRR0 issues the EPAR instruction to determine if it has addressability to the global resource serialization address space. (ISGGFRR0 receives control with the same addressability that existed when the SETFRR was issued. In most cases, the global resource serialization address space is not accessible.) If the global resource serialization address space is not accessible, ISGGFRR0 copies all necessary information, including a copy of the 200-byte workarea, into the workarea obtained in SQA (CSA). ISGGFRR0 then issues a PC instruction to obtain the necessary addressability.		
8 Some callers hold no locks, others hold the local lock of the global resource serialization address space and others hold both a local and the CMSEQDQ lock. The failing process might not have been holding the locks necessary to perform resource repair. If no locks are held, ISGGFRR0 obtains the local lock of the global resource serialization address space and the CMSEQDQ lock. If only a local lock is held, ISGGFRR0 obtains the CMSEQDQ lock. If both locks are held, ISGGFRR0 does not obtain any locks. (*Note:* ISGGFRR0 uses SETLOCK for lock requests. ISGGFRR0 does not check for potential hierarchy violations.)		
9 ISGGFRR0 ensures that there are no storage errors associated with the global resource serialization vector table extension (GVTX) and that the acronym is correct. The GVTX contains information about global resource serialization control blocks that is essential for resource validation and repair. Recovery is not possible if the GVTX is inaccessible. In this case, processing continues at step 13.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

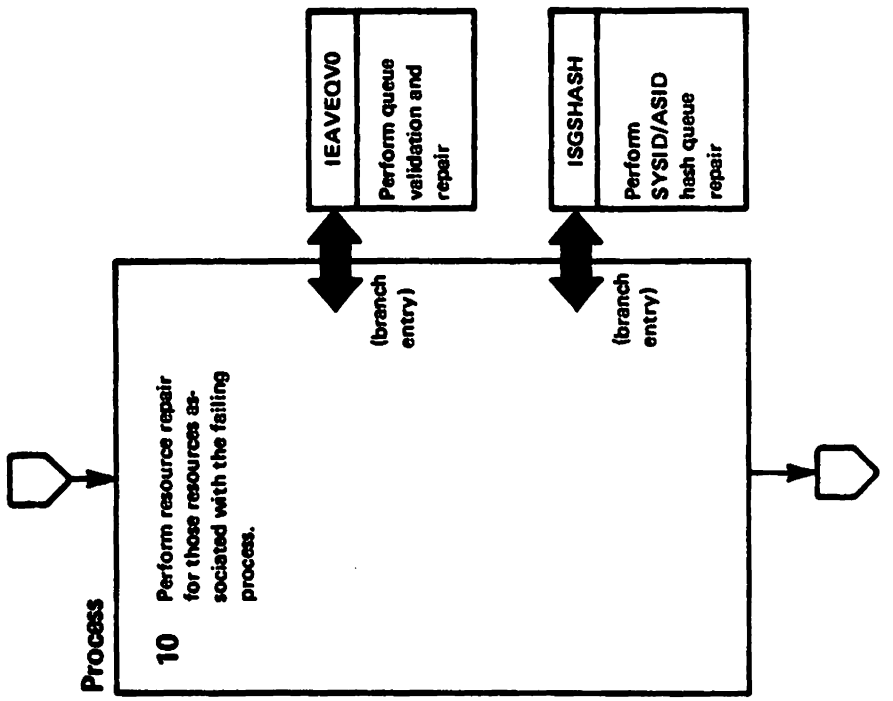


Diagram GRS-29. ISGGFRR0 – ENQ/DEQ/RESERVE Recovery Routine (Part 7 of 12)

LY28-1695-0 (c) Copyright IBM Corp. 1987

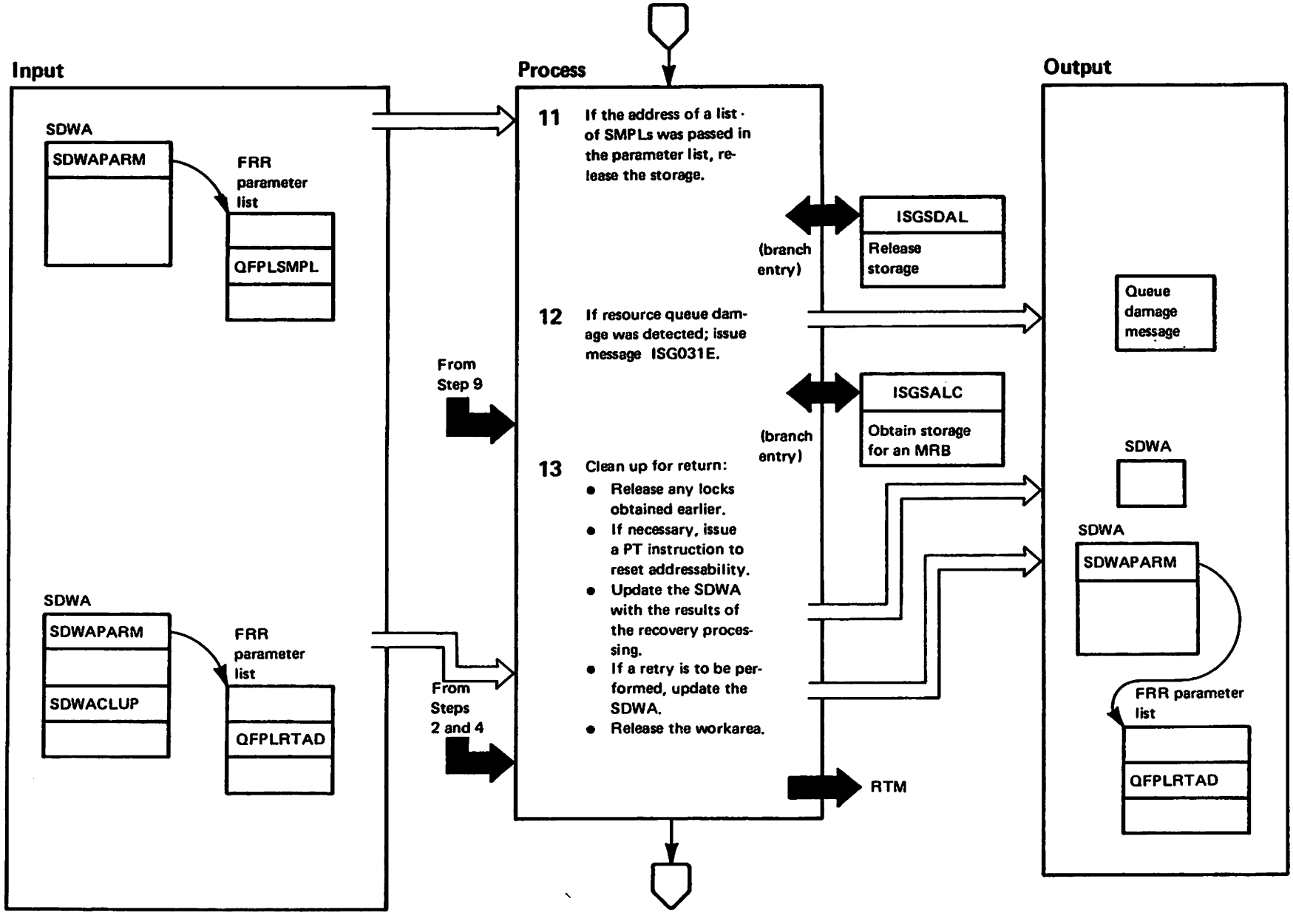
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-29. ISGGFRR0 – ENQ/DEQ/RESERVE Recovery Routine (Part 8 of 12)

Extended Description	Module	Label	
10 Depending on which process failed, ISGGFRR0 attempts resource validation and repair for the following resources in the order listed: Hash table queues (GVTXLQHT, GVTXGQHT) SYSID/ASID hash queue (GVTXSAHT) ASCB resource queues (ASCBLQEL, ASCBGQEL) ASCB synchronization queue (ASCBGSYN) Process queue (GVTPROCQ) Resource pool table queues (GVTXLRPT, GVTXGRPT) Global QWB queue (an entry in the GRPT) Count of inactive PEXBs (GVTXIACT) SMPL queue (FIXSMPLQ) ASCB request count (ASCBREQ)			ISGGFRR0 notes errors in a queue element (QEL) chain in the queue control block (QCB) from which the QELs are chained. It notes errors in a QCB synonym chain in the queue hash table entry (QHTE) from which the QCBs are chained. (Subsequent ENQ or RESERVE requests that require addition of new elements to a damaged queue will be abnormally terminated. DEQ requests will be allowed to proceed.)
IEAVEQV0 calls element verification routines for the following queue elements: PEXB QEL QCB QWB	IEAVEQV0		
If ISGGFRR0 could not determine which module failed, it attempts validation/repair for all resources except ASCBCREQ.			
If IEAVEQV0 finds an error in a single-threaded queue, it truncates the queue. When a double-threaded queue contains an error, IEAVEQV0 uses backward chain pointers to splice together as much of the queue as possible. IEAVEQV0 records all queue repair actions in an area which ISGGFRR0 copies into the variable recording area (VRA) portion of the SDWA.			
Repair of the SYSID/ASID hash queue or ASCB resource queues is different. ISGGFRR0 completely rebuilds these queues using the previously validated hash table queues to do this.	ISGSHASH		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-29. ISGGFRR0 – ENQ/DEQ/RESERVE Recovery Routine (Part 9 of 12)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-29. ISGGFRR0 – ENQ/DEQ/RESERVE Recovery Routine (Part 10 of 12)

Extended Description	Module	Label
11 If the 24-byte FRR parameter list contains the address of a list of storage management parameter list(s) (SMPLs), ISGGFRR0 calls module ISGSDAL to release the cells defined in the SMPLs.	ISGSDAL	
12 If damage was detected in the hash table queues, ISGGFRR0 notifies the operator by issuing message ISG031E. ISGGFRR0 invokes module ISGSALC to obtain storage for a message request block (MRB). After the MRB has been placed on the command request queue, a cross memory post is performed to notify ISGCMDR of the message request.	ISGSALC ISGCMDR IEA0PT01	
13 ISGGFRR0 performs cleanup before returning to RTM. • If ISGGFRR0 obtained any locks, it releases them via SETLOCK. • If a PC was issued in step 7, ISGGFRR0 issues a PT instruction to reestablish addressability to the SDWA and the 200-byte workarea. • ISGGFRR0 copies into the SDWA the output data area (ODA) used by IEAVEQV0 to record queue damage. In addition, it copies into the SDWAVRA miscellaneous processing flags and a bit string that identifies the damaged resources. • Retry is not performed when: – The problem is due to a user error – The name of the failing module is unknown – The 24-byte parameter list has the recursion flag set – ISGGFRR0 was entered for cleanup-only – The processing aborted flag is set – No retry address is available In all other cases, ISGGFRR0 updates the SDWA to request a retry. • If a workarea was obtained earlier, ISGGFRR0 uses a branch entry FREEMAIN to release it.		
ISGGFRR0 returns to RTM. RTM performs any actions requested in the SDWA, for example, it records the SDWA in LOGREC, frees the SDWA, and possibly retries.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

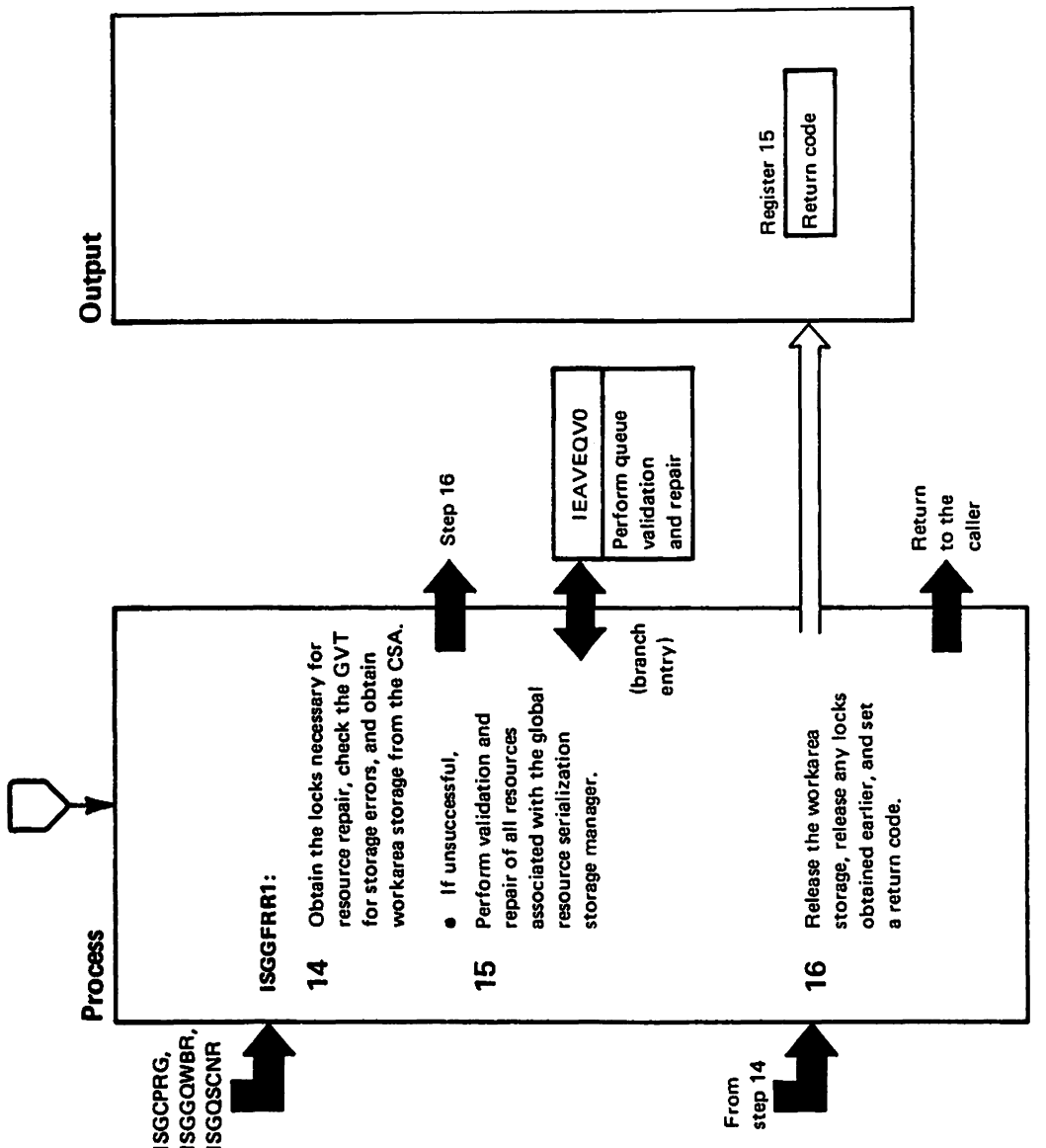


Diagram GRS-29. ISGGFRR0 – ENQ/DEQ/RESERVE Recovery Routine (Part 11 of 12)

LY28-1695-0 (c) Copyright IBM Corp. 1987

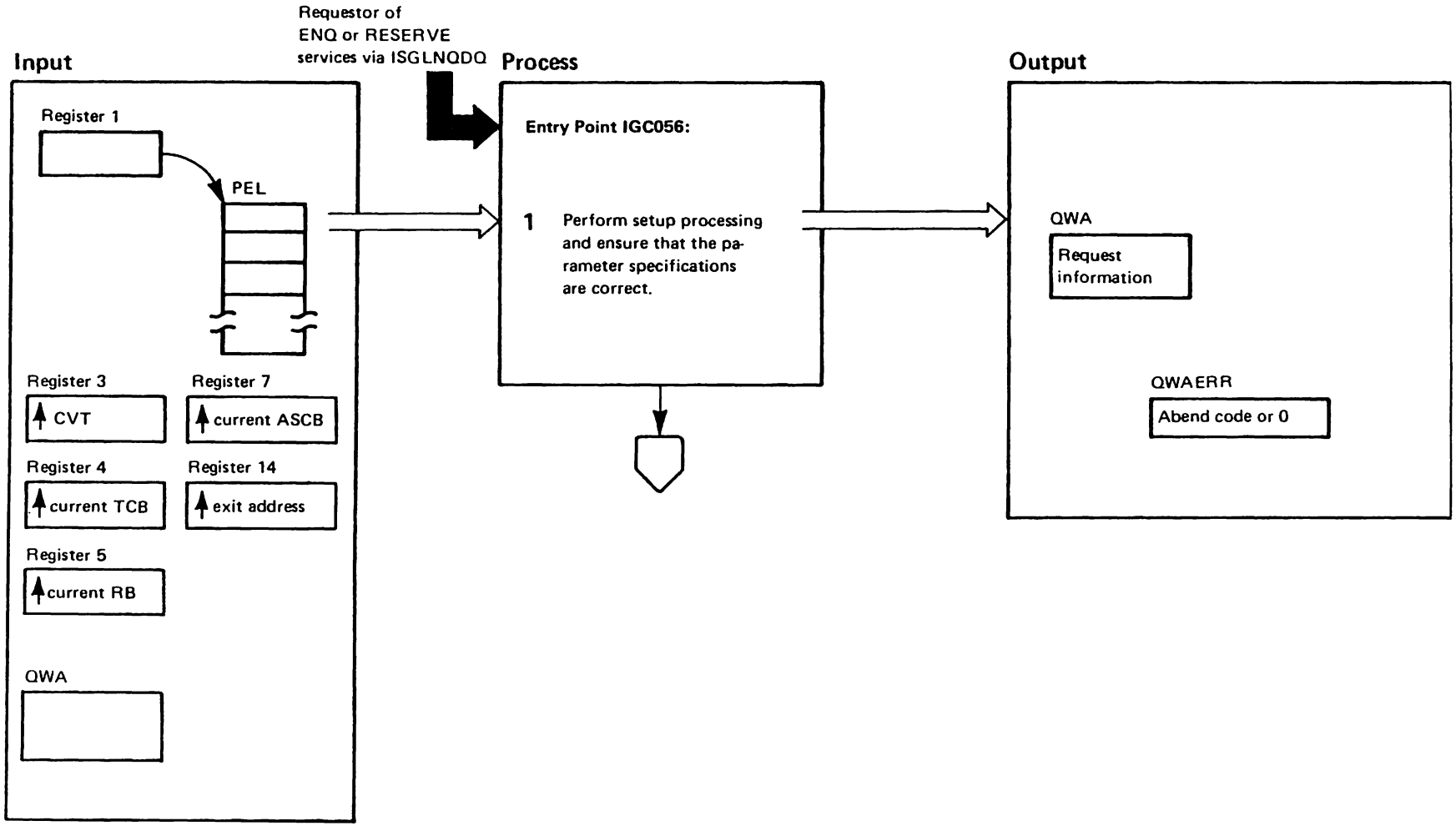
Extended Description	Module	Label
ISGGFRR1:		
Routines that call ISGSALC and ISGSDAL may use ISGGFRR1 to validate and repair the resources used by ISGSALC and ISGSDAL. Note that the callers of ISGGFRR1 must have established addressability to the global resource serialization address space.		
14 Some callers hold no locks, others hold the local lock of the gloabl resource serialization address space and others hold both a local and the CMSEQDQ lock. If no locks are held, ISGGFRR1 obtains the local lock of the global resource serialization address space and the CMSEQDQ lock. If only a local lock is held, ISGGFRR1 obtains the CMSEQDQ lock. If both locks are held, ISGGFRR1 does not obtain any locks. (*Note:* ISGGFRR1 uses SETLOCK for lock requests. ISGGFRR1 does not check for potential hierarchy violations.)		
ISGGFRR1 uses a brach entry GETMAIN to conditionally request storage for a workarea. Storage is requested from subpool 239 (an SQA subpool allocated from the CSA.) If storage cannot be obtained, ISGGFRR1 bypasses resource repair.		
15 If the proper locks were obtained, ISGGFRR1 validates and repairs the following resources used by the global resource serialization storage manager: Resource pool table (RPT) queues (GVTXLRPT, GVTXGRPT) Global QWB queue (an entry in the GRPT) Count of inactive PEXBs (GVTXIACT) Global and local SMPLs in the GVTX (GVTXGSMP, GVTXLSMP) These resources are a subset of those repaired by ISGGFRR0. Refer to the extended descritpion of step 10 for an explanation of how the resources are repaired.	IEAVEQV0	

Extended Description	Module	Label
16 ISGGFRR1 uses a branch entry FREEMAIN to release the workarea obtained earlier. If ISGGFRR1 obtained any locks in order to perform resource repair, it releases the locks via SETLOCK. When control is returned to the caller, register 15 contains a return code as follows: 0 = Resource validation/repair is complete. 4 = ISGGFRR1 was unable to perform validation/repair.		

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-30. ISGGNQDQ – ENQ/RESERVE Processing (Part 1 of 24)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

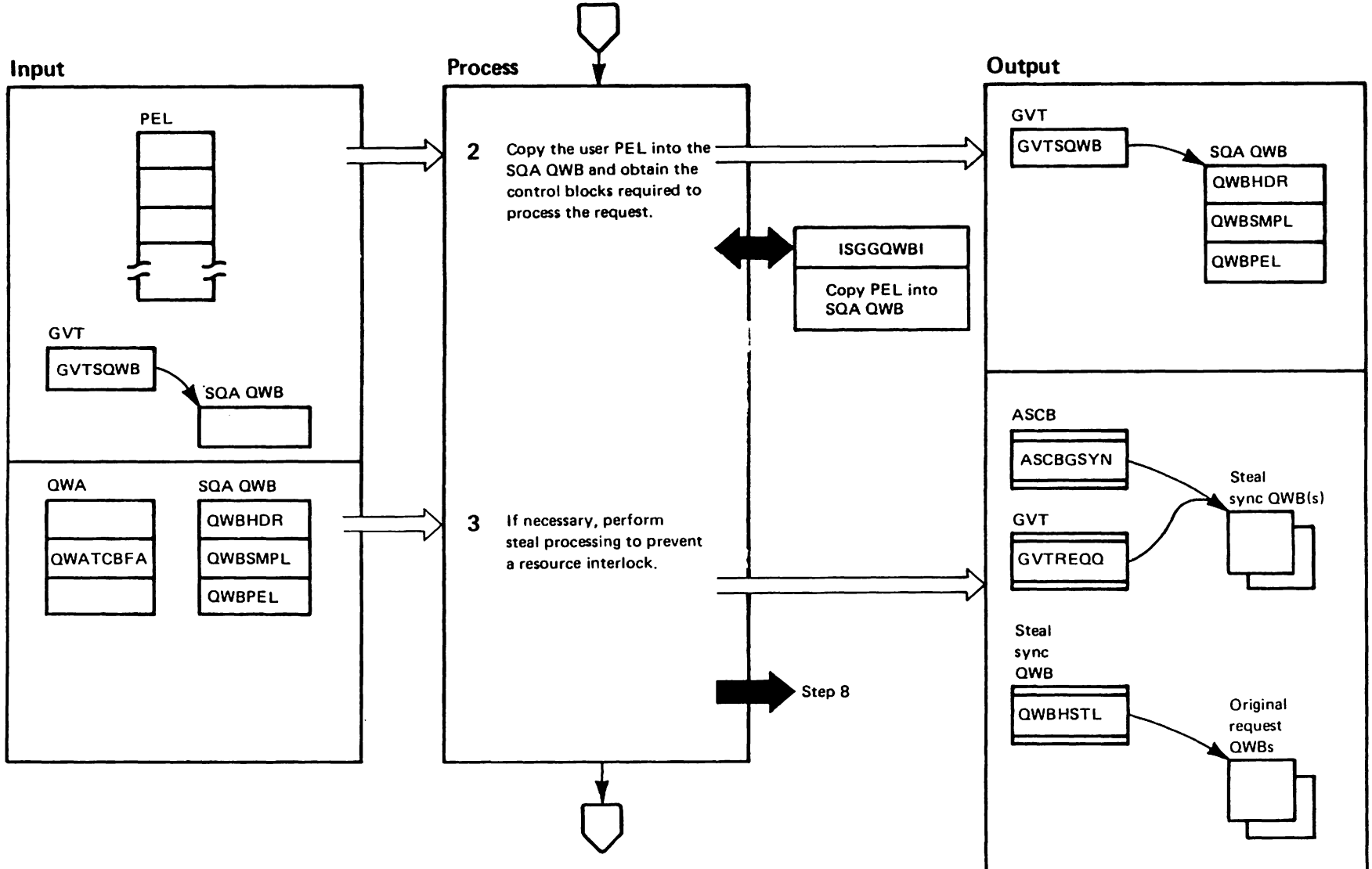
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-30. ISGGNQDQ – ENQ/RESERVE Processing (Part 2 of 24)

Extended Description	Module	Label
ISGGNQDQ processes ENQ/RESERVE requests for specified resources. There are three major sections to ENQ/RESERVE processing. ISG056 is the initial entry point for all ENQ/RESERVE requests, subroutine XPROCENQ performs the actual processing, and entry point ISGGNQ00 is utilized by ISGGRP00 for processing global requests.		
At entry point IGC056, ISGGNQDQ first determines if a request is only for local resources, only for global resources, or for a mixture of local and global resources. The processing of local and global requests differs in that requests for local resources can be processed immediately while requests for global resources cannot be processed until the other systems active in the global resource serialization ring have been informed of this request. For local requests, ISGGNQDQ calls subroutine XPROCENQ to perform the ENQ immediately. For global requests, ISGGNQDQ calls the QWB-copy routine (ISGGQWBC) to build a queue workblock (QWB) for each global request and then places the QWBs on the request queue (GVTREQQ).		
After the QWB built by ISGGNQDQ for a global request has been passed around the global resource serialization ring, IGGRP00 calls ISGGNQDQ at entry point ISGGNQ00 to process the global request. ISGGNQDQ calls XPROCENQ to process the request. ISGGNQDQ then returns to ISGGRP00.		
Subroutine XPROCENQ searches the global and local hash tables and finds the appropriate hash table slots for the requested resources. XPROCENQ then processes the ENQ/RESERVE requests.		
In some cases XPROCENQ also needs to perform steal processing. When a resource is requested by a task that is part of an abending task structure, and the resource is owned by another task in this same task structure, XPROCENQ initiates a resource steal because the abending task is not able to release the resource.		
If the resource request is for a global resource, XPROCENQ builds a sync QWB to be sent around the ring (to be sure that there are no outstanding requests for this resource.) If it is necessary to actually steal the resource, XPROCENQ builds a DEQ QWB and places the DEQ QWB followed by the request QWB on the request queue.		
If the resource request is for a local resource, XPROCENQ steals the resource without notifying the other systems.	ISGGNQDQ	IGC056
Entry Point IGC056:		
1 ISGGNQDQ establishes an FRR, obtains the global requestor's local lock and the CMSEQDQ lock, and initializes the queue workarea (QWA). ISGGNQDQ checks whether the parameters conflict and whether the caller is authorized to request the specified functions. ISGGNQDQ abends requestors when they fail any of these checks.		XSETUP

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-30. ISGGNQDQ – ENQ/RESERVE Processing (Part 3 of 24)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

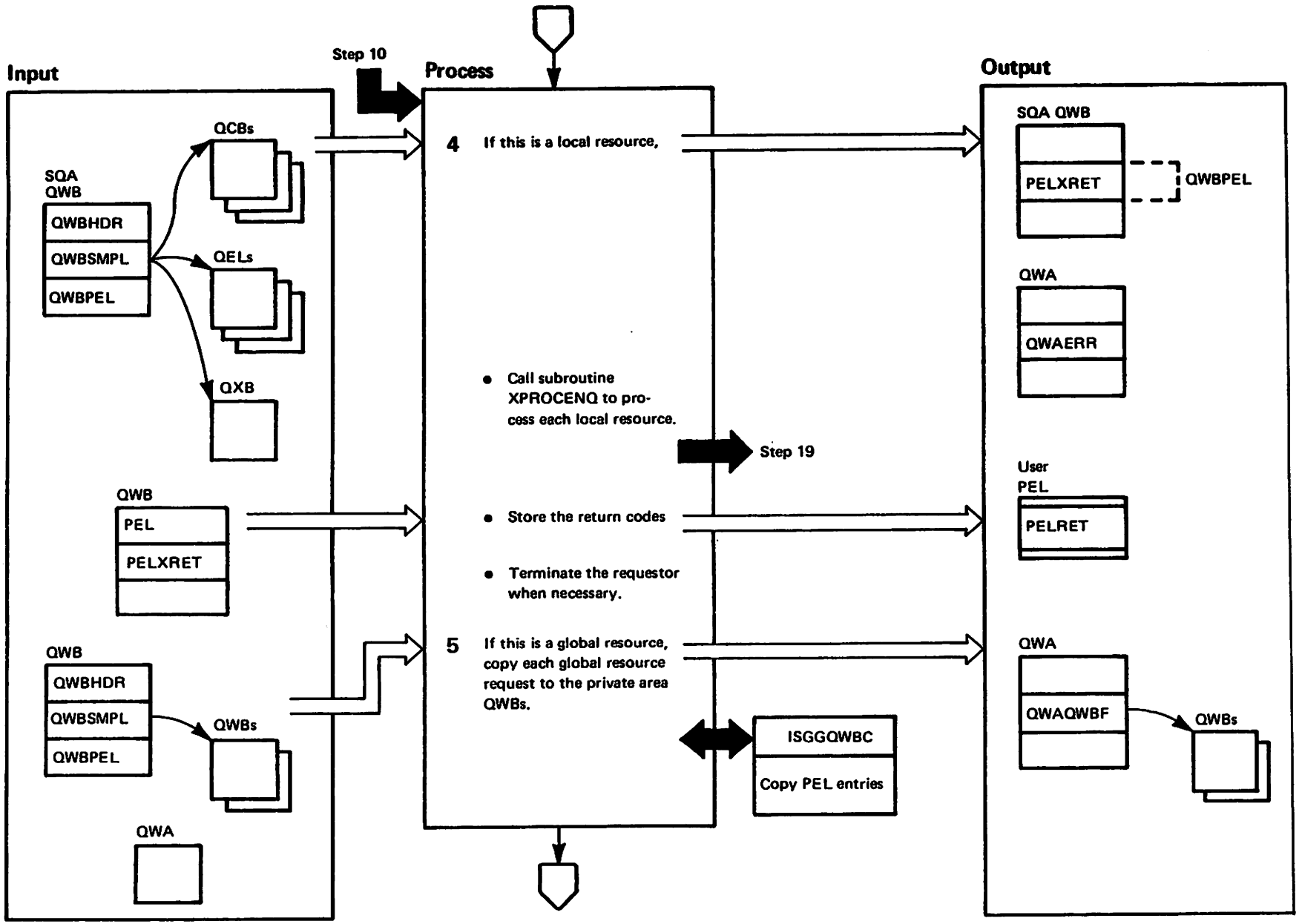
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-30. ISGGNQDQ – ENQ/RESERVE Processing (Part 4 of 24)

Extended Description	Module	Label
2 ISGGNQDQ invokes the global resource serialization queue work block initialization routine (ISGGQWBI) to • Copy the parameter element list (PEL) to the system queue area (SQA) QWB • Call installation exit routines (ISGGREX0) to identify the requests as being for either local or global resources • Establish addressability to the global resource serialization address space • Obtain QCBs, QELs, and QXBs	ISGGQWBI ISGGREX0	
Upon return from ISGGQWBI, ISGGNQDQ verifies that the requests represented by the PEL entries do not exceed the concurrent request limit. If an unconditional requester exceeds the limit, ISGGNQDQ issues an ABEND; if a conditional requester exceeds the limit, ISGGNQDQ notifies the requester of this fact via the appropriate return code.		
3 When a resource is requested by a task that is part of an abending task structure, and the resource is owned by another task in this same task structure, there can be an interlock. If ISGGNQDQ finds this situation, it solves the problem by stealing the resource from the owning task. Global steal processing is performed in 3 stages.		
Stage 1 – ISGGNQDQ constructs a sync QWB containing a pointer to the original request's QWB(s). The sync QWB ensures that the request and processing queues are purged of any outstanding ENQs for this resource before the steal is attempted. (That is, because QWBs are processed in the order in which they are queued, when the sync QWB appears on the process queue, ISGGNQDQ is assured that all preceding requests have been processed.)		
Stage 2 – After the sync QWB is processed, XPROCENQ processes the original request's QWB. XPROCENQ steals the requested local resources if this is a request with both local and global resources requested, and builds steal DEQ QWB(s) for all of the requested global resources. It places all the original request's QWB(s) after any DEQ QWB(s) on the process queue.		
Stage 3 – ISGGRP00 processes the original request's QWB(s) (ISGGRP00 calls ISGGNQDQ at entry point ISGGNQ00 to do the processing.)		
ISGGNQDQ starts the steal processing by calling ISGGQWBC to copy each global resource (and any local resources that are also present) from the SQA QWB into the private area QWBs. The private area QWBs were obtained earlier and chained from the SQA QWB SMPL. When all the PEL entries have been copied, ISGGQWBC initializes a sync QWB.	ISGGQWBC	
ISGGNQDQ moves the sync QWB to the request queue only when no other syncs are outstanding. This ensures that only one sync request is processed at a time. When a sync request is already being processed, ISGGQWBC places the current sync on the end of the ASCB sync queue. ISGGNQDQ (at subroutine XPROCENQ) processes the QWB after previous sync requests complete.		
ISGGNQDQ goes to step 8 to branch enter WAIT while the sync QWB is passed around the ring. (This completes stage 1 processing.)		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-30. ISGGNQDQ – ENQ/RESERVE Processing (Part 5 of 24)



 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-30. ISGGNQDQ – ENQ/RESERVE Processing (Part 6 of 24)

Extended Description	Module	Label
4 ISGGNQDQ calls subroutine XPROCENQ to process the local requests (steps 19-26 describe XPROCENQ's processing.) The QWBSMPL points to the QCB, QEL, and QXB control blocks. ISGGNQDQ passes this input to the subroutine XPROCENQ.		XPROCENQ
• ISGGNQDQ uses the SQA QWB PEL as the input PEL. (Note that in the case of steal processing, the input PEL is located in a private area QWB not a SQA QWB.)		
• After each PEL entry is processed, ISGGNQDQ moves the return codes to the user's PEL. ISGGNQDQ issues an ABEND if it is necessary to do so (determined by XPROCENQ).		XENQSTRC
5 If this is a global resource, ISGGNQDQ calls ISGGQWBC to copy each global resource from the SQA QWB into the private area QWBs. ISGGNQDQ had previously obtained the private area QWBs and chained them out of the QWB SMPL.	ISGGQWBC	

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

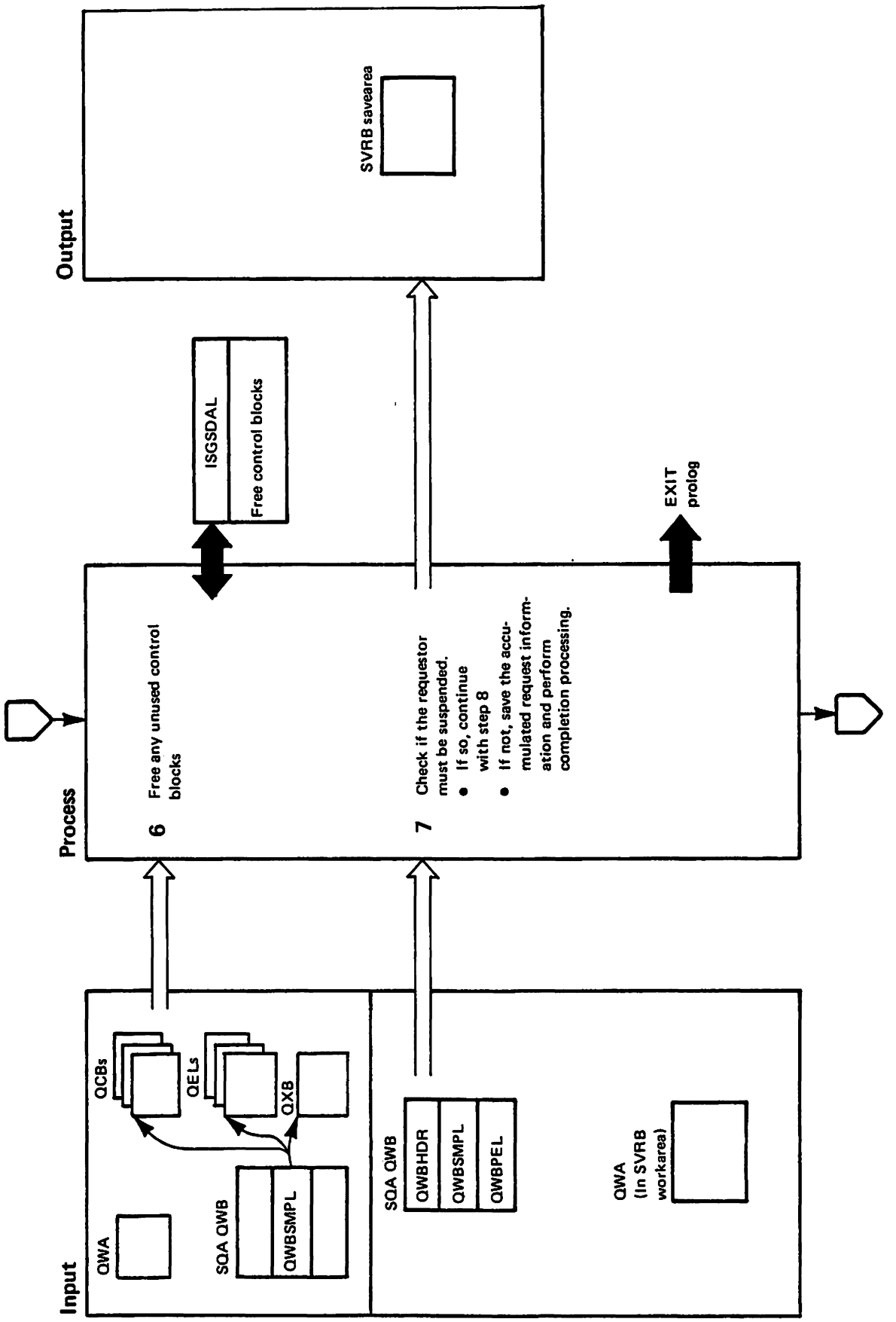


Diagram GRS-30. ISGGNQDQ - ENQ/RESERVE Processing (Part 7 of 24)

LY28-1695-0 (c) Copyright IBM Corp. 1987

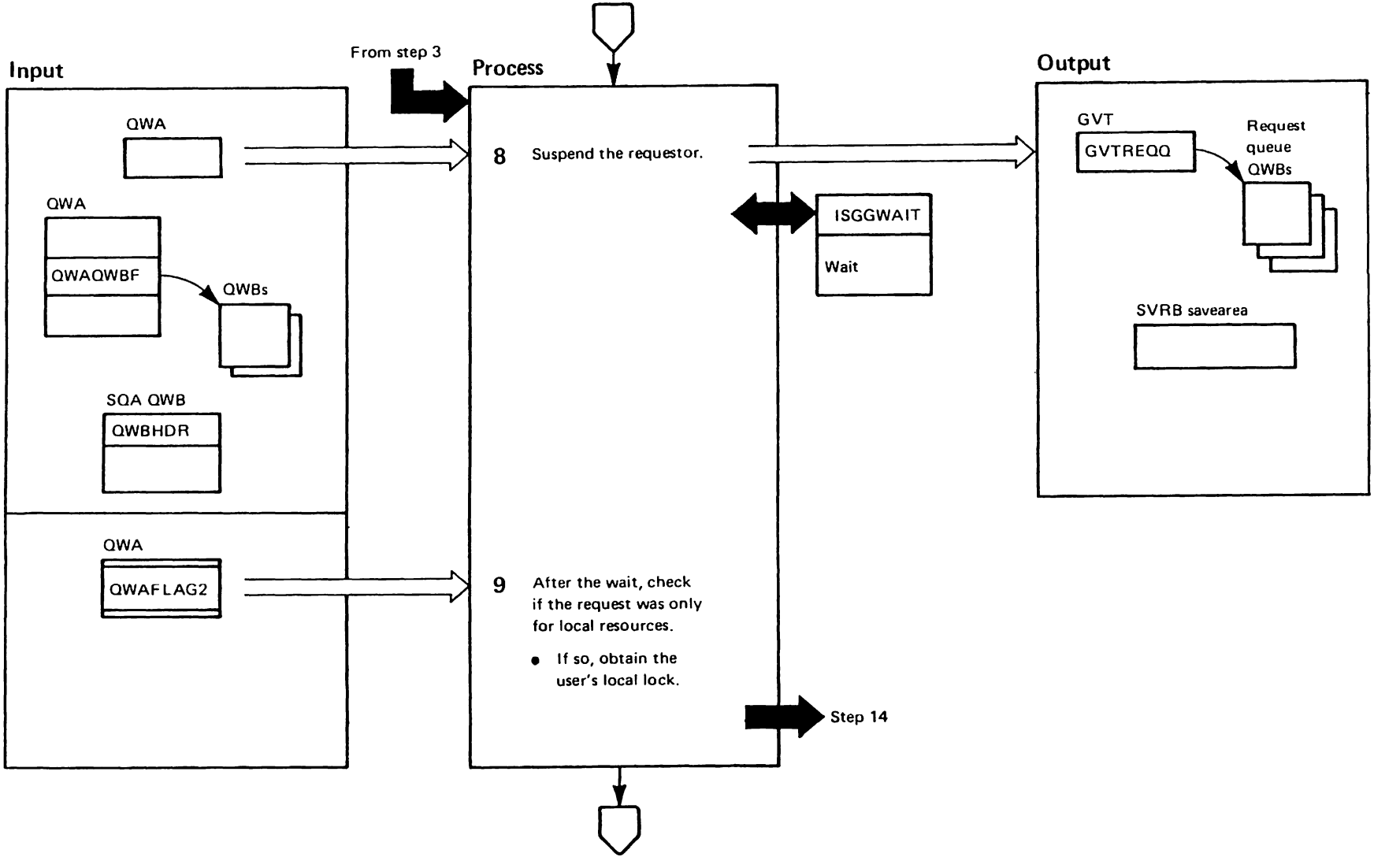
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-30. ISGGNQDQ – ENQ/RESERVE Processing (Part 8 of 24)

Extended Description	Module	Label
6 When all the input PEL entries have been processed, ISGGNQDQ calls ISGSDAL to free any unused control blocks (QCBs, QELs and the QXB.)	ISGSDAL	
7 ISGGNQDQ does not suspend the requestor if any of the following conditions are met for each local resource requested (requests for global resources always result in suspension): • The resource was immediately available. • The resource was not immediately available but ECB= or RET=USE was indicated. • RET=HAVE was indicated and the requestor currently owns the resource. • RET=TEST or CHNG was indicated. If the requestor must be suspended, processing continues at step 8. Otherwise, ISGGNQDQ moves the QWA into the SVRB extended savearea. This enables the completion routine to reference the data after the QWA serialization is released. To complete the request, ISGGNQDQ • Reestablishes addressability to the home address space. • Invokes STATUS if step must complete (SMC) was indicated. • Releases the locks and deletes the FRR.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-30. ISGGNQDQ – ENQ/RESERVE Processing (Part 9 of 24)



 (c) Copyright IBM Corp. 1987

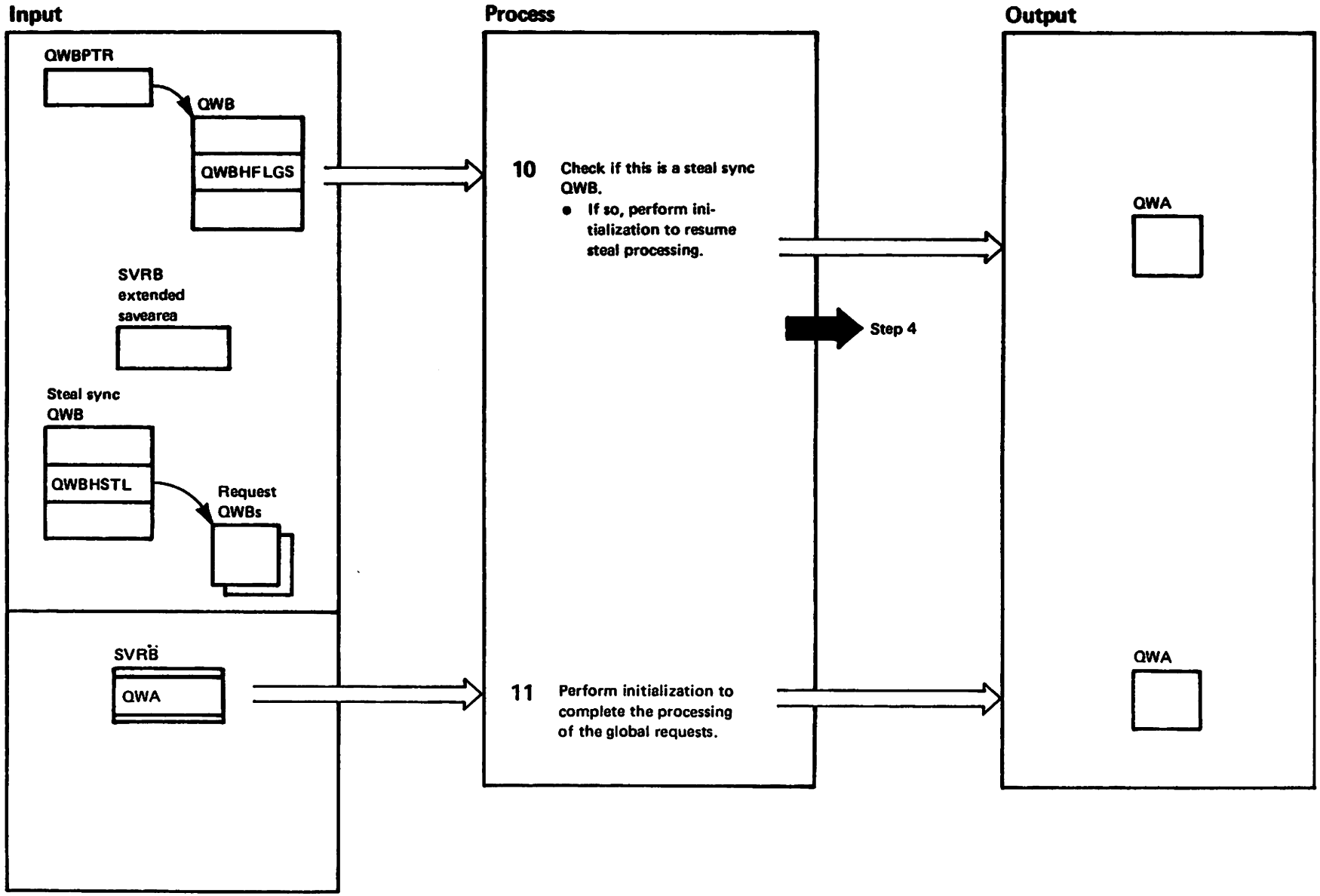
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Extended Description	Module	Label
8 ISGGNQDQ suspends the requestor if any of the following conditions are met: • A global resource is present. • A local resource was not immediately available and RET=NONE was specified or RET=HAVE was specified and the requestor was not the owner of the resource. • Stage 1 steal processing needs to wait until the sync QWB is processed. To suspend a requestor, ISGGNQDQ: • Places the QWBs on the request queue so that the request will be serialized with the other systems in the global resource serialization ring. • Increases the task global resource count (TCBGRES) by the number of global resources requested by this task. • Copies the QWA into the SVRB extended savearea (for global resources, some of the QWA information is copied into the private area QWB) prior to the WAIT. • Releases the CMSEQDQ lock. The local lock is retained since it is required by the WAIT interface (which will release the local lock.) • Sets register 0 to indicate either a short or long wait. Global requests are always considered short waits. For local resources, ISGGNQDQ checks whether the long-wait bit was set during resource processing. If the bit is set to one, ISGGNQDQ indicates in register 0 that this is to be a long wait. ISGGNQDQ calls the wait service routine to suspend the requestor.	ISGGWAIT	
9 If global resource serialization is not active, or if it is active but the request did not specify any global resources, ISGGNQDQ continues at step 14 where it prepares to return to the caller.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-30. ISGGNQDQ – ENQ/RESERVE Processing (Part 11 of 24)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-30. ISGGNQDQ – ENQ/RESERVE Processing (Part 12 of 24)

Extended Description	Module	Label

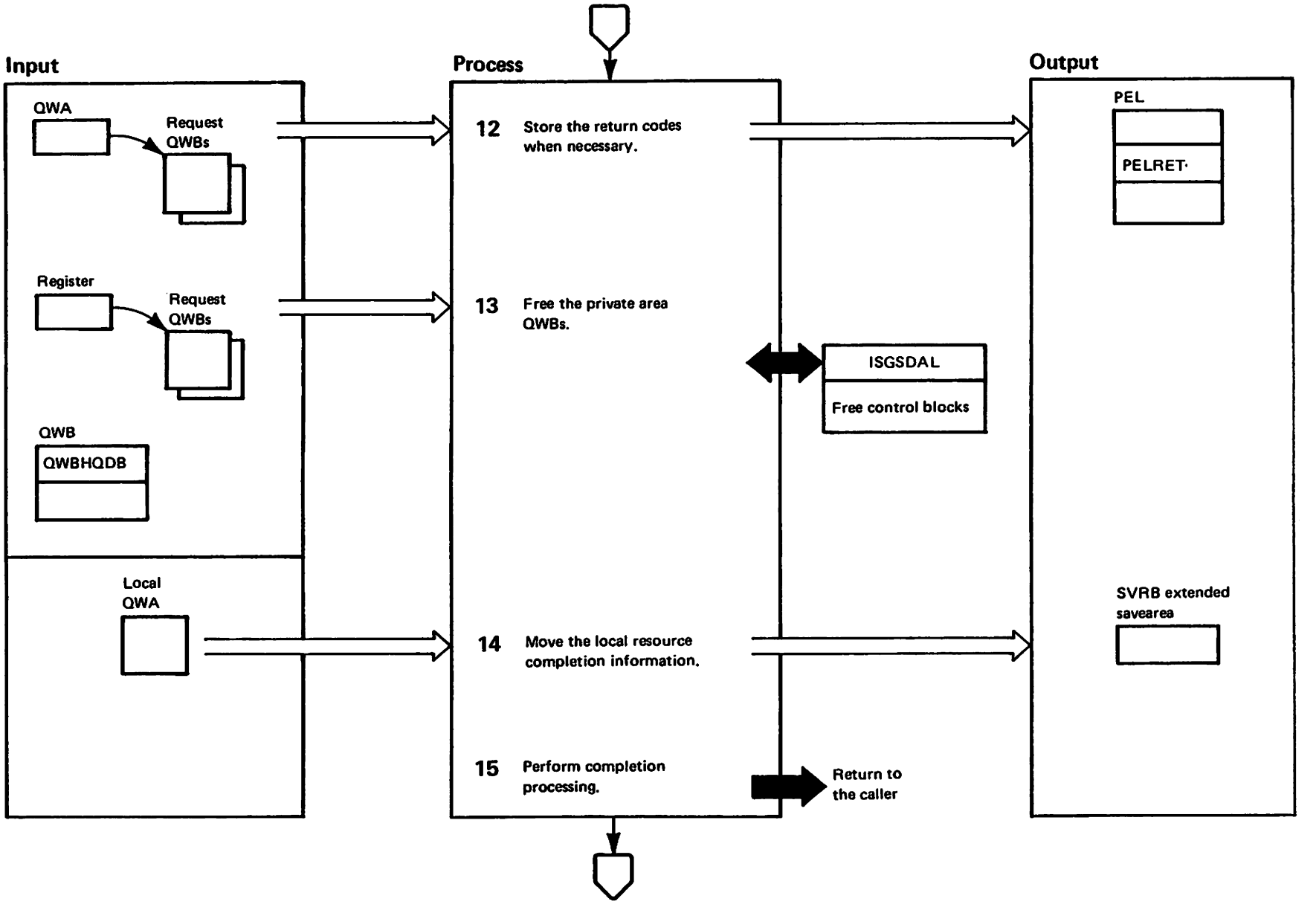
10 Steal processing placed a sync QWB on the request queue and waited for it to be processed (see step 3). The POST from ISGGRP00 to ISGGWAIT causes steal processing to resume. ISGGNQDQ performs the following initialization functions:

- Acquires the global resource serialization local lock to serialize the global resource queues.
- Acquires the CMSEQDQ lock to serialize the QWB pool and the sync request queue.
- Copies the data saved prior to the wait from the SVRB extended savearea back into global QWA. This data is needed to process the request.
- Decreases the task global resource count (TCBGRES) by the number of global resource requests that are on the queue for global ENQ requests (ISGGRP00 has not put any QELs on the queue.)
- Locates the QWBs that are chained out of the steal synchronization QWB. These private area QWBs are the input for stage 2 steal processing.

11 In order to complete the processing of the global requests, ISGGNQDQ obtains the requestor's local lock and the CMSEQDQ lock (in order to free the QWB) and copies the data saved in the SVRB extended savearea back into the QWA.

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-30. ISGGNQDQ – ENQ/RESERVE Processing (Part 13 of 24)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-30. ISGGNQDQ – ENQ/RESERVE Processing (Part 14 of 24)

Extended Description	Module	Label

12 ISGGNQDQ moves the return codes from the QWB PEL entry into the requestor's PEL entry. ISGGNQDQ also issues an ABEND when the return code indicates that one is needed.

13 ISGGNQDQ frees the QWBs defining this request. (Private area QWBs will not exist unless a request was for a global resource.)

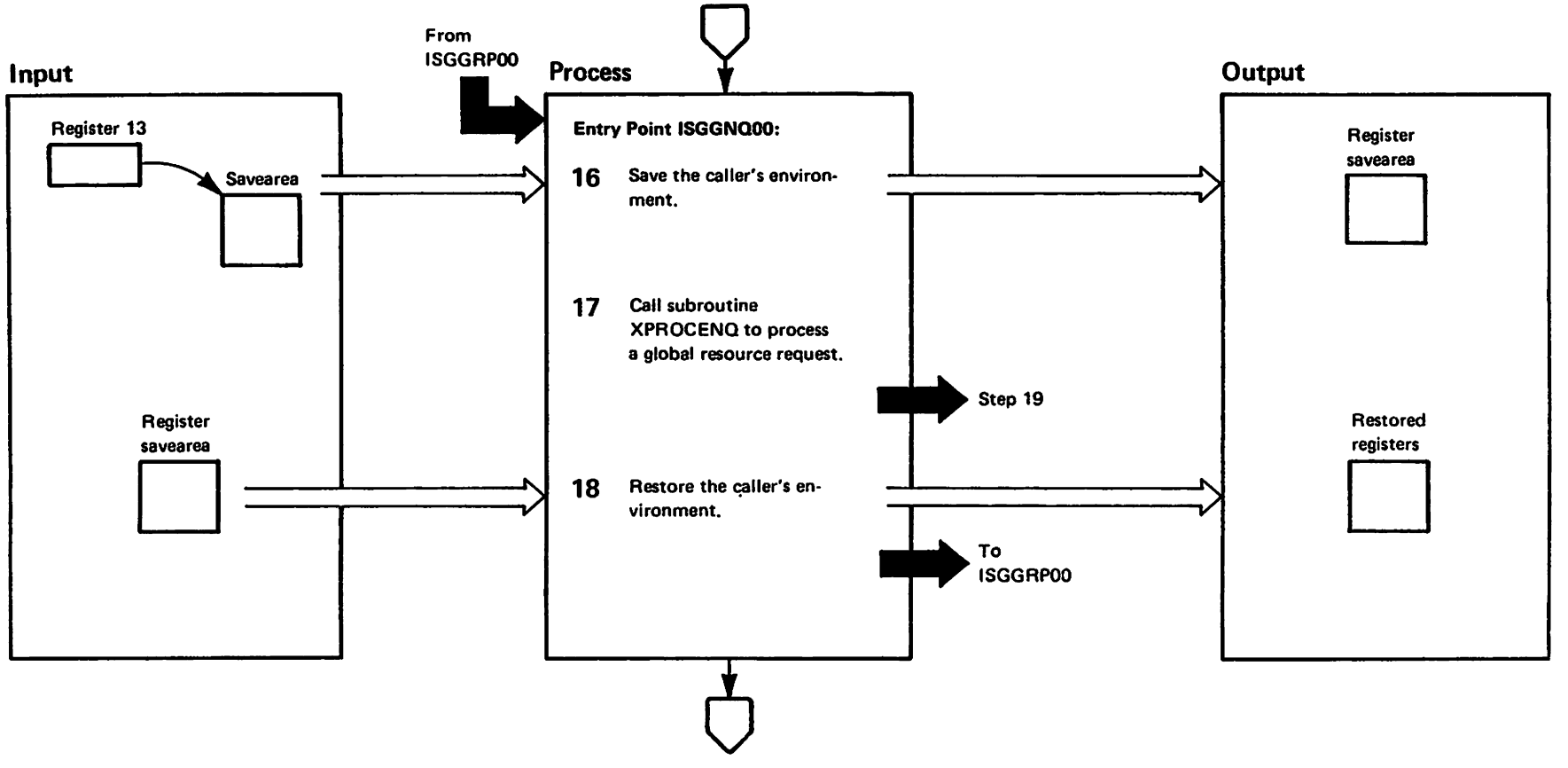
14 ISGGNQDQ moves the QWB to the SVRB extended savearea. This is necessary so that the completion data can be referenced after addressability is reestablished to the home address space and the locks are released.

15 ISGGNQDQ performs the following completion processing:

- Reestablishes addressability to the home address space
- Invokes STATUS when step must complete (SMC) was indicated
- Releases the locks and deletes the FRR

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-30. ISGGNQDQ – ENQ/RESERVE Processing (Part 15 of 24)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

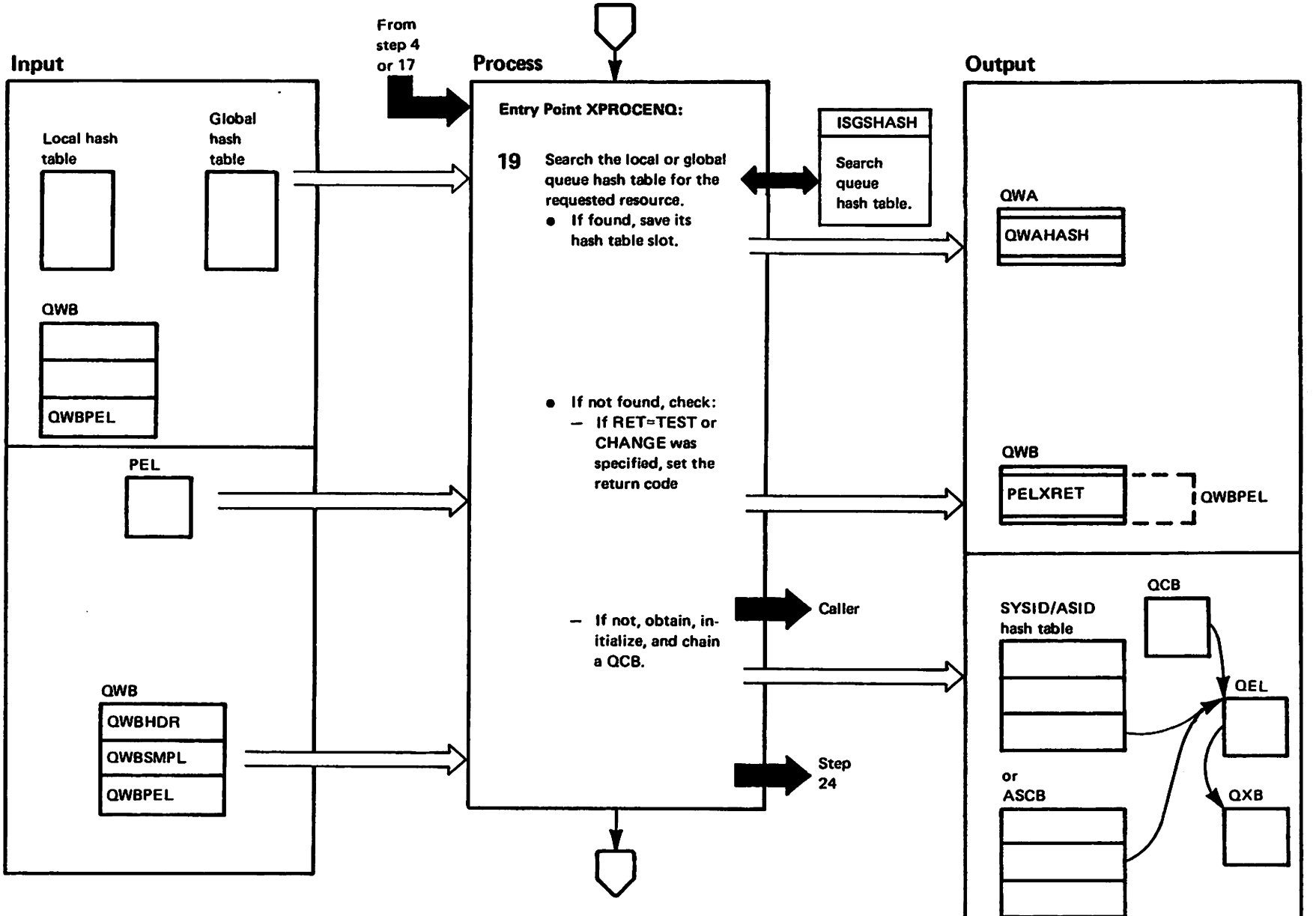
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-30. ISGGNQDQ – ENQ/RESERVE Processing (Part 16 of 24)

Extended Description	Module	Label
Entry Point ISGGNQ00:		
16 ISGGRP00 uses entry point ISGGNQ00 as an interface to reach subroutine XPROCENQ. ISGGNQDQ saves ISGGRP00's registers and the savearea address before calling XPROCENQ.	ISGGNQDQ	ISGGNQ00
17 XPROCENQ processes a global request. Steps 19-26 describe XPROCENQ's processing.		XPROCENQ
18 ISGGNQDQ restores ISGGRP00's registers and returns to ISGGRP00.	ISGGRP00	

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-30. ISGGNQDQ – ENQ/RESERVE Processing (Part 17 of 24)



LY28-1695-0 (c) Copyright IBM Corp. 1987

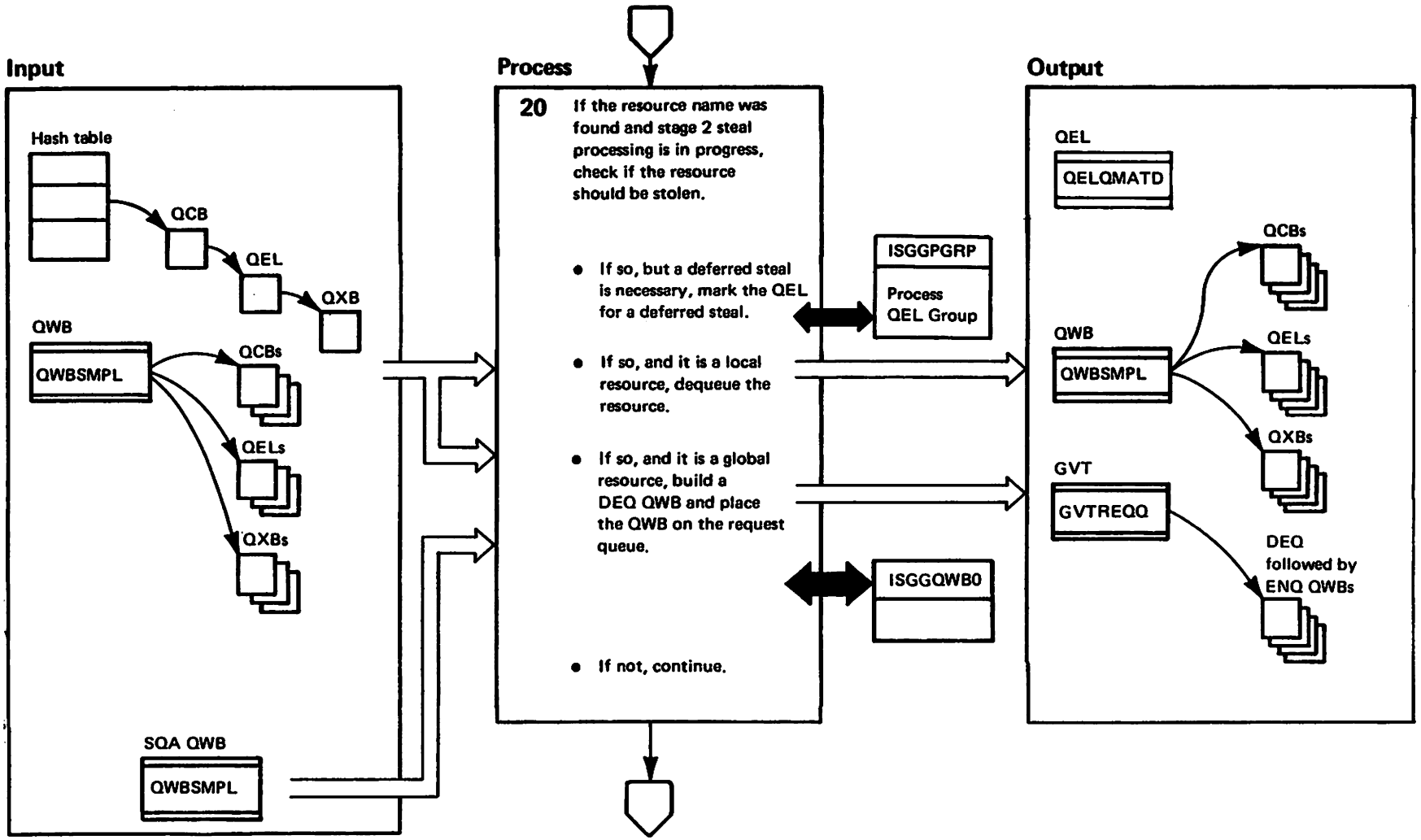
"Restricted Materials of IBM"
Licensed Materials – Property of IBM

Extended Description	Module	Label
Entry Point XPROCENQ:		
19 The subroutine XPROCENQ calls ISGSHASH to search the resource queues for the requested resource (represented by a QCB.) If a local resource has been requested, ISGSHASH returns a slot from the local hash table. If a global resource has been requested, ISGSHASH returns a slot from the global hash table.	ISGAHASH	ISGSGLH
The subroutine XPROCENQ uses the hash table slot to queue this resource to the hash table for subsequent processing.		
If the resource name is not found queued out of this hash table slot, a QCB might need to be added to the hash table.		
• If the requestor specified RET=CHNG, and ENQ for this resource should have already been done. Since the resource was not found queued out of the hash table, the requested change cannot be done. XPROCENQ sets a return code of 8 in the PELXRET and returns to the caller. • If the requestor specified RET=TEST, not finding the resource indicates that the resource is available. The subroutine XPROCENQ sets a return code of zero in the PELXRET and returns to the caller.		
For all other types of requests, XPROCENQ obtains, initializes and chains a QCB to the appropriate hash table entry.		
XPROCENQ takes the QCB from control blocks that it obtains from the global resource serialization storage manager (ISGSALC). ISGGNQDQ calls ISGSALC before the subroutine XPROCENQ. ISGSALC allocates a QXB and one or more QCBs and QELs. The storage management parameter list (SMPL) points to the allocated control blocks.		
Note: The return code is only set if the request originated from the current system. Each system in the ring sets its own return codes.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

Diagram GRS-30. ISGGNQDQ – ENQ/RESERVE Processing (Part 19 of 24)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

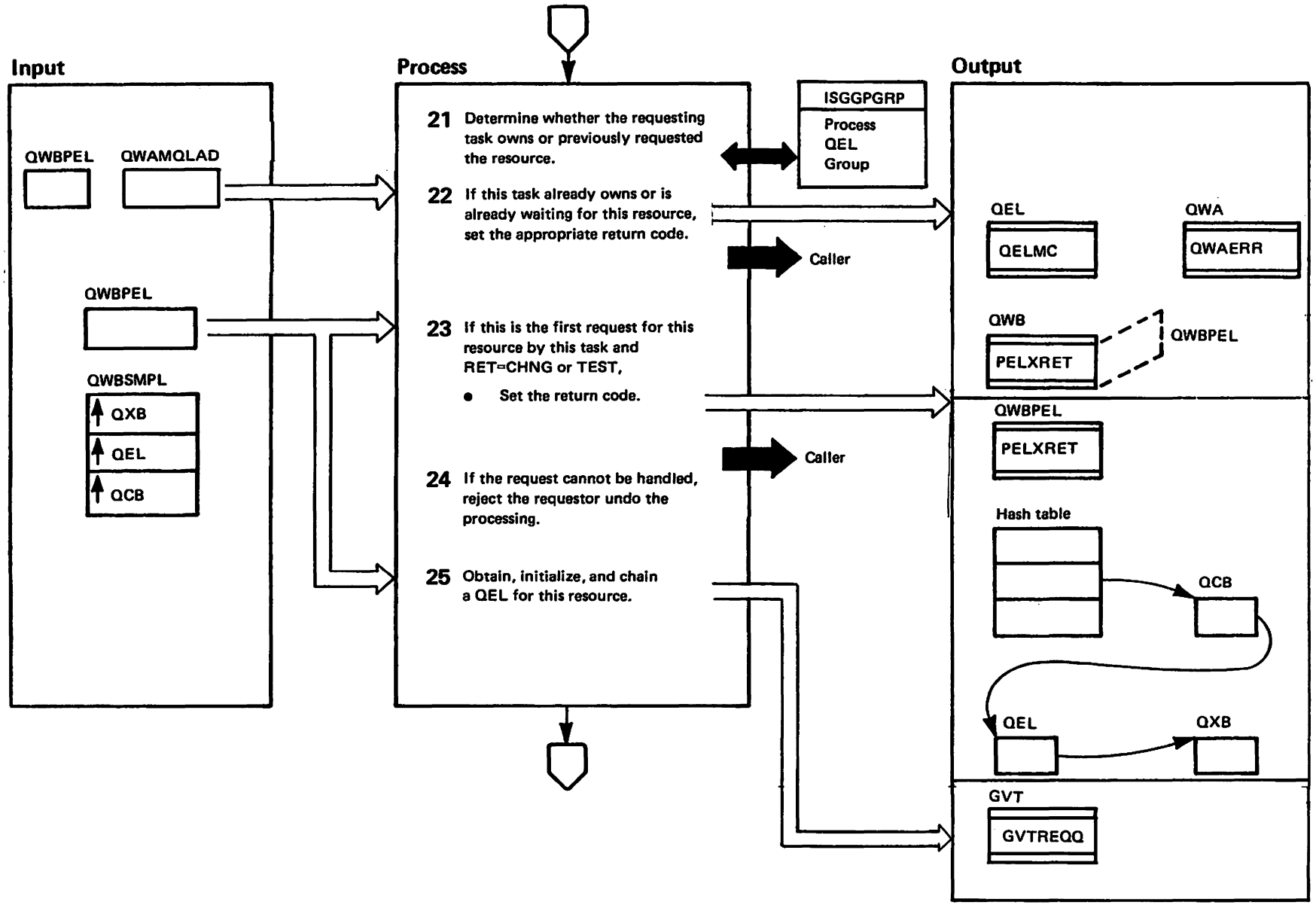
LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-30. ISGGNQDQ – ENQ/RESERVE Processing (Part 20 of 24)

Extended Description	Module	Label
20 This processing is required to ensure that no permanent waits caused by ENQ suspensions occur during RTM processing. If the requesting task is abending and a global resource was requested, ISGGNQDQ invokes the subroutine XPROCENQ twice: first to build steal DEQ QWBs if necessary (stage 2 processing), and then to process the ENQ request (stage 3 processing). QWASTLC controls steal processing. When it is zero (during stage 2 processing), XPROCENQ attempts the needed steal by building DEQ QWBs. When it is one (during stage 3 processing), the steal has been completed and XPROCENQ processes the original ENQ request. During stage 2, HASID=SASID=the ENQ requestor's address space, and PASID=the global resource serialization address space. During stage 3, HASID=SASID=PASID=the global resource serialization address space. XPROCENQ scans the resource's QEL chain. If the end of the queue is reached and no steals are necessary, XPROCENQ return to the caller. If XPROCENQ finds the QEL, and the SYSID/ASID matches the requestor's SYSID/ASID, the QEL is a steal possibility. XPROCENQ examines TCBs in the ENQ requestor's address space. XPROCENQ issues SAC ON to access the requestor's address space and SAC OFF to restore access to the global resource serialization address space. In the requestor's address space, XPROCENQ checks if the QEL-TCB is in the same abend TCB tree as the requestor. If the QEL is not, XPROCENQ issues a SAC OFF to re-establish addressing to the global resource serialization address space. XPROCENQ continued to search the QEL queue for steal possibilities. If XPROCENQ finds the QEL in the requestor's TCB tree, a steal is needed. XPROCENQ marks the QEL for a deferred steal (if a MASID-QEL currently points at the QEL that must be stolen) or performs the steal. XPROCENQ calls ISGGPGRP (function MASIDSCN) to make this decision.	ISGGPRGP	
If the QEL is for a local resource, XPROCENQ performs the steal by calling subroutine XDEQQEL, which dequeues the resource. XPROCENQ continues to search the QEL chain for other steal possibilities. If the QEL is for a global resource, the DEQ cannot be executed until after the DEQ request has been presented to each system in the global resource serialization ring. This is necessary to ensure the consistency of the hash table data. Therefore, DEQ requests are generated in this step and passed to other systems in the global resource serialization ring. XPROCENQ calls subroutine XGLDEQG, which calls ISGGQWB0 (entry point ISGGQWB4) to build a DEQ QWB. XPROCENQ places this QWB on the request queue. XPROCENQ continues to search the QEL queue for steal possibilities until it reaches the end of the queue. It then returns to the caller.	XDEQQEL ISGGQWB0	 ISGGQWB4

Diagram GRS-30. ISGGNQDQ – ENQ/RESERVE Processing (Part 21 of 24)



LY28-1695-0 (c) Copyright IBM Corp. 1987

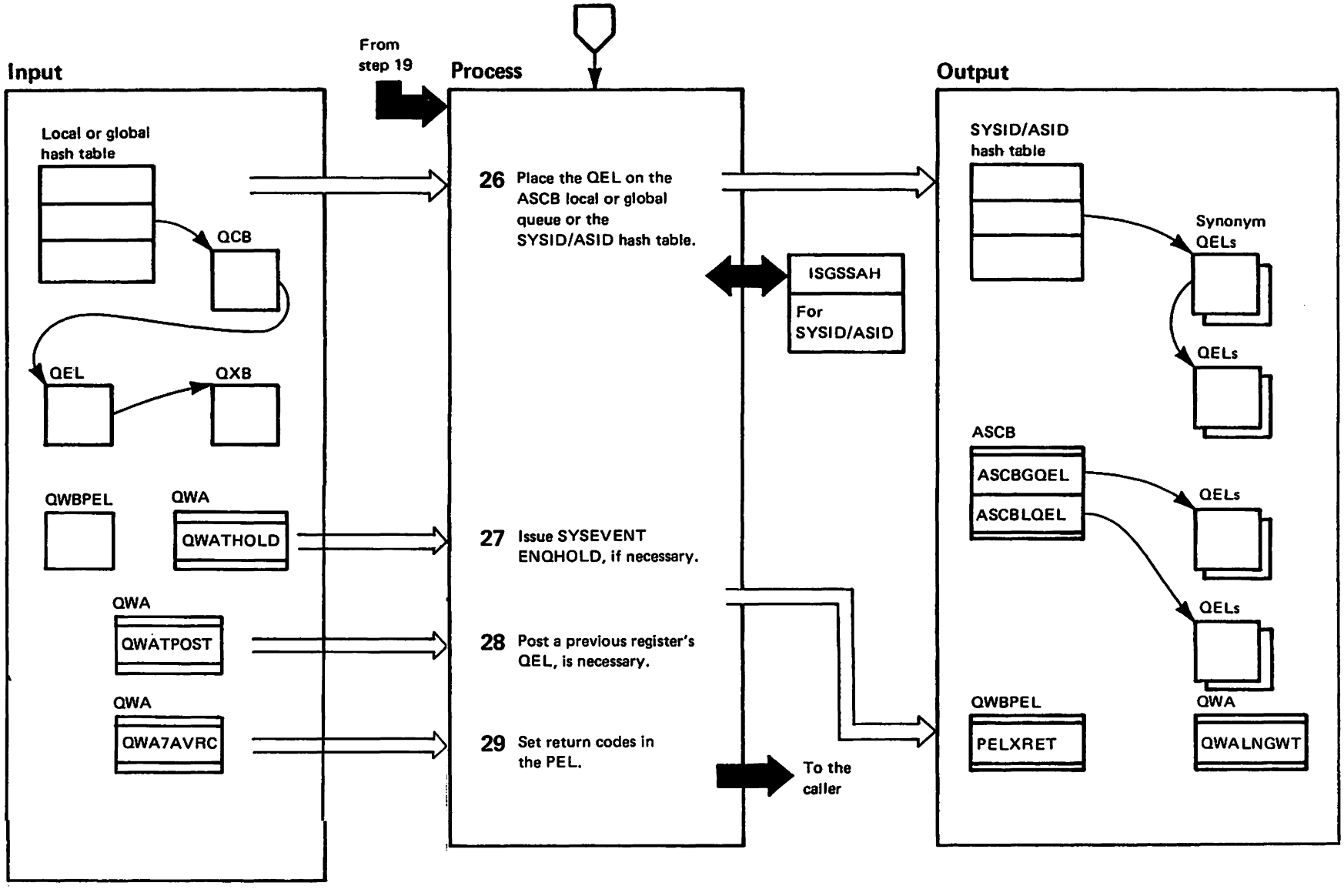
"Restricted Materials of IBM"
Licensed Materials – Property of IBM

Extended Description	Module	Label
21 XPROCENQ calls the QEL group processing routine (ISGGPGRP) to perform the function ENQSCAN. ISGGPGRP determines whether the requesting task owns the resource or previously requested the resource. ISGGPGRP also determines whether the requesting task is allowed to use the resource (flag QWA7AURC) and whether the request is illegal (flag QWA7ABMR) or cannot be handled (flag QWA7COEX).	ISGGPGRP	
22 When the requestor asks for the same resource a second time, XPROCENQ sets a return code. The return code is determined by whether the requestor owns the resource of is waiting for the resource, and by what RET=parameter was specified. • When the requestor owns the resource and specified RET=HAVE and SMC=STEP, XPROCENQ sets the RMC indicator in the current QEL and sets a return code of 8 in PELXRET. • When the requestor owns the resource and specified RET=CHNG, XPROCENQ attempts to change the resource's status from shared to exclusive. If no other requestor is presently sharing the resource, XPROCENQ changes the resource's status to exclusive and sets a return code of zero in PELXRET. If the requestor already owns the resource exclusively, XPROCENQ sets a return code of zero in PELXRET. If another requestor is presently sharing the resource, the resource's status cannot be changed. XPROCENQ sets a return code of 4 in PELXRET to show the request failed. • When the requestor owns the resource and specified RET=HAVE, RET=USE, RET=TEST, or ECB=, XPROCENQ sets a return code of 8 in PELXRET. • When the requestor owns the resource and specified RET=NONE, XPROCENQ sets an abend code of X'138' in QWAERR. • When the requestor is waiting for the resource and specified RET=HAVE, USE, CHNG, TEST, or ECB= , XPROCENQ sets a return code of 20. • When the requestor is waiting for the resource and specified RET=NONE, XPROCENQ sets an abend code of X'138' in QWAERR.		
23 Because this is the first request for this resource, XPROCENQ cannot test or change the status of the resource. XPROCENQ sets a return code of zero for a TEST and an 8 for a CHNG in the PELXRET and returns to the caller.		
24 If the request cannot be handled (flag QWA7C0EX is on), a local-resource ENQ must be rejected or a global-resource ENQ must be completed and then undone. This is done by calling subroutine XGLDEQG, which initializes a DEQ QWB and places it on the ring processing request queue (GVTREQQ).		
25 XPROCENQ obtains, initializes, and chains a QEL to this QCB (to represent this requestor.) If this request does not already have a QXB, XPROCENQ obtains a QXB from the SMPL and chains it to the QEL.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

Diagram GRS-30. ISGGNQDQ -- ENQ/RESERVE Processing (Part 23 of 24)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

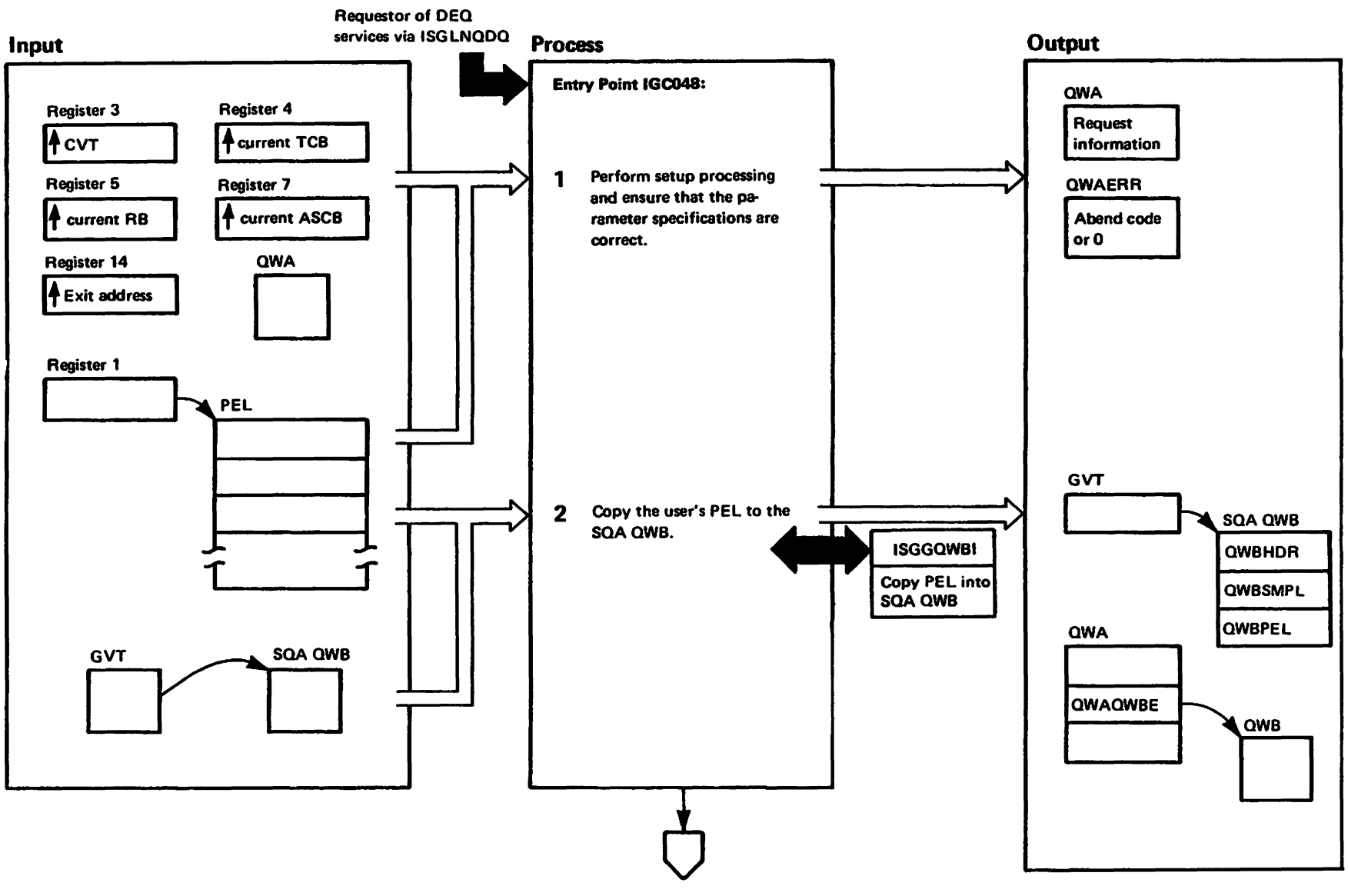
"Restricted Materials of IBM"
Licensed Materials – Property of IBM

Diagram GRS-30. ISGGNQDQ – ENQ/RESERVE Processing (Part 24 of 24)

Extended Description	Module	Label
26 XPROCENQ obtains, initializes, and chains a QEL to this QCB to represent the requestor. XPROCENQ also chains the QEL to: • The ASCB local QEL queue if this is a local request. • The ASCB global QEL queue if this is a global request. • The SYSID/ASID hash table if this is a global request from another system. On behalf of ISGGRP00, XPROCENQ decreases by one the count of global resource requests (QWAGRES) for each QEL that was not placed in the queue for a global resource request. The task global resource count (TCBGRES) will be decreased by the value in QWAGRES after ISGGRP00 posts ISGGWAIT. If this request does not already have a QXB, XPROCENQ obtains a QXB from the SMPL and queues it to the QEL. For RET=HAVE or USE, and ECB= requests, XPROCENQ sets a return code of zero in the PELXRET. *Note:* The return code is only set if the request originated from the current system. Each system in the ring sets its own return codes. Serialization of the local queue is through use of the CMSEQDQ lock. Serialization of the global queue is through use of the local lock of the global resource serialization address space. If both queues must be serialized, both locks must be held. The caller is responsible for this serialization.		
27 Flags set by module ISGGPGRP determine which SYSEVENTs are issued. XPROCENQ issues SYSEVENTs only for address spaces in the system that is issuing the SYSEVENT. Each system issues SYSEVENTs for its own address space.		
28 Flags set by module ISGGPGRP determine what QELs are posted. A QEL is posted by reducing the wait count in the corresponding QXB. When the QXB wait count is reduced to zero, ISGGNQDQ posts the ECB or SVRB of the requestor. Each system posts ECBs or SVRBs for tasks in its own address spaces.		
29 Flags set by module ISGGPGRP determine what return code is placed in the PELX.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-31. ISGGNQDQ – DEQ Processing (Part 1 of 12)



LY28-1655-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

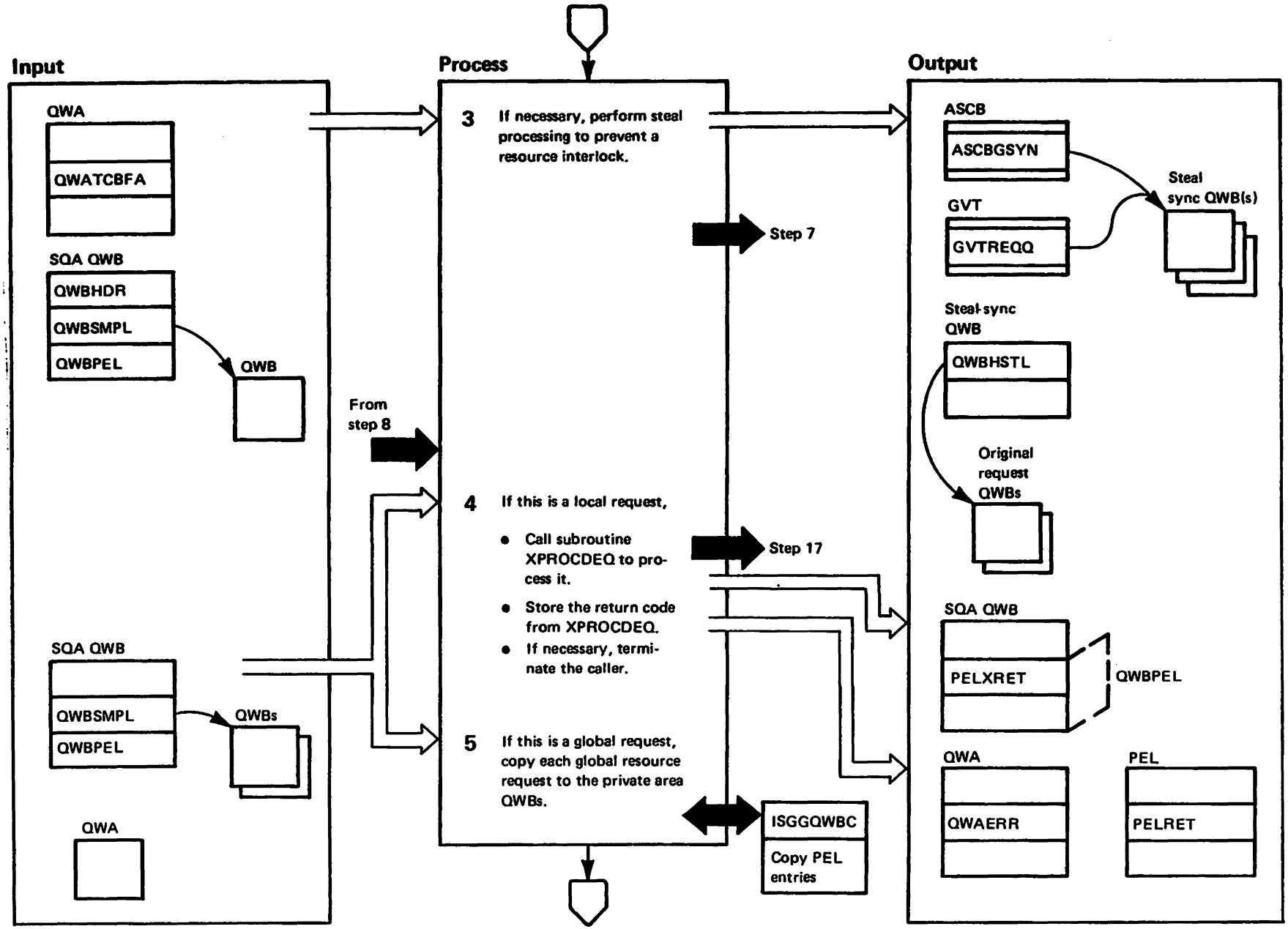
LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

Diagram GRS-31. ISGGNQDQ – DEQ Processing (Part 2 of 12)

Extended Description	Module	Label
ISGGNQDQ processes DEQ requests for specified resources. There are three major sections to DEQ processing. IGC048 is the initial entry point for all DEQ requests, subroutine XPROCDEQ performs the actual processing, and entry ISGGDQ00 is utilized by ISGGRP00 for global requests. At entry point IGC048, ISGGNQDQ first determines if a request is only for local resources, only for global resources, or for a mixture of local and global resources. The processing of local and global requests differs in that requests for local resources can be processed immediately while requests for global resources cannot be processed until the other systems active in the global resource serialization ring have been informed of this request. For local requests ISGGNQDQ calls subroutine XPROCDEQ to perform the DEQ immediately. For global requests ISGGNQDQ calls QWB-copy routine (ISGGQWBC) to build a queue workblock (QWB) for each global request and then ISGGNQDQ places the QWBs on the request queue (GVTREQQ). After the QWB built by ISGGNQDQ for a global request has passed around the global resource serialization ring, ISGGRP00 calls ISGGNQDQ (at entry point ISGGDQ00) to process the DEQ request. ISGGNQDQ calls XPROCDEQ to process the request. Subroutine XPROCDEQ searches the global and local hash tables and finds the appropriate table slots for the requested resources. XPROCDEQ then processes the DEQ requests. **Entry Point IGC048:** **1** ISGGNQDQ establishes an FRR, obtains the requestor's local lock and the CMSEQDQ lock, and initializes the queue workarea (QWA). ISGGNQDQ checks whether the parameters conflict and whether the caller is authorized to request the specified authorized functions. ISGGNQDQ abends requestors when they fail any of these checks.		XSETUP
2 ISGGNQDQ invokes the global resource serialization queue work block initialization routine (ISGGQWBI) to copy the parameter element list (PEL) to the system queue area (SQA) QWB, establish addressability to the global resource serialization address space, and obtain private QWBs if there are global resources.	ISGGQWBI	

Diagram GRS-31. ISGGNQDQ – DEQ Processing (Part 3 of 12)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

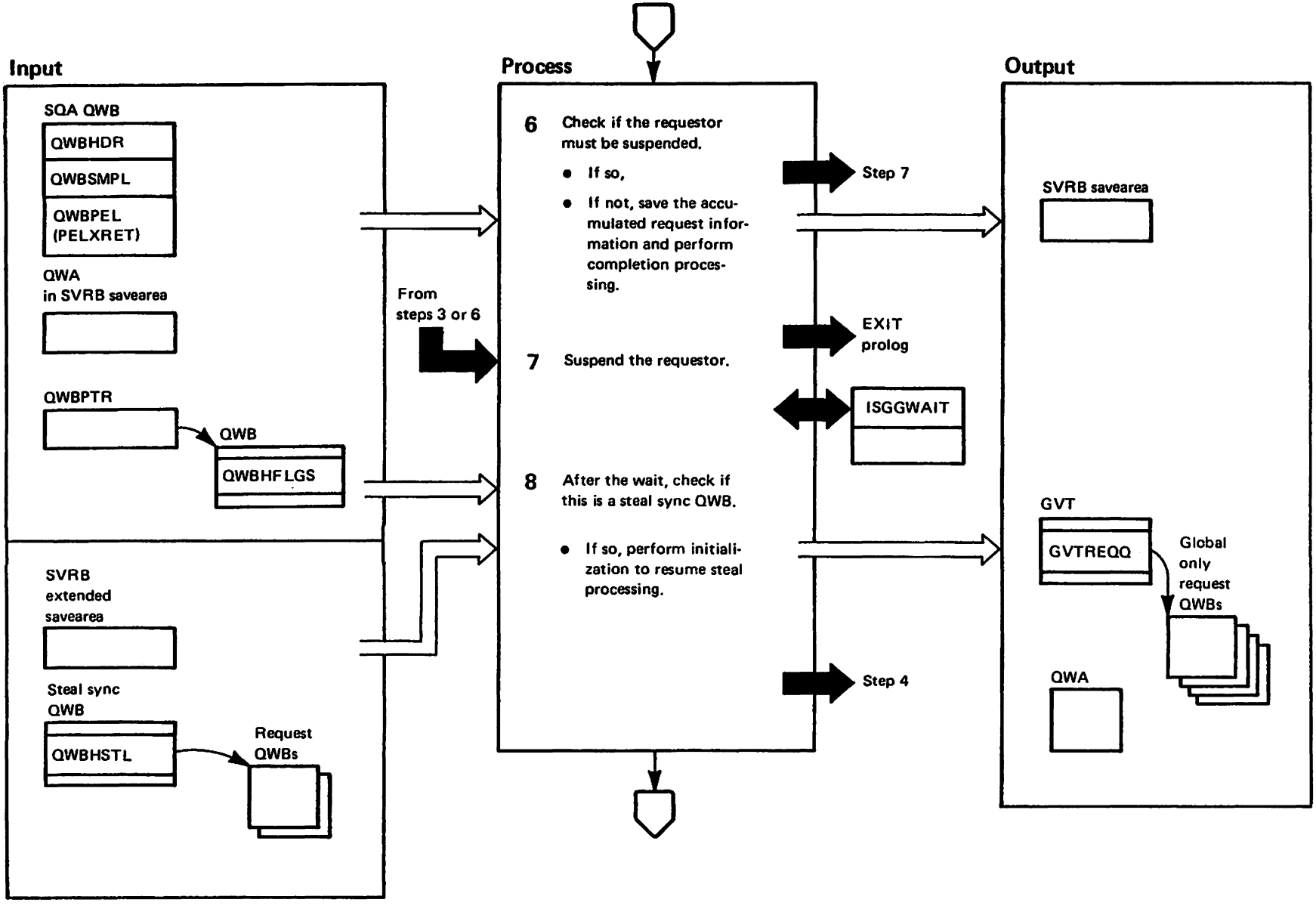
Extended Description	Module	Label
3 When a resource is requested by a task that is part of an abending task structure, and the resource is owned by another task in this same task structure, there can be an interlock over the resource. When a DEQ request is for a local resource, it is not necessary to perform steal processing. However, when a DEQ request is for a global resource, steal processing consists of sending a sync QWB around the ring so that any outstanding ENQs complete before the DEQ request starts. Because the resource is being released, no actual steal is necessary (as is done in ENQ steal processing). DEQ steal processing is done in 3 stages. Stage 1 – ISGGNQDQ constructs a sync QWB containing a pointer to the original request QWB(s). The sync QWB ensures that the request and processing queues are purged of any outstanding ENQs for this resource before the DEQ request starts. Stage 2 – After the sync QWB has processed, the original request queue QWB is processed. XPROCDEQ processes any local resources. ISGGNQDQ copies any global resource requests to global resource serialization private area QWBs. When all resource requests have been copied, ISGGNQDQ places these QWBs on the request queue. Stage 3 – ISGGRP00 processes the global request QWB(s) (ISGGRP00 calls ISGGNQDQ at entry point ISGGDQ00 to do the processing).		
ISGGNQDQ starts the steal processing by calling ISGGQWBC to copy each local and global resource from the SQA QWB into the private area QWBs. The private area QWBs were obtained earlier and chained from the SQA QWB SMPL. When all the PEL entries have been copied, ISGGQWBC initializes a sync QWB. ISGGQWBC moves the sync QWB to the request queue only when no other syncs are outstanding. This ensures that only one sync request is processed at a time. When a sync request is already being processed, ISGGQWBC places the current sync QWB on the end of the ASCB sync queue. ISGGNQDQ (subroutine XPROCDEQ) processes the QWB after previous sync requests	ISGGQWBC	

Extended Description	Module	Label
3 (continued) complete. ISGGNQDQ goes to step 7 to branch enter WAIT while the sync QWB is passed around the ring. (This completes stage 1 steal processing.)		
4 ISGGNQDQ calls subroutine XPROCDEQ to process the local requests (steps 17-18 describe XPROCDEQ's processing.) • ISGGNQDQ uses the SQA QWB PEL as the input PEL. (Note that in the case of steal processing, the input PEL is located in a private area QWB not a SQA QWB.) The SQA QWBSMPL points to the QWB control blocks obtained earlier. ISGGNQDQ passes this input to XPROCDEQ.		XPROCDEQ
• If XPROCDEQ detected an error, it places the abend code in QWAERR and ISGGNQDQ discontinues the PEL scan, performs cleanup, and issues an ABEND.		XDEQSTRC
• If no error was detected, ISGGNQDQ places any return codes in the requestor's PEL.		
5 ISGGNQDQ calls ISGGQWBC to copy each global resource from the SQA QWB into private area QWBs. ISGGNQDQ has previously obtained the private area QWBs and chained them out of the QWB SMPL.	ISGGQWBC	

"Restricted Materials of IBM" Licensed Materials – Property of IBM

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-31. ISGGNQDQ – DEQ Processing (Part 5 of 12)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

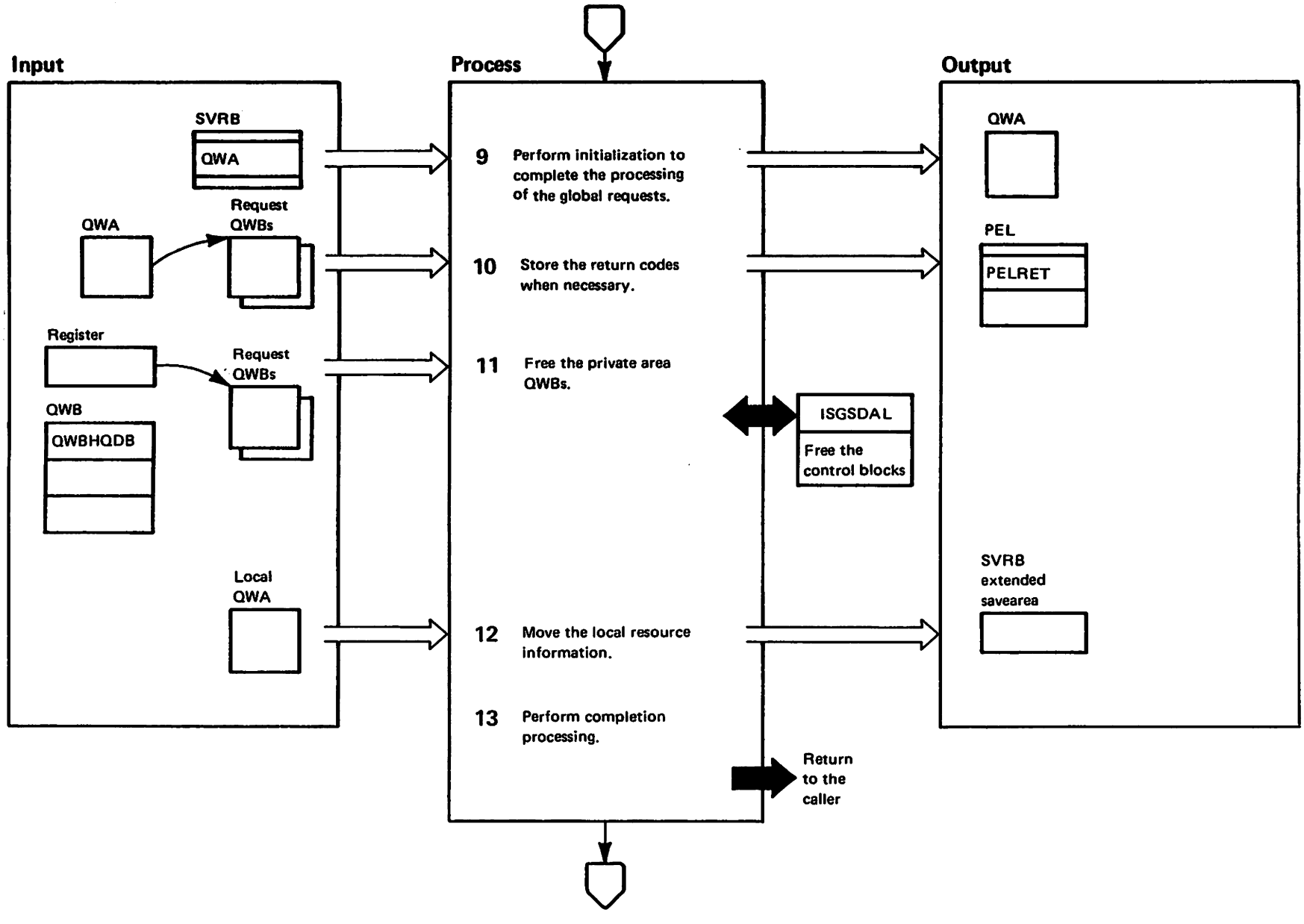
"Restricted Materials of IBM"
Licensed Materials – Property of IBM

Diagram GRS-31. ISGGNQDQ – DEQ Processing (Part 6 of 12)

Extended Description	Module	Label
6 ISGGNQDQ does not suspend the caller if all of the requested resources are local resources. ISGGNQDQ issues a PT instruction to the caller's address space. ISGGNQDQ then does the following: ISGGNQDQ sets the following indicators for completion processing in the global resource serialization address space: • Locks held • RMC indicator • SPOST indicator Note that ISGGNQDQ moves the accumulated request information to the SVRB extended savearea to ensure that the correct data is available for completion processing after the QWA serialization is lost. To complete the request, ISGGNQDQ performs the following completion processing in the requestor's address space. • Reestablishes addressability to the home address space. • Issues an SPOST, if necessary. • Invokes STATUS if reset must complete (RMC) was indicated. • Releases the locks and deletes the FRR.		
7 ISGGNQDQ suspends the requestor if a global resource is requested or stage 1 steal processing is waiting for the sync QWB to be processed. • If this was a mixed resource request, ISGGNQDQ places the address of the private area QWB and the address of the QXB in the SVRB extended savearea. The wait is covered by an ESTAE, ISGGEST0, which cleans up the existing requestor if an error occurs (the FRR is deleted when WAIT is entered). • ISGGNQDQ releases the CMSEQDQ lock. The local lock is retained since it is required by the WAIT interface. The WAIT occurs in the global resource serialization address space under the requestor's TCB. The POST will occur when one QWB is processed by the GRP.		LOCALCOM
8 Steal processing placed a sync QWB on the request queue and waited for it to be processed (see step 3). The POST from ISGGRP00 to ISGGWAIT causes steal processing to resume. ISGGNQDQ performs the following functions: • Acquires the global resource serialization local lock to serialize the global resource queues. • Acquires the CMSEQDQ lock to serialize the QWB pool and the sync request queue. • Decreases the task global resource count (TCBGRES) by the number of global resource requests on the queue for global DEQ requests from which a QEL was removed. • Locates the QWBs that are chained from the steal sync QWB. These private area QWBs are the input for stage 2 steal processing.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-31. ISGGNQDQ – DEQ Processing (Part 7 of 12)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

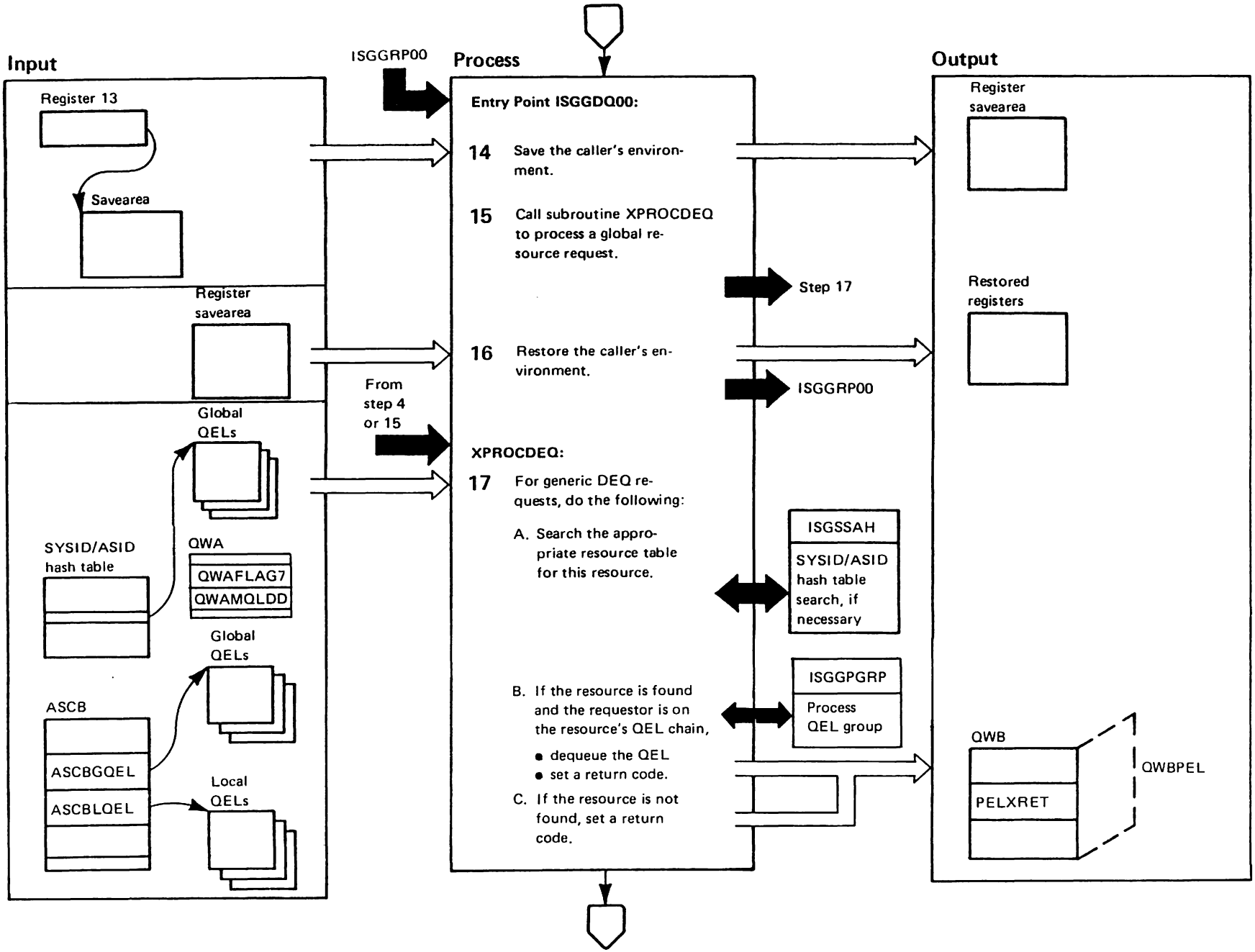
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-31. ISGGNQDQ – DEQ Processing (Part 8 of 12)

Extended Description	Module	Label
9 This step is performed after ISGGRP00 has updated the global-resource QCBs, QELs, and QXBs (by calling ISGGDQ00). This step is executed because the ISGGRP00 posted the suspended SVRB in the requestor's address space. The posted SVRB must store return codes in the requestor's PEL, free the QWB(s) in the private area of the global resource serialization address space, and then exit to the DEQ requestor. In order to complete the processing of the global requests, ISGGNQDQ obtains the requestor's local lock and the CMSEQDQ lock (in order to free the QWB) and copies the data saved in the SVRB extended savearea back into the QWA.		
10 ISGGNQDQ moves the return codes from the QWB PEL entry to the requestor's PEL entry. ISGGNQDQ will also issue an ABEND when the return codes indicate that one is needed.		
11 ISGGNQDQ calls ISGSDAL to free the QWBs defining this request. (Private area QWBs will not exist unless a request was for a global resource).	ISGSDAL	
12 ISGGNQDQ moves the local QWA to the SVRB extended savearea. This is necessary so that the completion data can be referenced after addressability is reestablished to the home address space and the locks are released.		
13 ISGGNQDQ performs the following completion processing: • Reestablishes addressability to the home address space. • Invokes STATUS when reset must complete (RMS) was indicated. • Releases the locks and deletes the FRR.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-31. ISGGNQDQ – DEQ Processing (Part 9 of 12)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-31. ISGGNQDQ – DEQ Processing (Part 10 of 12)

Extended Description	Module	Label
Entry Point ISGGDQ00:		
14 ISGGRP00 uses entry point ISGGDQ00 as an interface to reach subroutine XPROCDEQ. ISGGNQDQ executes under a task in the global resource serialization address space. It updates global resource QCBs, QELs, and QXBs and then returns them to ISGGRP00. ISGGNQDQ saves ISGGRP00's registers and the savearea address before calling XPROCDEQ.	ISGGNQDQ	ISGGDQ00
15 XPROCDEQ processes a global request. Steps 17-18 describe XPROCDEQ's processing.		XPROCDEQ
16 ISGGNQDQ restores ISGGRP00's registers and returns to ISGGRP00.	ISGGRP00	
17 For each PEL entry in the request, XPROCDEQ does the following:		
A. If the request originated from the current system, XPROCDEQ searches the ASCB QEL queues. These are queues of QELs representing both local and global resource requests for the address space defined by this ASCB. There exists a separate queue for local requests and for global request.		
If the request originated from another system in the global resource serialization ring, XPROCDEQ calls ISGSSAH to obtain the hash table slot that points to the QELs for those global resources requested by other systems in the global resource serialization ring. For each QEL defined for this SYSID/ASID, a match occurs when: • The SYSID/ASIDs are equal and • The QNAME equals the QNAME in the QCB		
B. When the resource is found, XPROCDEQ determines if the requestor owns the resource or is waiting for the resource by calling the QEL group processing routine (ISGGPGRP). Users are not permitted to DEQ a resource unless they own it or specified ECB= on the corresponding ENQ. If the requestor owns the resource, XPROCDEQ dequeues it now.	ISGGPGRP	

Extended Description	Module	Label
C. If the request originated from the current system and is a global resource, XPROCDEQ increases the count of global resources (QWAGRES) by one for each QEL removed from the queue. This count is used to decrease the task global resource count (TCBGRES) after ISGGRP00 posts ISGGWAIT.		
XPROCDEQ sets return codes in PELXRET when the request is processed on the requesting system and RET=HAVE was specified.		
Return code=0 – The resource was found and the requestor owned the resource or the resource was found and the requestor was waiting but specified ECB= on the corresponding ENQ. In both cases the resource was dequeued.		
Return code=4 – The resource was found but the requestor is a waiter who did not specify ECB= on the corresponding ENQ.		
Return code=8 – The resource QEL for this request was not found.		
Return code=NONE– RET=NONE was specified or this was an internally generated DEQ.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

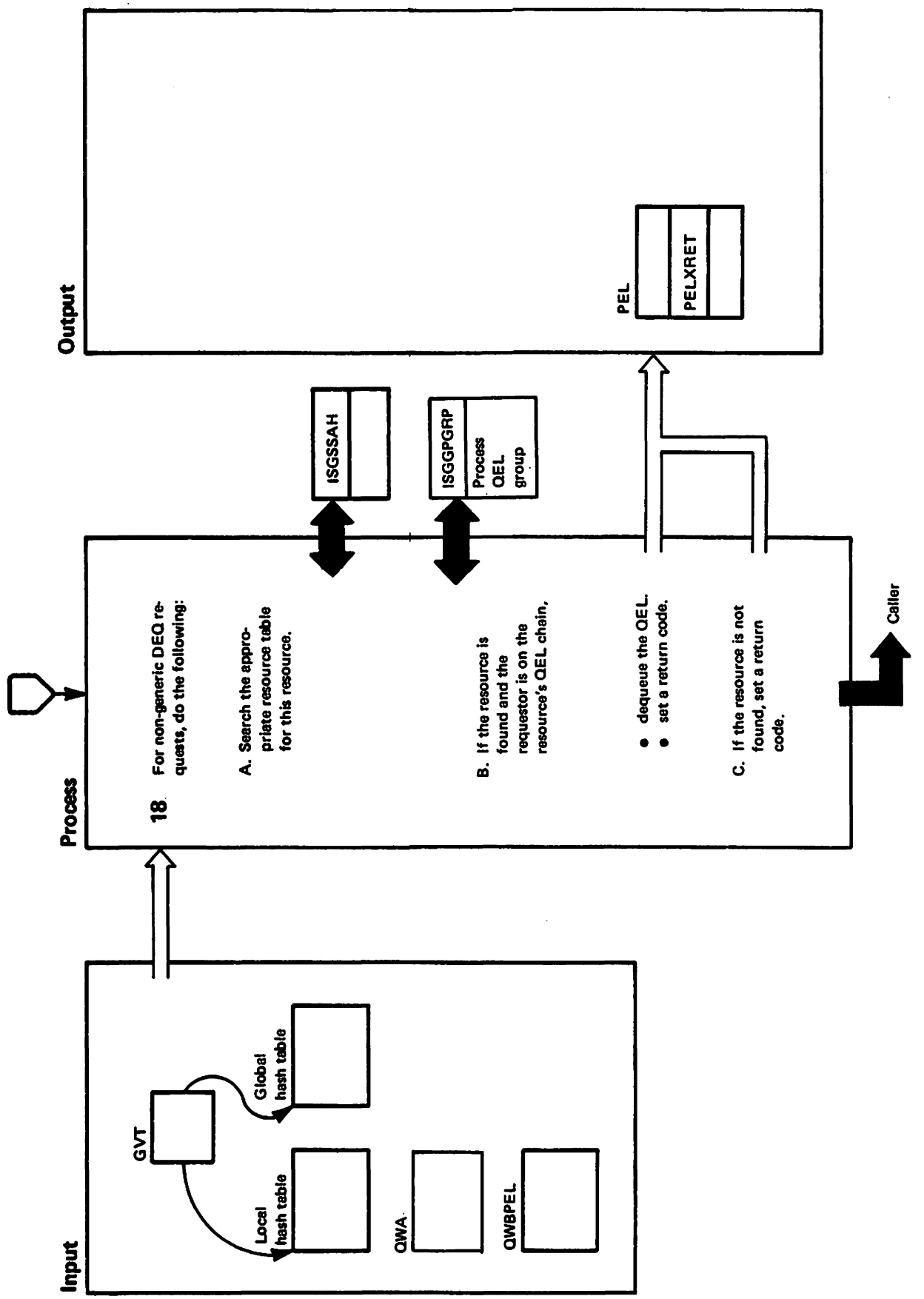


Diagram GRS-31. ISGGNQDQ – DEQ Processing (Part 11 of 12)

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-31. ISGGNQDQ – DEQ Processing (Part 12 of 12)

Extended Description | **Module** | **Label**

18 For non-generic DEQ requests, XPROCDEQ does the following:

A. If the current request represents a request for a local resource, the local hash table is searched; otherwise the global hash table is searched. A match occurs when:

- The scopes are equal and
- The SYSID/ASIDs are equal and
- The input resource equals a QCB resource name.

B. DEQs are not permitted unless the resource is owned, or one of the following conditions is met:

- The DEQ requestor previously issued an ECB=ENQ or RESERVE.
- The DEQ represents an internal global resource serialization DEQ.
- The DEQ requestor previously issued a MASID=ENQ or RESERVE.

If the DEQ is permitted, XPROCDEQ dequeues the resource here.

Return codes are set in PELXRET when the request is processed on the requesting system and RET=HAVE was specified.

Return code=0 – The resource was found and the requestor owned the resource or the resource was found and the requestor was waiting but specified ECB= .

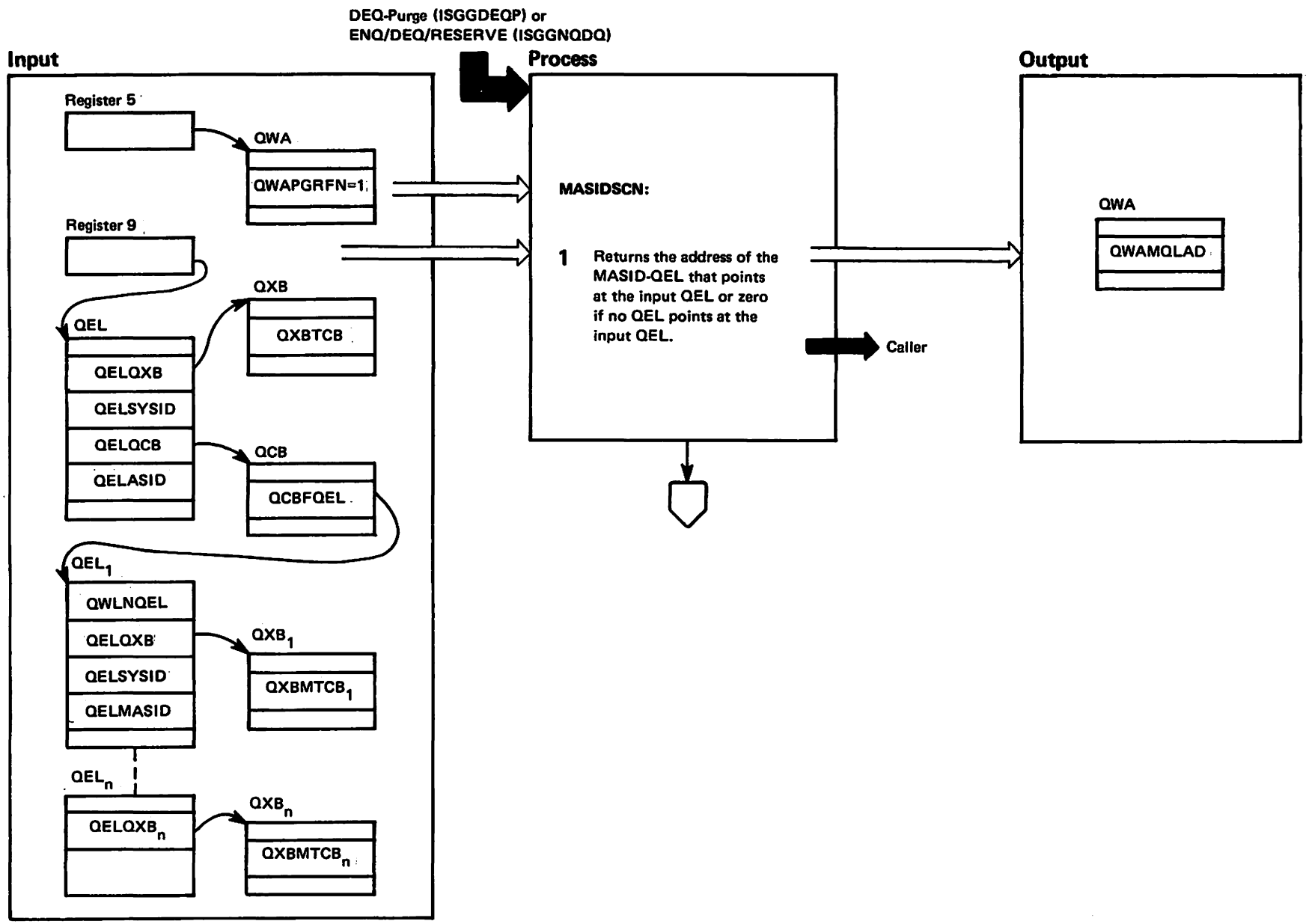
Return code=4 – The resource was found but the requestor is a waiter who did not specify ECB= or MASID=.

Return code=8 – The resource QEL for this request was not found.

Return code=NONE – RET=NONE was specified or this was an internally generated DEQ.

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-32. ISGGPGRP – QEL Group Processing Routine (Part 1 of 10)



"Restricted Materials of IBM"
Licensed Materials – Property of IBM

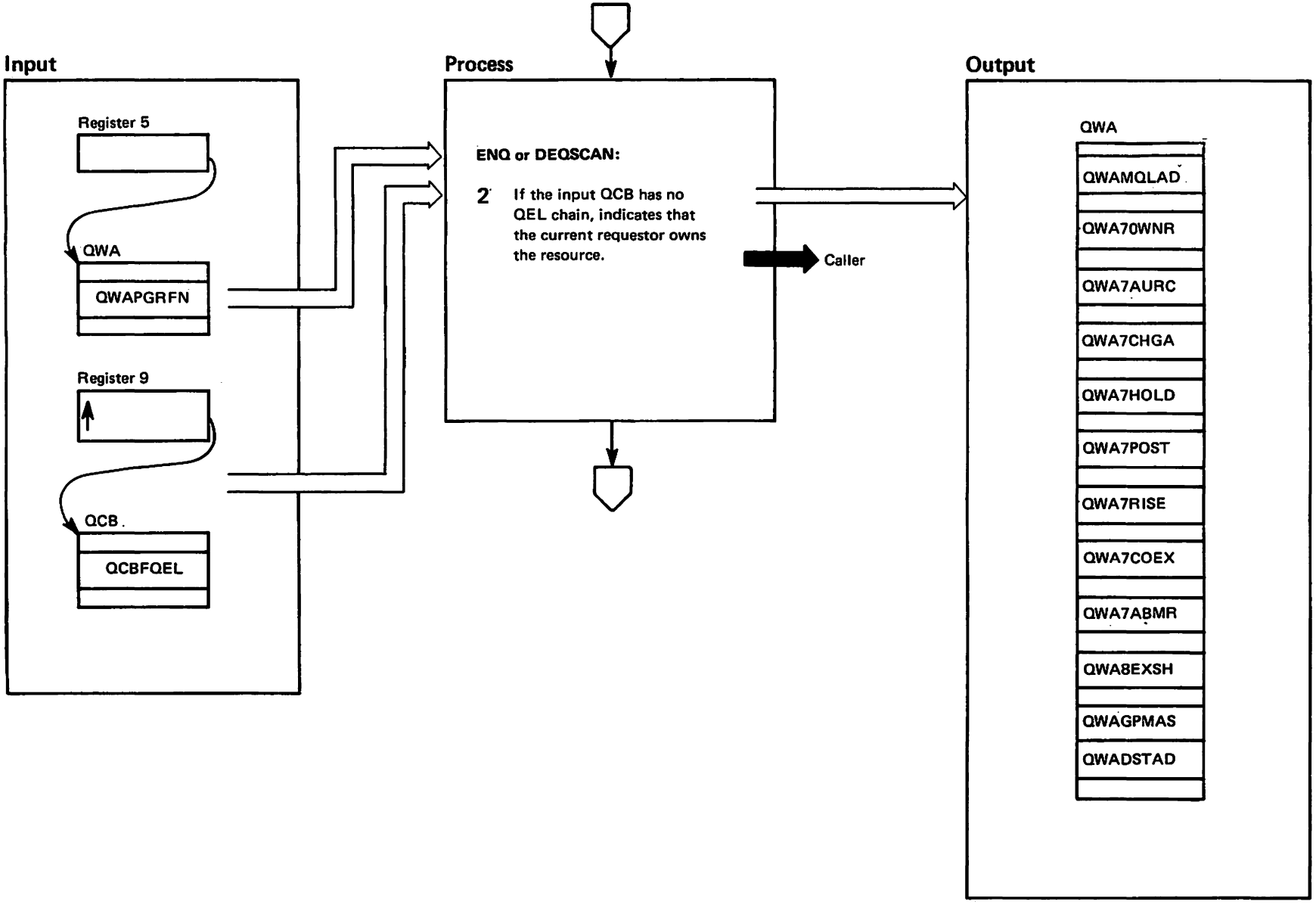
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-32. ISGGPGRP – QEL Group Processing Routine (Part 2 of 10)

Extended Description	Module	Label
ISGGPGRP processes QEL groups for task requests. ISGGPGRP determines whether: • A QEL exists for a task • The task requested exclusive or shared control • The task owns the resource ISGGPGRP transfers this information to the caller by setting the appropriate flags and filling address pointers. A QCB represents a resource; each QCB has its own QEL chain that consists of one or more QELs. A QEL represents a task that requested the resource represented by the QCB. The first QEL on the QEL chain represents the task that issued the oldest outstanding request for the resource. The last QEL on the QEL chain represents the task that issued the newest outstanding request for the resource. When a task issues a DEQ for the resource, its QEL is removed from the QEL chain. A task can appear, at most, once on the QEL chain of a QCB. An ENQ or RESERVE is rejected if the requesting task already appears on the QEL chain of the requested resource. Each QEL has a flag (QELSHR) that indicates whether the task requested exclusive control or shared control. The QEL chain can be divided into QEL groups, representing tasks that can share use of the resource. A QEL group consists of either (a) one exclusive control QEL, or (b) any number of successive shared control QELs. Successive shared-control QELs are considered a single group, because they represent tasks that can share the resource. The task (or tasks) in the first QEL group are the task (or tasks) that own the resource. QEL groups other than the first group represent tasks that must wait for ownership of the resource. A task owns the resource if it appears in the first QEL group. This is indicated by output flag QWA7OWNR.		
A task can use the resource if it owns the resource, or if it "points at" the QEL of a task that owns the resource; this is indicated by the otuput flag QWA7AURC. A QEL "points at" another QEL via the QELMASID and QXBMTCB fields; these fields are non-zero only if the requesting task used the MASID= and MTCB= operands of ENQ (or RESERVE) when the QEL was created. **MASIDSCN function:** 1 If the input function code is MASIDSCN, ISGGPGRP searches for the address of a MASID QEL that points at the input QEL and returns it to the caller. If none of the MASID QELs point at the input QEL, ISGGPGRP returns a zero to the caller. The input QEL has the address of its QCB. ISGGPGRP searches the QEL chain of this QCB for a QEL that meets the following conditions: (a) it has the same SYSID as the input QEL, and (b) its QELMASID value equals QELASID of the input QEL, and (c) its QXBMTCB value equals QXBTCB of the input QEL. If ISGGPGRP finds such a QEL, ISGGPGRP sets QWAMQLAD to point at it. If ISGGPGRP cannot find such a QEL, ISGGPGRP sets QWAMQLAD to zero. ISGGPGRP then returns to its caller.	ISGGPGRP	

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-32. ISGGPGRP -- QEL Group Processing Routine (Part 3 of 10)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

Diagram GRS-32. ISGGPGRP – QEL Group Processing Routine (Part 4 of 10)

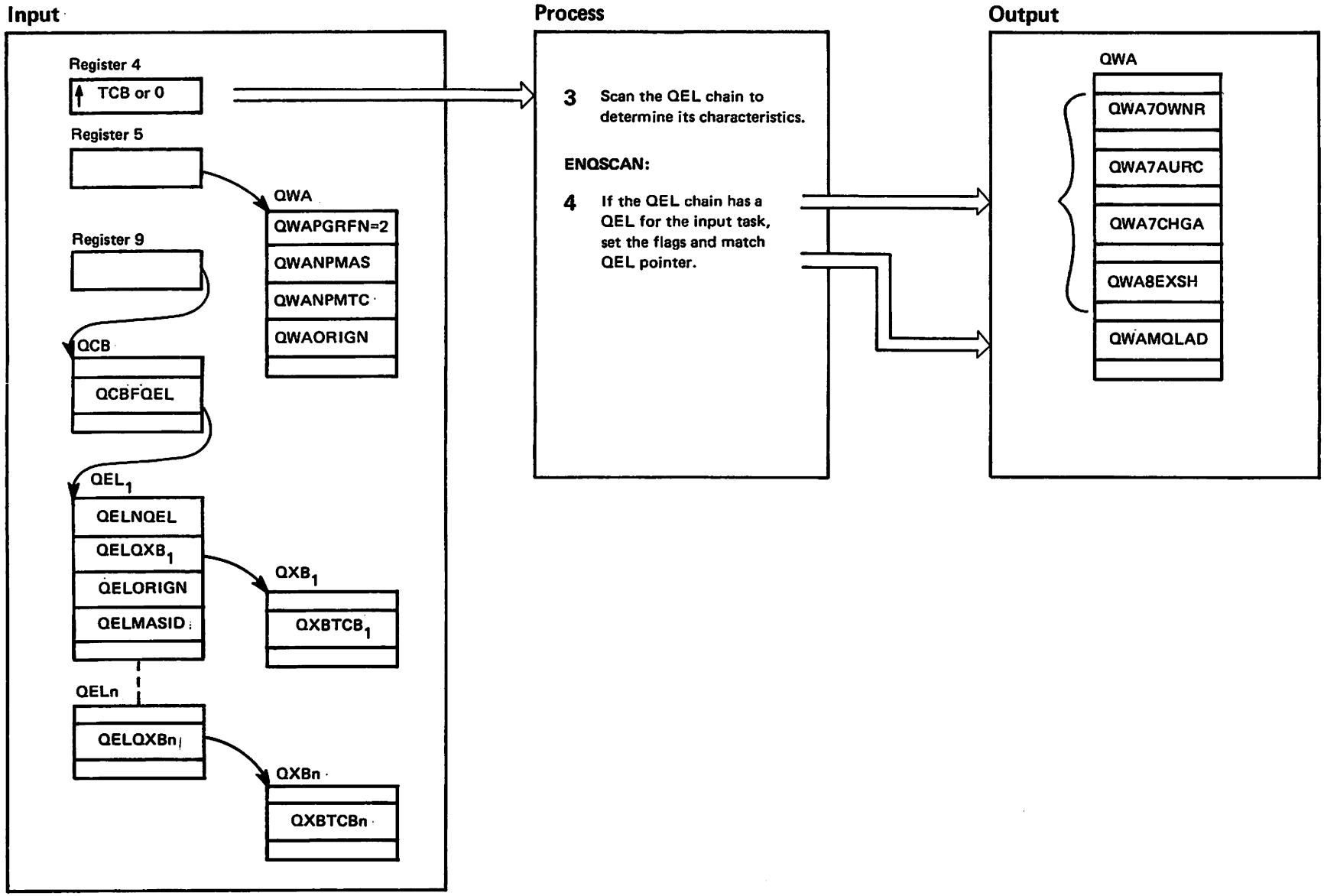
Extended Description | **Module** | **Label**

ENQSCAN and DEQSCAN function:

2 If the requesting task does not appear on the QEL chain, ISGGPGRP indicates this by setting QWAMQLAD to zero. ISGGPGRP turns QWA7OWNR on to indicate that any new QEL added to the QEL chain will describe a task that owns the resource. ISGGPGRP turns QWA7AURC on to indicate that the task described by the newly-added QEL can use the resource. Then ISGGPGRP returns to the caller. If the caller is performing an ENQ or a RESERVE, the caller can add a new QEL to the QEL chain. If the caller is performing a DEQ, the caller can reject the DEQ, because the requesting task is not on the QEL chain.

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-32. ISGGPGRP – QEL Group Processing Routine (Part 5 of 10)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-32. ISGGPGRP – QEL Group Processing Routine (Part 6 of 10)

Extended Description	Module	Label

3 ISGGPGRP searches the QEL chain to determine the following:

- Whether the QEL chain has a match-QEL for the input task (QWAMQLAD)
- Whether the input MASID and MTCB operands, if there are any, match the ASID/TCB values in some QEL that is on the QEL chain (QWAGPMAS)
- How many QELs are in each of the first three QEL groups on the QEL chain, and what is the address of the first QEL of each group
- Whether a fourth QEL group exists

ENQSCAN function:

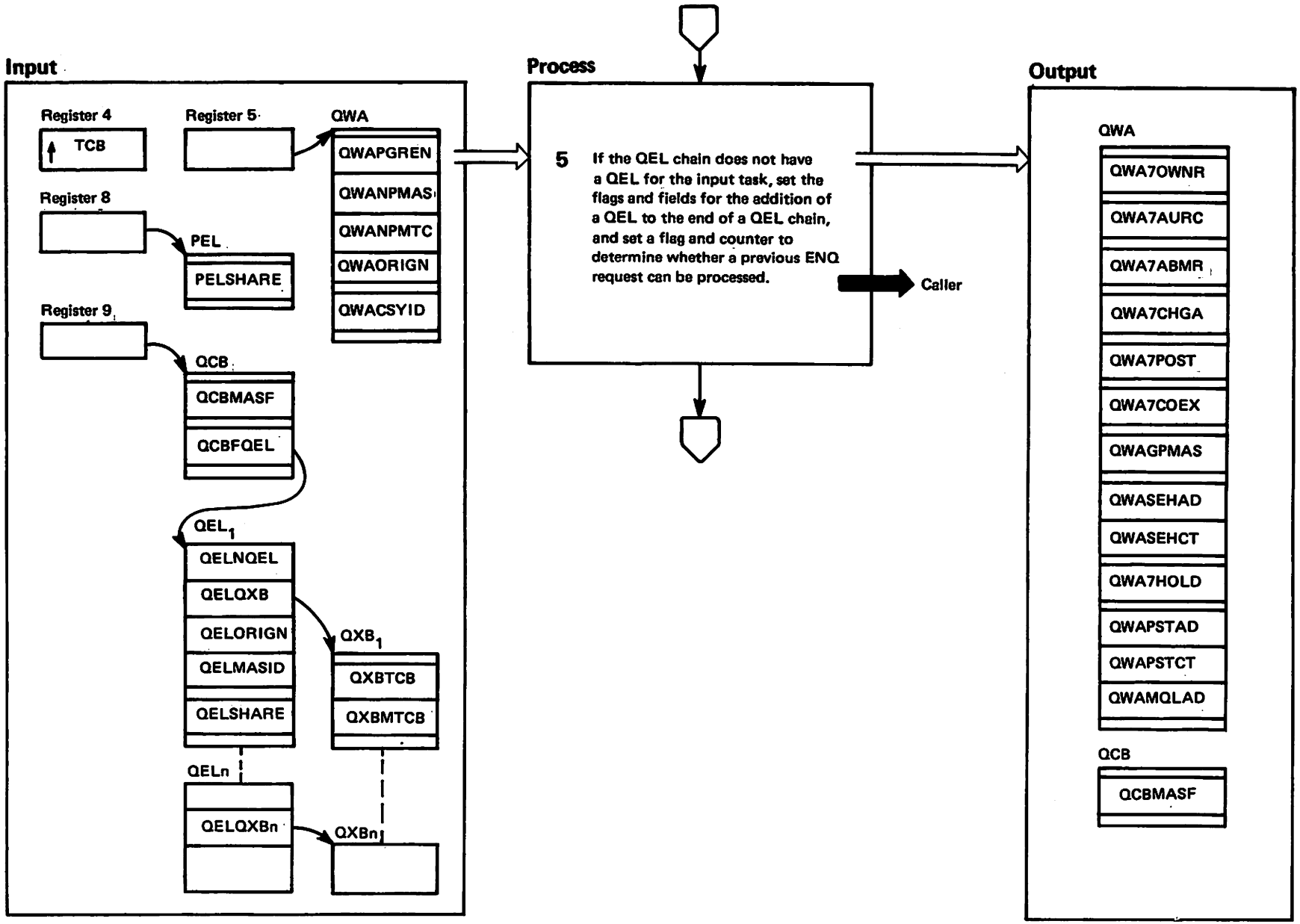
4 If the QEL chain has a QEL for the input task, ISGGPGRP sets flags to indicate whether the input task:

- Owns the resource
- Can use the resource
- Is the sole owner of the resource
- Has exclusive control of the resource

A task has exclusive control of the resource if it owns the resource and appears in an exclusive control QEL. A task is the sole owner of the resource if it appears in the only QEL of the first QEL group. ISGGPGRP also sets the match-QEL pointer.

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-32. ISGGPGRP – QEL Group Processing Routine (Part 7 of 10)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-32. ISGGPGRP – QEL Group Processing Routine (Part 8 of 10)

Extended Description **Module Label**

5 If the QEL chain does not have a QEL for the input task, ISGGPGRP sets flags to indicate how the QEL chain will appear after the caller adds a QEL for the requesting task to the end of the QEL chain. The flags show whether the requesting task:

- Owns the resource
- Can use the resource
- Caused contention with other tasks that are using the resource
- Points to another QEL

ISGGPGRP also sets a flag and counter to tell the caller whether a previous request can now be satisfied. Then ISGGPGRP returns control to the caller.

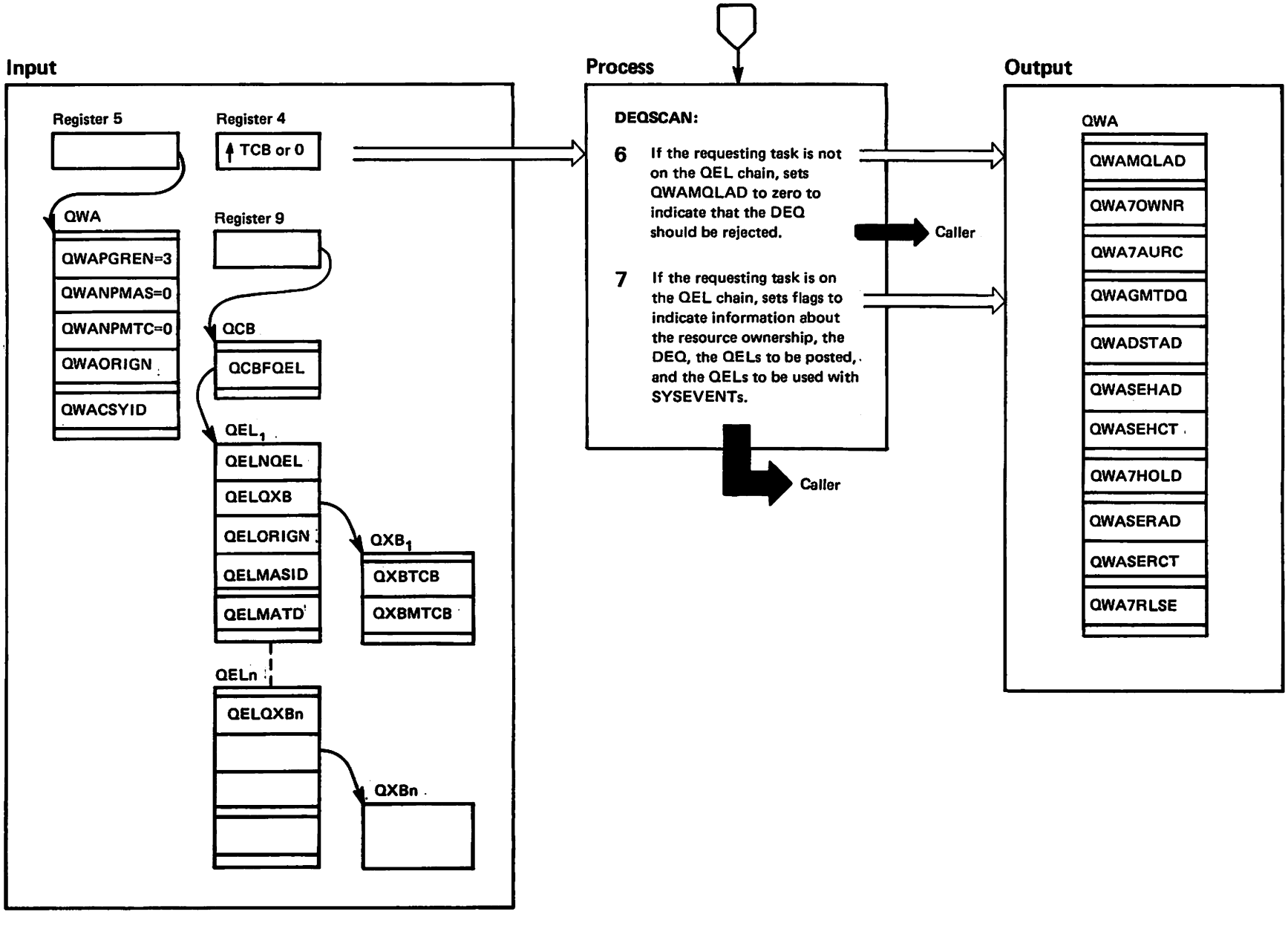
When a request causes contention, ISGGPGRP sets certain flags to tell its caller to issue ENQHOLD SYSEVENTs for the tasks of the first QEL group. A request causes contention if the new QEL, which the caller will add to the end of the QEL chain, is the first QEL of the second QEL group.

The new QEL should "point at" another QEL if the new ENQ (or RESERVE) uses MASID= or MTCB= operands that match the ASID and TCB of some task that appears in a QEL on the QEL chain. If the new QEL should "point at" another QEL, ISGGPGRP sets QWAGPMAS to the value that must be place in the QELMASID field of the new QEL.

A previous ENQ request can be satisfied if the previous ENQ requested exclusive-control and used the MASID= or MTCB= operands to "point at" a shared control QEL that shared ownership of the resource. One such request can be satisfied when all shared control QELs of the first QEL group are "pointed at" by exclusive control QELs. This can occur only when ISGGPGRP processes an exclusive control ENQ (or RESERVE) with MASID= or MTCB= operands.

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-32. ISGGPGRP – QEL Group Processing Routine (Part 9 of 10)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

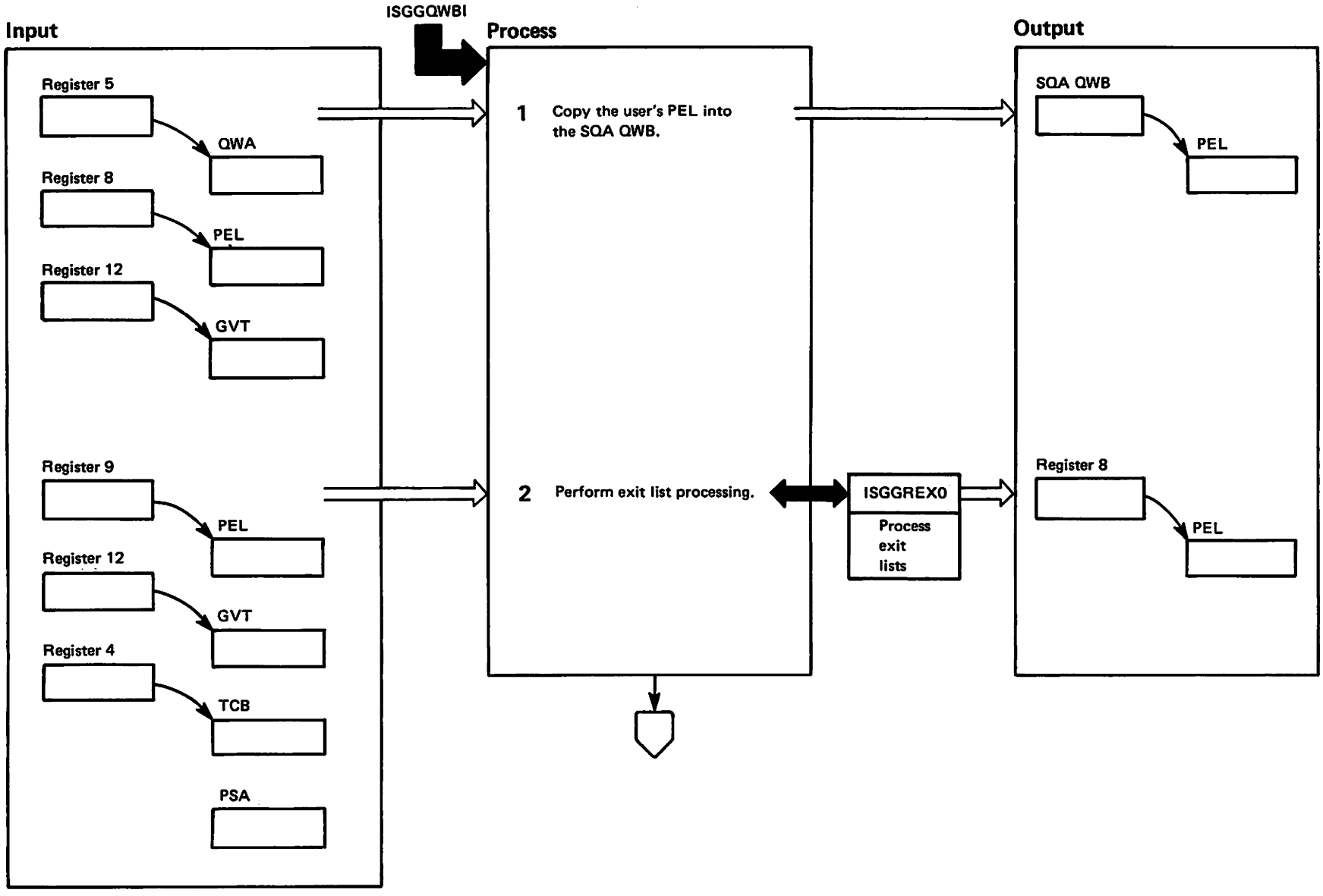
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-32. ISGGPGRP -- QEL Group Processing Routine (Part 10 of 10)

Extended Description	Module	Label
DEQSCAN function:		
6 If the requesting task is not on the QEL chain, ISGGPGRP returns to the caller with QWAMQLAD equal to zero to indicate that the DEQ should be rejected.		
7 If the requesting task is on the QEL chain, ISGGPGRP sets the flags to indicate whether:		
• The requesting task owns the resource		
• The DEQ is illegal		
The DEQ is illegal if it removes a QEL that is "pointed at" by some other QEL on the QEL chain.		
ISGGPGRP sets flags, counts, and addresses to indicate to the caller which QELs should be posted and which QELs should be used with SYSEVENTs.		
The caller posts QELs if:		
• The caller will be removing the only QEL of the first QEL group		
• The caller will be removing an exclusive control QEL that makes up the second QEL group, thus allowing shared control QELs in the third QEL group to share the resource with shared control QELs in the first QEL group		
• The caller will be removing a QEL from the first QEL group, and all remaining QELs in the first QEL group are "pointed at" by exclusive control QELs.		
The caller issues a SYSEVENT ENQRLSE if the caller will be removing a QEL from the QEL chain and if the QEL was previously used with a SYSEVENT ENQHOLD. If the caller removes the QWAMQLAD QEL from the QEL-chain and the removed QEL is the only QEL in its group, the caller issues a SYSEVENT ENQHOLD and a SYSEVENT ENQRLSE to reflect the contention that exists.		
ISGGPGRP performs a deferred steal if the DEQ removes a QEL which "points at" some other QEL, which was previously marked for a deferred steal (flag QELMATD). ISGGPGRP sets field QWADSTAD to "point at" the QEL that was marked for the deferred steal.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-33. ISGGQWBI -- Queue Work Block Initialization Routine (Part 1 of 6)



"Restricted Materials of IBM"
Licensed Materials - Property of IBM

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-33. ISGGQWBI – Queue Work Block Initialization Routine (Part 2 of 6)

Extended Description	Module	Label
ISGGQWBI, the queue work block initialization routine, moves the requestor's parameter list (PEL) to the system queue area (SQA) queue work block (QWB) and establishes addressability to the global resource serialization address space.		
1 ISGGQWBI copies the user's parameter element (PEL) into the SQA QWB. If the entire PEL cannot be contained in the SQA QWB, ISGGQWBI invokes subroutine XGETQWB to do the following:	ISGGQWBI	XINITQWB
• Establish addressability to the global resource serialization address space via a program call (PC) to entry point ISGGED02.		
• Obtain private area QWB extensions to contain the remainder of the request.		
2 If global resource serialization is active (GVTGRSNA='0'B), ISGGQWBI performs exit list processing as follows:		
• If the request has SCOPE=STEP, local resource processing occurs; exit list processing is not performed.		
• If the request has SCOPE=SYSTEM, ISGGQWBI calls the ISGGSIEX entry point of ISGGREX0 to scan the SYSTEM Inclusion RNL. If a matching resource name is found, the request becomes a global request (SCOPE=SYSTEMS); otherwise the request remains local.	ISGGREX0	ISGGSIEX
• If the request has SCOPE=SYSTEMS or a matching resource name was found in the SYSTEM Inclusion RNL, ISGGQWBI calls the ISGGSEEX entry point of ISGGREX0 to scan the SYSTEMS Exclusion RNL. If a matching resource name is found, the request becomes a local request (SCOPE=SYSTEM); otherwise the request remains global.	ISGGREX0	ISSGSEEX
• If the request is a RESERVE, ISGGQWBI calls the ISGGRCEX entry point of ISGGREX0 to scan the RESERVE Conversion RNL. If a matching resource name is found, the hardware RESERVE is suppressed and the request is serialized by a global ENQ; otherwise, the request remains a RESERVE.	ISGGREX0	ISGGRCEX
During initialization if a matching name was not found in the SYSTEMS Exclusion RNL, ISGGQWBI excludes RESERVE (SCOPE=SYSTEMS) requests from global processing by converting the request to a local ENQ with a hardware RESERVE. ISGGQWBI also issues the message 'ISG066I – RESOURCE NAMED xxx,yyy TEMPORARILY EXCLUDED FROM GLOBAL PROCESSING'.	ISGGQWBI	
ISGGQWBI excludes any DEQ (SCOPE=SYSTEMS) associated with an excluded RESERVE from global processing by treating it as a local resource.		
If the global resource serialization is not active, ISGGQWBI treats all requests as local requests.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

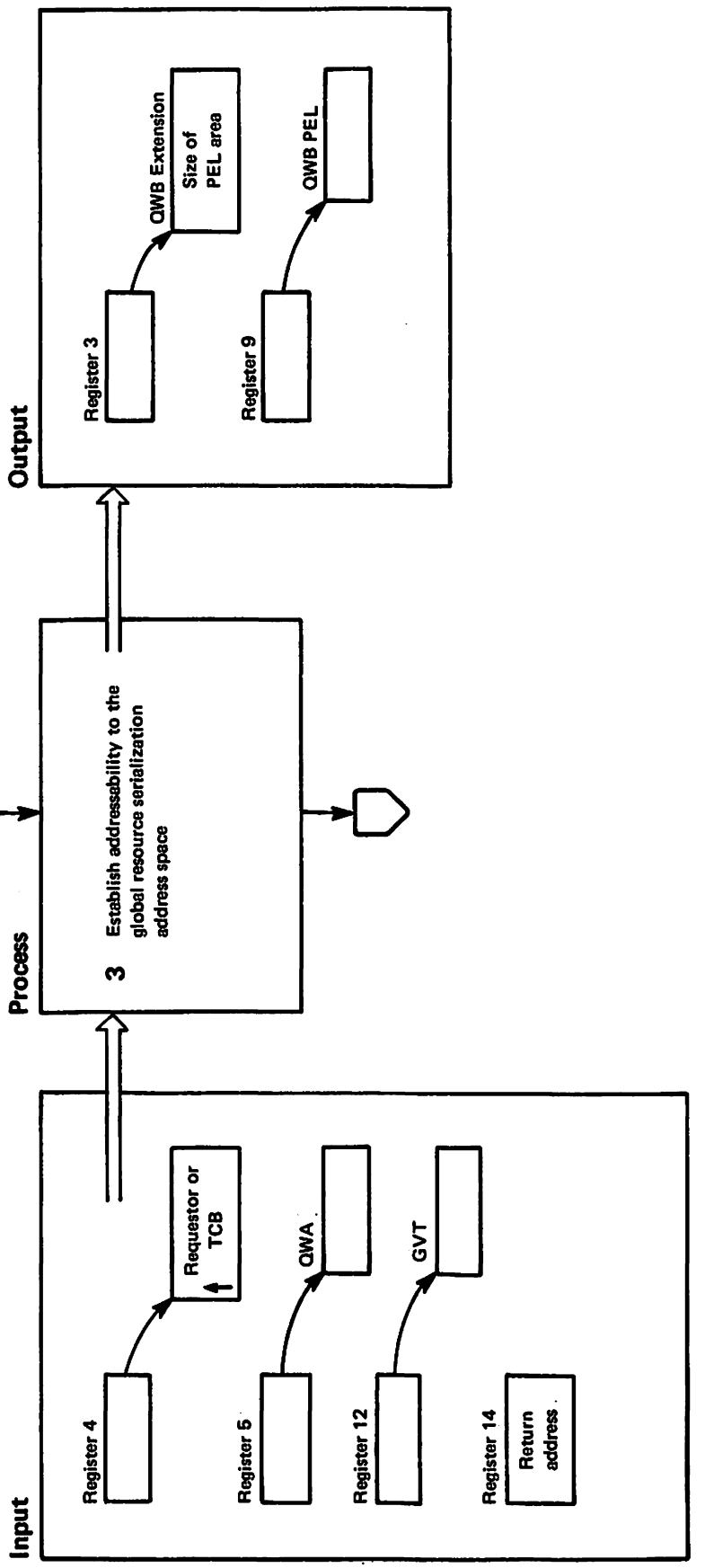


Diagram GRS-33. ISGGQWBI – Queue Work Block Initialization Routine (Part 3 of 6)

LY28-1695-0 (c) Copyright IBM Corp. 1987

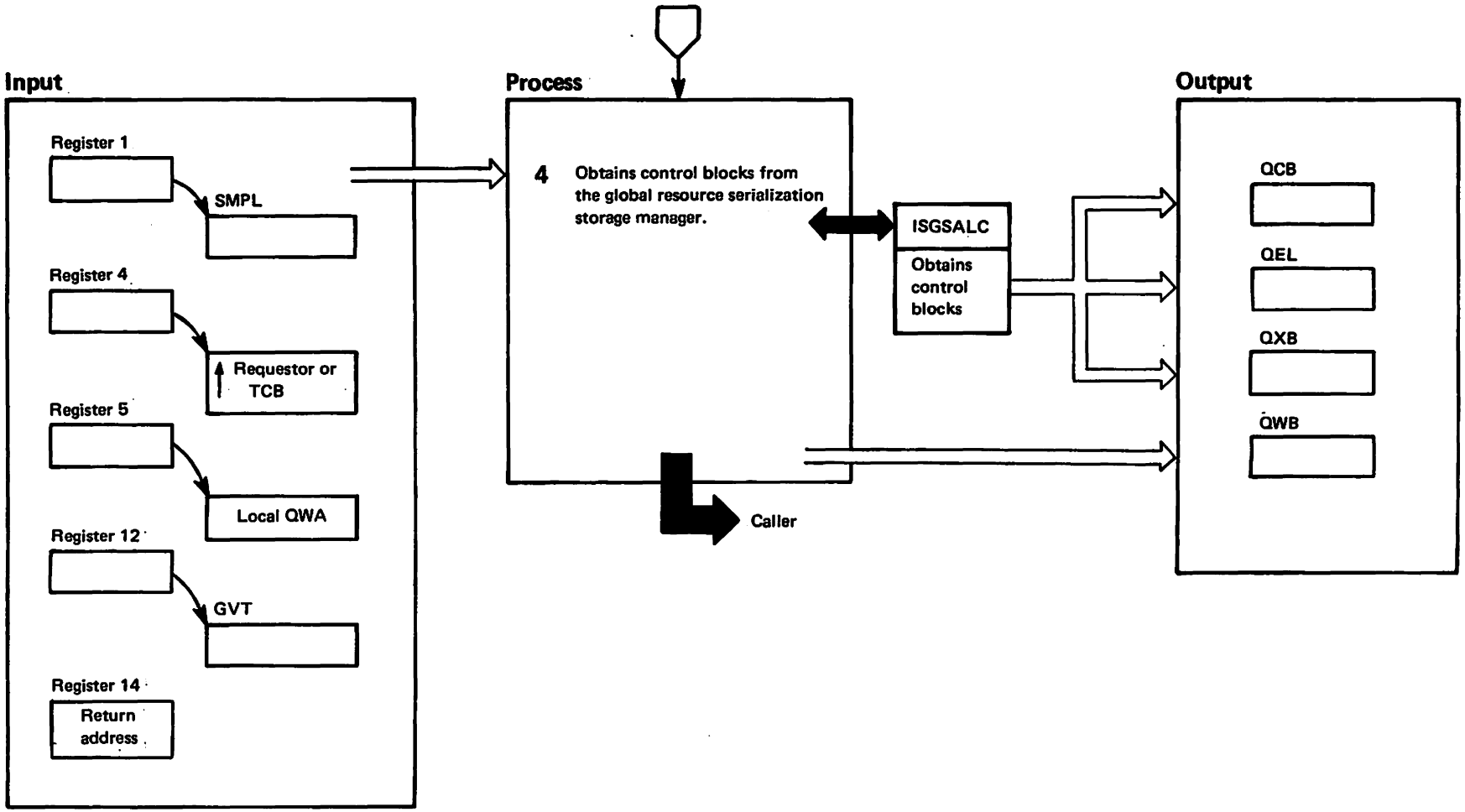
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-33. ISGGQWBI – Queue Work Block Initialization Routine (Part 4 of 6)

Extended Description	Module	Label
3 ISGGQWBI determines if addressability to the global resource serialization address space was established. If not, ISGGQWBI establishes addressability to the global resource serialization address space via a PC to entry point ISGGED01.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-33. ISGGQWBI – Queue Work Block Initialization Routine (Part 5 of 6)



"Restricted Materials of IBM"
Licensed Materials – Property of IBM

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-33. ISGGQWBI – Queue Work Block Initialization Routine (Part 6 of 6)

Extended Description

Module Label

4 ISGGQWBI obtains control blocks from the global resource serialization storage manager (ISGSALC). The control blocks obtained are defined by the input SMPL, and the first QWB will be initialized when the global resources are present.

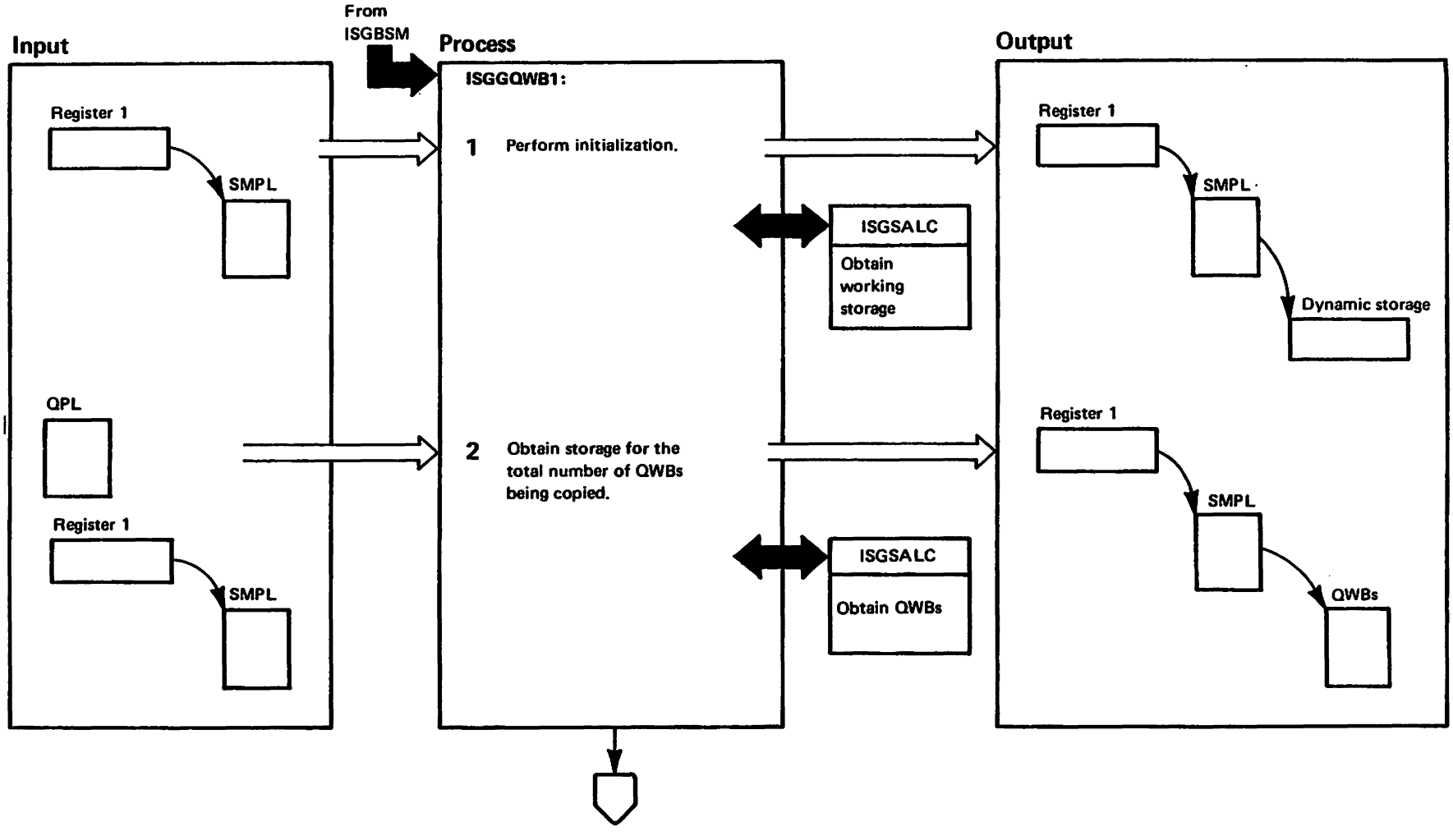
ISGSALC

Recovery Operation

When ISGGQWBI is executing, recovery is provided by the module ISGGFRR0.

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-34. ISGGQWB0 – Queue Work Block Service Routine (Part 1 of 20)



"Restricted Materials of IBM"
Licensed Materials – Property of IBM

LY28-1695-0 (c) Copyright IBM Corp. 1987

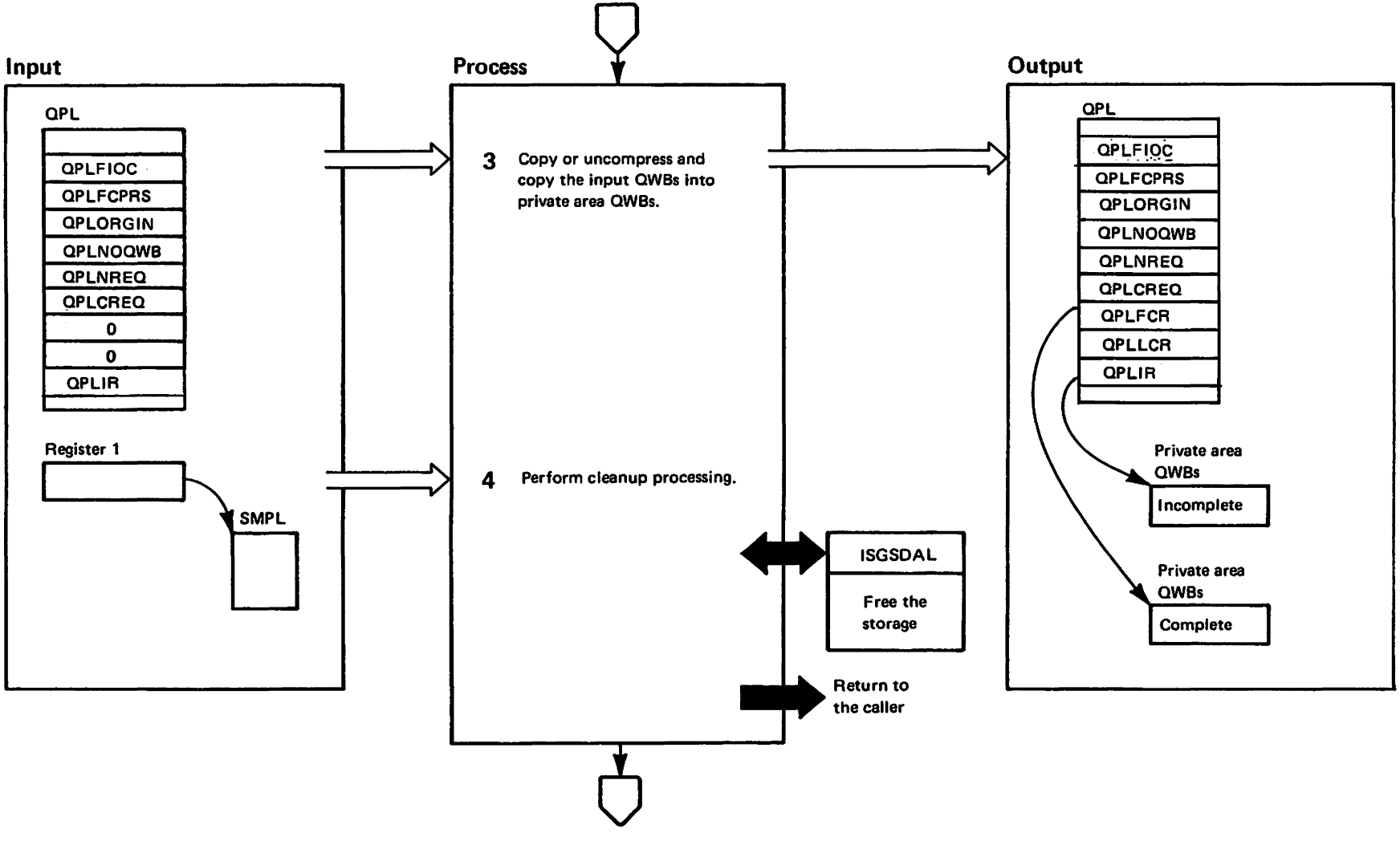
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-34. ISGGQWB0 — Queue Work Block Service Routine (Part 2 of 20)

Extended Description	Module	Label
ISGGQWB0 is a series of service routines. Callers enter ISGGQWB0 at one of six entry points to obtain, return, or initialize queue work blocks (QWBs). See each entry point for more detailed descriptions.		
Entry Point ISGGQWB1		
ISGBSM calls ISGGQWB1 for one of two functions: 1. Copy of (optionally) uncompress and copy the ring system authority (RSA) QWBs to the global resource serialization private area QWBs (copy into the system). 2. Copy or (optionally) compress and copy the resource serialization private area QWBs to the ring system authority (RSA) QWBs (copy out of the system). Input to this routine when copying into the system is a queue parameter list (QPL). The QPL defines the function (copy into or out of the system), compression code (copy or uncompress and copy), location of QWBs and number of QWBs. This parameter list also contains pointers to complete and incomplete request queues.	ISGGQWB0	ISGGQWB1
When a request is incomplete, ISGGQWB1 assumes that subsequent RSAs will contain QWBs defining this request until the request has been completed and copied to the private QWBs.		
1 Initialization consists of establishing addressability, establishing ISGGQWBR as the FRR, obtaining the local lock of the home address space and the CMSEQDQ lock, and invoking ISGSALC to obtain dynamic storage. The address of the storage management parameter list (SMPL) is passed to ISGSALC in register 1.	ISGSALC	
2 ISGGQWB1 determines a "Copy into the system" function and determines how many QWBs need to be copied (QPLNOQWB). It then invokes ISGSALC to obtain storage from the QWB pool for that many QWBs. The address of the SMPL is passed to ISGSALC in register 1. ISGSALC holds the CMSEQDQ lock for serialization of the QWB pool.	ISGGQWB0 ISGSALC	 ISGGQWB1

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-34. ISGGQWB0 — Queue Work Block Service Routine (Part 3 of 20)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials — Property of IBM

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-34. ISGGQWB0 -- Queue Work Block Service Routine (Part 4 of 20)

Extended Description	Module	Label
3 ISGGQWB1 copies or uncompresses and copies the RSA QWBs into the global resource serialization private area QWBs. Virtual addresses within the PEL area of the QWB are re-initialized to reflect these new virtual addresses. The input parameter list (QPL) identifies that this is a copy from the RSA to the global resources serialization private area (QPLFIOC), whether to copy or uncompress and copy the QWB (QPLFCPRS), and whether there is an outstanding incomplete request (QPLIR not zero). If there is an outstanding incomplete request ISGGQWB1 copies or uncompresses and copies the input QWBs associated with the incomplete request ihto private area QWBs. The incomplete request becomes complete when all of the QWBs associated with the request have been initialized. ISGGQWB1 places a request on the completed-request-queue when all of the associated QWBs are not in the input area.	ISGGQWB0	ISGGQWB1
4 ISGGQWB1 invokes ISGSDAL to free the dynamic storage. The address of the SMPL is passed to ISGSDAL in register 1. ISGGQWB1 then releases the CMSEQDQ lock and the local lock, and deletes the FRR. Note that ISGGQWB1 only releases the locks that it obtained.	ISGSDAL	
On return, register 1 points to the input QPL that contains pointers to the first and last completed requests (QWBs) and pointers to any incomplete requests (QWBs).	ISGGQWB0	ISGGQWB1

LY28-1695-0 (c) Copyright IBM Corp. 1987

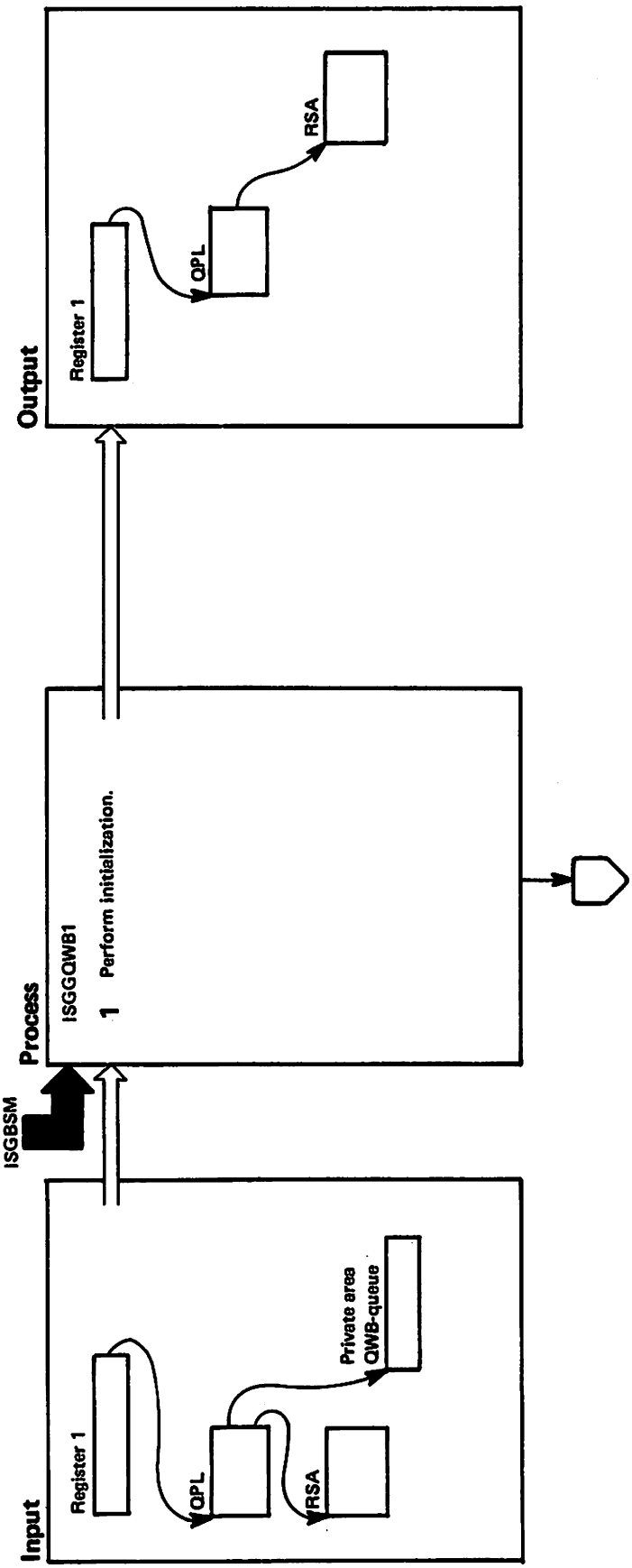


Diagram GRS-34. ISGGQWB0 – Queue Workblock Service Routine (Part 5 of 20)

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

Diagram GRS-34. ISGGQWB0 – Queue Workblock Service Routine (Part 6 of 20)

Extended Description	Module	Label
Entry Point ISGGQWB1		
ISGBSM calls ISGGQWB1 to copy or (optionally) compress and copy the global resources serialization private area QWBs to the ring status authority (RSA).	ISGGQWB0	ISGGQWB1
The input parameter list (QPL) identifies that this is a copy from the global resources serialization private area to the RSA (QPLFIOC).		
1 Initialization consists of establishing addressability, establishing ISGGQWBR as the FRR, obtaining the local lock of the home address space and the CMSEQDQ lock.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

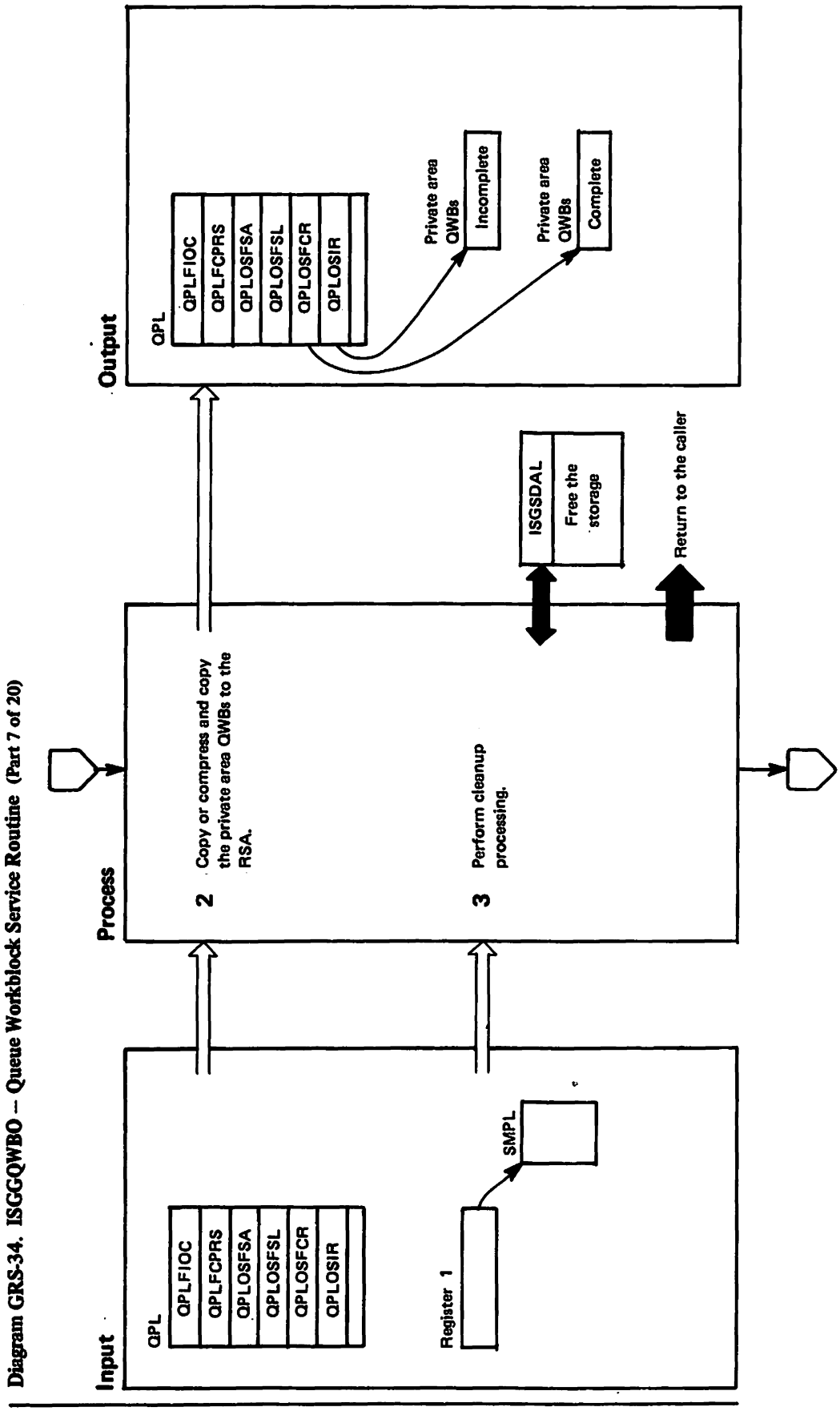


Diagram GRS-34. ISGGQWBO -- Queue Workblock Service Routine (Part 7 of 20)

LY28-1695-0 (c) Copyright IBM Corp. 1987

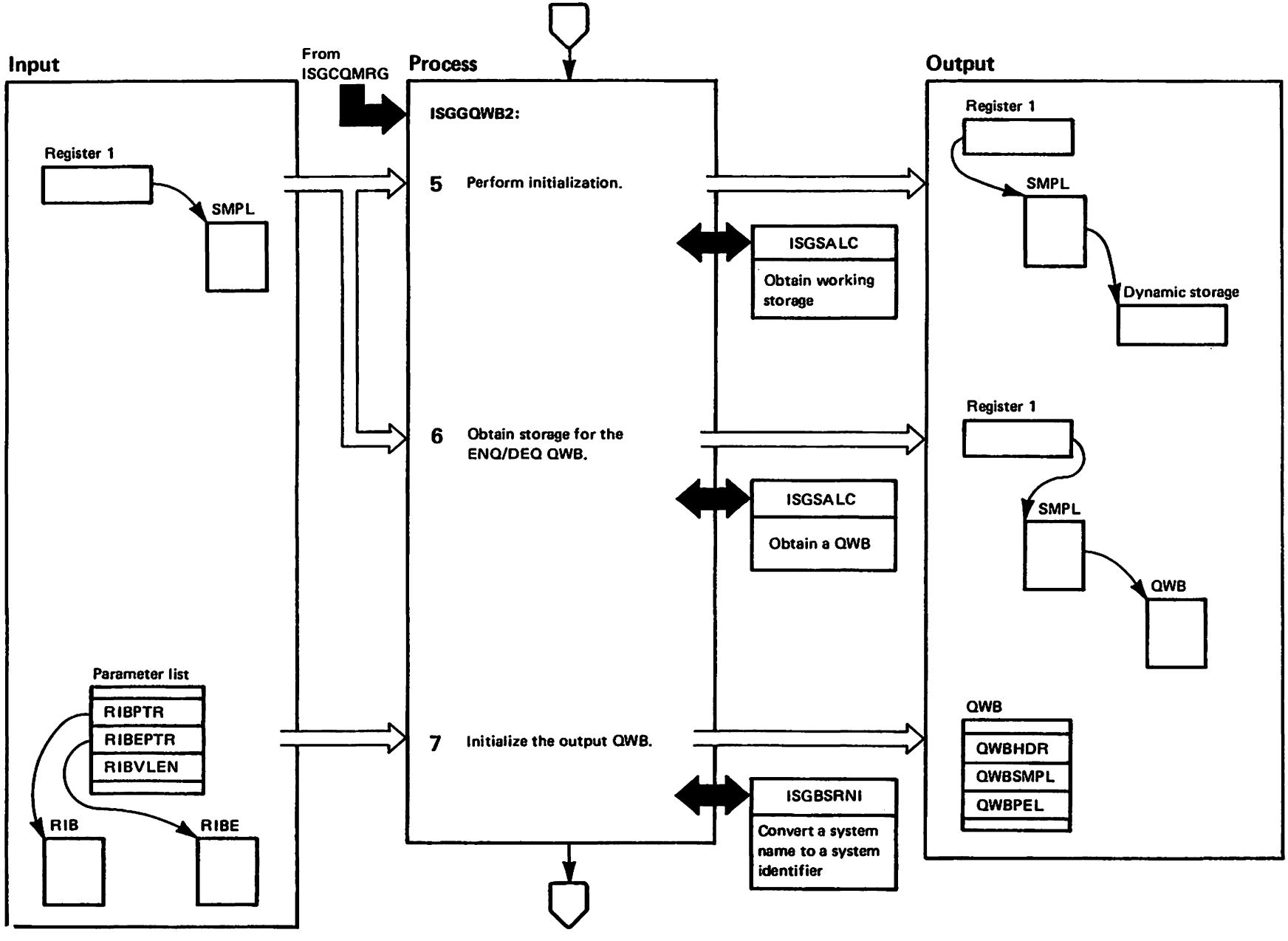
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-34. ISGGQWB0 – Queue Workblock Service Routine (Part 8 of 20)

Extended Description	Module	Label
2 ISGGQWB1 copies or compresses and copies the global resource serialization private area QWBs to the RSA. Virtual addresses are converted to displacements within the RSA. The input parameter list (QPL) identifies whether to copy or compress and copy the QWB (QPLFCPRS), a pointer to the RSA (QPLOSFSA), a pointer to the first QWB on the input queue (QPLOSFCR), and whether this is an outstanding incomplete request (QPLOSCPRS not zero). If this is an outstanding incomplete request ISGGQWB1 copies or compresses and copies the private area QWBs associated with the incomplete request into the RSA. The incomplete request becomes complete when all of the QWBs associated with the request have been copied to the RSA (QPLOSFCR non-zero). An incomplete request is initiated when some portion of the first QWB on the input queue will not fit in the RSA.	ISGGQWB0	
3 ISGGQWB1 invokes ISGSDAL to free the dynamic storage. The address of the SMPL is passed to ISGSDAL in register 1. ISGGQWB1 then releases the CMSEQDQ lock and deletes the FRR. On return, register 1 points to the output QPL that contains the count of QWBs copied to the RSA (QPLOSOQCT), pointers to the first and last complete request (QPLOSFCR and QPLOSLCR), and pointer to the extension to be copied or zero (QPLOSIR).		ISGGQWB1

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-34. ISGGQWB0 – Queue Work Block Service Routine (Part 9 of 20)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

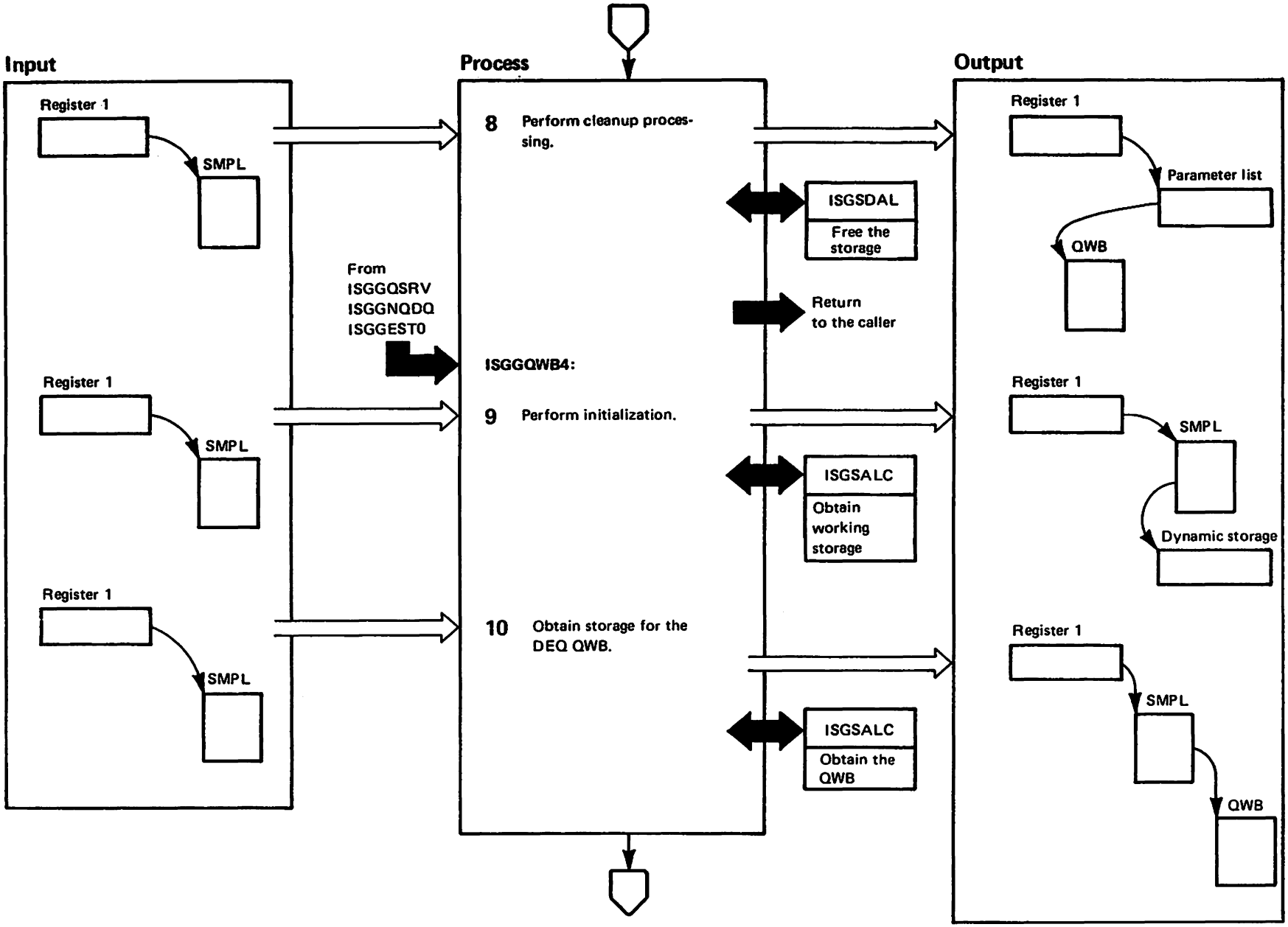
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-34. ISGGQWB0 – Queue Work Block Service Routine (Part 10 of 20)

Extended Description	Module	Label
Entry Point ISGGQWB2		
ISGCQMRG calls ISGGQWB2 in the global resource serialization address space to build a QWB for an ENQ or DEQ request from an input resource information block (RIB) or resource information block extension (RIBE). The input parameter list, pointed to by register 1, contains pointers to the RIB and RIBE and indicates either an ENQ or DEQ request.	ISGGQWB0	ISGGQWB2
This routine is only invoked to build QWBs for global resources.		
ISGGQWB2 marks each DEQ request as an unconditional internal DEQ (that is, the DEQ will be processed regardless of current ownership).		
5 Initialization consists of establishing addressability, establishing ISGGQWBR as the FRR, obtaining the local lock of the home address space and the CMSEQDQ lock, and invoking ISGSALC to obtain dynamic storage. The address of the storage management parameter list (SMPL) is passed to ISGSALC in register 1.	ISGSALC	
6 ISGGQWB2 invokes the storage manager (ISGSALC) to obtain storage from the QWB pool for a QWB. The address of the SMPL is passed to ISGSALC in register 1. ISGSALC holds the CMSEQDQ lock for serialization of the QWB pool.	ISGGQWB0 ISGSALC	ISGGQWB2
7 ISGGQWB2 fills in the QWB using data from the RIB and RIBE. ISGGQWB2 invokes ISGBSRNI to obtain the system identifier.	ISGGQWB0	ISGGQWB2
If this is an ENQ request, ISGGQWB2 initializes the SMPL to obtain a QCB, QEL, and QXB from the storage manager.		
If this is a DEQ request, ISGGQWB2 sets these entries to zero (control blocks are not obtained during DEQ processing). However, the QWB SMPL entry is set to 1 to allow for returning the DEQ QWB to the storage manager.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-34. ISGGQWB0 – Queue Work Block Service Routine (Part 11 of 20)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

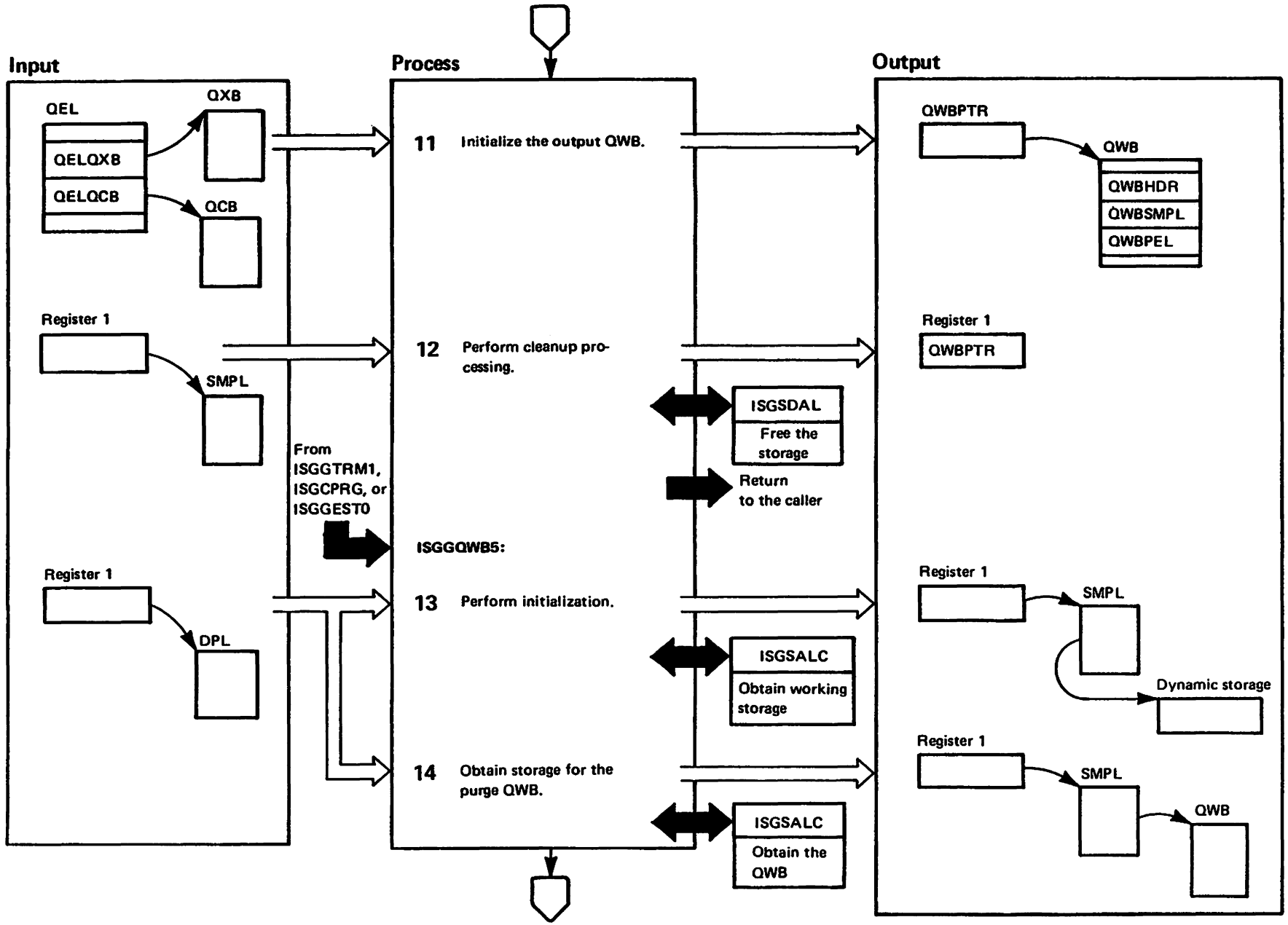
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-34. ISGGQWB0 – Queue Work Block Service Routine (Part 12 of 20)

Extended Description	Module	Label
8 ISGGQWB2 calls ISGSDAL to free the dynamic storage. The address of the SMPL is passed to ISGSDAL in register 1. ISGGQWB2 releases the locks it obtained and deletes the FRR. ISGGQWB2 puts the address of the QWB it just built into the input parameter list and returns to the caller.	ISGSDAL	
Entry Point ISGGQWB4		
ISGGQSRV, ISGGNQDQ, and ISGGEST0 invoke ISGGQWB4 to build a QWB for a DEQ request from the data described by an input QEL (pointed to by register 1). This DEQ QWB is marked as internally-generated to ensure that the DEQ occurs regardless of ownership checks.	ISGGQWB0	ISGGQWB4
9 Initialization consists of establishing addressability, establishing ISGGQWBR as the FRR, obtaining the local lock of the home address space and the CMSEQDQ lock, and invoking ISGSALC to obtain dynamic storage. The address of the storage management parameter list (SMPL) is passed to ISGSALC in register 1.	ISGSALC	
10 ISGGQWB4 invokes ISGSALC to obtain storage from the QWB pool for a QWB. The address of the SMPL is passed to ISGSALC in register 1. ISGSALC holds the CMSEQDQ lock for serialization of the QWB pool.	ISGGQWB0 ISGSALC	ISGGQWB4

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-34. ISGGQWB0 – Queue Work Block Service Routine (Part 13 of 20)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

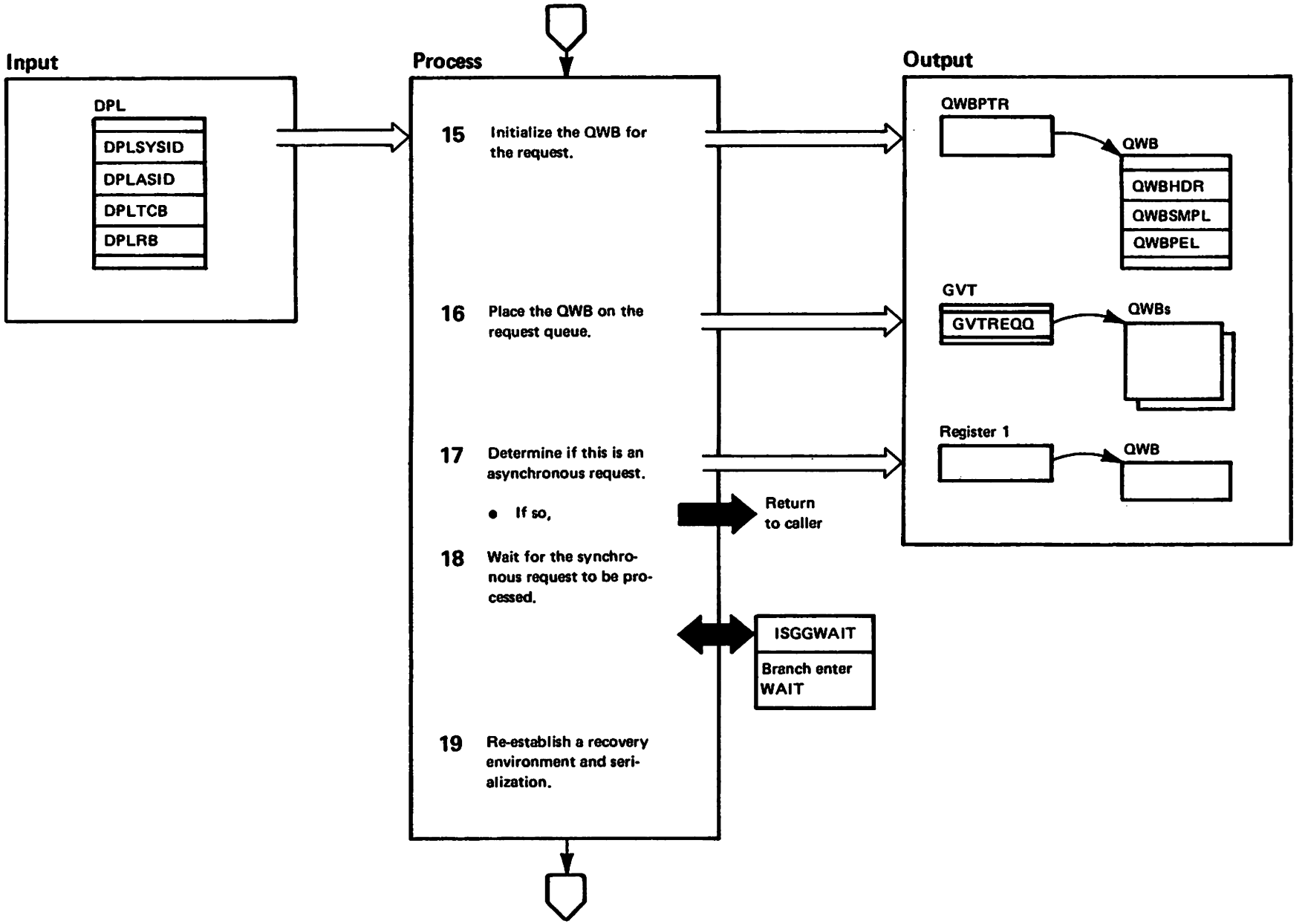
"Restricted Materials of IBM"
Licensed Materials – Property of IBM

Diagram GRS-34. ISGGQWB0 – Queue Work Block Service Routine (Part 14 of 20)

Extended Description	Module	Label
11 ISGGQWB4 builds the DEQ QWB using data from the input QEL. The QELQXB and QELQCB fields contain the addresses of the QXB and QCB respectively. The QXB contains pointers to the TCB and SVRB or ECB, the jobname and some flags. The QCB contains the scope, qname, rname, and more flags.	ISGGQWB0	ISGGQWB4
12 ISGGQWB4 invokes ISGSDAL to free the dynamic storage. The address of the SMPL is passed to ISGSDAL in register 1. ISGGQWB4 releases the locks it obtained and deletes the FRR. ISGGQWB4 returns to the caller with the private area QWB address in register 1.	ISGSDAL	ISGGQWB4
Entry Point ISGGQWB5		
Three routines invoke ISGGQWB5 in the global resource serialization address space: • ISGCPRG invokes it to perform a synchronous SYSID purge • ISGGTRM1 invokes it to perform a synchronous TCB or ASID purge • ISGGEST0 invokes it to perform a synchronous request, which ensures that all previous requests have been processed The DEQ purge list (DPL) pointed to by register 1, indicates the system, ASID, or TCB to be purged and whether the request is to be a synchronous or asynchronous request. In the case of a synchronous request, a WAIT is issued to the current RB until the purge has completed. When the global resource processor has processed the request and issued a POST to the RB defined in the QWB, processing continues. In the case of an asynchronous request, the request is placed on the request queue and ISGGQWB5 returns to the caller.	ISGGQWB0	ISGGQWB5
13 Initialization consists of establishing addressability, establishing ISGGQWBR as the FRR, obtaining the local lock of the home address space and the CMSEQDQ lock, and invoking ISGSALC to obtain dynamic storage. The address of the storage management parameter list (SMPL) is passed to ISGSALC in register 1.	ISGSALC	
14 ISGGQWB5 invokes ISGSALC to obtain storage from the QWB pool for a QWB. The address of the SMPL is passed to ISGSALC in register 1. ISGGQWB5 holds the CMSEQDQ lock for serialization of the QWB pool.	ISGGQWB0 ISGSALC	ISGGQWB5

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-34. ISGGQWB0 – Queue Work Block Service Routine (Part 15 of 20)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

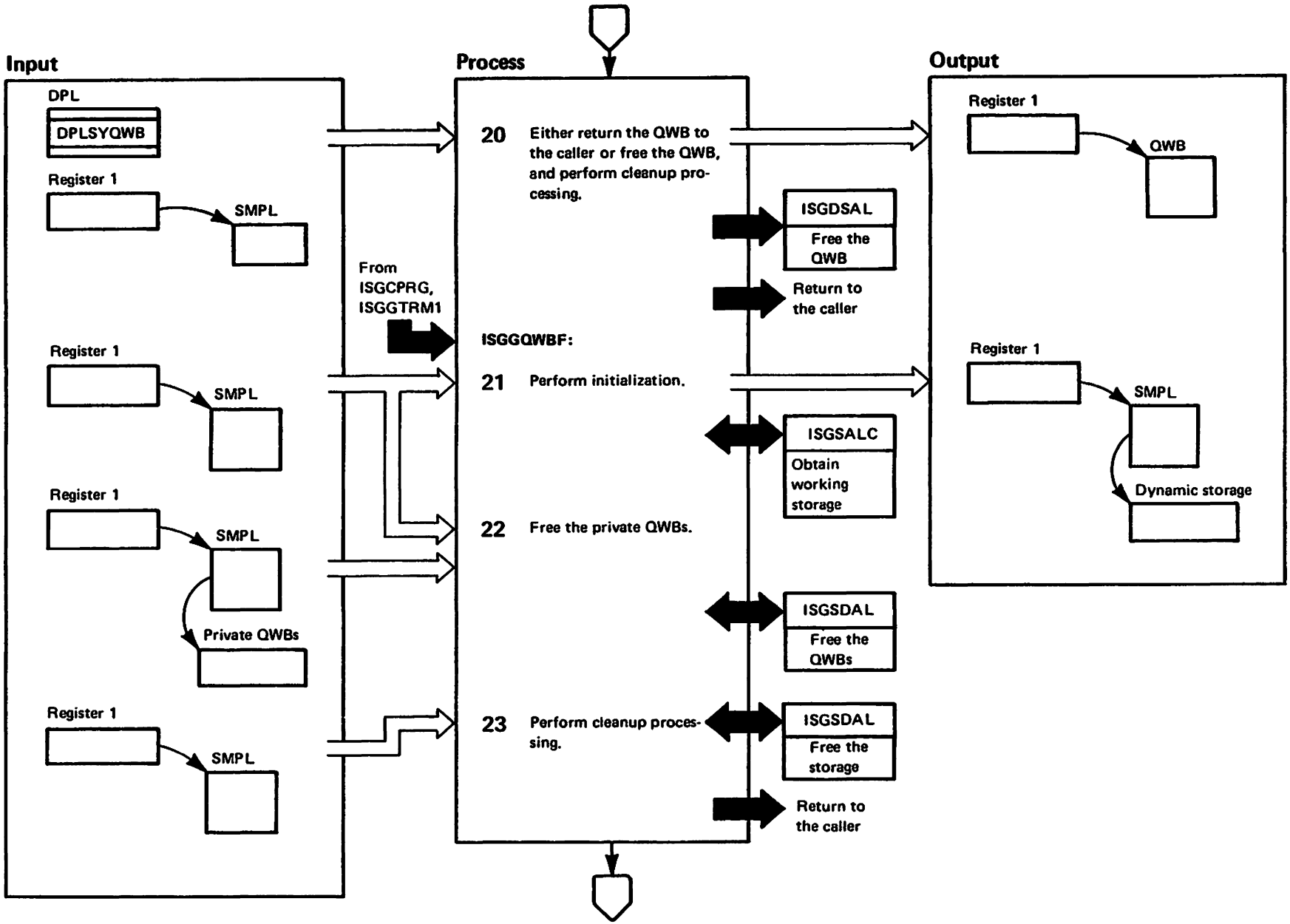
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-34. ISGGQWB0 – Queue Work Block Service Routine (Part 16 of 20)

Extended Description	Module	Label
15 ISGGQWB5 initializes the output QWB using data in the DPL. The output QWB is initialized for a SYSID purge, a TCB purge, an ASID purge, or a synchronization request.	ISGGQWB0	ISGGQWB5
16 ISGGQWB5 places the QWB on the request queue.		
17 If ISGGQWB5 determines that this is an asynchronous request (DPLASYNC=1), it returns control to the caller.		
18 For synchronous requests, ISGGQWB5 saves the registers and releases the CMSEQDQ lock. ISGGQWB5 then invokes ISGGWAIT to branch enter WAIT. After the global resource processor executes, it issues an RB POST to reactivate this routine.	ISGGWAIT	
19 ISGGQWB5 reestablishes an FRR and obtains the local lock of the home address space and the CMSEQDQ lock. ISGGQWB5 decreases the task global resource count (TCBGRES) by the number of global resources for which a QEL was removed from the queue by ISGGRP00.	ISGGQWB0	ISGGQWB5

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-34. ISGGQWB0 – Queue Work Block Service Routine (Part 17 of 20)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

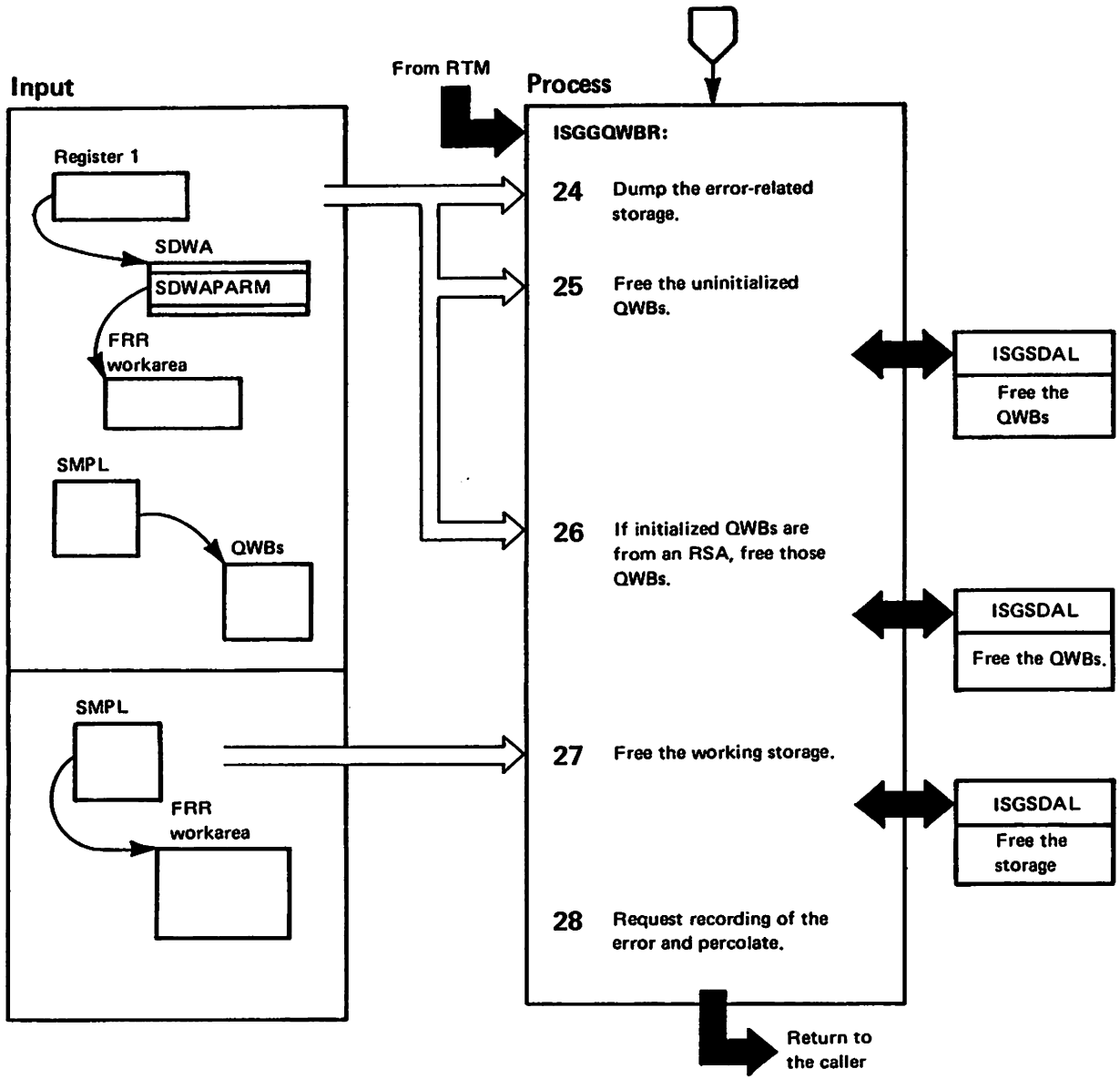
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-34. ISGGQWB0 -- Queue Work Block Service Routine (Part 18 of 20)

Extended Description	Module	Label
20 After the synchronous request has been processed, ISGGQWB5 determines if the QWB is to be returned to the caller. If DPLSVQWB=1, register 1 will point to the QWB to return to the caller. If DPLSVQWB=0, ISGGQWB5 calls the storage manager (ISGSDAL) to return the QWB to the QWB pool, if necessary. The address of the SMPL is passed to ISGSDAL in register 1.	ISGSDAL	
ISGGQWB5 then frees the working storage, releases the locks it obtained, and deletes the recovery environment.	ISGGQWB0	ISGGQWB5
Entry Point ISGGQWBF		
ISGGTRM1 and ISGCPRG invoke ISGGQWBF to free a private area QWB. Input to this routine is the address of the first QWB on the chain of QWBs to be freed. This routine must be invoked with the current addressability, to the global resource serialization address space.	ISGGQWB0	ISGQWBF
21 Initialization consists of establishing addressability, establishing ISGGQWBR as the FRR, obtaining the global resource serialization local lock and the CMSEQDQ lock, and invoking ISGSALC to obtain dynamic storage. The address of the storage management parameter list (SMPL) is passed to ISGSALC in register 1.	ISGSALC	
22 ISGGQWBF initializes the storage management parameter list (SMPL) to define the QWBs to be freed. ISGGQWBF then invokes the storage manager (ISGSDAL) to free the input QWBs. The address of the SMPL is passed to ISGSDAL in register 1. ISGGQWBF holds the CMSEQDQ lock for serialization of the QWB pool.	ISGGQWB0 ISGSDAL	ISGGQWBF
23 ISGGQWBF invokes ISGSDAL to free the dynamic storage. The address of the SMPL is passed to ISDSDAL in register 1. ISGGQWBF releases the locks it obtained and deletes the FRR.	ISGGQWB0	ISGGQWBF

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-34. ISGGQWB0 – Queue Work Block Service Routine (Part 19 of 20)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-34. ISGGQWB0 – Queue Work Block Service Routine (Part 20 of 20)

Extended Description	Module	Label
Entry Point ISGGQWBR		
ISGGQWBR provides recovery for all entry points to ISGGQWB0. Its function is to dump, clean up, record, and percolate.	ISGGQWB0	ISGGQWBR
24 ISGGQWBR initializes the SDWA to provide recovery data. It then calls the branch entry interface to the SDUMP service to dump the storage related to the failure. To ensure that the storage is dumped before processing continues, ISGGQWBR requests the suspend function of the SDUMP interface.		
25 If there are uninitialized QWBs to free, ISGGQWBR initializes the storage management parameter list (SMPL) to identify those QWBs. ISGGQWBR invokes ISGSDAL to free the QWBs. The address of the SMPL is passed to ISGSDAL in register 1.	ISGSDAL	
26 If initialized QWBs exist in the original input parameter list (RSA), ISGGQWBR initializes the SMPL to identify those QWBs. ISGGQWBR then invokes ISGSDAL to free the QWBs. The address of the SMPL is passed to ISGSDAL in register 1.	ISGSDAL	ISGGQWBR
27 ISGGQWBR calls ISGSDAL to free the dynamic workarea used by the failing function. The address of the SMPL is passed to ISGSDAL in register 1.	ISGSDAL	
28 ISGGQWBR returns control, requesting the locks to be freed and the error to be recorded in the SDWA, and percolates.	ISGGQWB0	

LY28-1695-0 (c) Copyright IBM Corp. 1987

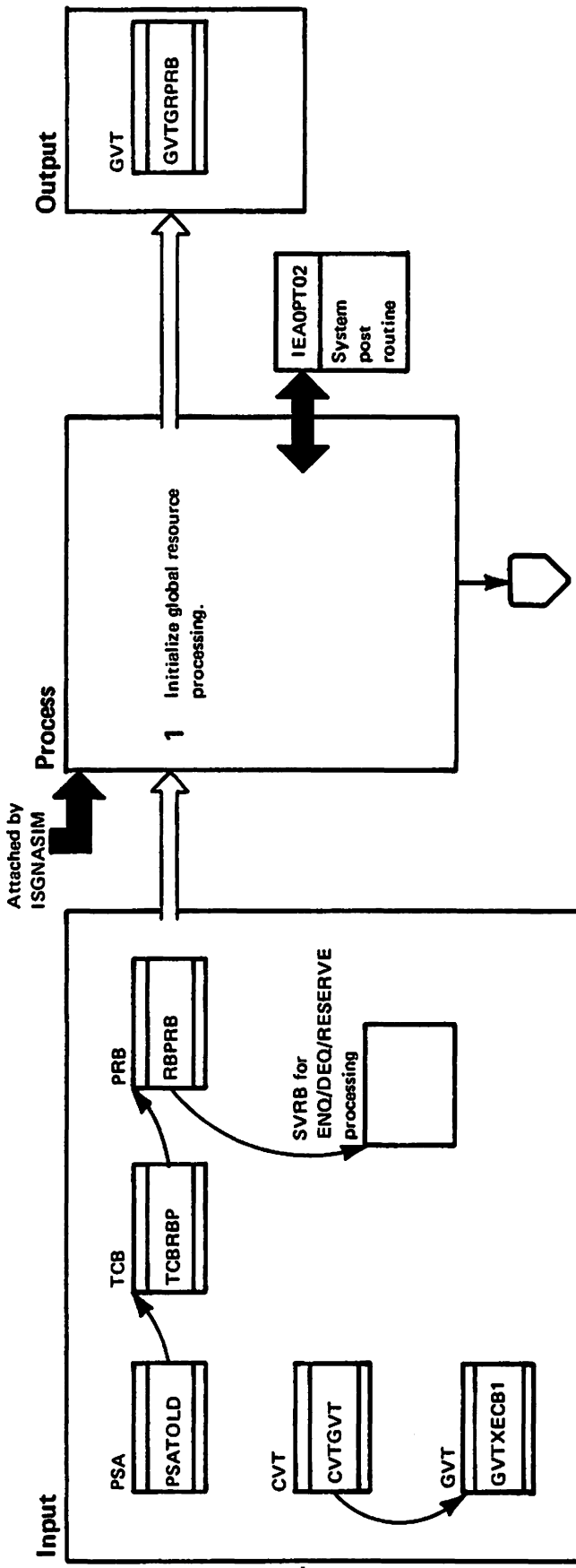


Diagram GRS-35. ISGGRP00 – Global Resource Processor (Part 1 of 20)

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-35. ISGGRP00 – Global Resource Processor (Part 2 of 20)

Extended Description	Module	Label
When initializing the global resource serialization address space, ISGNASIM attaches ISGGRP00 to prepare for processing global ENQ/DEQ/RESERVE requests. ISGGRP00 then enters a wait until ISGBSR puts ENQ/DEQ/RESERVE requests (in the form of queue work blocks, QWBs) on the process queue and notifies ISGGRP00 that there are requests to be processed. For each QWB on the process queue, ISGGRP00 removes the QWB from the queue in first-in-first-out order, determines the request type it represents, and processes it accordingly. When the process queue is empty, ISGGRP00 returns to a wait until posted.		
1 To initialize global resource processing, ISGGRP00:		
• Establishes GPRESTAE as its ESTAE to provide recovery while in a wait state.		
• Obtains the local lock of the global resource serialization address space to serialize the global queues and control blocks. Serialization is necessary because other global resource serialization functions can be executing concurrently with ISGGRP00. The local lock is also required to call the system POST routine (IEA0PT02) and to serialize the GVTNONE (GRS=NONE) flag with the global resource serialization option processor (ISGNGRSP).		
• Establishes ISGGFRR0 as its FRR to provide recovery when processing the process queue elements.		
• Places the address of ISGGRP00's RB in the GVTGRPRB field. ISGBSR posts the RB in that field when elements are placed on the process queue.		
• Calls IEA0PT02 to inform ISGNASIM that ISGGRP00 is now initialized and can be posted to handle requests on the process queue. IEA0PT02 posts the ECB in the GVTXECB1 field.	IEA0PT02	

LY28-1695-0 (c) Copyright IBM Corp. 1987

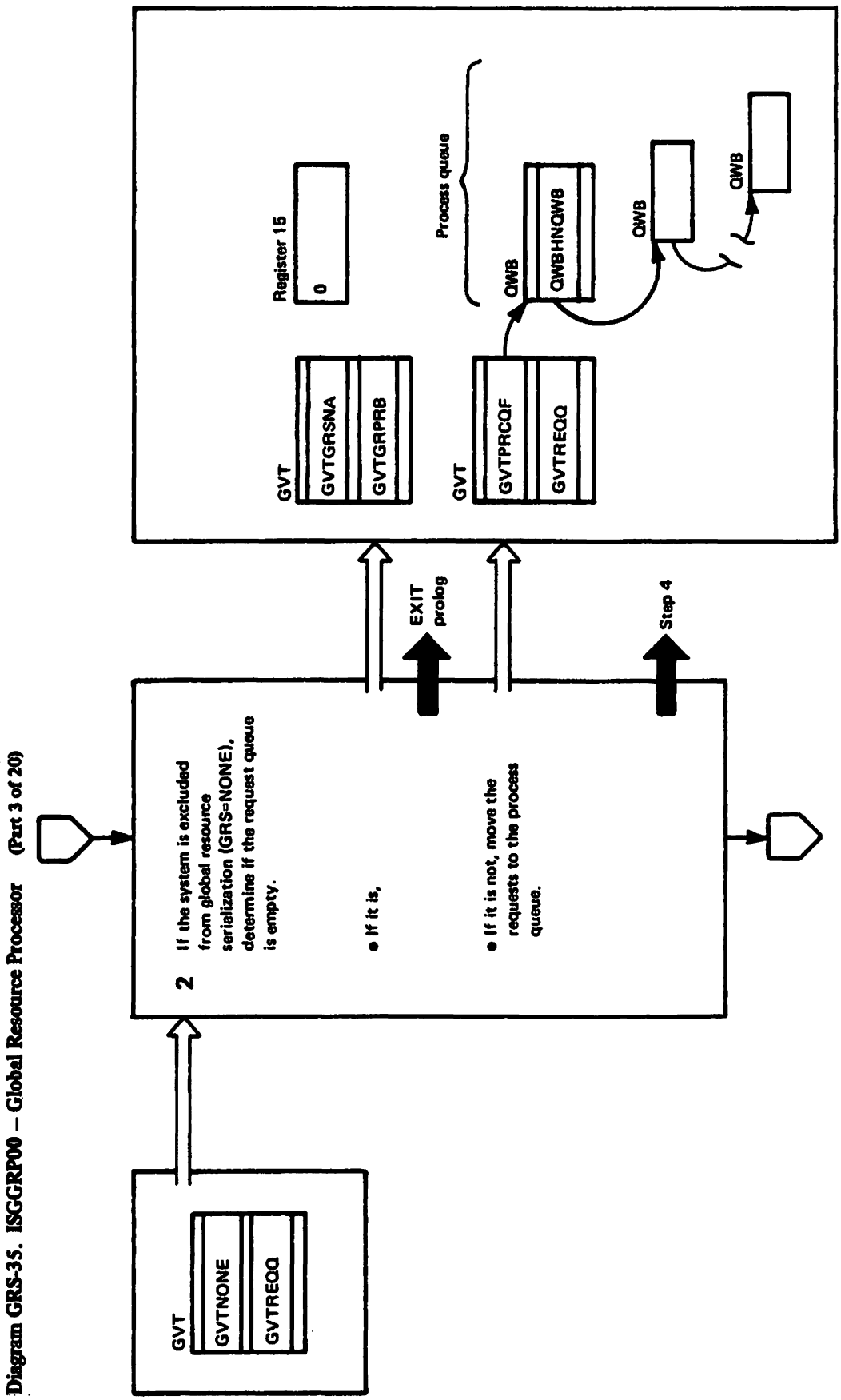


Diagram GRS-35. ISGGRP00 – Global Resource Processor (Part 3 of 20)

LY28-1695-0 (c) Copyright IBM Corp. 1987

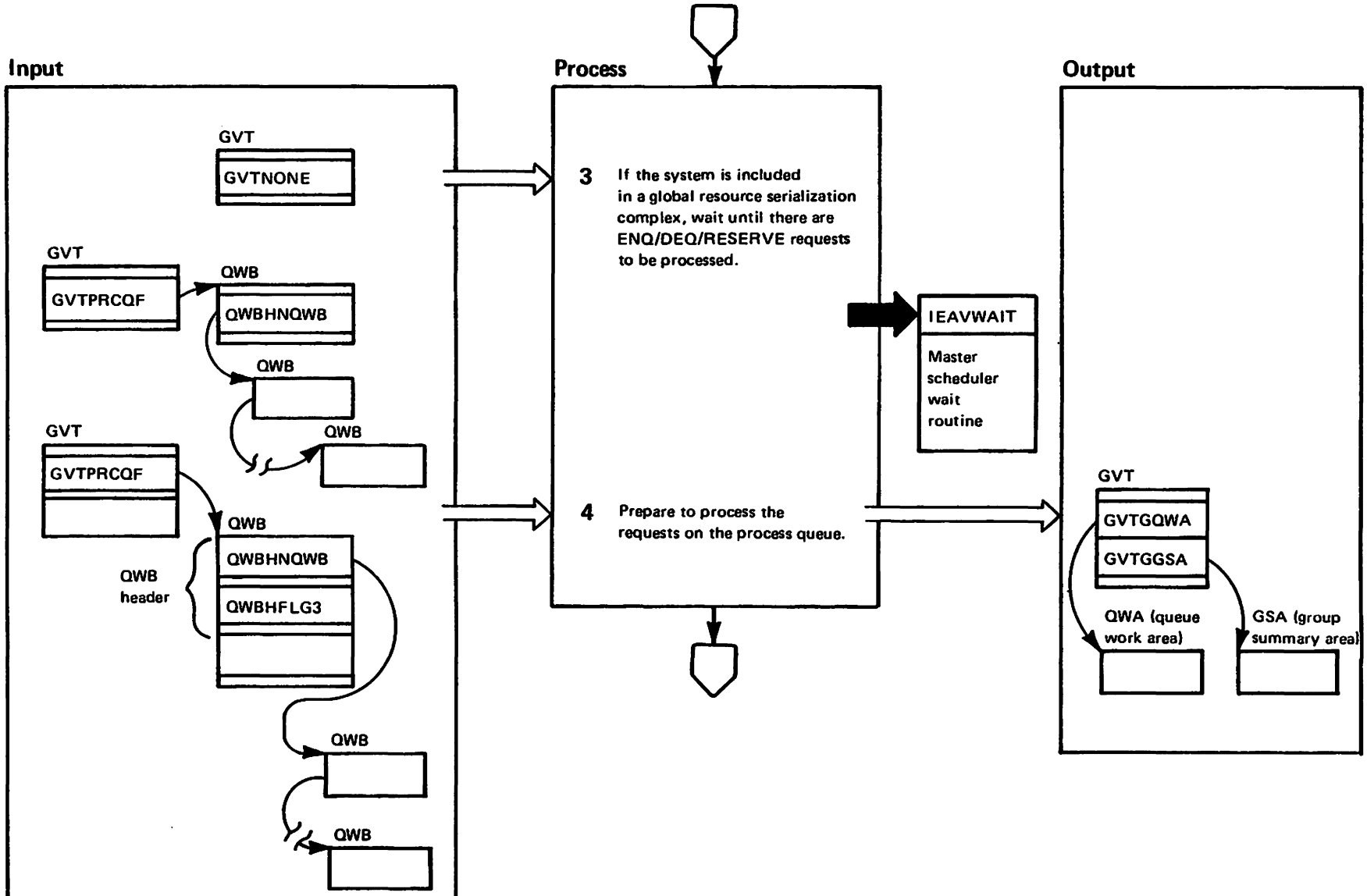
Extended Description	Module	Label
2 If the GRS=NONE option was specified, the system is excluded from global resource processing. Although this means that the system will not pass or receive global resource requests to or from other systems, the request queue might contain requests because: • Global resources were requested and queued during NIP processing before the GRS system parameter was resolved. • The system specified GRS=JOIN or START and requests were placed on the queue before the GRS parameter was resolved. After that, the installation was unable to include the system in a global resource serialization ring or to start a new ring and, therefore, responded with the GRS=NONE option. When the request queue contains entries (GVTREQQ≠0), ISGGRP00: • Moves the QWBs to the process queue. • Indicates that the request queue is empty by setting the GVTREQQ field to zero. After the existing requests are processed, ISGGRP00 notes that the request queue is empty and terminates global resource processing. • Processes the requests as described in steps 5 to 11. Note that although the resources might have been requested as global resources, they are treated as local requests because global resource serialization is inactive.		

Extended Description	Module	Label
2 (continued) When the request queue is empty (GVTREQQ=0), ISGGRP00 terminates global resource processing. To do so, it: • Obtains the CMSEQDQ lock (if it is not already held) to serialize the global-sharing-not-active flag (GVTGRSNA). • Clears the RB address in the GVTGRPRB field since ISGGRP00 can no longer be posted. • Indicates that global sharing is inactive by setting the GVTGRSNA bit to one. • Releases the CMSEQDQ lock. • Deletes the FRR. • Releases the local lock. • Deletes the ESTAE. • Sets a return code of zero to indicate successful termination. • Branches to EXIT prolog.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

Diagram GRS-35. ISGGRP00 – Global Resource Processor (Part 5 of 20)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

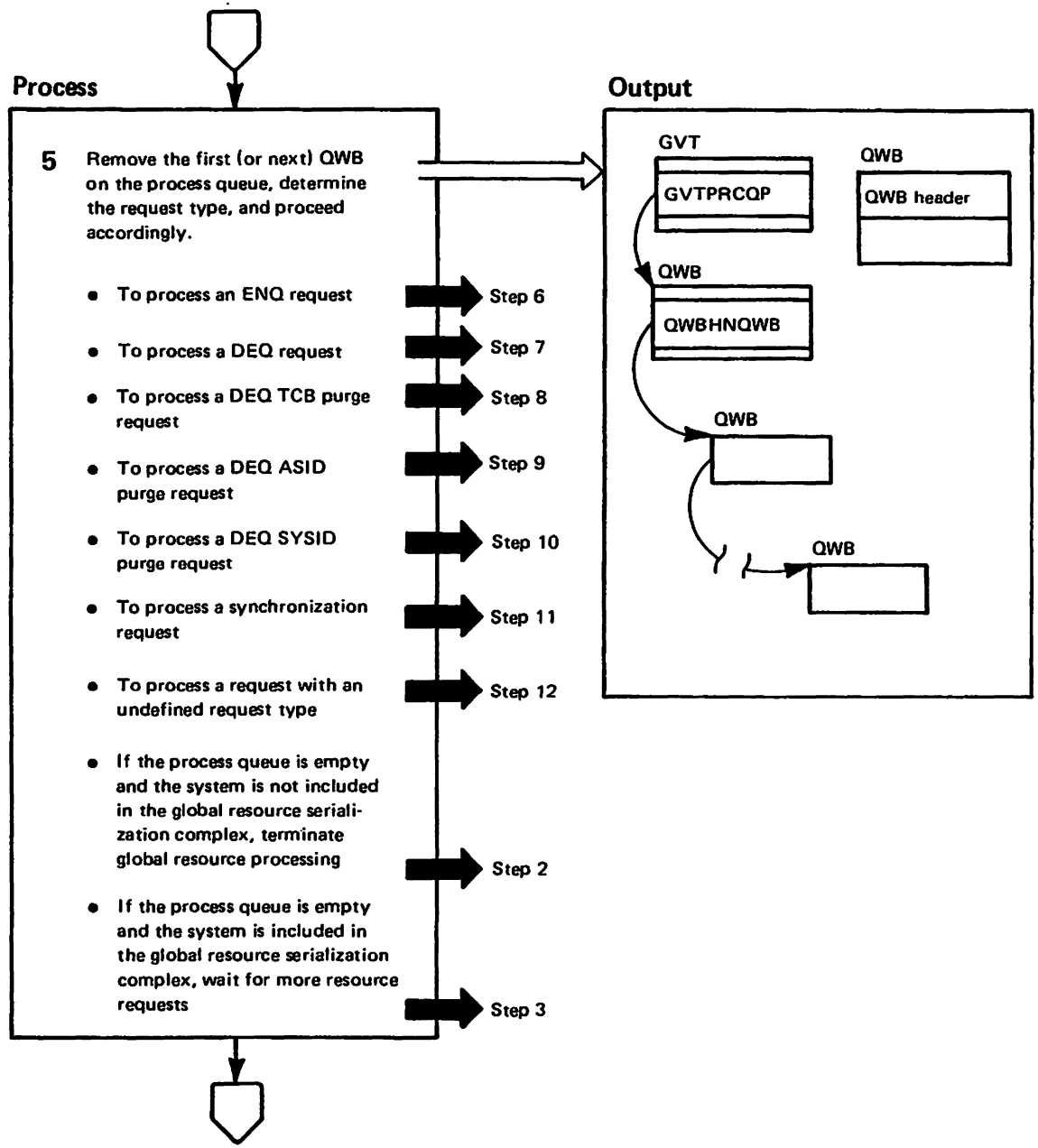
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-35. ISGGRP00 – Global Resource Processor (Part 6 of 20)

Extended Description	Module	Label
3 When GRS=JOIN or START (GVTNONE≠0), ISGGRP00 releases the FRR and calls IEAVWAIT, which puts ISGGRP00 in a wait until requests (QWBs) are placed on the process queue. When this happens, ISGBSR posts ISGGRP00 via the GVTGRPRB field and ISGGRP00 resumes processing at the next step.	IEAVWAIT	
4 To prepare for processing the ENQ/DEQ/RESERVE requests on the process queue, ISGGRP00:		
• Obtains the local lock of the global resource serialization address space to serialize the global work areas and control blocks.		
• Establishes ISGGFRR0 as its recovery routine.		
• Clears the queue work area (QWA) and group summary area (GSA).		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-35. ISGGRP00 -- Global Resource Processor (Part 7 of 20)



"Restricted Materials of IBM"
Licensed Materials - Property of IBM

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-35. ISGGRP00 – Global Resource Processor (Part 8 of 20)

Extended Description	Module	Label

5 ISGGRP00 removes the first (or next) element (QWB) from the process queue. The QWB header contains request-related information required to process the request. ISGGRP00 examines the QWB header's flag field (QWBHFLG3) to determine which request type it represents and proceeds accordingly. Possible request types and the step describing how ISGGRP00 processes each are:

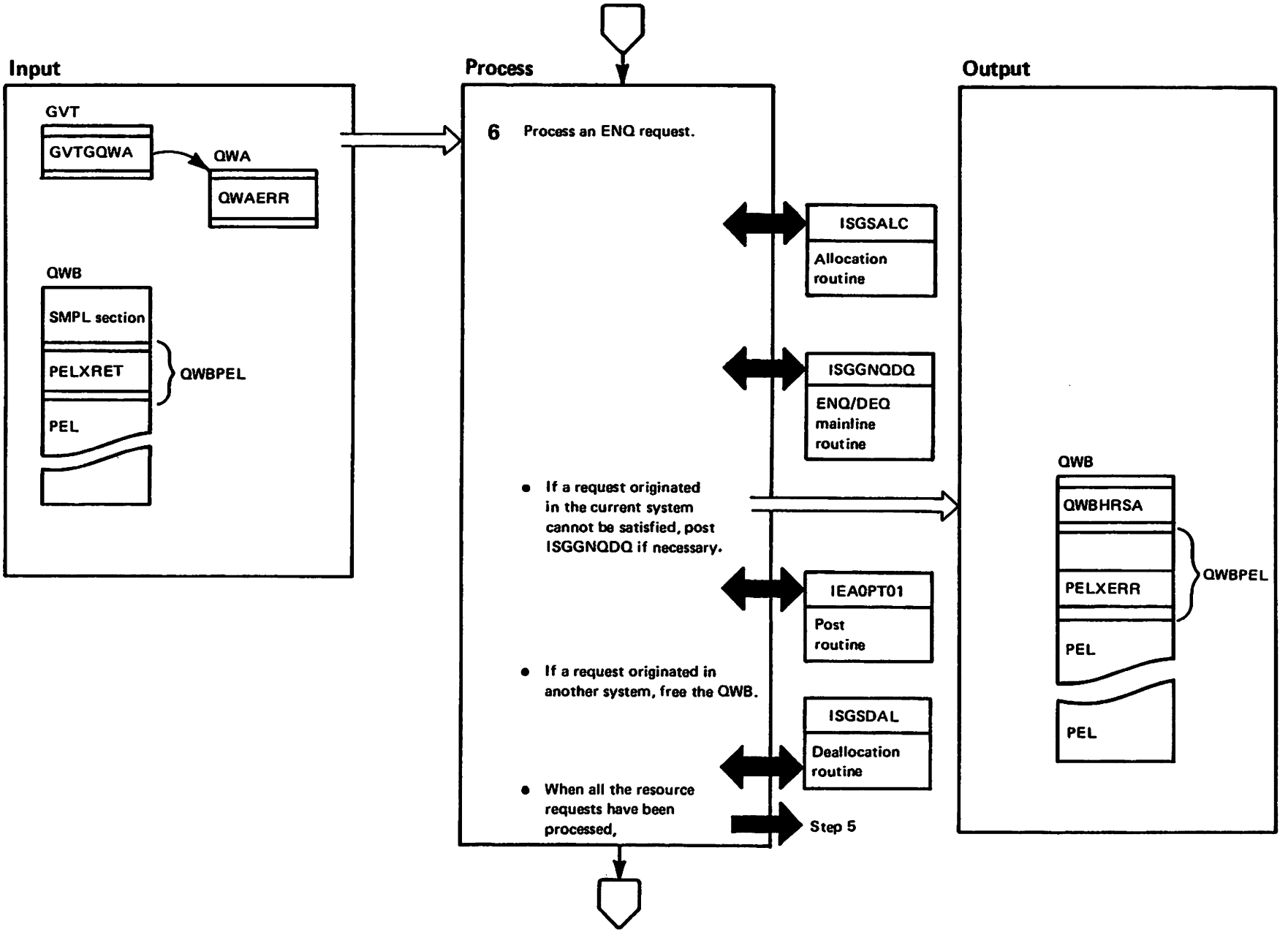
ENQ – step 6
DEQ – step 7
DEQ TCB purge – step 8
DEQ ASID purge – step 9
DEQ SYSID purge – step 10
Synchronization – step 11
Undefined QWB – step 12

If the process queue is empty and the system has requested that it not be included in a global resource serialization complex (GRS=NONE), ISGGRP00 deletes the FRR and continues at step 2 where it terminates global resource processing.

If the process queue is empty and the system is included in a global resource serialization complex (GRS=JOIN or START), ISGGRP00 continues at step 3 where it enters a wait until posted that more elements have been placed on the process queue.

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-35. ISGGRP00 – Global Resource Processor (Part 9 of 20)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

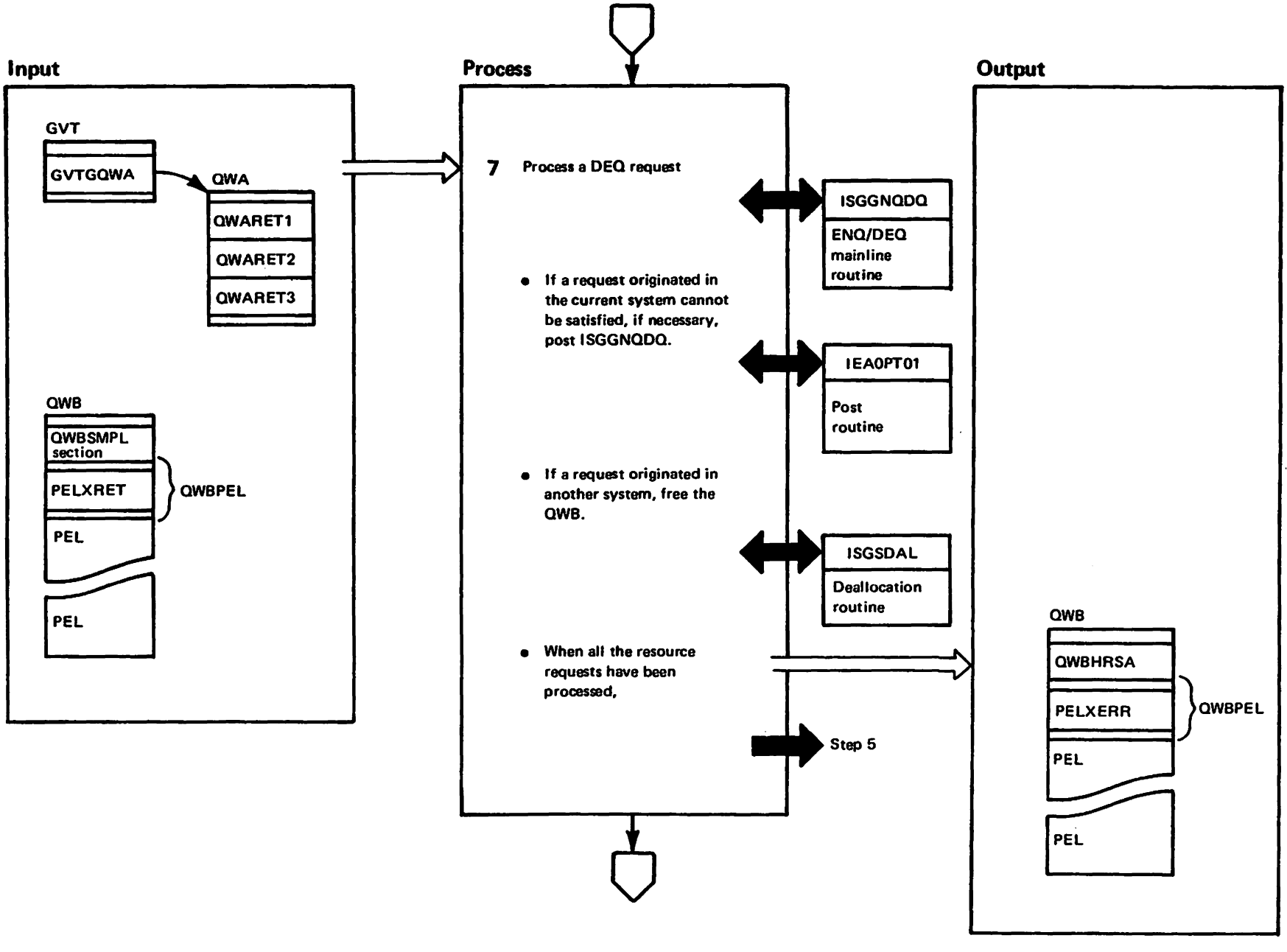
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-35. ISGGRP00 – Global Resource Processor (Part 10 of 20)

Extended Description	Module	Label
ENQ Processing		
6 The QWBBASIC section of the QWB contains one or more parameter element entries (PELs). Each PEL entry represents a resource on which the requestor wants to enqueue. ISGGRP00 calls the allocation routine (ISGSALC) to obtain the control blocks (QCBs, QELs, and QXBs) required to enqueue on all of the resources specified in the PELs. The SMPL section of the QWB defines the types and amounts of storage required.	ISGSALC	
For each PEL entry in the QWB (or QWB extension if the PEL cannot be contained in a QWB), ISGGRP00 calls the mainline ENQ/DEQ routine (ISGGNQDQ) at entry point ISGGNQ00 to enqueue the requestor on the specified resource.	ISGGNQDQ	ISGGNQ00
After ISGGNQDQ returns control, ISGGRP00 examines the PELXRET and QWAERR fields to determine whether or not the requestor should be abended. If not, ISGGRP00 processes the next PEL entry in the QWB.		
If ISGGRP00 is to abend the requestor or has processed all the PEL entries and the request originated in this system, ISGGRP00:		
• Frees the unused QELs, QXB, and/or QCB.		
• Copies the QWARSA section of the QWA into the header section of the QWB (QWBHRSA). This saves for the caller the return or abend code that ISGGNQDQ placed into the QWA. Although ISGGRP00 copies the QWARSA into the QWB for all requests, the information in the QWB is only used when the request originated in the current system.		
• If the request is not an internal request(one generated by a global resource serialization module), calls IEA0PT01 to post ISGGWAIT to finish processing the request.	IEA0PT01	
If the request originated in another system, ISGGRP00 frees the request QWB and any unused control blocks.	ISGSDAL	
When all the PELs entries related to one request have been processed, ISGGRP00 returns to step 5 to process the next QWB.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-35. ISGGRP00 – Global Resource Processor (Part 11 of 20)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

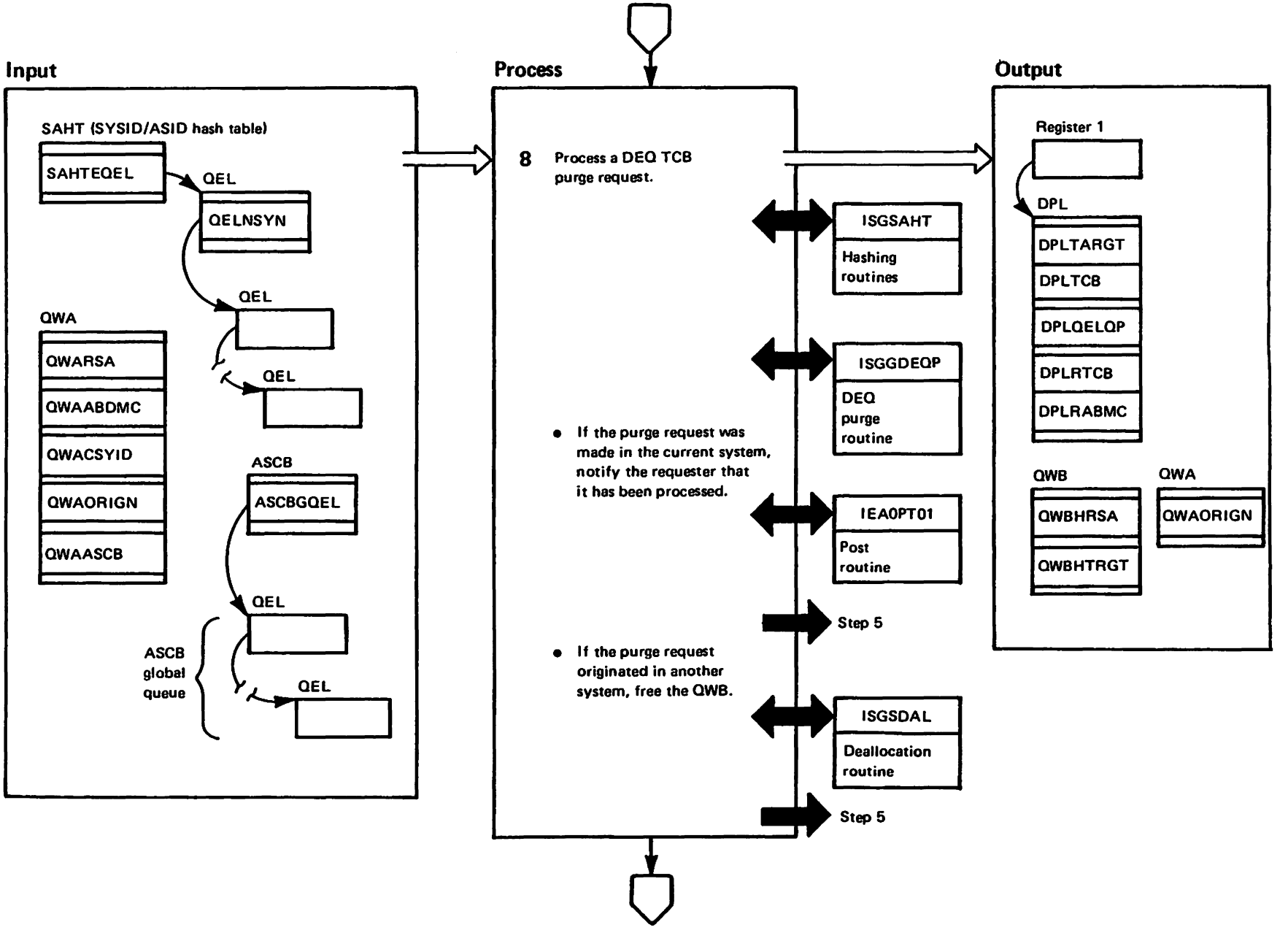
"Restricted Materials of IBM"
Licensed Materials – Property of IBM

Diagram GRS-35. ISGGRP00 – Global Resource Processor (Part 12 of 20)

Extended Description	Module	Label
DEQ Processing		
7 The input QWB contains one or more PEL entries. Each PEL entry represents a resource to be dequeued. For each PEL entry, ISGGRP00 calls the mainline ENQ/DEQ routine (ISGGNQDQ) at entry point ISGGDQ00 to dequeue the requestor from the specified resource.	ISGGNQDQ	ISGGDQ00
After ISGGNQDQ returns control, ISGGRP00 determines if this was a generic or unconditional DEQ request.		
• If this was a generic DEQ request, ISGGRP00 uses the return codes generated by local and global processing to determine the final return code.		
• If this was an unconditional DEQ request with a non-zero return code, ISGGRP00 saves the abend code for mainline DEQ processing (ISGGNQDQ).		
ISGGNQDQ will issue the ABEND for this requestor after the POST from ISGGRP00. If ISGGRP00 did not encounter an abend condition, it processes the next PEL entry in the QWB request.		
If ISGGRP00 is to abend the requestor or has processed all the PEL entries and the request originated in this system, ISGGRP00:		
• Frees any control blocks accumulated in the SMPL as a result of ISGGNQDQ (at entry point ISGGDQ00) processing.		
• Copies the QWARSA section of the QWA into the header section of the QWB. This saves for the caller the return or abend code that ISGGNQDQ placed into the QWA. Although ISGGRP00 copies the QWARSA into the QWB for all requests, the information in the QWB is only used when the request originated in the current system.		
• If the request is not an internal request (one generated by a global resource serialization module), calls IEAVOPT01 to post ISGGWAIT to finish processing the request.	IEAVOPT01	
If the request originated in another system, ISGGRP00 frees the request QWB and any other control blocks.	ISGSDAL	
When all the PELs related to one request have been processed, ISGGRP00 returns to step 5 to process the next QWB.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-35. ISGGRP00 – Global Resource Processor (Part 13 of 20)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

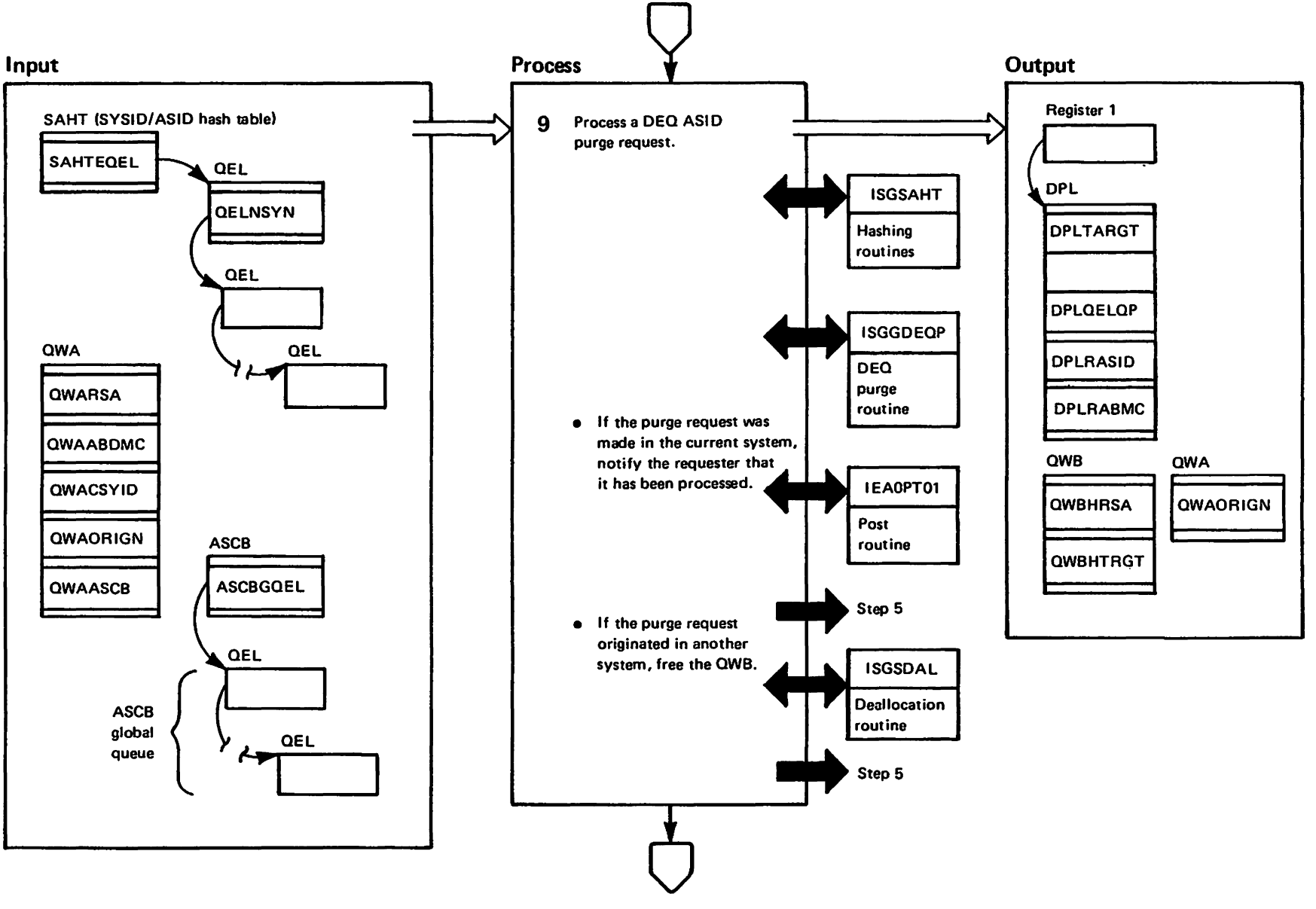
LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-35. ISGGRP00 – Global Resource Processor (Part 14 of 20)

Extended Description	Module	Label
DEQ TCB Purge Processing		
8 The ENQ/DEQ/RESERVE resource termination manager (ISGGTRM1) issues a DEQ TCB purge request to purge the global resources that are owned by that TCB. The TCB to be purged is represented by queue elements (QELs) on two queues:		
• If the resources for a task in the current system are being purged, the TCB is represented by QELs on the ASCB global QEL queue.		
• If the task is in a system other than the current one, the QEL queue to be purged is chained from a QEL on a synonym chain pointed to by a slot in the SYSID/ASID hash table (SAHT). To locate the appropriate synonym QEL, ISGGRP00 first calls the hash routine (ISGSAHT) to find the hash slot address which points to the appropriate synonym chain. It then searches the synonym chain for the synonym QEL with a SYSID/ASID matching the input SYSID/ASID (QELORIGN=QWBHTRGT). The QELs queued from the synonym QEL owned by this TCB represent the resources to be purged.	ISGSAHT	
To purge the resources, ISGGRP00:		
• Initializes a DEQ purge list (DPL) with information about the resources to be purged, including the address of either the ASCB global QEL queue or the synonym QEL queue. (See the output section of the diagram for details.)		
• Calls ISGGDEQP to purge the resources. The DPL is input to ISGGDEQP.	ISGGDEQP	
• Saves the requestor's RSA (the QWARSA field) in QWBHRSA. ISGGRP00 does this for every request processed, even though information in the QWB is used only when the request originated in the current system.		
• If the purge request was made in the current system, calls the post routine (IEA0PT01) to post ISGGWAIT which returns to ISGGQWB0 which returns to ISGGTRM1.	IEA0PT01	
• If the request came from a system other than the current one, calls ISGSDAL to free the QWB. ISGGRP00 obtains and holds the CMSEQDQ lock during this processing.	ISGSDAL	
ISGGRP00 returns to step 5 to process the next QWB.		
Note that the DEQ TCB request allows the ENQ/DEQ/RESERVE resource termination manager to be sure that the terminating task has no outstanding global resource requests on the request, staging, or process queue. Since the QWB representing the DEQ TCB request is queued and therefore processed after any requests the terminating task might have made, the terminating task's requests must have already been processed. Any resources allocated to the task would be represented by QELs on either the SYSID/ASID hash table queues or the ASCB global QEL queue.		

Diagram GRS-35. ISGGRP00 – Global Resource Processor (Part 15 of 20)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

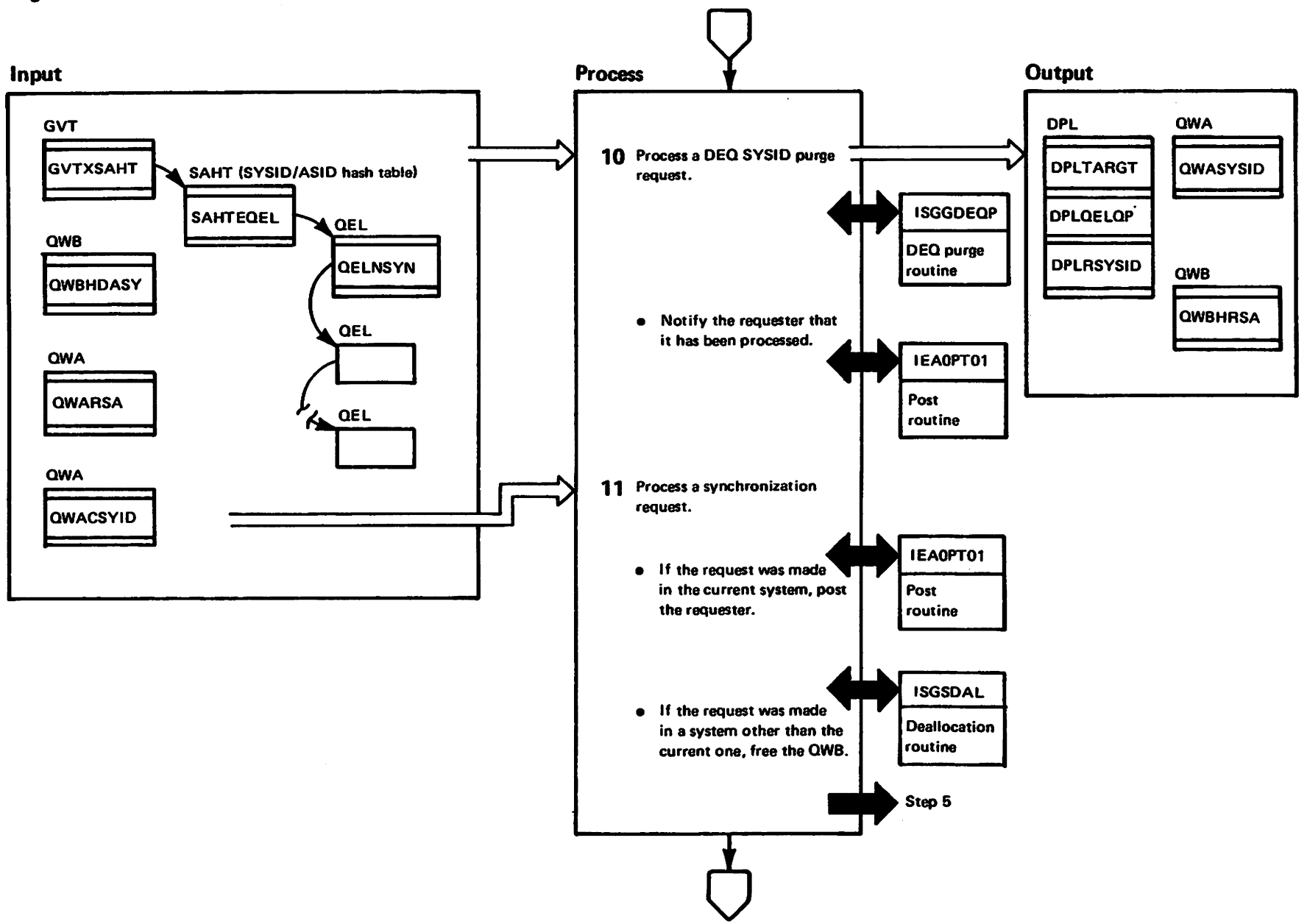
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-35. ISGGRP00 – Global Resource Processor (Part 16 of 20)

Extended Description	Module	Label
DEQ ASID Purge Processing		
9 The ENQ/DEQ/RESERVE resource termination manager (ISGGTRM1) issues a DEQ ASID purge request to purge the gloabl resources that are owned by the ASID specified in the request. ISGGRP00 processes a DEQ ASID purge request the same way that it processes DEQ TCB purge requests with the exception that the TCB address is ignored and the SYSID/ASID alone becomes the search argument for resources to be dequeued. Step 6 describes that processing.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-35. ISGGRP00 – Global Resource Processor (Part 17 of 20)



LY28-1695-0 (c) Copyright IBM Corp. 1987

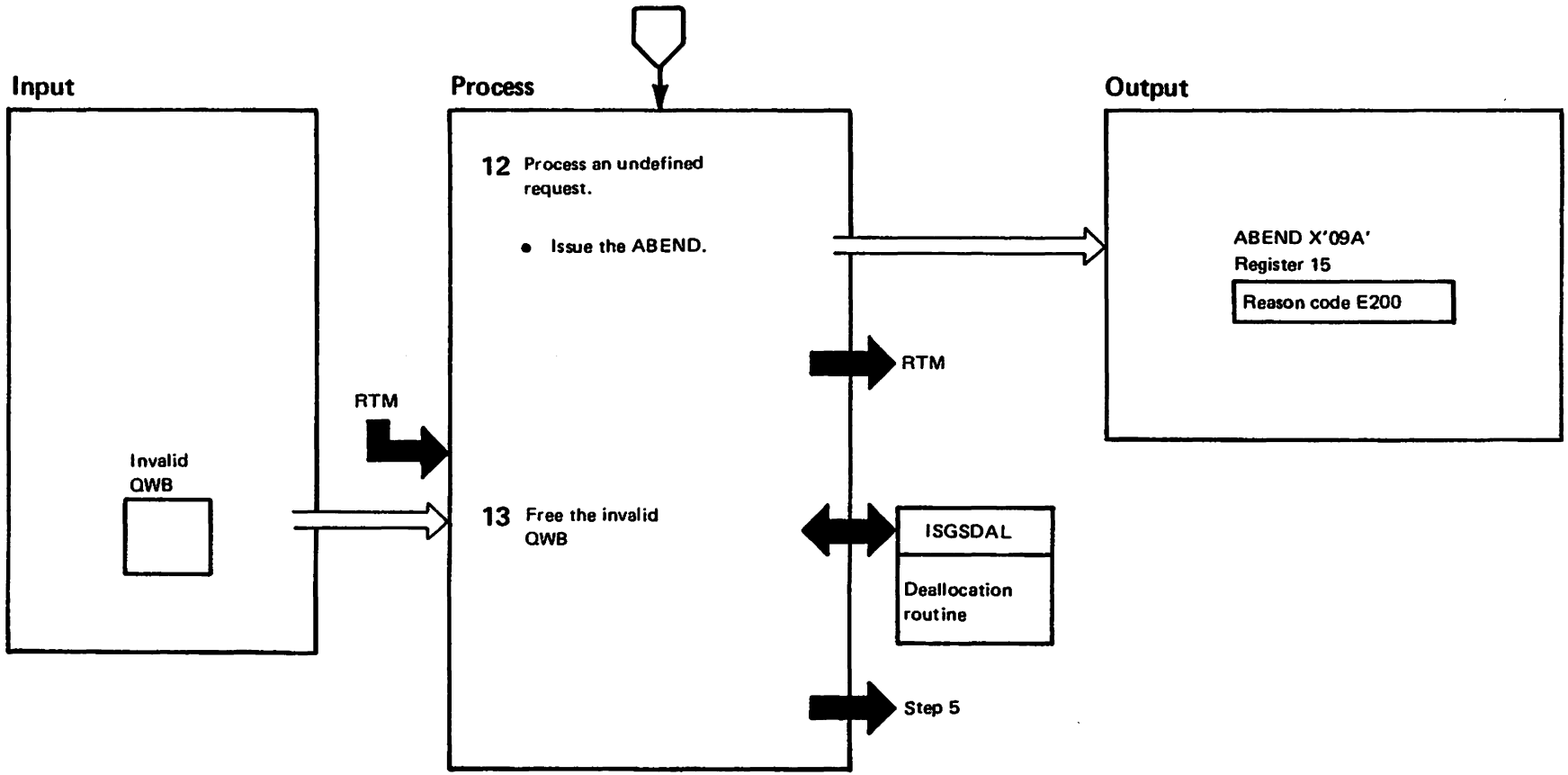
"Restricted Materials of IBM"
Licensed Materials – Property of IBM

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Extended Description	Module	Label
DEQ SYSID Purge Processing		
10 ISGGRP00 satisfies a DEQ SYSID purge request by purging all global resources allocated to tasks in the system identified by the input SYSID. (The caller must purge local resources.) ISGGRP00 searches all the synonym chains queued from the SYSID/ASID hash table for synonym QELs having the same SYSID as specified in the input (QELSYSID=QWBHDASY). For each match found, ISGGRP00 initializes a DEQ purge list (DPL) and calls ISGGDEQP to purge the QEL queue chained from the synonym QEL. The DPL is input to ISGGDEQP.	ISGGDEQP	
When all the synonym chains have been processed, ISGGRP00: • Saves the requestor's RSA (the QWARSA field) in the QWB • Obtains and holds the CMSEQDQ lock during this processing. • Calls IEA0PT01 to post the requestor	IEA0PT01	
Note: A requestor may not do a SYSID purge on itself. ISGGRP00 returns to step 5 where it processes the next QWB.		
Synchronization Processing		
11 ISGGRP00 determines whether or not the synchronization request originated in the current system. If it did, ISGGRP00 calls IEA0PT01 to post the requestor. The requestor then knows that all requests made prior to the synchronization request have been processed (that is, they are not outstanding on the request, staging, or process queue).	IEA0PT01	
If the request originated on a system other than the current one, ISGGRP00 calls ISGSDAL to free the QWB.	ISGSDAL	
ISGGRP00 returns to step 5 where it processes the next QWB.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-35. ISGGRP00 – Global Resource Processor (Part 19 of 20)



"Restricted Materials of IBM"
Licensed Materials – Property of IBM

LY28-1695-0 (c) Copyright IBM Corp. 1987

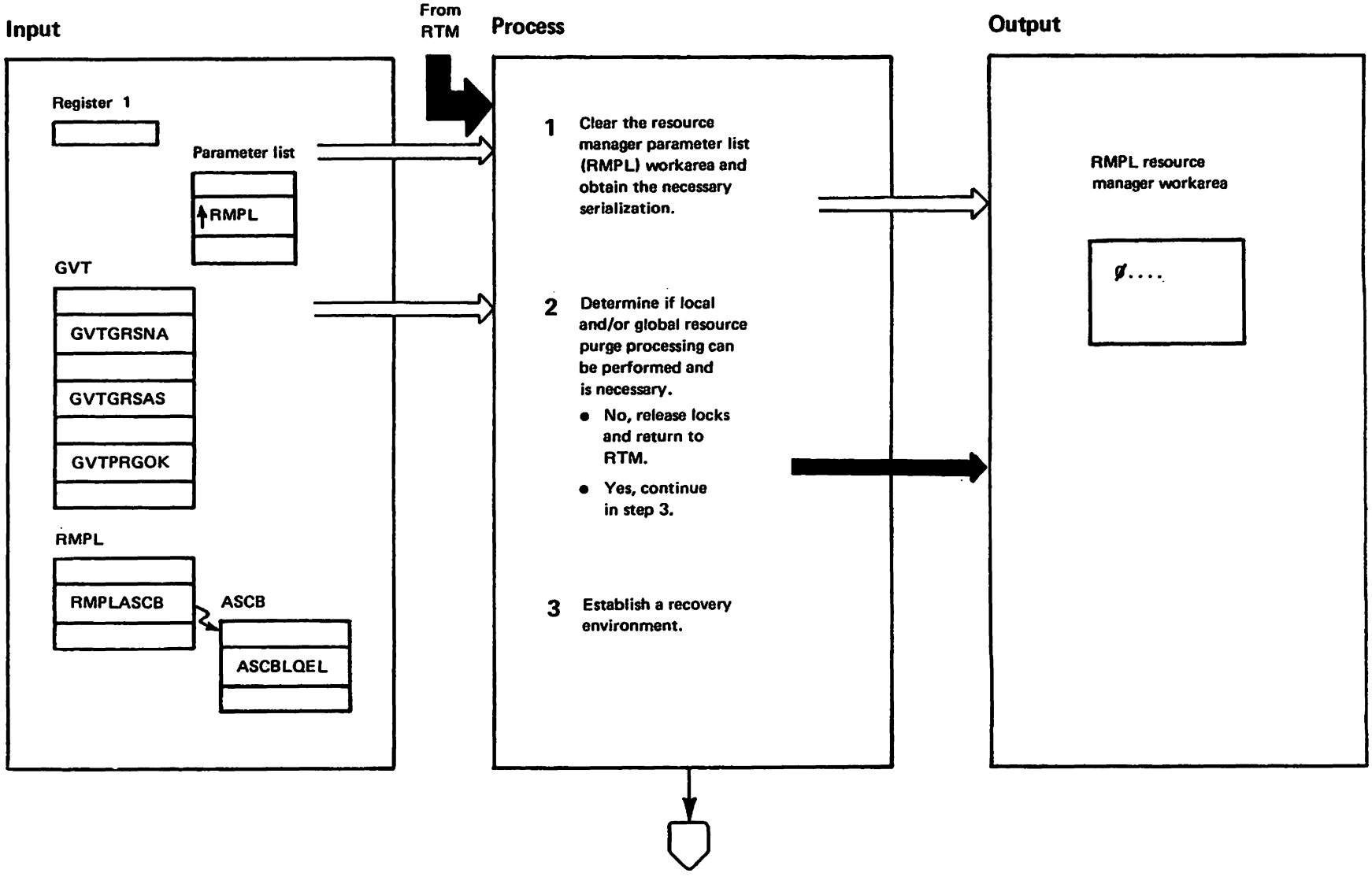
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-35. ISGGRP00 – Global Resource Processor (Part 20 of 20)

Extended Description	Module	Label
12 When none of the request type flags are on in QWBHFLG3, the QWB is invalid. ISGGRP00 issues an ABEND X'09A' with a reason code X'E200' set in register 15. (ISGGFRR0 retries at label GRPRTRY2 in ISGGRP00).		
13 At retry label GRPRTRY2, ISGGRP00 calls ISGSDAL to return the invalid QWB to the storage manager. ISGGRP00 returns to step 5 where it processes the next QWB.	ISGSDAL	

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-36. ISGGTRM0 – ENQ/DEQ/RESERVE Termination Resource Manager (Part 1 of 4)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

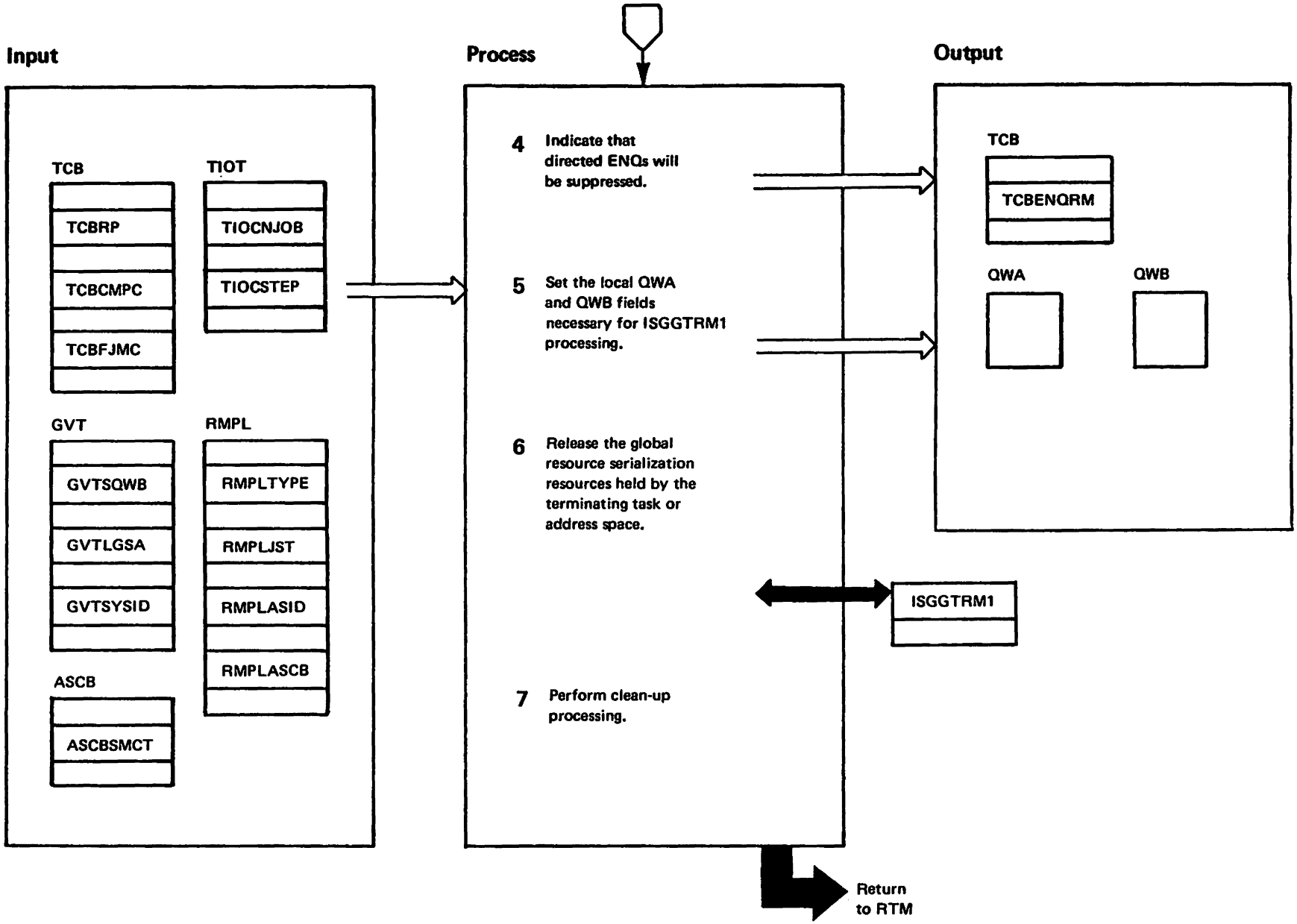
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-36. ISGGTRM0 – ENQ/DEQ/RESERVE Termination Resource Manager (Part 2 of 4)

Extended Description	Module	Label
ISGGTRM0, the ENQ/DEQ/RESERVE termination resource manager, receives control from RTM when a task or address space is being terminated. Input to ISGGTRM0 is a parameter list, pointed to by register 1, that contains the address of the RMPL. ISGGTRM0 receives control in the terminating address space during normal and abnormal task termination and in the master address space during normal and abnormal address space termination. ISGGTRM0 determines if there are resources needing to be purged. If so, it prepares them for purging and calls ISGGTRM1 to purge them. After ISGGTRM1 purges the resources, it returns to ISGGTRM0 for clean up processing. Note that this routine does not clean up any global resources acquired by the terminating task or address space until the global resource serialization address space has been initialized and the global resource processor (ISGGRP00) has run.		
1 ISGGTRM0 copies the resource manager parameter list (RMPL) workarea into the RMPL resource manager workarea (RMPLRMWA) and sets it to zeroes. ISGGTRM0 then obtains the local lock of the current address space and the CMS ENQ/DEQ lock. Holding these locks allows ISGGTRM0 to serialize the local queue workarea (QWA) and local resource processing.	ISGGTRM0	
2 If the global resource serialization address space is not initialized (GVTGRSAS= '0'B), then purge processing cannot be performed. ISGGTRM0 releases the locks and returns control to RTM. If there are no global resources on the local resource queue (ASCBLQEL) and the task owns no global resources for task termination (TCBGRES=0) or the global resource serialization address space is not initialized, purge processing is not necessary. ISGGTRM0 releases the locks and returns control to RTM.		
3 ISGGTRM0 issues a SETFRR macro instruction to establish ISGGFRR0 as its recovery routine.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-36. ISGGTRM0 – ENQ/DEQ/RESERVE Termination Resource Manager (Part 3 of 4)



 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-36. ISGGTRM0 – ENQ/DEQ/RESERVE Termination Resource Manager (Part 4 of 4)

Extended Description **Module** **Label**

4 ISGGTRM0 sets the TCBENQRM bit to 1. Setting this bit while holding the CMS ENQ/DEQ lock ensures that the ENQ/DEQ routine (ISGGNQDQ) suppresses all ENQs directed ot the terminating task. (ISGGNQDQ abends all callers directing ENQ requests to a terminating task.)

5 ISGGTRM0 sets all the fields necessary for processing in ISGGTRM1. The necessary QWA fields are:

QWACMS	QWACSYS
QWAFRR	QWAQWBHS
QWAREQLL	QWAGSA
QWACSYID	QWASTPNM
QWAJOBNM	QWAJSTEP
QWASYSID	QWACOMPC
QWAASID	QWARB
QWAASCB	QWAGBLRS

If a task that is in "step-must-complete" mode is being terminated, ISGGTRM0 also sets the QWARMC bit in the QWA so that, when ISGGTRM1 completes processing, ISGGTRM0 will issue a STATUS macro with the RMC option.

If either a task or address space that is in "step-must-complete" mode is being abmormally terminated, ISGGTRM0 sets the QWAABDMC bit in the local QWA so that a message is set up during local or global resource purge processing.

The QWB field necessary for ISGGTRM1 processing is QWBSMPL.

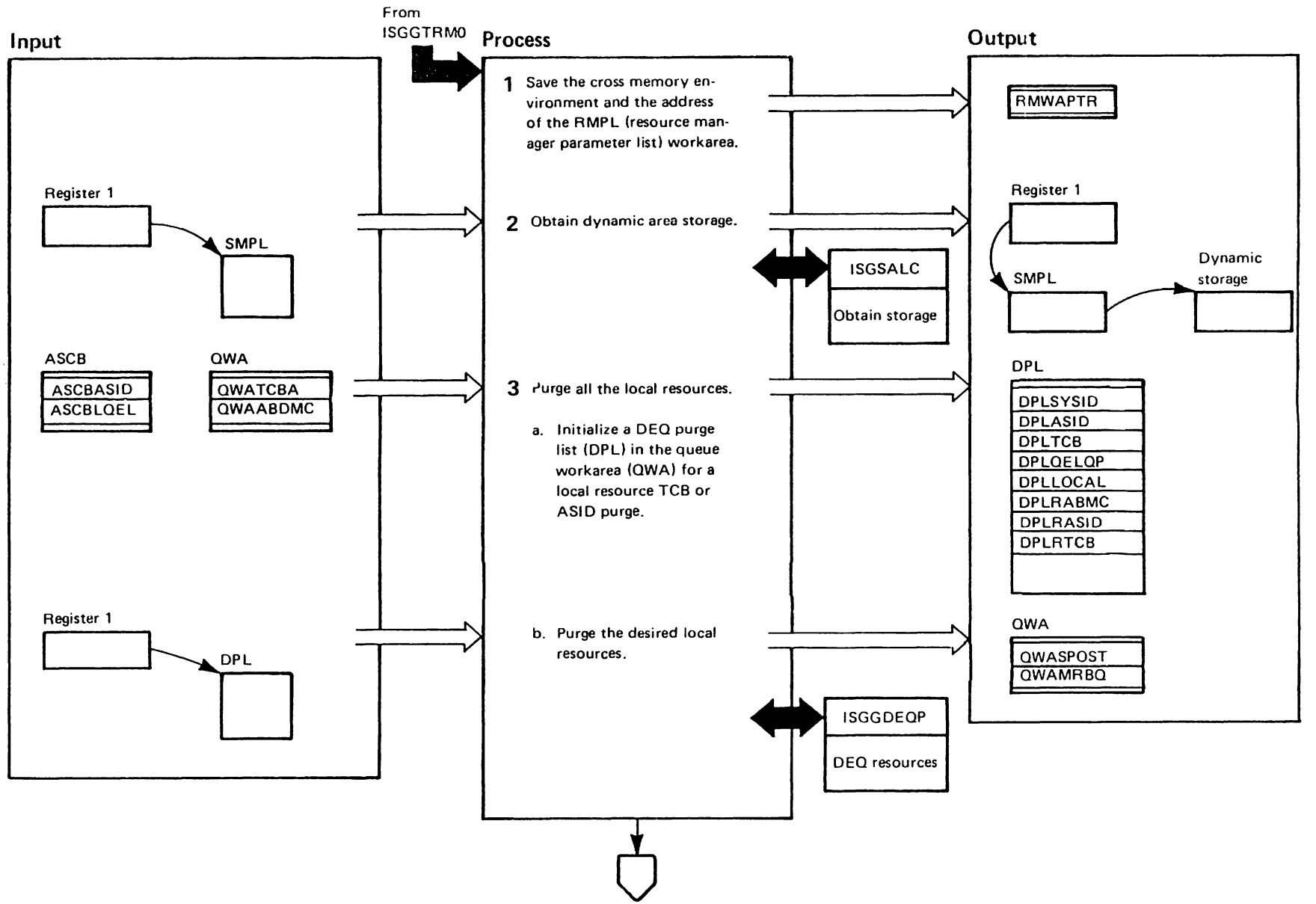
6 ISGGTRM0 issues a program call (PC) instruction to ISGGTRM1, which purges the global resource serialization resources held by the terminating task or address space. ISGGTRM0 passes the address of the RMPL workarea in register 1. The RMPL workarea is used to save registers and flags during the PC instruction.

7 When ISGGTRM1 completes processing, it issues a program transfer (PT) instruction to return control to this step. ISGGTRM0 does the following clean up tasks:

- Releases the CMS ENQ/DEQ lock.
- Deletes the recovery routine.
- Releases the local lock.
- Issues a STATUS macro with the RMC (reset-must-complete) option when necessary to reset a task dispatchable.
- Issues an SPOST macro when necessary to synchronize all outstanding cross memory POSTs.

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-37. ISGGTRM1 – ENQ/DEQ/RESERVE Termination Resource Manager (Part 1 of 6)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

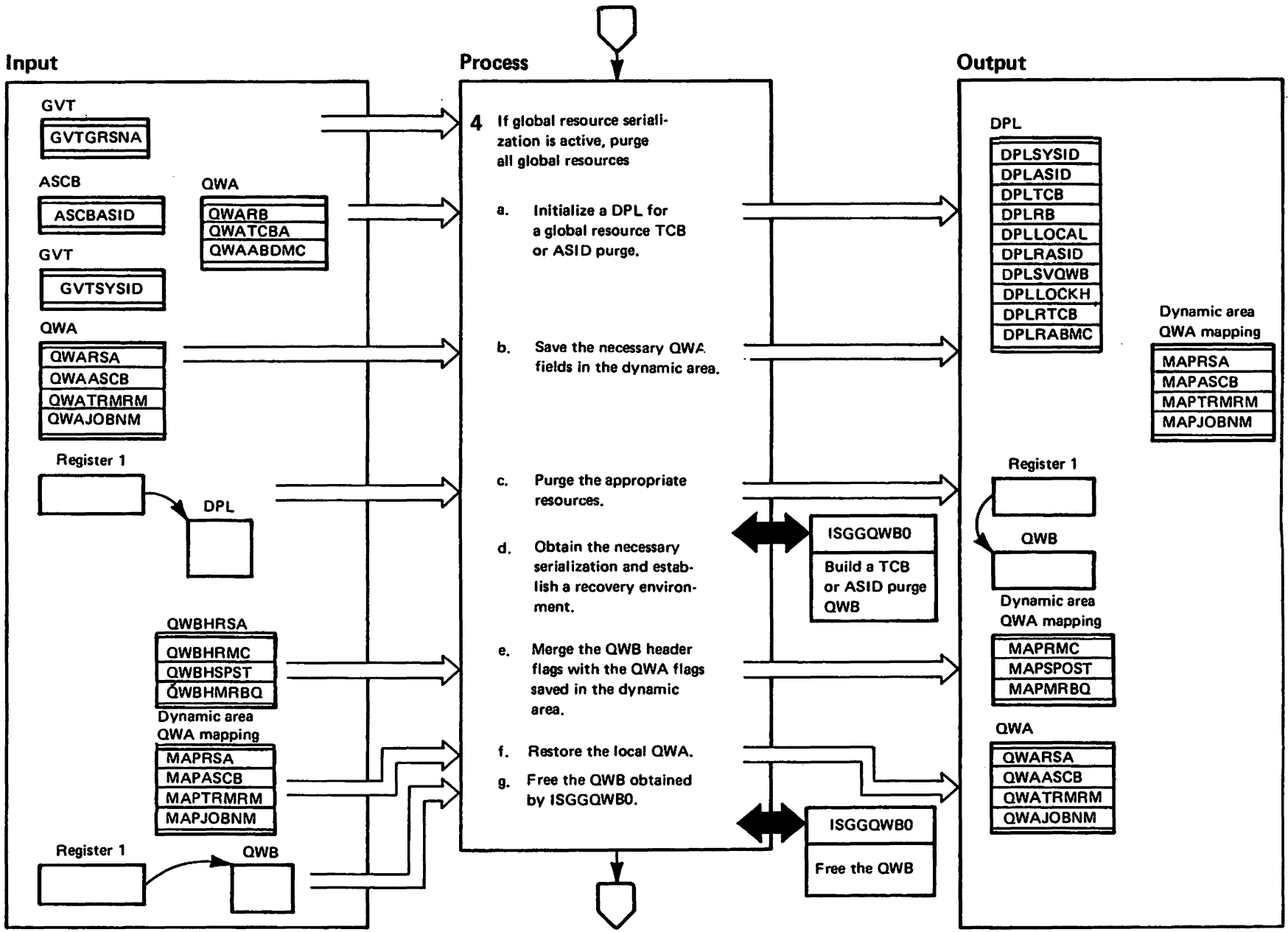
Diagram GRS-37. ISGGTRM1 – ENQ/DEQ/RESERVE Termination Resource Manager (Part 2 of 6)

Extended Description	Module	Label
ISGGTRM1 receives control from ISGGTRM0 via a PC instruction to purge all local and global ENQ/RESERVE resources acquried by the terminating task or address space. The input to this routine is the address of the RMPL workarea in register 1.		
1 ISGGTRM1 issues a PCLINK STACK macro to save the STACK entry that ISGGTRM0 created for ISGGTRM1. ISGGTRM1 saves the RMPL workarea address in RMWAPTR so it can pass this address back to ISGGTRM0.	ISGGTRM1	
2 ISGGTRM1 obtains a dynamic area to be used as working storage for this routine. ISGGTRM1 establishes addressability to the dynamic area and sets the area to zeroes.	ISGSALC	
3 To purge local resources:		
a. ISGGTRM1 initializes the following fields in the DEQ purge list (DPL): • DPLSYSID = 0 for a local purge • DPLASID= the ASID of the address space in which termination is occurring • DPLTCB= pointer to a TCB or 0. If a task is being terminated, ISGGTRM1 uses the TCB address that ISGGTRM0 passed in the QWATCBA field. If an address space is being terminated, ISGGTRM1 sets this field to 0. • DPLQELQP= pointer to a queue element (QEL) of the queue containing the resources to be purged. ISGGTRM1 purges local resources from the ASCB local resource queue (ASCBLQEL). • DPLLOCAL (the local/global flag)= 1 to indicate a local purge • DPLRASID (the ASID purge flag)= 1 or DPLRTCB (the TCB purge flag)= 1 to indicate either an ASID or TCB purge • DPLRABMC (the must-complete flag)= 1 if the TCB or ASID failed in the must-complete mode. This flag is set so that the appropriate error message will be set up.	ISGGTRM1	

Extended Description	Module	Label
3 (continued) b. ISGGTRM1 calls the DEQ processor (ISGGDEQP) to purge all local resources as requested, passing the DPL in register 1. Before returning to ISGGTRM1, ISGGDEQP sets an indicator (QWASPOST) showing whether an SPOST (synchronization of outstanding cross memory POSTs) is necessary. ISGGDEQP also places in the QWAMRBQ field the beginning address of a queue of messages to be issued.	ISGGDEQP	

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-37. ISGGTRM1 -- ENQ/DEQ/RESERVE Termination Resource Manager (Part 3 of 6)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

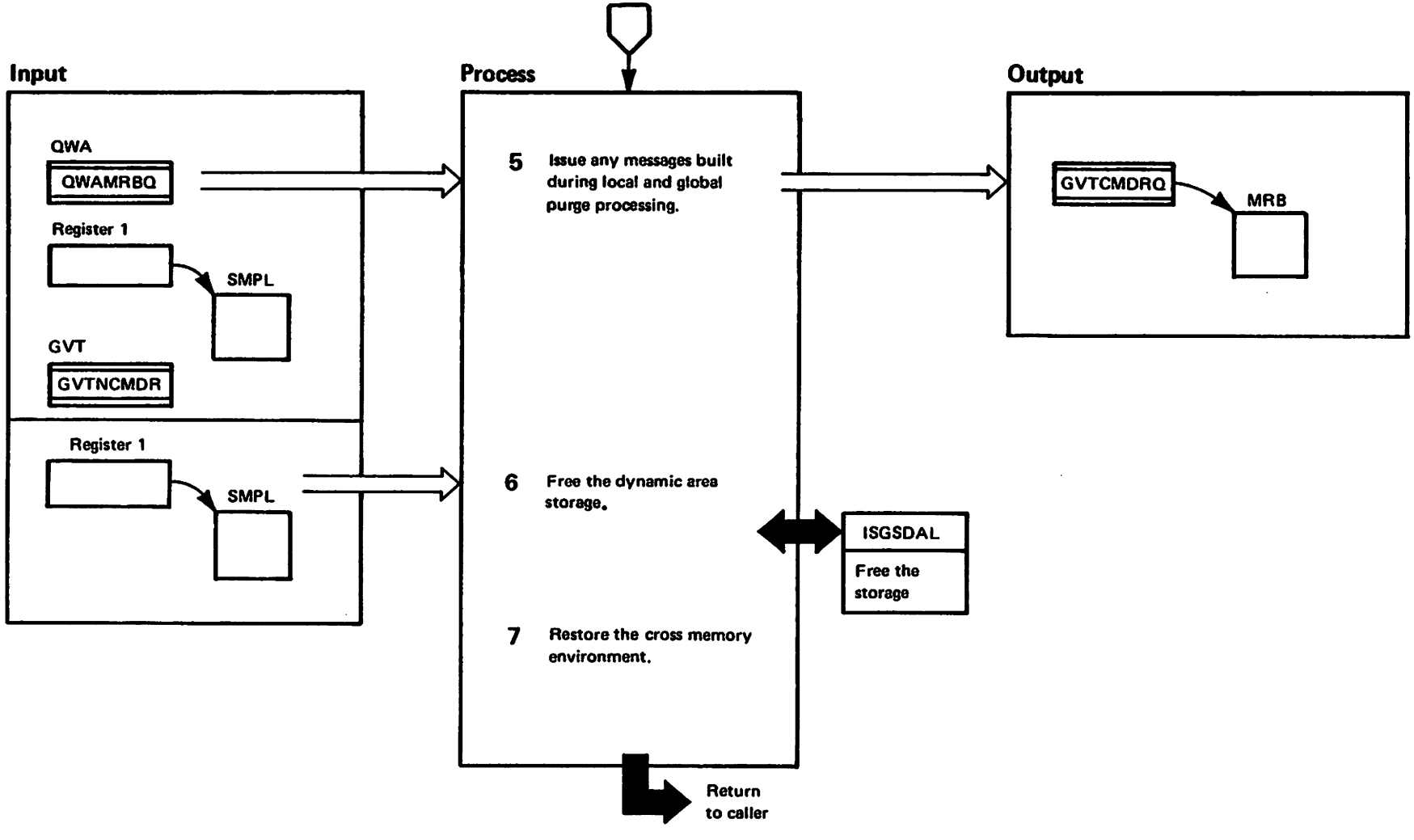
"Restricted Materials of IBM"
Licensed Materials – Property of IBM

Diagram GRS-37. ISGGTRM1 – ENQ/DEQ/RESERVE Termination Resource Manager (Part 4 of 6)

Extended Description	Module	Label
4 When global resource serialization is not active (GVTGRSNA=0) or when the task owns no global resources for task termination (QWAGBLRS=0), global purge processing is not required and ISGGTRM1 skips this step. Otherwise, ISGGTRM1 purges all global resources if purge processing is allowed (GVTPRGOK=1). (The CMS ENQ/DEQ lock that ISGGTRM0 obtained serializes the GVTGRSNA field.)	ISGGTRM1	
ISGGTRM1 invokes ISGGQWB0 (entry point ISGGQWB5) to present the purge request (either TCB or ASID) to each system in the global resource serialization ring.	ISGGQWB0	ISGGQWB5
a. ISGGTRM1 initializes the following data in the DEQ purge list (DPL): • DPLSYSID= the current system ID • DPLASID= the ASID of the address space in which termination is occurring • DPLTCB= pointer to a TCB or 0. If a task is being terminated, ISGGTRM1 uses the TCB address that ISGGTRM0 passed in the QWATCBA field. If an address space is being terminated, ISGGTRM1 sets this field to 0. • DPLRB= address of the current RB. ISGGQWB5 uses this address when issuing a WAIT macro. • DPLLOCAL (the local/global flag)= 0 to indicate a global purge • DPLRASID (the ASID purge flag)= 1 or DPLRTCB (the TCB purge flag)= 1 to indicate either an ASID or TCB purge • DPLSVQWB (the save QWB flag)= 1 to tell ISGGQWB5 to return the QWB (queue work block) it obtains • DPLRABMC (the must-complete flag)= 1 if the TCB or the ASID failed in the must-complete mode • DPLLOCKH (the lock-held flag)= 1 to indicate that locks are held on entry to ISGGQWB5		
b. ISGGTRM1 saves the following local QWA fields in the dynamic area: QWARSA, QWAASCB, QWATRMRM, and QWAJOBNM.		
c. ISGGTRM1 calls ISGGQWB0 at entry point ISGGQWB5 to place the purge request on the request queue and to issue a WAIT macro. ISGGQWB5 releases the local lock of the home address space and the CMS ENQ/DEQ lock and releases all FRRs on the stack before issuing the WAIT macro. The global resource processor (ISGGRP00) periodically checks this queue. When it finds the queued purge request, it purges the appropriate resources and posts ISGGQWB5 when finished. ISGGQWB5 then returns the address of the purge QWB to ISGGTRM1.	ISGGQWB0	ISGGQWB5
d. ISGGTRM1 obtains the local lock of the home address space and the CMS ENQ/DEQ lock to restore serialization that ISGGQWB5 released. It also re-establishes ISGGFRR0 as its FRR.		
e. The global resource processor (ISGGRP00) initialized the fields in the QWB header. These include QWBHSPST, QWBHRMC, and QWBHMRBQ. ISGGTRM1 merges these fields with the QWA flags that were saved in the dynamic area's QWA mapping.		
f. ISGGTRM1 moves the QWA and QWB fields saved in the dynamic area QWA mapping back into the local QWA.		
g. ISGGTRM1 calls ISGGQWB0 at entry point ISGGQWBF to free the QWB obtained and returned by ISGGQWB5.	ISGGQWB0	ISGGQWBF

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-37. ISGGTRM1 – ENQ/DEQ/RESERVE Termination Resource Manager (Part 5 of 6)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

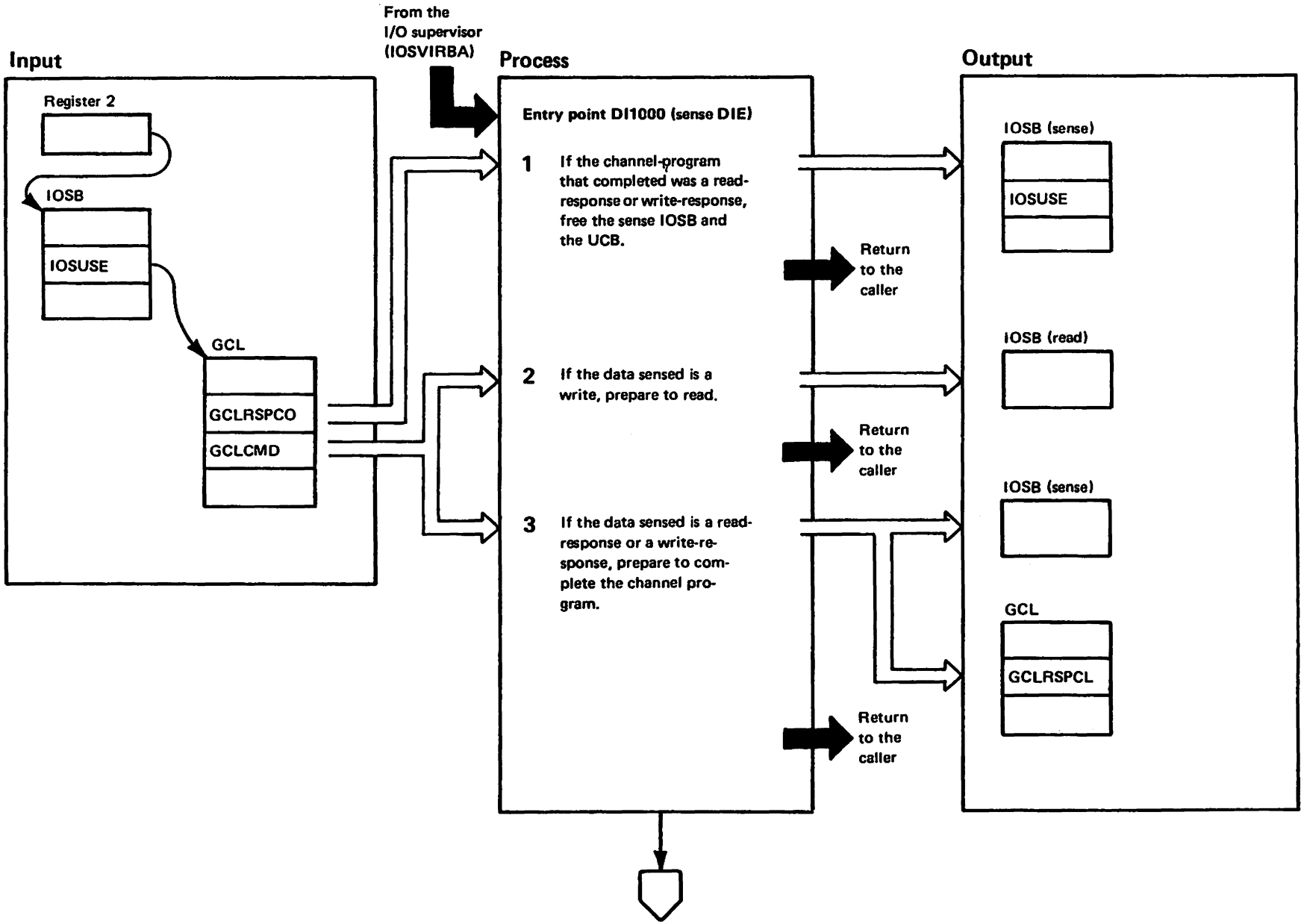
"Restricted Materials of IBM"
Licensed Materials – Property of IBM

Diagram GRS-37. ISGGTRM1 – ENQ/DEQ/RESERVE Termination Resource Manager (Part 6 of 6)

Extended Description	Module	Label
5 If any messages were set up during local and global purge processing (QWAMRBQ is not zero), ISGGTRM1: • Obtains the storage for the header messages from the global resource serialization storage manager (ISGSALC). Two message request blocks (MRBs) are obtained for task termination, one MRB for address space termination. • Builds the necessary header messages. • Places the header messages at the beginning of the chain of MRBs built during local and global purge processing. The QWAMRBQ field points to the MRB chain. • Places the chain of MRBs onto the command request queue pointed to by the GVTCMDRQ field. • If the global resource serialization command router (ISGCMDR) is active, issues a cross address space POST for the ECB to notify ISGCMDR of work. Since ISGCMDR checks the command request queue when it becomes active, any MRBs placed on the queue by this routine get processed, even if ISGGTRM1 does not issue a POST.	ISGGTRM1	
6 ISGGTRM1 calls the global resource serialization deallocation routine (ISGSDAL) to release the dynamic area storage. ISGGTRM1 passes in register 1 the address of the storage manager parameter list (SMPL), which contains the address of the storage to be freed.	ISGSDAL	
7 ISGGTRM1 issues a PCLINK UNSTACK macro to retrieve the STACK entry created for this routine. ISGGTRM1 then issues a PT (program transfer) instruction to ISGGTRM0 to perform cleanup processing. Input to ISGGTRM0 is the address of the RMPL workarea, which ISGGTRM1 saved earlier in RMWAPTR.	ISGGTRM1	

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-38. ISGJDI – Global Resource Serialization CTC Driver DIE (Part 1 of 12)



"Restricted Materials of IBM"
Licensed Materials - Property of IBM

LY28-1695-0 (c) Copyright IBM Corp. 1987

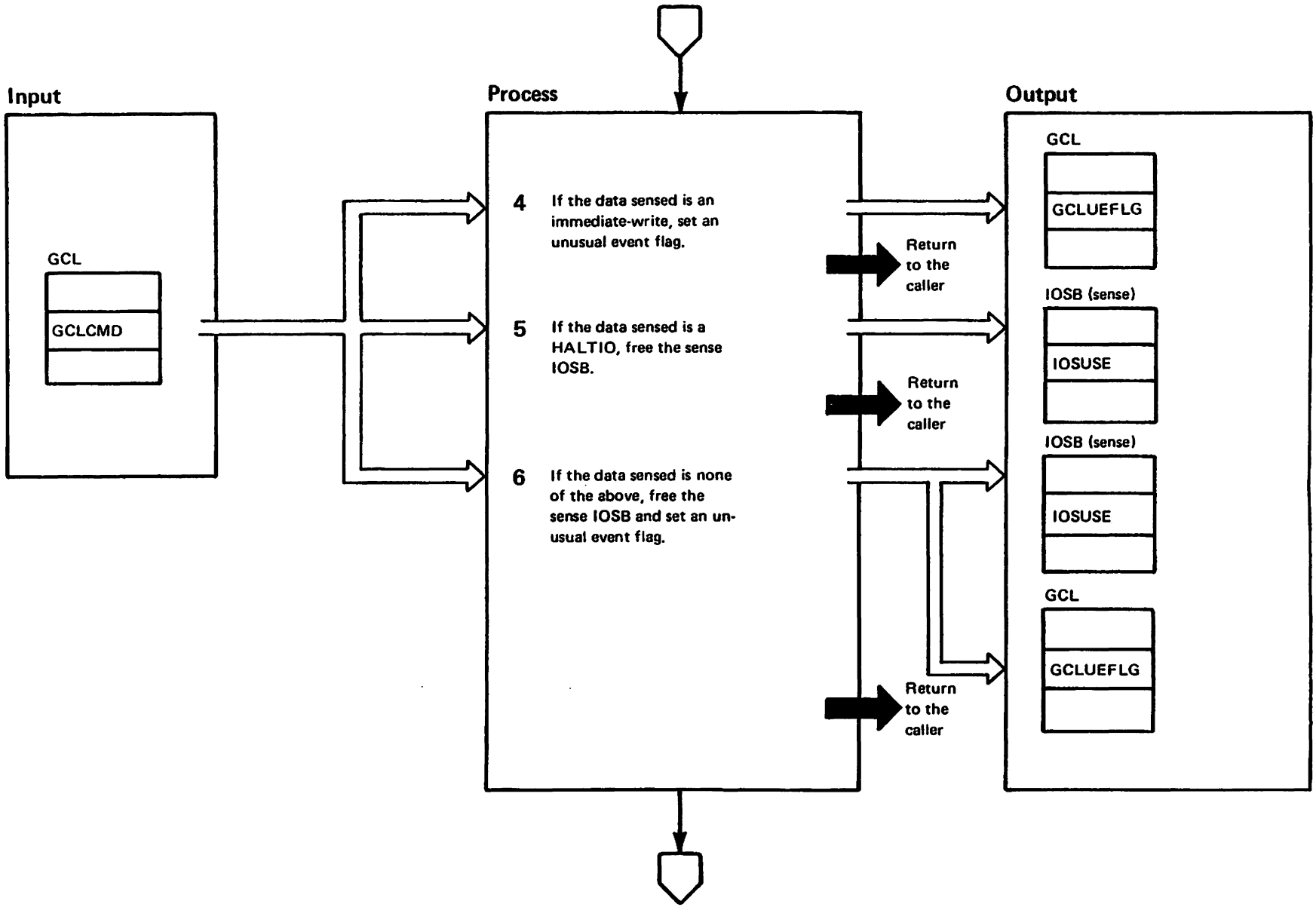
"Restricted Materials of IBM"
Licensed Materials – Property of IBM

Diagram GRS-38. ISGJDI – Global Resource Serialization CTC Driver DIE (Part 2 of 12)

Extended Description	Module	Label
ISGJDI contains various entry points that are either disabled interrupt exits (DIEs) or error-handling exits. These entry points receive control from the I/O supervisor (IOS) to complete the processing of a channel program. The channel programs processed are sense, read, write, read-response, and write-response. Except for the sense channel program, which IOS initiates, the CTC driver initiates these channel programs to enable two systems, connected to a CTC, to communicate.		
The DIE entry points (DI1000, DI2000, and DI3000) complete the processing of sense (including read-response and write-response), write, and read channel programs, respectively. The error-handling entry points (ABN0000, NRM0000, and PGAD000) receive control from IOS only if an ISGJDI DIE entry point (DI2000 or DI3000) returned to IOS requesting further processing to handle an error, as indicated by setting register 15 to 0.		
During processing, ISGJDI references and updates the global resource serialization CTS driver link control block (GCL) and the I/O supervisor block (IOSB). On each processor, there is one GCL for a CTC and three IOSBs for a GCL. The IOSBs are the sense IOSB, the read IOSB, and the write IOSB. IOS or the global resource serialization CTC driver initiates the sense IOSB; the global resource serialization CTC driver initiates both the read IOSB and the write IOSB.		
Each entry point in ISGJDI establishes addressability to module ISGJRCV which is the functional recovery routine (FRR) for ISGJDI.		
Note: A read channel program consists of a read CCW and a write response CCW is all the systems in the main ring are of a pre MVS/220 level or the main ring has systems of a mixed level. If all systems in the main ring are of an MVS/220 level or higher, a read channel program consists of a read CCW.		
A write channel program consists of a write CCW and a read response CCW if all the systems in the main ring are of pre MVS/220 level or the main ring has systems of a mixed level. If all systems in the main ring are of an MVS/220 level or higher, a write channel program consists of a write CCW.		
Entry point DI1000 processes the sense IOSB received from IOS. Step 1 describes the processing performed if the sense IOSB was inititiated during a previous pass through this entry point (refer to step 3). Steps 2 through 6 describe the processing performed if the sense IOSB was initiated by IOS.	ISGJDI	DI1000
1 If the completed I/O is a read-response or a write-response resulting from DI1000 having initiated the sense IOSB to process a read or a write channel program that did not complete its function successfully (refer to step 3), DI1000 frees the sense IOSB (by turning off a bit in the IOUSE field of the IOSB) and frees the UCB (by invoking the IOSLEVEL-RESET service). Control returns to IOS indicating that no further processing is requested (register 15=8).		
2 If the data sensed by IOS is a write CCW operation code (meaning that the remote system is sending a message or data to this system via the CTC), DI1000 prepares for a read. It frees the sense IOSB, obtains and initializes the read IOSB, and returns to IOS requesting the start of I/O (register 15=4).		DI8000 DI1000
If DI1000 is unable to locate a buffer or finds that the CTC is now offline, DI1000 sets an unusual event flag in the GCL and returns to IOS, requesting the start of I/O (register 15=4).		
If DI1000 is unable to obtain the read IOSB or a buffer, it sets unusual event flags in the GCL and returns to IOS indicating that no further processing is requested (register 15=8).		UEREPT DI1000
3 If the data sensed by IOS is a read-response CCW operation code or a write-response CCW operation code, this system's channel program did not successfully complete its function. (A read channel program consists of a read CCW and a write response CCW is all the systems in the main ring are of a pre MVS/220 level or the main ring has systems of a mixed level. If all systems in the main ring are of an MVS/220 level or higher, a read channel program consists of a read CCW. A write channel program consists of a write CCW and a read response CCW if all the systems in the main ring are of pre MVS/220 level or the main ring has systems of a mixed level. If all systems in the main ring are of an MVS/220 level or higher, a write channel program consists of a write CCW.)		
DI1000 reinitializes the sense IOSB with a write-response channel program to satisfy the sensed read-response CCW or with a read-response channel program to satisfy the sensed write-response CCW. It then places a length value in the GCL to indicate to the remote system that this system's original channel program did not complete successfully. DI1000 returns to IOS requesting the start of I/O (register 15=4).		DI8100 DI1000

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-38. ISGJDI – Global Resource Serialization CTC Driver DIE (Part 3 of 12)



"Restricted Materials of IBM"
Licensed Materials - Property of IBM

LY28-1695-0 (c) Copyright IBM Corp. 1987

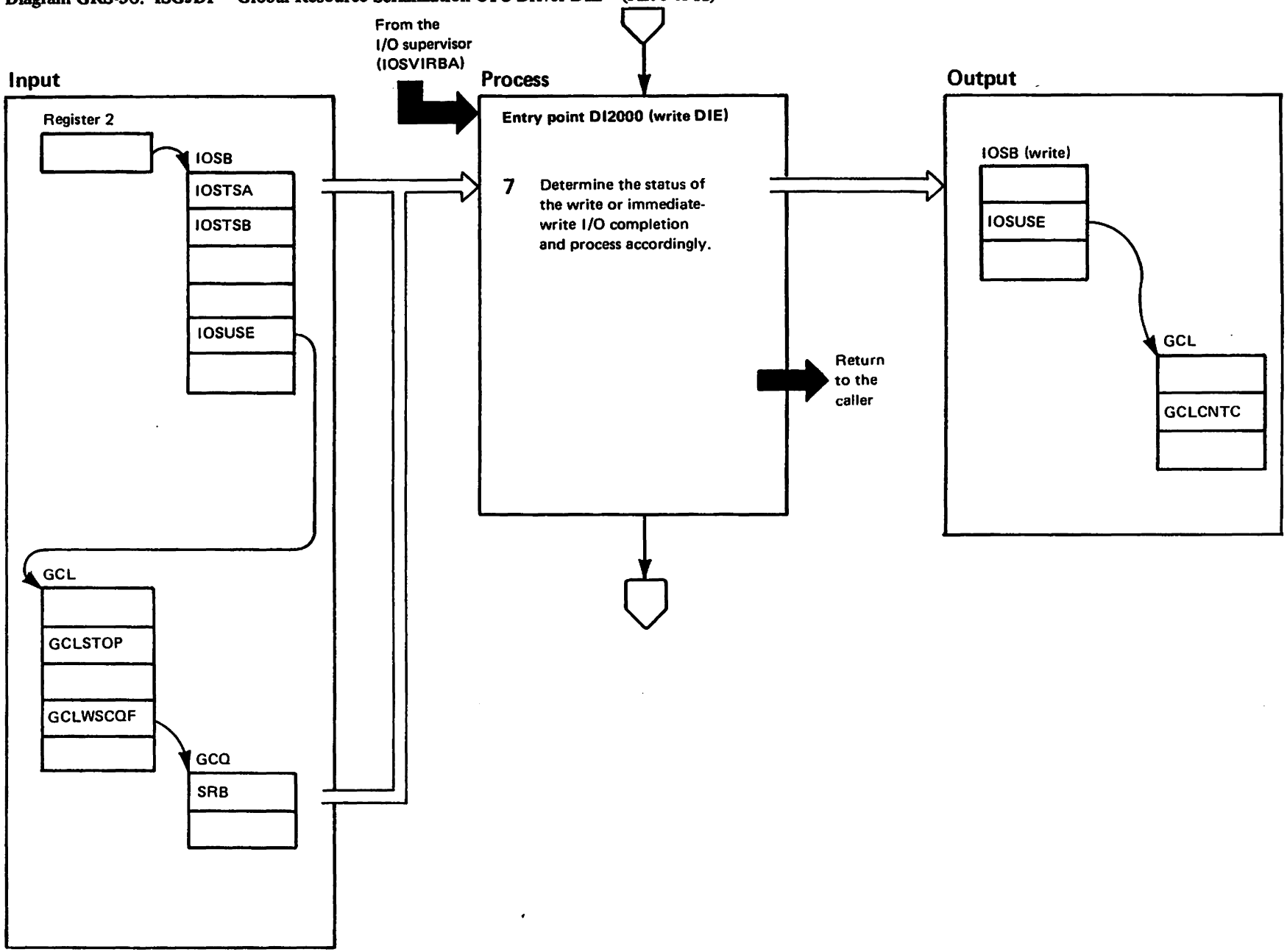
"Restricted Materials of IBM"
Licensed Materials – Property of IBM

Diagram GRS-38. ISGJDI – Global Resource Serialization CTC Driver DIE (Part 4 of 12)

Extended Description	Module	Label
4 If the data sensed by IOS is an immediate-write CCW operation code (meaning that the remote system wants this system to identify itself), DI1000:		
• Sets an unusual event flag in the GCL to indicate that an immediate-write was sensed		UEREPT
• Frees the sense IOSB		DI1000
Control returns to IOS with the indication that no further processing is requested (register 15=8).		
5 If the data sensed by IOS is a HALTIO CCW operation code, DI1000 frees the sense IOSB and returns to IOS indicating that no further processing is requested (register 15=8).		
6 If the data sensed is not any of those described in steps 1 through 5, DI1000 frees the sense IOSB, sets an unusual event flag in the GCL, and returns to IOS indicating that no further processing is requested (register 15=8).		UEREPT DI1000

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-38. ISGJDI – Global Resource Serialization CTC Driver DIE (Part 5 of 12)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

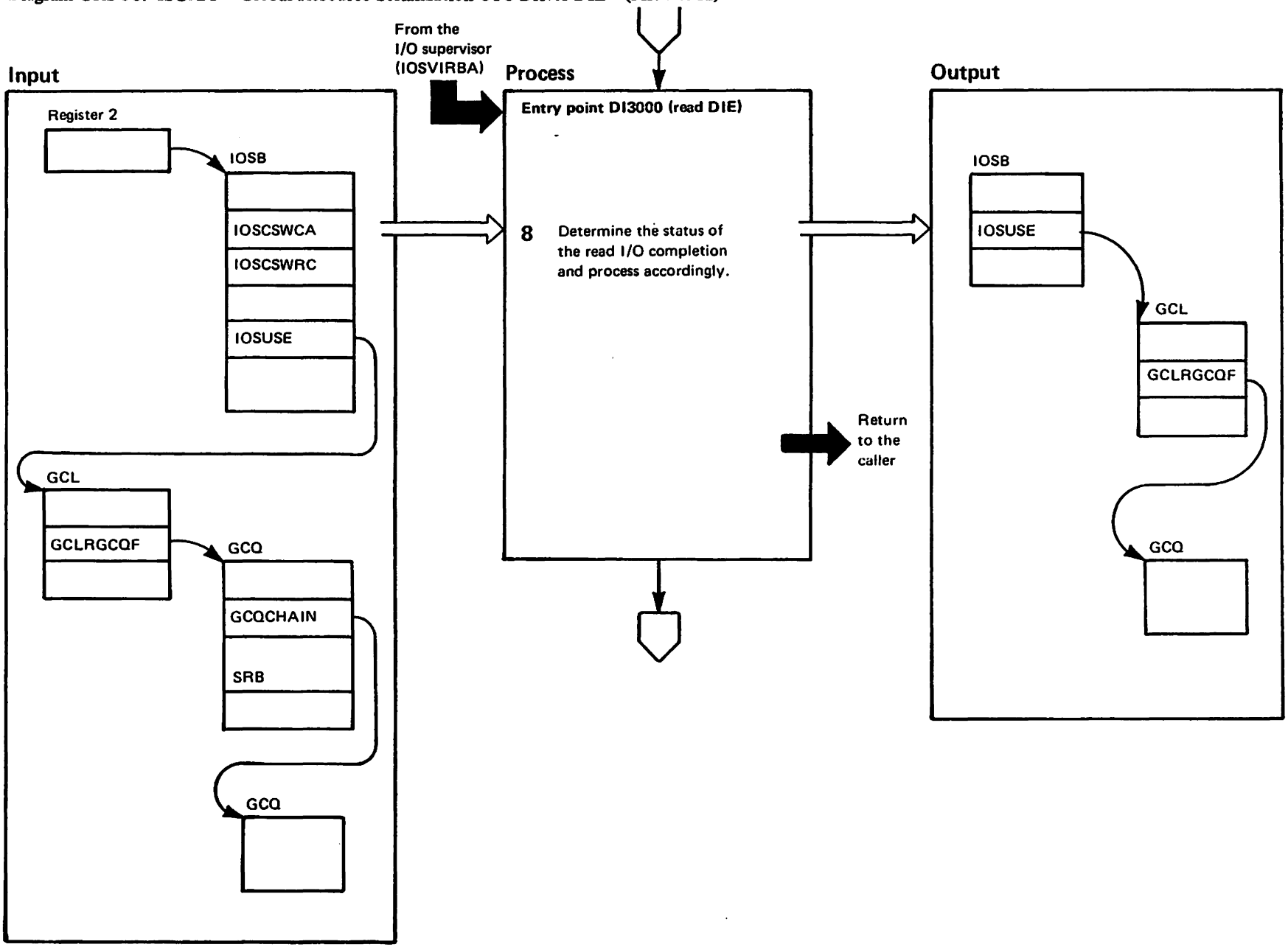
"Restricted Materials of IBM"
Licensed Materials – Property of IBM

Diagram GRS-38. ISGJDI – Global Resource Serialization CTC Driver DIE (Part 6 of 12)

Extended Description	Module	Label
Entry point DI2000 processes I/O completions (from IOS) for write and immediate-write channel programs which use the write IOSB. (A write channel program consists of a write CCW and a read-response CCW.)		DI2000
7 If a write or immediate-write completes without error (no unit exception, no unit check, and no status bits on), DI2000: • Frees the write IOSB • Removes the global resource serialization CTC driver queueing element (GCQ) from the write queue • Schedules the SRB (if supplied) • Adds one to the I/O write-completed count in the GCL Control returns to IOS indicating that no further processing is requested (register 15=8).		
If a write completes with an error and no HALTIO is in progress, DI2000 returns to IOS indicating that further processing is requested (register 15=0). Otherwise (a HALTIO is in progress), DI2000 frees the write IOSB, adds one to the write-completed count in the GCL, and returns to IOS indicating that no further processing is requested (register 15=8).		
If an immediate-write completes with error, DI2000 returns to IOS indicating that further processing is requested (register 15=0).		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-38. ISGJDI – Global Resource Serialization CTC Driver DIE (Part 7 of 12)



 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

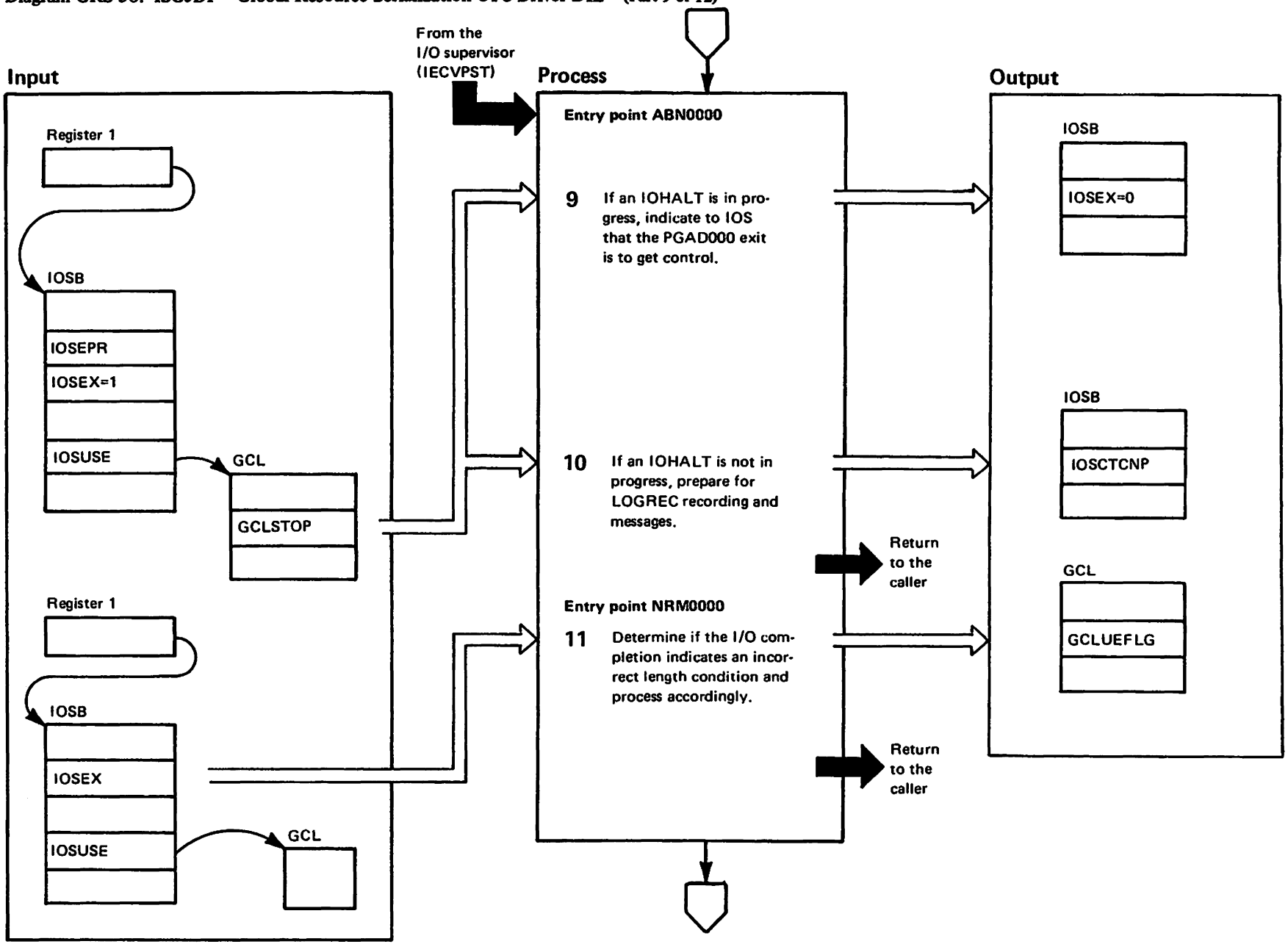
"Restricted Materials of IBM"
Licensed Materials – Property of IBM

Diagram GRS-38. ISGJDI – Global Resource Serialization CTC Driver DIE (Part 8 of 12)

Extended Description	Module	Label
Entry point DI3000 processes I/O completions (from IOS) for read channel programs which use the read IOSB. (A read channel program consists of a read CCW and a write-response CCW).		DI3000
8 If the read completed its function without error, DI3000 frees the read IOSB, removes the global resource serialization CTC driver queueing element (GCQ) from the read queue, and schedules the SRB (if one was supplied). Control returns to IOS indicating that no further processing is requested (register 15=8).		
If the read completed with an error or did not complete the read function, DI3000 returns to IOS indicating that further processing is requested (register 15=0).		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-38. ISGJDI – Global Resource Serialization CTC Driver DIE (Part 9 of 12)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

Diagram GRS-38. ISGJDI – Global Resource Serialization CTC Driver DIE (Part 10 of 12)

Extended Description	Module	Label
Entry point ABN0000 is the error-handling exit used by IOS to process all abnormal I/O completions, except those indicating an incorrect length condition.		ABN000
9 If an IOHALT is in progress (cleanup of the CTC is taking place), ABN0000 sets the IOSEX flag to 0 and returns to IOS. No LOGREC recording, issuing of messages, or retries will be done. IOS then enters the PGAD000 exit to free the IOSB.		
10 If no IOHALT is in progress, ABN000 leaves the IOSEX flag on and indicates to IOS that no retry of I/O is to be done (IOSCTCNR=1). Control returns to IOS for LOGREC recording and for issuing messages. (IOS then enters the PGAD000 termination exit to free the IOSB.)		
Entry point NRM0000 is the error-handling exit used by IOS to process all I/O completions that indicate an incorrect length condition.		NRM0000
11 NRM0000 checks the IOSEX flag to determine if the completed I/O had an incorrect length condition. If NRM0000 finds that IOSEX=1, an incorrect length condition exists and NRM0000 returns to IOS. Otherwise, NRM0000 sets an unusual event flag in the GCL before returning to IOS.		
When IOS gets control, it enters the PGAD000 exit to free the IOSB.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

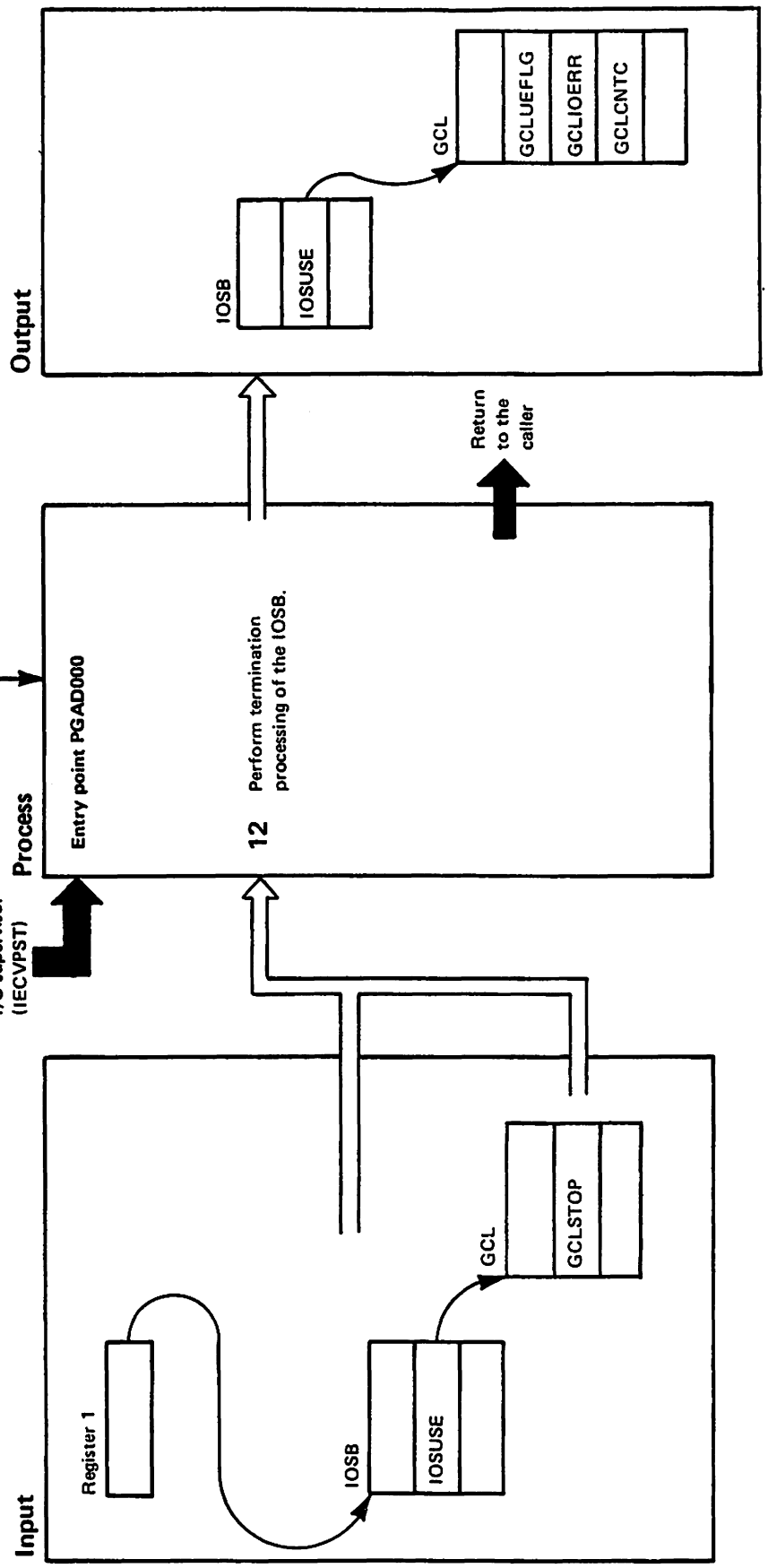


Diagram GRS-38. ISGJDI – Global Resource Serialization CTC Driver DIE (Part 11 of 12)

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-38. ISGJDI – Global Resource Serialization CTC Driver DIE (Part 12 of 12)

Extended Description	Module	Label
Entry point PGAD000 is the error-handling termination exit IOS uses to allow the global resource serialization CTC driver to free the IOSB. PGAD000 is entered after either the ABN0000 or the NRM0000 entry point has executed.		PGAD000
12 PGAD000 frees the IOSB (read or write) associated with the completed I/O. For a write I/O completion, PGAD000 adds one to the write-completed count in the GCL. For a read or a write I/O completion, PGAD000 sets an unusual event flag in the GCL if no HALTIO is in progress (GCLSTOP=0). For an I/O error PGAD000 sets an I/O error indicator (GCLIOERR=1). PGAD000 returns to IOS.		UEREPT PGAD000

Recovery Processing:

If an error occurs while ISGJDI is executing, RTM gives control to ISGJRCV, which is the functional recovery routine (FRR) for ISGJDI. ISGJRCV:

- Fills in the SDWA with module identification data
- Verifies the IOSB, GCL, and GCQ control blocks
- Fills in the variable recording area (SDWAVRA)
- Issues the SDUMP macro to obtain a dump
- Marks the GCL as inoperative
- Frees the IOSB
- Schedules the SRB, if applicable, to handle unusual event processing
- Returns to RTM

LY28-1695-0 (c) Copyright IBM Corp. 1987

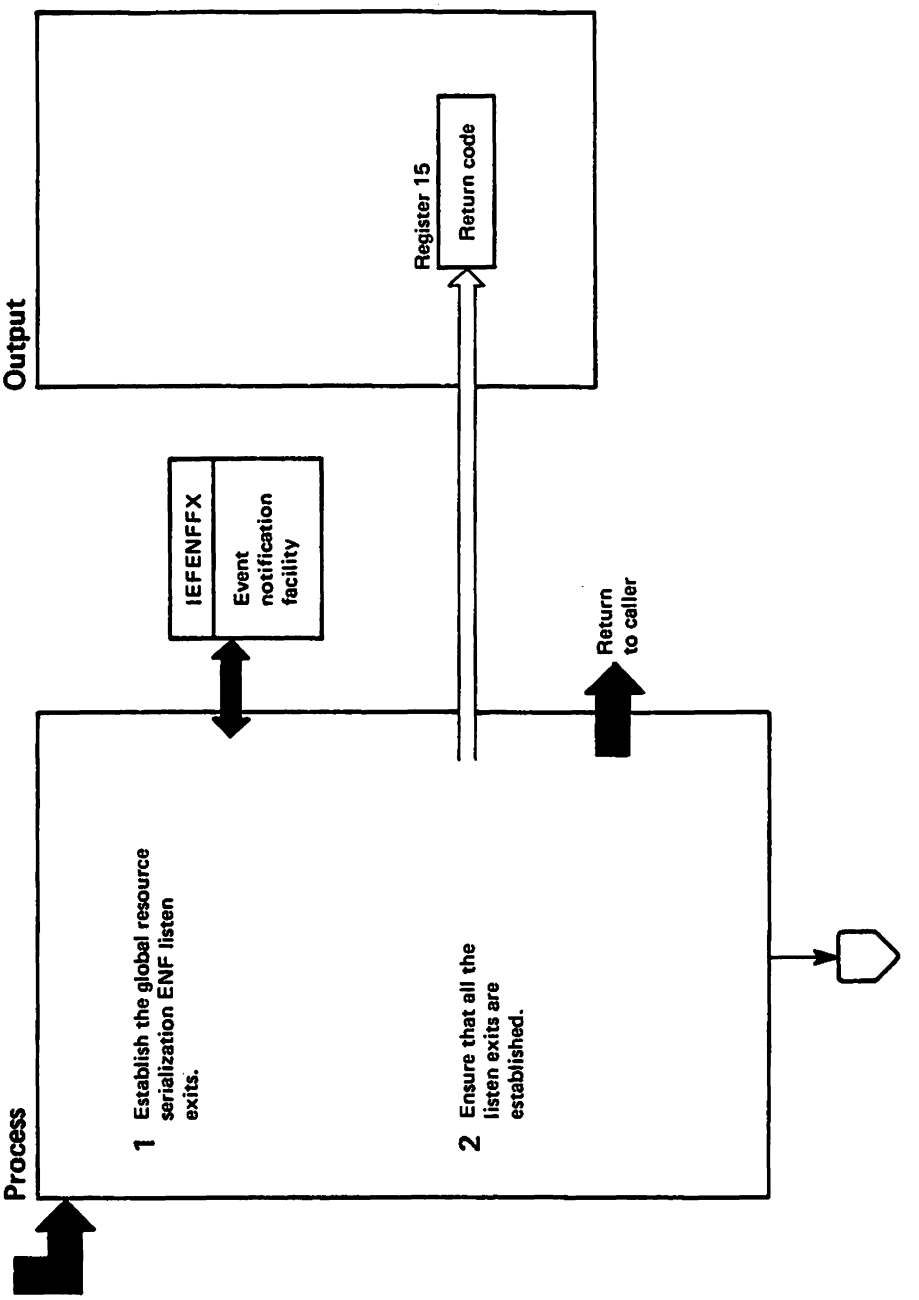


Diagram GRS-39. ISGJENF0 - Global Resource Serialization Event Notification Exits (Part 1 of 8)

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

Diagram GRS-39. ISGJENF0 – Global Resource Serialization Event Notification Exits (Part 2 of 8)

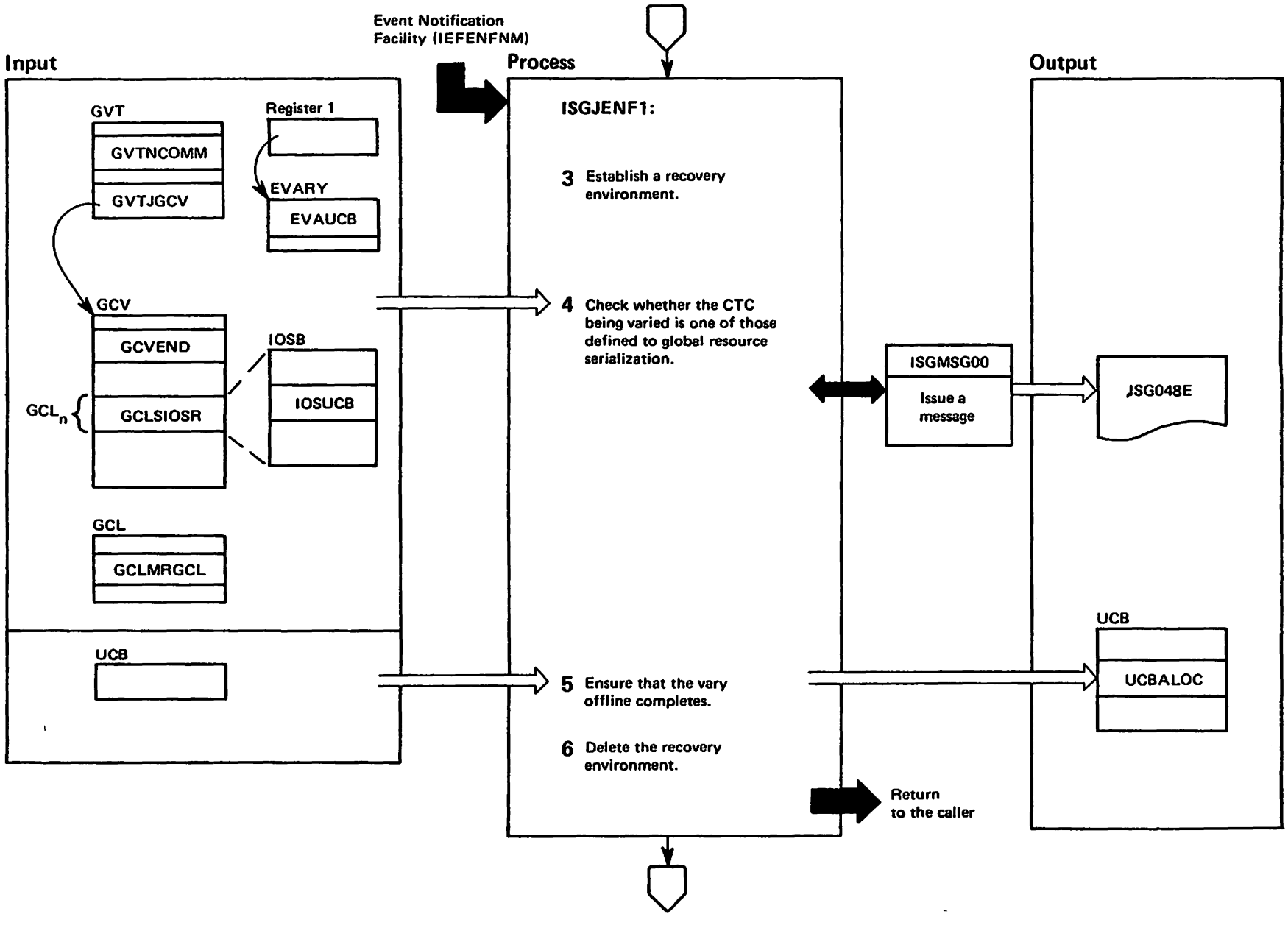
Extended Description	Module	Label
During ring processing initialization, ISGBTC invokes ISGJENF0. ISGJENF0 establishes event notification exits to "listen" for processing conditions associated with varying a CTC online or offline. The "listen" exits notify the operator that the CTCs involved in global resource serialization are being varied offline or online; their processing is described later in this diagram.		
1 Issue the ENFREQ macro instruction to invoke the event notification facility (ENF) in order to establish the following global resource serialization ENF "listen" exits that will apply to CTC devices only: • ISGJENF1 – used to listen for a device pending offline condition. • ISGJENF2 – used to listen for a vary device offline completion condition. • ISGJENF3 – used to listen for a vary device online condition.	ISGJENF0	
2 Verify that the event notification facility successfully established all the global resource serialization ENF "listen" exits; if it did, set a return code of zero; otherwise, set a return code of four. Then return to the caller. The caller, ISGBTC, checks the return code to determine if global resource serialization should become active.		

Recovery Processing for ISGJENF0

ISGJENF0 does not establish any recovery routine for itself but relies on the recovery established by its caller, ISGBTC.

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-39. ISGJENF0 – Global Resource Serialization Event Notification Exits (Part 3 of 8)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

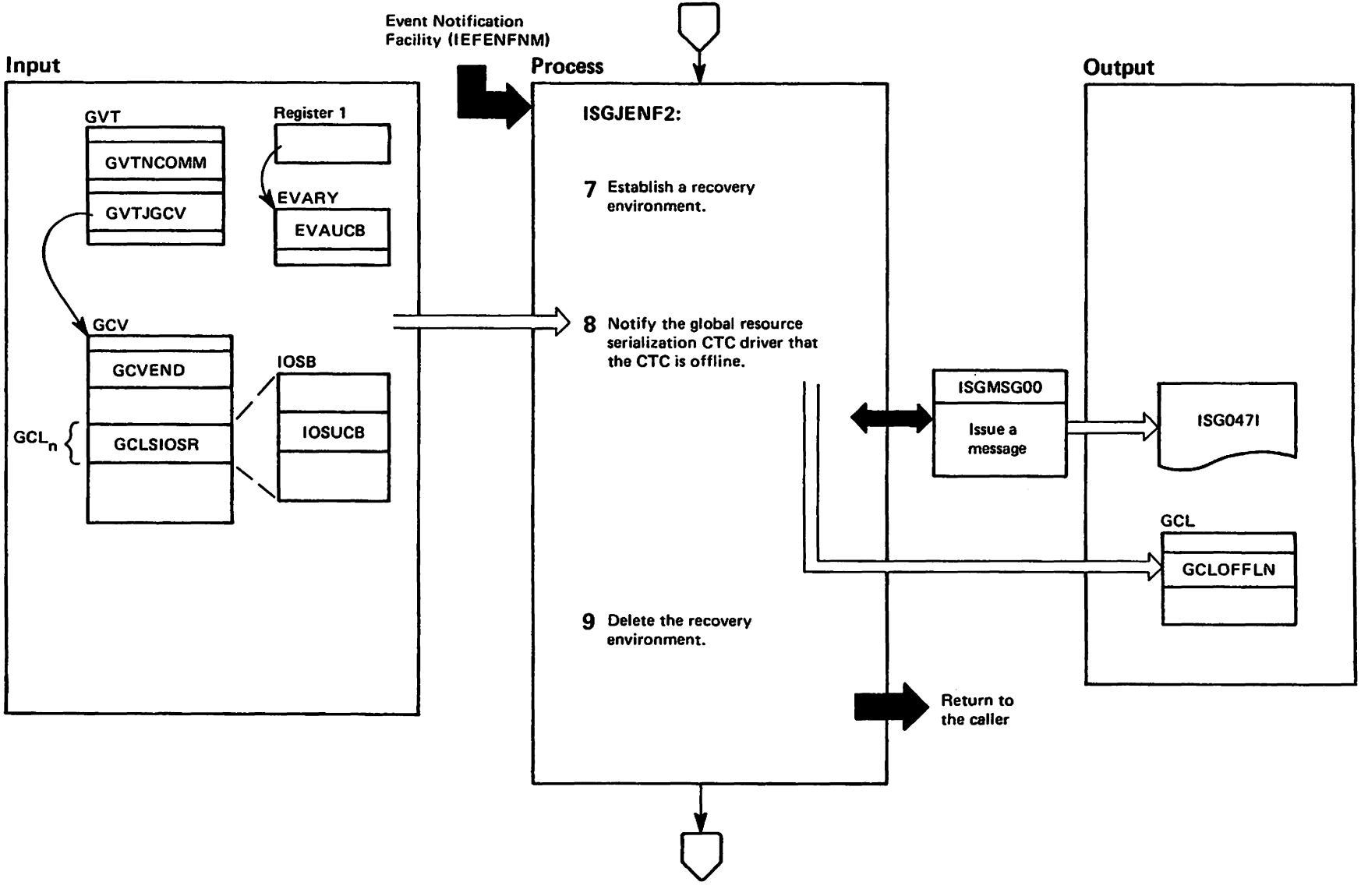
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-39. ISGJENF0 – Global Resource Serialization Event Notification Exits (Part 4 of 8)

Extended Description	Module	Label
ISGJENF1		
This ENF exit allows a VARY device,OFFLINE command to complete if the target of the request is a global resource serialization CTC and the CTC is not being used to send or receive the RSA message.		
3 Establish the ISGJENF0 entry point ISGJENFR as an ESTAE recovery routine. If the ESTAE is not established successfully, issue an ABEND macro instruction with a system completion code of X'09A' and a reason code identifying the nature of the error. If the ESTAE was established successfully, continue processing.		ISGJENFI
4 If ring processing is inactive (GVTNCOMM=1) or if the target of the VARY CTC, OFFLINE command is not a global resource serialization CTC (EVAUCB≠IOSUCB), then return to the caller; otherwise, check to see if the CTC is being used to send or receive the RSA message (GCLMRGCL=1).		
5 If the CTC is not being used to send or receive the RSA message, then ensure that the vary offline request completes by setting UCBAIOC=0; otherwise, invoke ISGMSG00 to issue message ISG0481.		
6 Issue the ESTAE macro instruction to delete the ESTAE recovery routine.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-39. ISGJENF0 – Global Resource Serialization Event Notification Exits (Part 5 of 8)



"Restricted Materials of IBM"
Licensed Materials - Property of IBM

LY28-1695-0 (c) Copyright IBM Corp. 1987

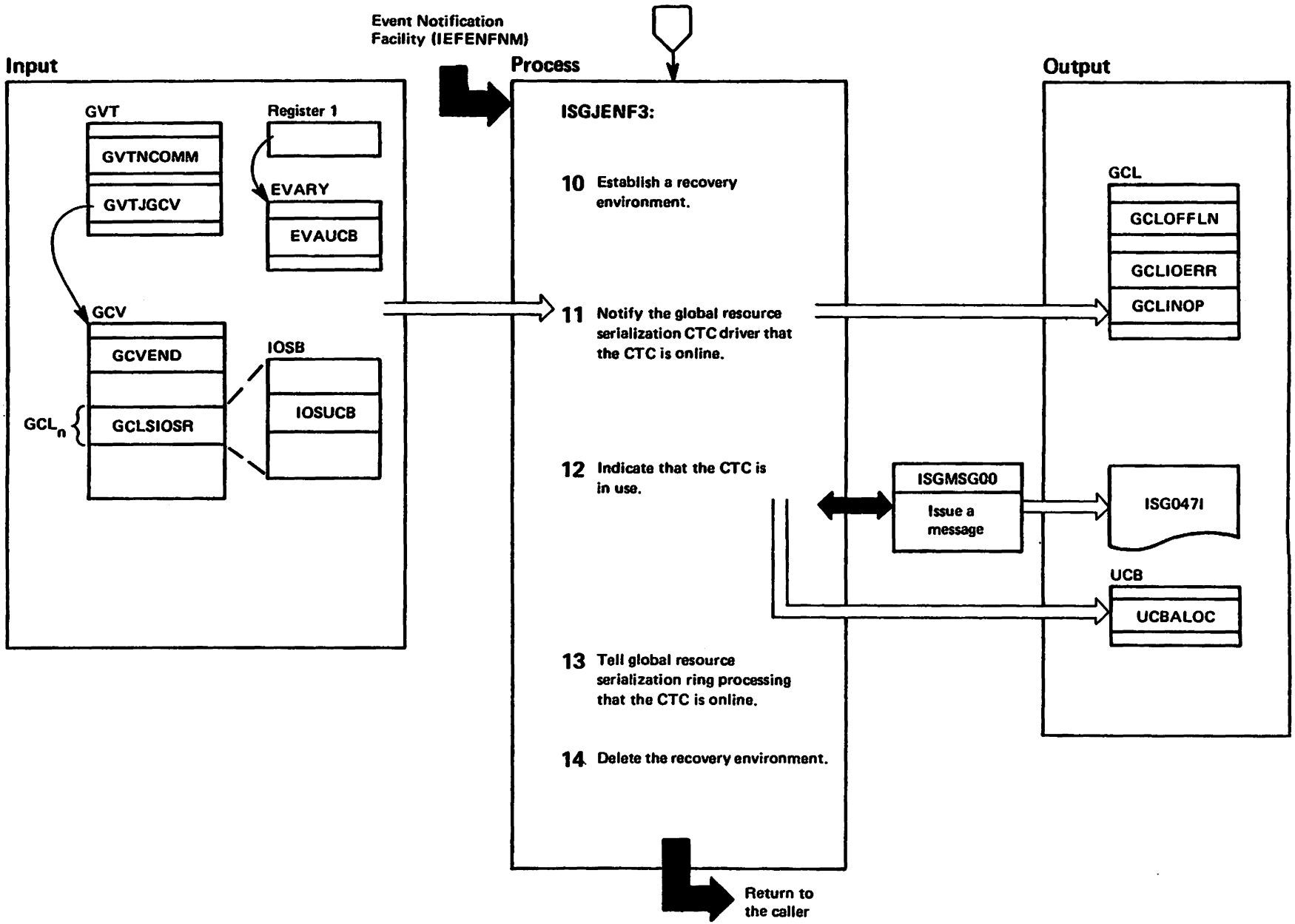
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-39. ISGJENF0 – Global Resource Serialization Event Notification Exits (Part 6 of 8)

Extended Description	Module	Label
ISGJENF2 This ENF exit notifies the operator and the global resource serialization CTC driver that a CTC defined to global resource serialization has been varied offline.		
7 Establish the ISGJENF0 entry point ISGJENFR as an ESTAE recovery routine. If the ESTAE is not established successfully, issue an ABEND macro instruction with a system completion code of X'09A' and a reason code identifying the nature of the error. If the ESTAE was established successfully, continue processing.		ISGJENF2
8 Return to the caller if ring processing is inactive (GVTNCOMM=1) or if the target of the VARY CTC, OFFLINE command is not a global resource serialization CTC (EVAUCB≠IOSUCB); otherwise, notify the global resource serialization CTC driver that the global resource serialization CTC is offline. Then invoke ISGMSG00 to issue message ISG047I.		
9 Issue the ESTAE macro instruction to delete the ESTAE recovery routine.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-39. ISGJENF0 – Global Resource Serialization Event Notification Exits (Part 7 of 8)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

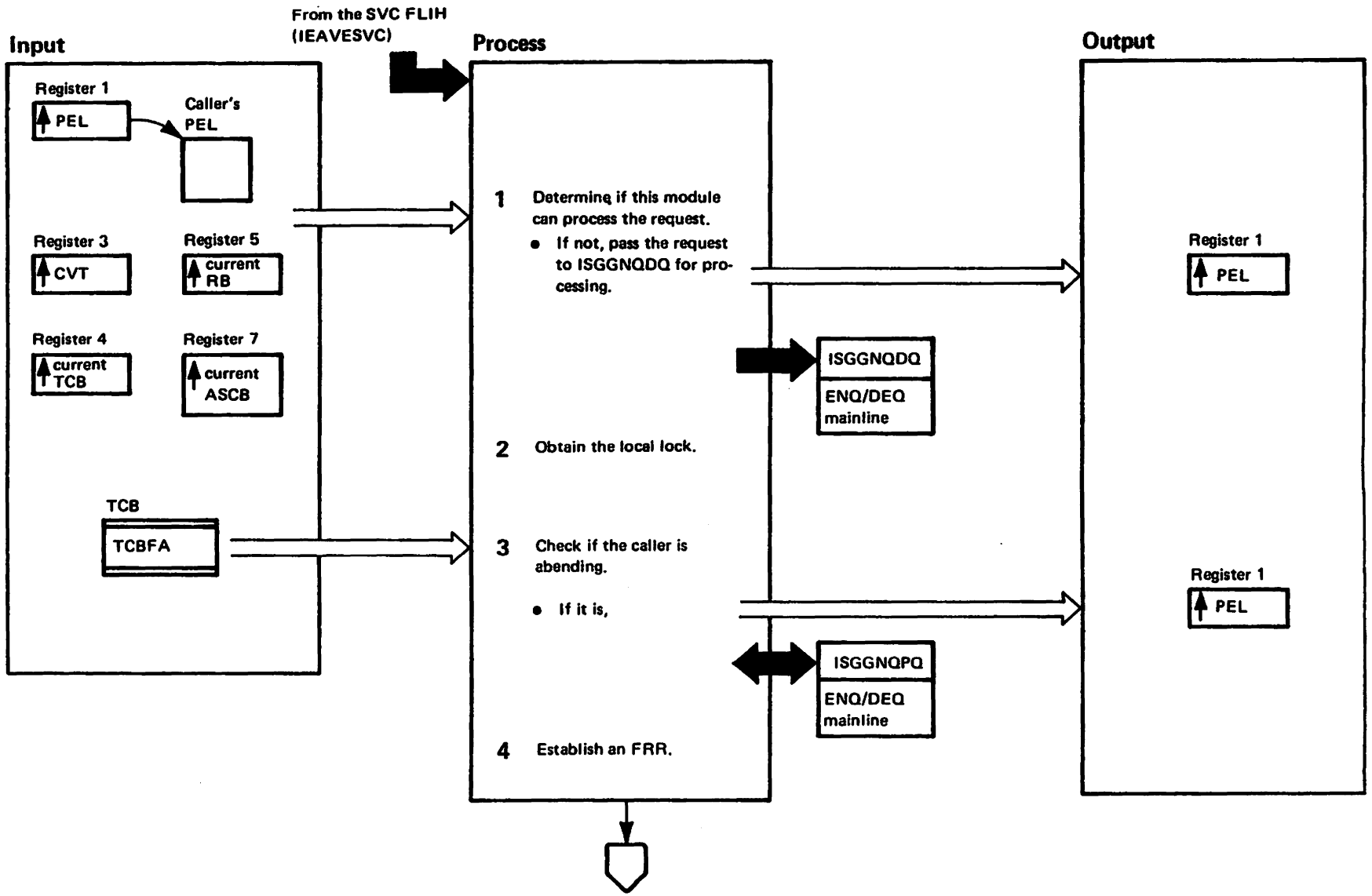
Diagram GRS-39. ISGJENF0 – Global Resource Serialization Event Notification Exits (Part 8 of 8)

Extended Description	Module	Label
ISGJENF3		
This ENF exit notifies the operator, the global resource serialization CTC driver, and the global resource serialization ring processor that a CTC defined to global resource serialization has been varied online.		
10 Establish the ISGJENF0 entry point ISGJENFR as an ESTAE recovery routine. If the ESTAE is not established successfully, issue an ABEND macro instruction with a system completion code of X'09A' and a reason code identifying the nature of the error. If the ESTAE was established successfully, continue processing.		ISGJENF3
11 Return to the caller if ring processing is inactive (GVTNCOMM=1) or if the target of the VARY CTC, ONLINE command is not a global resource serialization CTC (EVAUCB≠IOSUCB); otherwise, notify the global resource serialization CTC driver that the CTC is online by setting GCLOFFLN=0 and resetting the hardware and software error flags (GCLIOERR=0 and GCLINOP=0) to indicate to the global resource serialization CTC driver that there are no outstanding hardware or software errors on this CTC.		
12 Indicate that the CTC is being used by global resource serialization by setting the UCB allocated bit (UCBALOC=1). Invoke ISGMSG00 to issue message ISG047I.		
13 Schedule an unusual event as an SRB. The SRB notifies global resource serialization ring processing that the global resource serialization CTC has been varied online. Global resource serialization ring processing updates its control blocks that describe this CTC to indicate this online condition.		
14 Issue the ESTAE macro instruction to delete the ESTAE recovery routine.		

Extended Description	Module	Label
Recovery Processing		
The ISGJENFR entry point in ISGJENF0 is established as an ESTAE recovery routine by ISGJENF1, ISGJENF2, and ISGJENF3. On entry to ISGJENFR, check to see if an SDWA was supplied. If it was, record the error in SYS1.LOGREC and issue the SDUMP macro instruction to request a SVC dump. Then issue message ISG021I to notify the operator that an error occurred in global resource serialization event processing. Finally, set up to retry event processing at ISGJRTRY in ISGJENF0. If an SDWA is not available, no recording of the error to SYS1.LOGREC or dumping of storage (via SVC dump) takes place. Retry is still attempted at entry point ISGJRTRY in ISGJENF0 and error message ISG021I is still issued.		
If entry at ISGJENFR is because of a recursive error – that is, an error occurred after rhe retry routine (entry point ISGJRTRY) was executed – percolate the error to the next level of recovery.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-40. ISGLNQDQ – ENQ/DEQ Fast Path Routine (Part 1 of 16)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

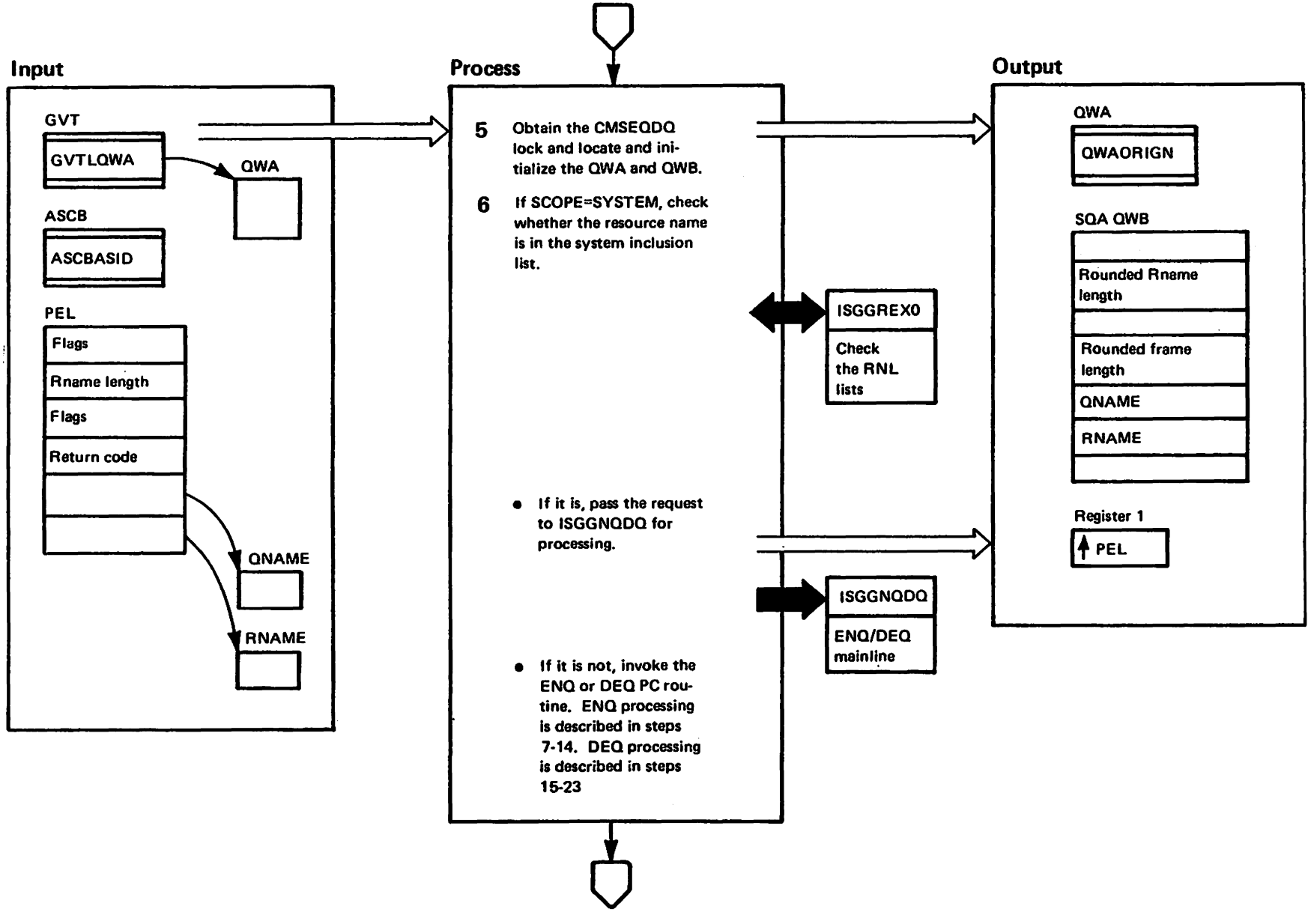
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-40. ISGLNQDQ – ENQ/DEQ Fast Path Routine (Part 2 of 16)

Extended Description	Module	Label
ISGLNQDQ provides a fast path for processing ENQ (SVC 56) and DEQ (SVC 48) requests that meet certain criteria. When an ENQ or DEQ SVC is issued, IEAVESVC gives ISGLNQDQ control at entry point IGC056FP or IGC048FP, respectively. If possible, ISGLNQDQ handles the request and returns control to the caller via EXIT prolog. If ISGLNQDQ cannot handle the request, it calls the ENQ/DEQ mainline routine (ISGGNQDQ) to process the request.		
ISGLNQDQ begins processing in the caller's address space. It performs initialization functions and copies the caller's parameter element list (PEL) into a queue work block (QWB) in common storage so that the global resource serialization address space can access it. ISGLNQDQ then issues a PC instruction to either the ENQ or DEQ PC routine (entry point ISGLNQ00 or ISGLDQ00 within ISGLNQDQ, respectively) and continues executing in the global resource serialization address space. This is where ISGLNQDQ performs the ENQ or DEQ processing. After the request is processed, ISGLNQDQ issues a PT instruction to transfer control to the caller's primary address space where it cleans up and exits.		
1 ISGLNQDQ checks if the request can be handled in the fast path. If not, ISGLNQDQ places the PEL address in register 1 and passes the request to ISGGNQDQ for processing. Only a request of the following type passes this test: • Caller in supervisor state • Single request (not a list) • Scope of STEP or SYSTEM (not SYSTEMS) • RET=NONE or RET=HAVE • Exclusive or shared • RMC=NONE or SMC=NONE • GENERIC=NO • TCB not specified • UCB not specified • ECB not specified	ISGGNQDQ	
2 ISGLNQDQ obtains the local lock of the caller's address space.		
3 ISGLNQDQ checks the TCB fail bit (TCBFA) to determine if the task is abending. If so, ISGLNQDQ places the PEL address in register 1 and passes the request to ISGGNQDQ for processing. This is done because the fast path does not contain the logic to check for possible interlocks in an abending task's family tree.	ISGGNQDQ	
4 ISGLNQDQ issues a SETFRR macro to establish ENQFRR or DEQFRR as its recovery routine. ISGLNQDQ saves recovery-related data and footprints in the FRR parameter area that the macro returns.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-40. ISGLNQDQ – ENQ/DEQ Fast Path Routine (Part 3 of 16)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

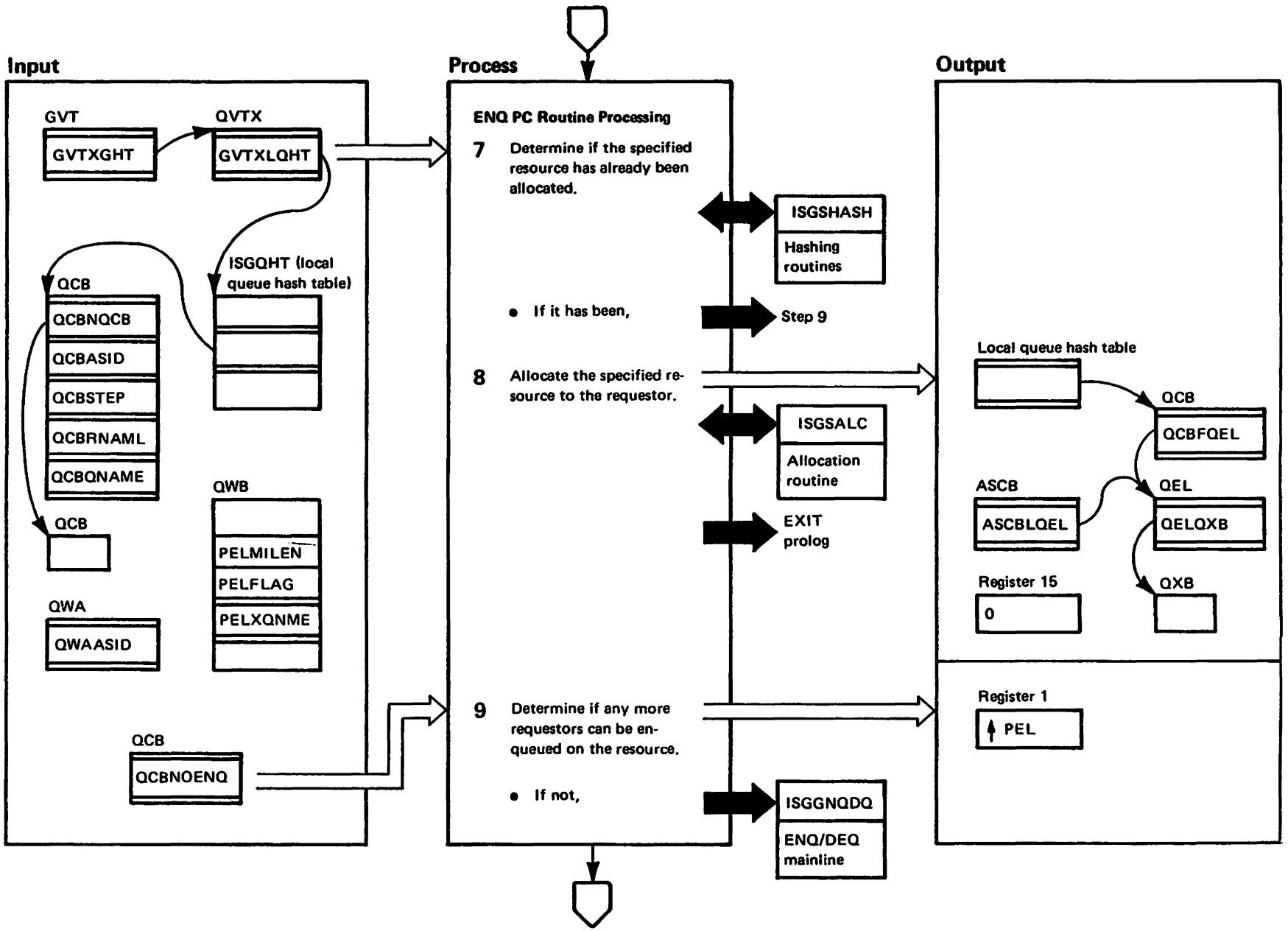
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-40. ISGLNQDQ – ENQ/DEQ Fast Path Routine (Part 4 of 16)

Extended Description	Module	Label
5 ISGLNQDQ obtains the CMSEQDQ lock to serialize the queue workareas (QWA and QWB) and the global resource serialization control blocks. ISGLNQDQ locates the QWA from the global resource serialization vector table (GVT). ISGLNQDQ stores the requestor's ASID (ASCBASID) in the QWAORIGN field. ISGLNQDQ moves the ENQ or DEQ request into the QWB. This is done so the information will be accessible from the global resource serialization address space (the QWB is in common storage).		
6 If the request specified SCOPE=SYSTEM and global resource serialization is active, ISGLNQDQ calls the global resource serialization resource exit routine (ISGGREX0) at entry point ISGGSIEX to determine if the resource name is in the system inclusion list. If it is, the request is treated as a global request and ISGLNQDQ passes the request to ISGGNQDQ for processing. Before branching to ISGGNQDQ, ISGLNQDQ (at label REJENQ1) releases the CMSEQDQ lock, deletes the FRR and places the PEL address in register 1. Note that it enters ISGGNQDQ holding the local lock.	ISGGREX0 ISGGNQDQ	ISGGSIEX
If the resource name for a DEQ request is not in the system inclusion list, ISGLNQDQ issues a PC instruction to ISGLDQ00, an entry point in ISGLNQDQ. If the resource name for an ENQ request is not in the system inclusion list, ISGLNQDQ checks that the request does not exceed the concurrent request limit. If it does, then ISGLNQDQ passes the request to ISGGNQDQ for processing. If it does not, then ISGLNQDQ issues a PC instruction to ISGLNQ00, an entry point in ISGLNQDQ.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-40. ISGLNQDQ – ENQ/DEQ Fast Path Routine (Part 5 of 16)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

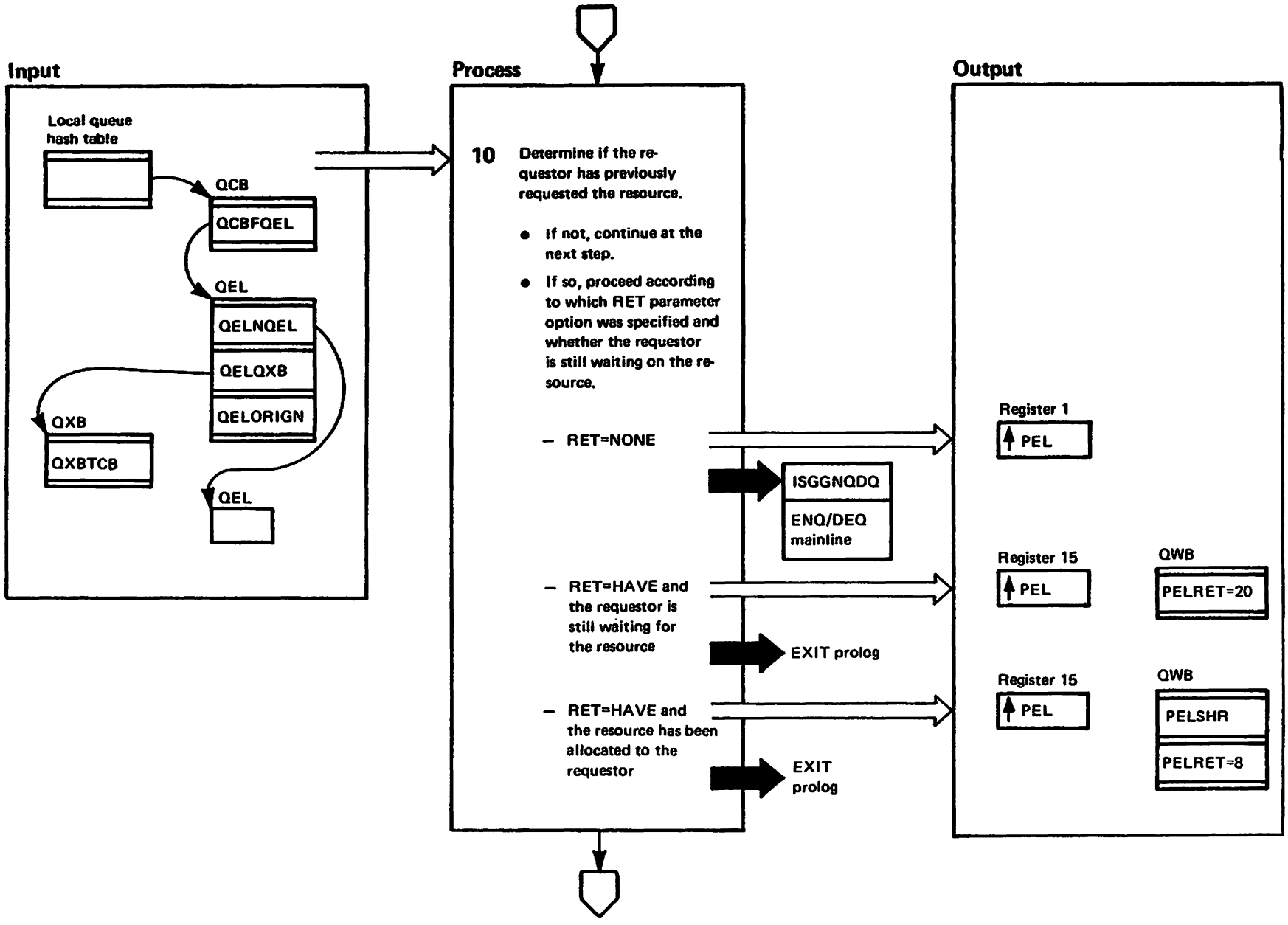
LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-40. ISGLNQDQ – ENQ/DEQ Fast Path Routine (Part 6 of 16)

Extended Description	Module	Label
ENQ Processing		
7 To determine if the reqeusted resource has already been allocated, ISGLNQDQ searches the appropriate local hash table synonym queue for a QCB having the same QNAME, RNAME length, RNAME, SCOPE, and ASID (if SCOPE=STEP) as specified in the parameter element list (PEL). (ISGLNQDQ calls the hash routine, ISGSHASH, to determine which local synonym queue to search.) If ISGLNQDQ finds a matching QCB, the resource has already been allocated. ISGLNQDQ continues at step 9 where it determines if the requestor can also be enqueued on the resource.	ISGSHASH	
8 If the resource is not already allocated (no QCB exists for it), ISGLNQDQ allocates it to the requesting task. To do so, ISGLNQDQ:		
• Calls ISGSALC to obtain storage for a QCB if there are no available cells in the currently allocated PEXBs, initializes the QCB with information about the resource being requested, and chains it to the appropriate hash table entry.	ISGSALC	
• Calls ISGSALC to obtain storage for a QEL and QXB, if there are no available cells in the currently allocated PEXBs. ISGLNQDQ initializes the QEL with information about the request type and the requestor. It puts the job name and pointers to the TCB and SVRB into the QXB. ISGLNQDQ then chains the QEL to the QCB and the QXB to the QEL. It also chains the QEL to the ASCB local QEL queue.		
• Issues a PT instruction to transfer control to label NQRET in ISGLNQDQ's mainline. There ISGLNQDQ releases the CMSEQDQ lock, sets a return code of zero, and branches to EXIT prolog. EXIT prolog releases the local lock, deletes the FRR, and returns to the caller.	ISGLNQDQ	NQRET
9 A recovery routine might have determined that no more requestors can be enqueued on the specified resource, in which case the recovery routine set the QCBNOENQ flag to one. If this has happened, ISGLNQDQ issues a PT instruction to transfer control to label NQRET1 in ISGLNQDQ's mainline. There ISGLNQDQ branches to label REJENQ1, places the PEL address in register 1, releases the CMSEQDQ lock, deletes the FRR, and branches to entry point ISC056 in ISGGNQDQ.	ISGGNQDQ	NQRET1 REJENQ1 ISC056

Diagram GRS-40. ISGLNQDQ -- ENQ/DEQ Fast Path Routine (Part 7 of 16)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

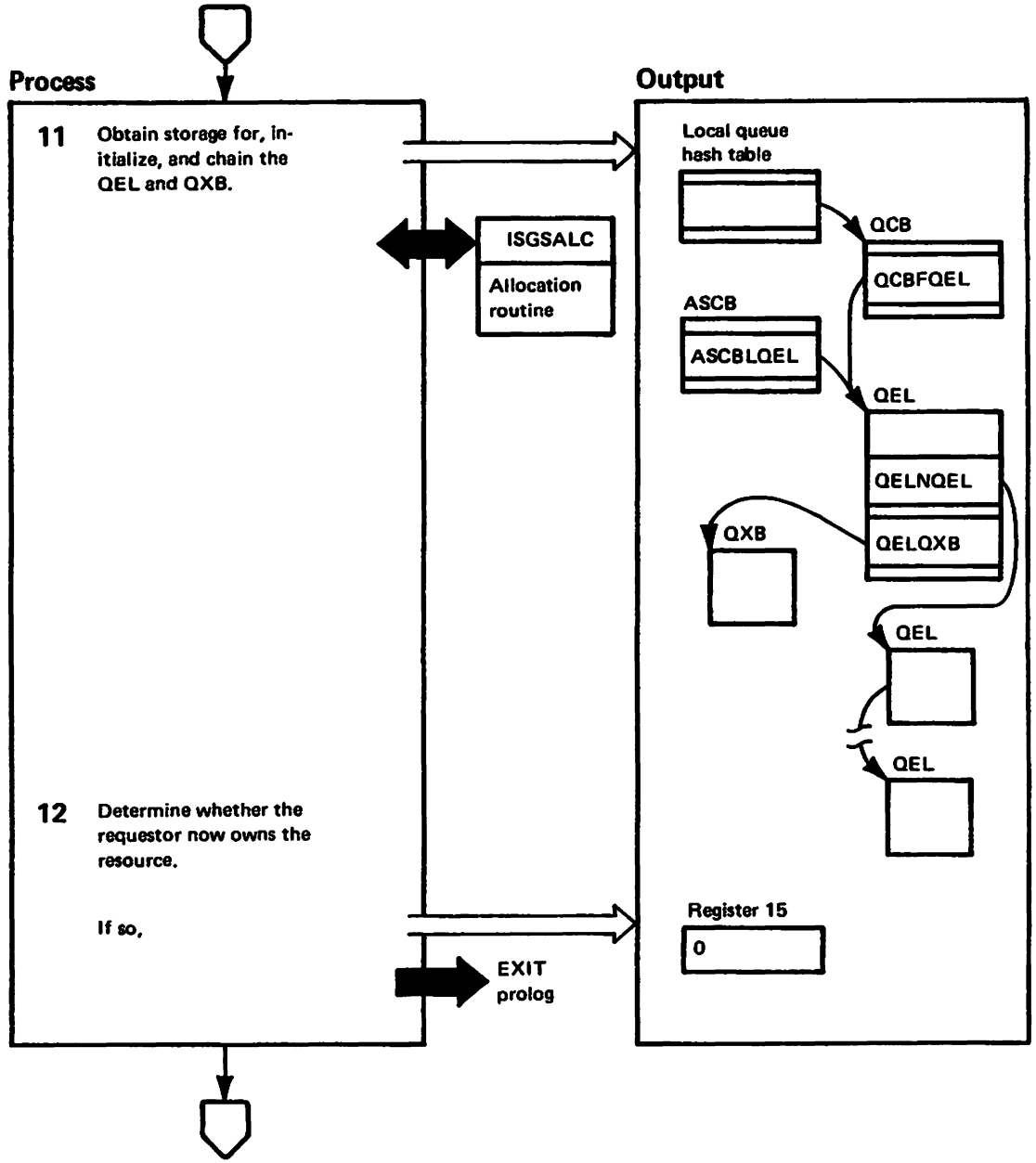
"Restricted Materials of IBM"
Licensed Materials – Property of IBM

Diagram GRS-40. ISGLNQDQ – ENQ/DEQ Fast Path Routine (Part 8 of 16)

Extended Description	Module	Label
10 ISGLNQDQ searches the QELs chained to the input QCB to determine if the requestor has previously requested the resource. If it has, ISGLNQDQ's actions depend on which RET parameter option is specified and whether the resource has been allocated or the task is still waiting for it. (If the requestor has not previously asked for the resource, see step 11.) If the requestor has previously asked for the resource and actually owns it, ISGLNQDQ issues a PT instruction to label NQRET2. If the requestor has previously asked for the resource and is still waiting for it, ISGLNQDQ issues a PT instruction to label NQRET3.		NQRET2 NQRET3
If RET=NONE at either label NQRET2 or NQRET3, ISGLNQDQ goes to label REJENQ1 where it loads the requestor's PEL address into register 1, restores the entry environment, releases the CMSEQDQ lock, deletes the FRR, and branches to ISGGNQDQ. ISGGNQDQ abends the requestor.	ISGGNQDQ	
If RET=HAVE and the task is still waiting (label NQRET3), ISGLNQDQ: • Places the requestor's PEL address into register 15 • Puts a return code of 20 into the PELRET field • Releases the CMSEQDQ lock • Branches to EXIT prolog, which releases the local lock, deletes the FRR, and returns to the caller.		
If RET=HAVE and the resource has been allocated to the requestor (label NQRET2), ISGLNQDQ: • Indicates whether the resource is allocated exclusively or shared by setting the PELSHR bit to 1 if the resource is shared or to zero if the resource is owned exclusively. • Puts the address of the requestor's PEL into register 15. • Puts a return code of 8 into the requestor's PELRET field. • Releases the CMSEQDQ lock. • Branches to EXIT prolog. EXIT prolog releases the local lock, deletes the FRR, and returns to the requestor.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-40. ISGLNQDQ – ENQ/DEQ Fast Path Routine (Part 9 of 16)



"Restricted Materials of IBM"
Licensed Materials – Property of IBM

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-40. ISGLNQDQ – ENQ/DEQ Fast Path Routine (Part 10 of 16)

Extended Description	Module	Label
11 ISGLNQDQ calls ISGSALC to obtain storage for a QEL and QXB if there are no available cells in the currently allocated PEXBs. ISGLNQDQ initializes the QEL with information about the request type and the requestor. It puts the job name and pointers to the TCB and SVRB into the QXB. ISGLNQDQ then chains the QEL to the QCB and the QXB to the QEL. It also chains the QEL to the ASCB local QEL queue.	ISGSALC	
12 If the resource can be shared and the task requested it shared, the task now owns the resource. (If the request specified exclusive and the task gains control of the resource, the task was the first requestor for the resource. This case is handled at step 8). ISGLNQDQ sets the return code of zero in register 15, releases the CMSEQDQ lock, and branches to EXIT prolog. EXIT prolog releases the local lock, deletes the FRR, and returns to the caller.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

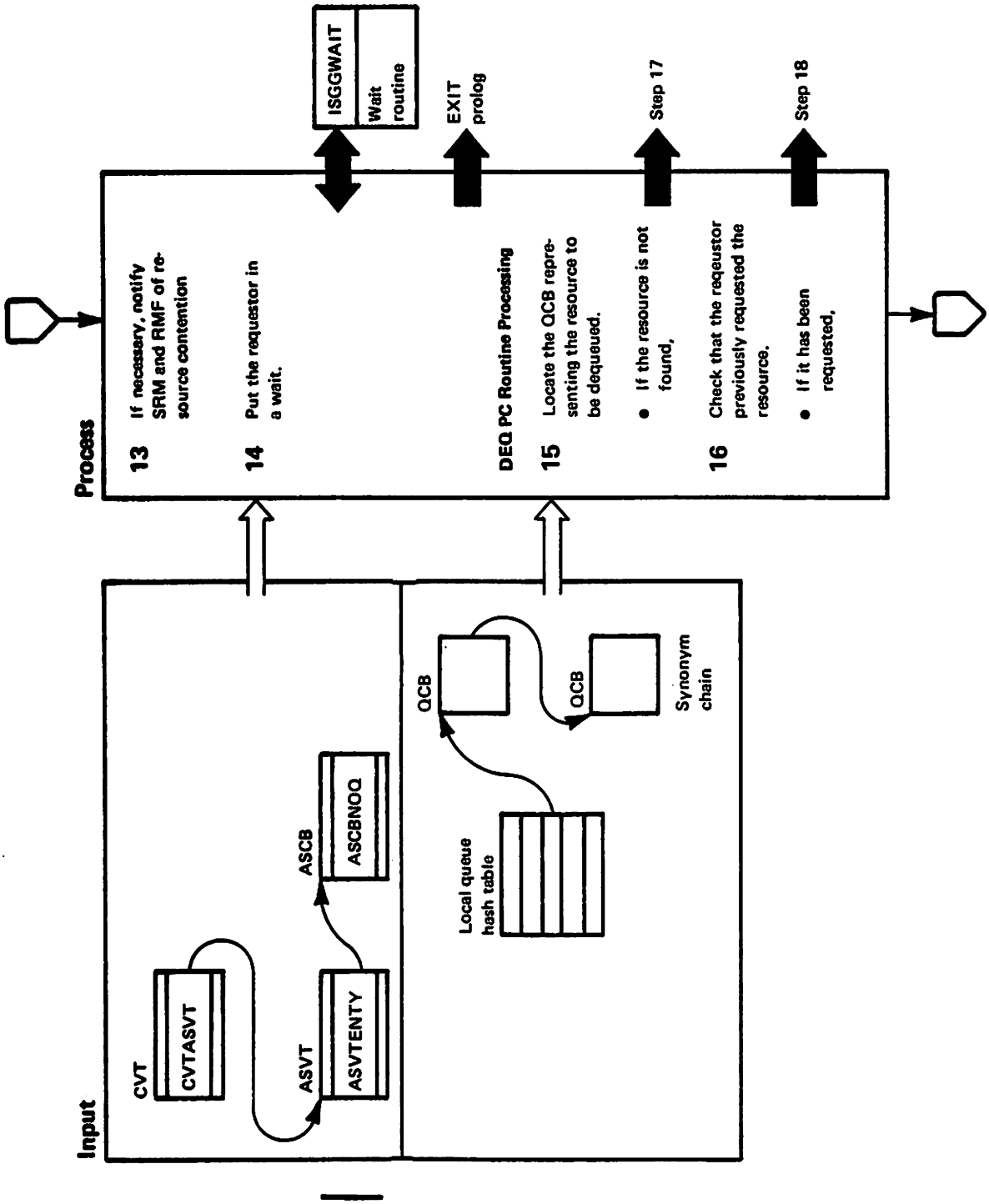


Diagram GRS-40. ISGLNQDQ - ENQ/DEQ Fast Path Routine (Part 11 of 16)

LY28-1695-0 (c) Copyright IBM Corp. 1987

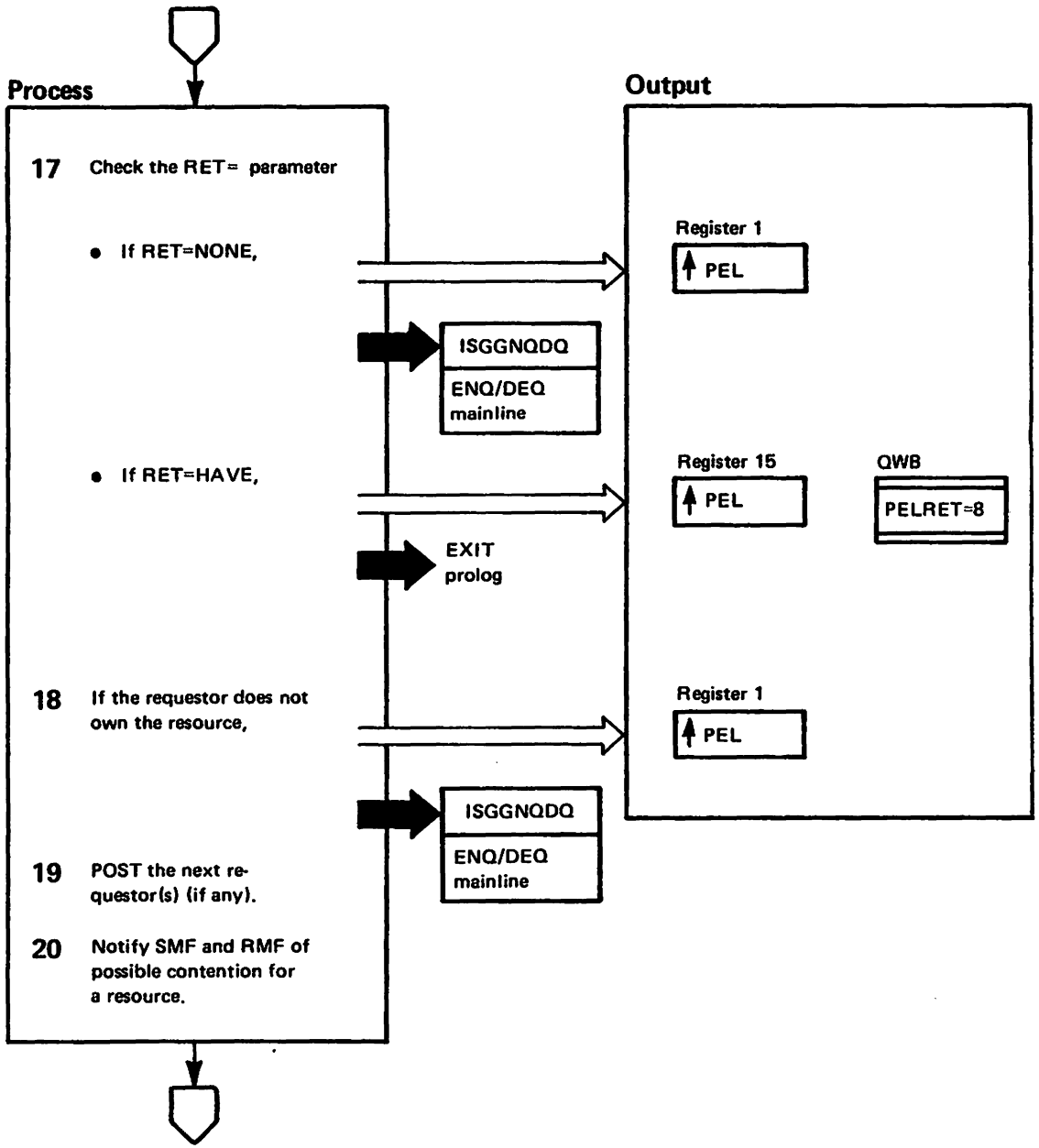
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-40. ISGLNQDQ – ENQ/DEQ Fast Path Routine (Part 12 of 16)

Extended Description	Module	Label
13 If the task cannot gain control of the resource and this is the first requestor not able to gain control of the resource with SCOPE=SYSTEM, ISGLNQDQ issues SYSEVENT ENQHOLD to notify SRM that the requesting task's processing is delayed. If RMF is active, ISGLNQDQ also calls RMF to inform it of resource contention.		
14 If any previous QELs on the QEL queue are associated with swapped out address spaces, ISGLNQDQ indicates that the requestor is to be put into a long wait (register 0=1). Otherwise, the requestor is put into a short wait. ISGLNQDQ then:		
• Releases the CMSEQDQ lock • Calls ISGGWAIT to put the current RB into a wait • Branches to EXIT prolog	ISGGWAIT	
15 ISGLNQDQ searches the hash table and synonym chain to find the QCB representing the resource to be dequeued. If the QCB is not found (the requestor had not previously enqueued on the resource), ISGLNQDQ issues a PT instruction to label DQRET2. Step 17 describes what ISGLNQDQ does after the PT instruction. If a matching QCB is found, see step 16.	ISGLNQDQ	DQRET2
16 If a matching QCB is found, ISGLNQDQ searches the QELs chained to the matching QCB to determine if the requestor had previously enqueued on the resource. If a QEL is not found, ISGLNQDQ issues a PT instruction to label DQRET2. Step 17 describes what ISGLNQDQ does after the PT instruction. If a matching QEL is found, ISGLNQDQ continues at step 18.		DQRET2

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-40. ISGLNQDQ – ENQ/DEQ Fast Path Routine (Part 13 of 16)



"Restricted Materials of IBM"
Licensed Materials - Property of IBM

LY28-1695-0 (c) Copyright IBM Corp. 1987

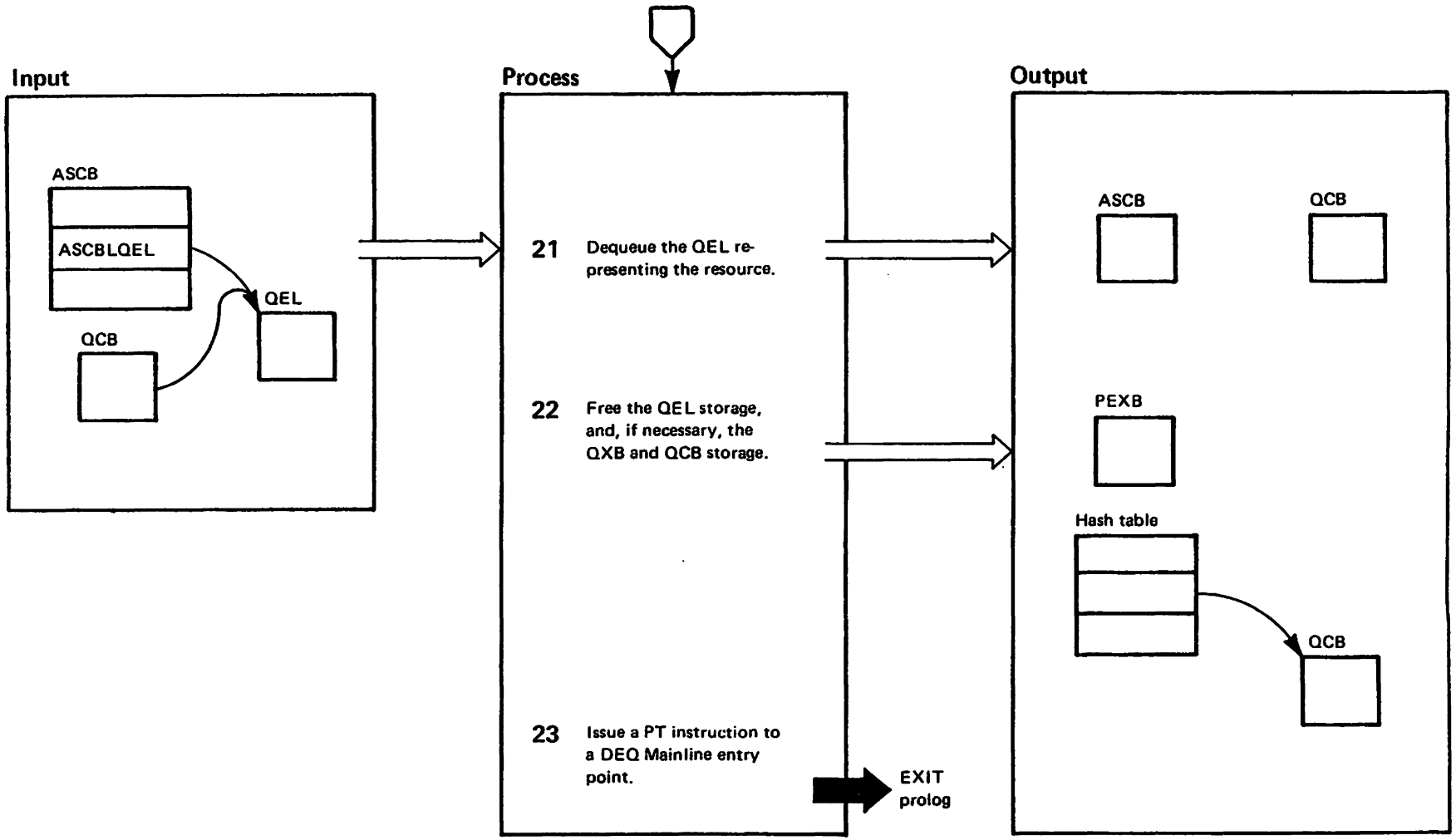
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-40. ISGLNQDQ – ENQ/DEQ Fast Path Routine (Part 14 of 16)

Extended Description	Module	Label
17 DQRET2 checks the RET parameter. If RET=NONE was specified, the request needs to be passed to ISGGNQDQ for an abend. DQRET2 resets the caller's registers, places the PEL address in register 1, releases the CMSEQDQ lock, deletes the FRR, and branches to ISGGNQDQ.	ISGGNQDQ	
If RET=HAVE was specified, DQRET2 sets a return code of 8 in the caller's PEL and places the address of the caller's PEL in register 15 to indicate that no DEQ was performed. DQRET2 then releases the CMSEQDQ lock and branches to EXIT prolog.		
18 ISGLNQDQ checks that the DEQ requestor currently owns the resource. If the requestor does not own the resource, ISGLNQDQ cannot dequeue it. ISGLNQDQ issues a PT instruction to label DQRET4 which branches to REJDEQ1. There ISGLNQDQ places the PEL address in register 1, releases the CMSEQDQ lock, deletes the FRR, and passes control to ISGGNQDQ.	ISGLNQDQ ISGGNQDQ	DQRET4 REJDEQ1
19 The resource owner is releasing the resource so if any programs are waiting for the resource, ISGLNQDQ issues a POST to them and they become the new owners. If the next requestor has requested exclusive ownership, ISGLNQDQ only issues a POST to that requestor. If the next requestor has requested shared ownership, ISGLNQDQ issues a POST to all the requestors up to but not including the first exclusive requestor.		
20 ISGLNQDQ notifies SRM to issue a SYSEVENT ENQRLSE if there is someone waiting for the resource being released. If the new owner has others still waiting for the resource, a SYSEVENT ENQHOLD is issued for the new owner. If RMF is active, ISGLNQDQ also calls RMF to inform it of a change in resource contention.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-40. ISGLNQDQ – ENQ/DEQ Fast Path Routine (Part 15 of 16)



"Restricted Materials of IBM"
Licensed Materials – Property of IBM

LY28-1695-0 (c) Copyright IBM Corp. 1987

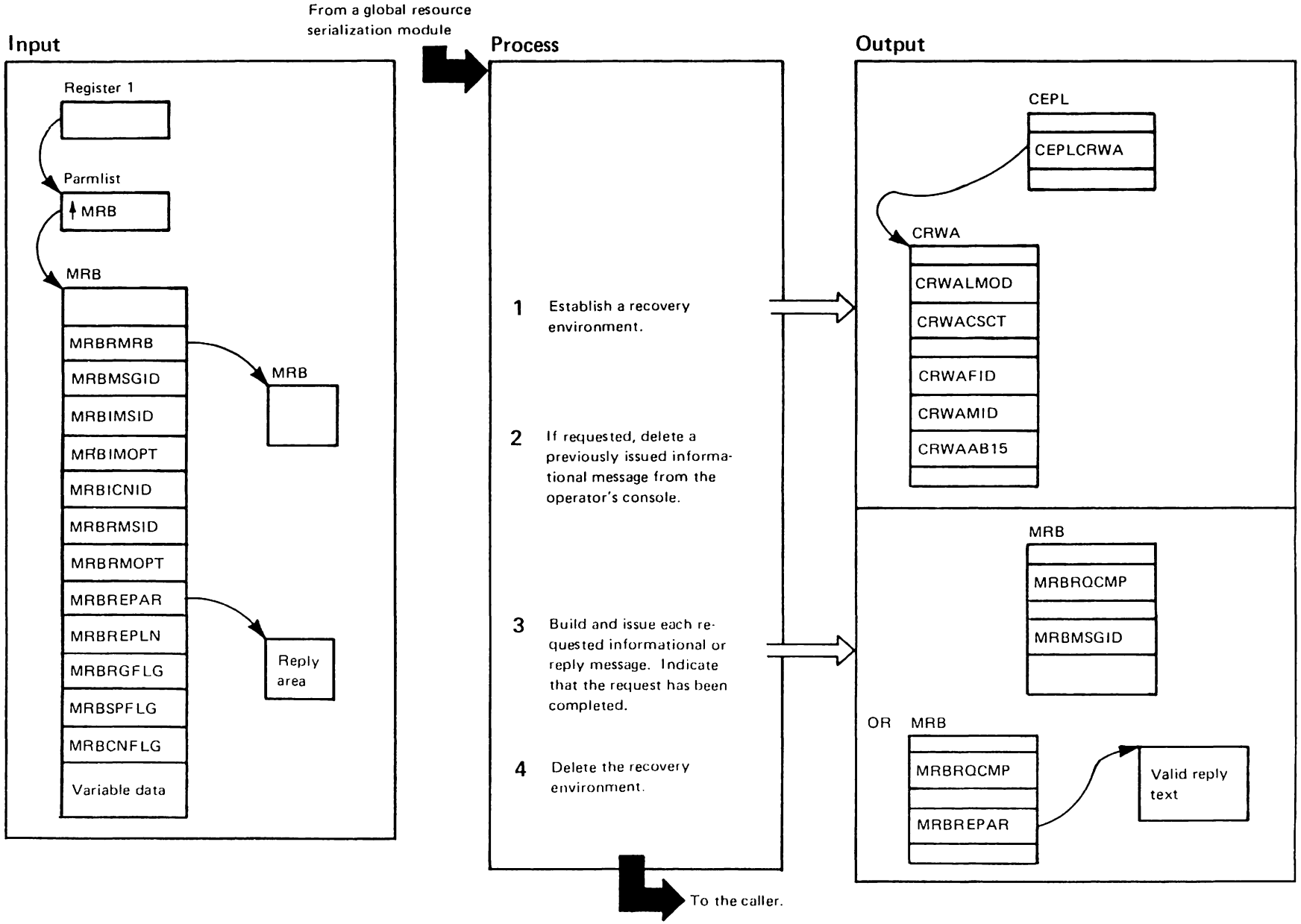
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-40. ISGLNQDQ – ENQ/DEQ Fast Path Routine (Part 16 of 16)

Extended Description	Module	Label
21 ISGLNQDQ unchains the QEL from the QCB and the ASCB local queue.		
22 ISGLNQDQ releases the storage occupied by the QEL. This is done by direct manipulation of the storage manager's control block, except in the case when all cells of a PEXB would then be free. In that case, ISGLNQDQ invokes the storage deallocation routine (ISGSDAL) to free the storage.	ISGSDAL	
If the wait count in the QXB is zero, ISGLNQDQ frees the QXB's storage (either by direct manipulation or via ISGSDAL).	ISGSDAL	
If the QCB has no more QELs chained from it, ISGLNQDQ unchains the QCB from the hash table and frees the QCB's storage (either directly or via ISGSDAL).	ISGSDAL	
23 ISGLNQDQ issues a PT instruction to DQRET if an SPOST is not needed or to label DQRET1 if an SPOST is needed. This PT instruction returns ISGLNQDQ to the caller's address space.	ISGLNQDQ	DQRET DQRET1
DQRET sets a return code of zero in register 15 to indicate the DEQ was successful. It releases the CMSEQDQ lock and branches to EXIT prolog.		
DQRET1 releases the CMSEQDQ lock (this is necessary before issuing an SPOST). DQRET1 deletes the FRR because it will no longer be valid after the local lock is released, and releases the local lock (also necessary before issuing an SPOST). DQRET1 then issues the SPOST. Upon return from the SPOST, DQRET1 branches to EXIT prolog.		

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-41. ISGMSG00 – Global Resource Serialization Message Processor (Part 1 of 2)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

Diagram GRS-41. ISGMSG00 – Global Resource Serialization Message Processor (Part 2 of 2)

Extended Description	Module	Label

ISGMSG00 receives control from a global resource serialization module via BALR whenever it needs to communicate with the operator. ISGMSG00 also receives control from ISGCMDR via an ATTACH whenever ISGCMDR encounters a message request block (MRB) on the global resource serialization command work queue. Upon entry, register 1 contains the address of a parameter list containing the address of an MRB.

1 ISGMSG00 initializes a command recovery work area (CRWA) with recovery information and places the CRWA on the CRWA queue. ISGMSG00 establishes ISGCRCV as its recovery routine via an ESTAE macro. (ISGMSG00 loads ISGCRCV prior to issuing the ESTAE.)

2 If an MRB contains a message ID for a previously issued informational message (MRBMSGID≠0), ISGMSG00 deletes the associated message from the operator's console.

3 ISGMSG00 establishes a loop to process each message request (each message request is represented by an MRB). The processing depends on if the MRB contains an informational message ID or a reply message ID.

If an informational message ID is provided in the MRB (MRBIMSID≠0), ISGMSG00 builds a WTO/MLWTO parameter list for the requested informational message. If the informational message is to be written to the operator, ISGMSG00 builds a WTO parameter list for a single line message and a MLWTO parameter list for a multi-line information message. If the informational message is to be written to the system log, ISGMSG00 builds a WTL parameter list for the message. The actual text of the line(s) in an informational message depends on the message option provided in the MRB (MRBIMOPT) and any variable data provided in the MRB. Once ISGMSG00 builds the WTO/MLWTO/WTL parameter list, ISGMSG00 issues the informational message to the appropriate operator console or to the system log.

Extended Description	Module	Label

3 (continued)

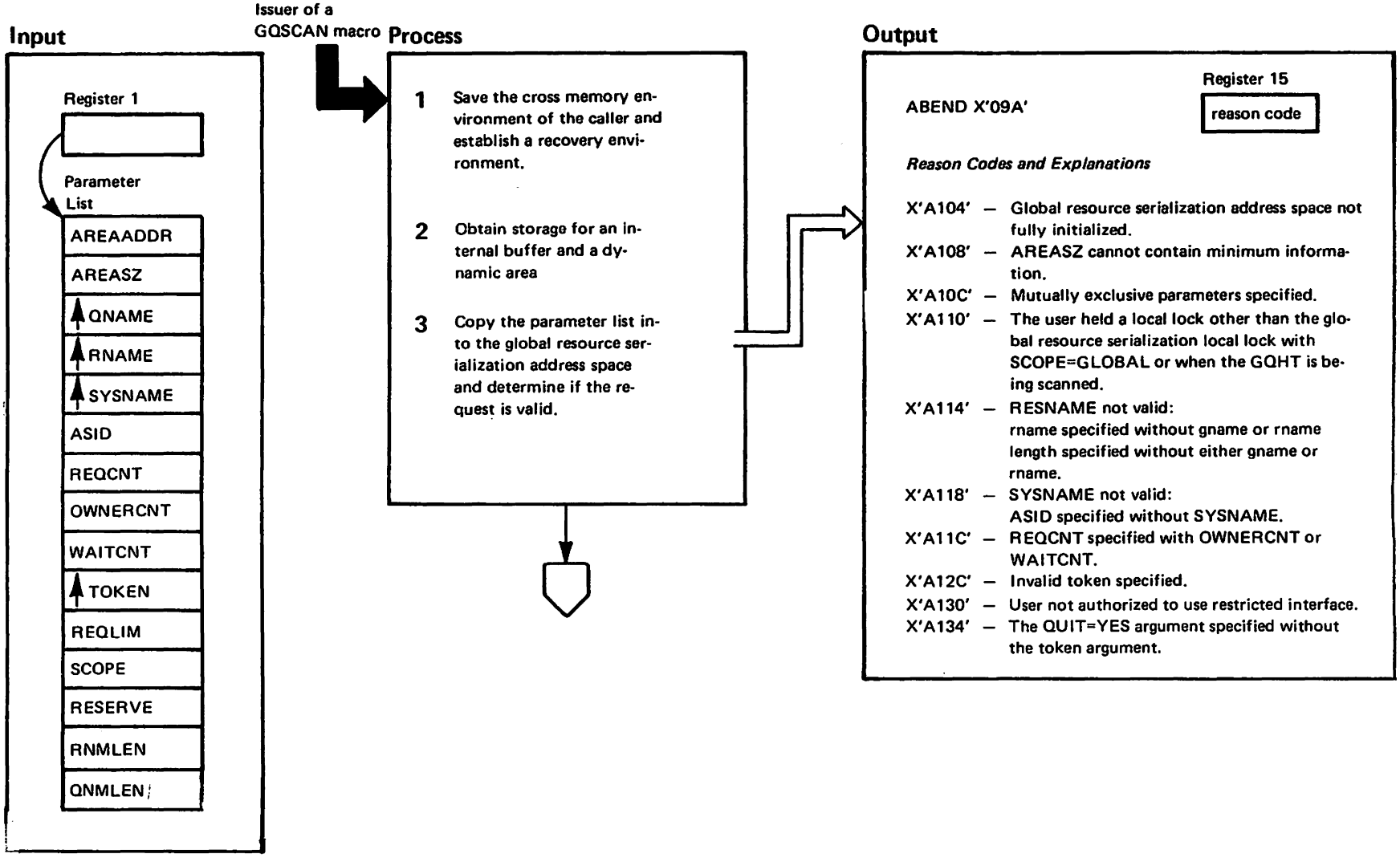
If a reply message ID is provided in the MRB (MRBRMSID≠0), ISGMSG00 builds a WTOR parameter list for the reply message. The actual text of the reply message depends on the message option provided in the MRB (MRBRMOPT) and any variable data provided in the MRB. Once ISGMSG00 builds the WTOR parameter list, it issues the reply message to the appropriate operator console. When the operator replies, ISGMSG00 validates the reply. If the reply is not appropriate for this particular message, ISGMSG00 reissues the reply message until a valid reply is received. Once a valid reply is received, ISGMSG00 places the reply in the reply area pointed to by MRBREPAR.

After processing the MRB, ISGMSG00 indicates that the message request has been processed by setting the MRBRQCMP bit to one. If MRBRMRB≠0 there are more messages to process and so ISGMSG00 repeats this step.

4 ISGMSG00 deletes the recovery routine (ISGCRCV via an ESTAE) and returns to the caller.

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-42. ISGQSCAN – Global Resource Serialization Queue Scanning Services (Part 1 of 6)



LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-42. ISGQSCAN – Global Resource Serialization Queue Scanning Services (Part 2 of 6)

Extended Description	Module	Label
Global resource serialization queue scanning service module (ISGQSCAN) receives control via a PC instruction from the issuer of the GQSCAN macro in key zero and supervisor state. ISGQSCAN scans the resource queues for information about resources specified on the GQSCAN macro and requestors of those resources. ISGQSCAN returns this information to the issuer of the GQSCAN macro. ISGQSCAN receives as input the parameter list shown in the input section of the diagram.		
1 ISGQSCAN saves the cross memory environment of the issuer of a GQSCAN macro via a PCLINK STACK macro. The PCLINK macro returns a stack element token (not the TOKEN pointed to in the parameter list) that uniquely identifies the stack element containing such information as the register contents and PSW key of the issuer of a GQSCAN macro. ISGQSCAN saves the stack element token in an unused register until the dynamic area storage is available. The functional recovery routine (FRR) for ISGQSCAN is ISGQSCNR. For more details see the Recovery Processing section at the end of this extended description.		
ISGQSCAN obtains the serialization required by ISGSALC, the global resource serialization storage allocation routine, to allocate the storage required by ISGQSCAN.		
2 ISGQSCAN calls ISGSALC to obtain storage for the internal buffer and its dynamic area from the pool extent blocks (PEXBs) in the resource queue area (RQA). ISGQSCAN uses the internal buffer to store the requested information until it can be copied into the area provided by the user. ISGQSCAN also releases the serialization, if any, obtained by ISGSALC in step 1.	ISGSALC	
3 ISGQSCAN copies the parameter list built by the GQSCAN macro from the user's address space to the global resource serialization address space. ISGQSCAN also copies data pointed to by fields of the parameter list into the global resource serialization address space. ISGQSCAN checks to see that the global resource serialization address space is active. If it is not active, ISGQSCAN issues a X'09A' ABEND with reason code X'A104'. ISGQSCAN also checks the parameter list to determine if the list specifies allowable combinations of parameters. (Combinations that are not allowed can be determined from the explanations of reason codes in the output section of the diagram.) ISGQSCAN also checks the following parameters to determine if their values are valid: REQLIM, REQCNT, OWNERCNT, AREASZ, and WAITCNT. If any parameters are invalid, ISGQSCAN terminates the requestor by issuing a X'09A' ABEND with one of the reason codes shown in the output section of the diagram.		SYNTXCHK

LY28-1695-0 (c) Copyright IBM Corp. 1987

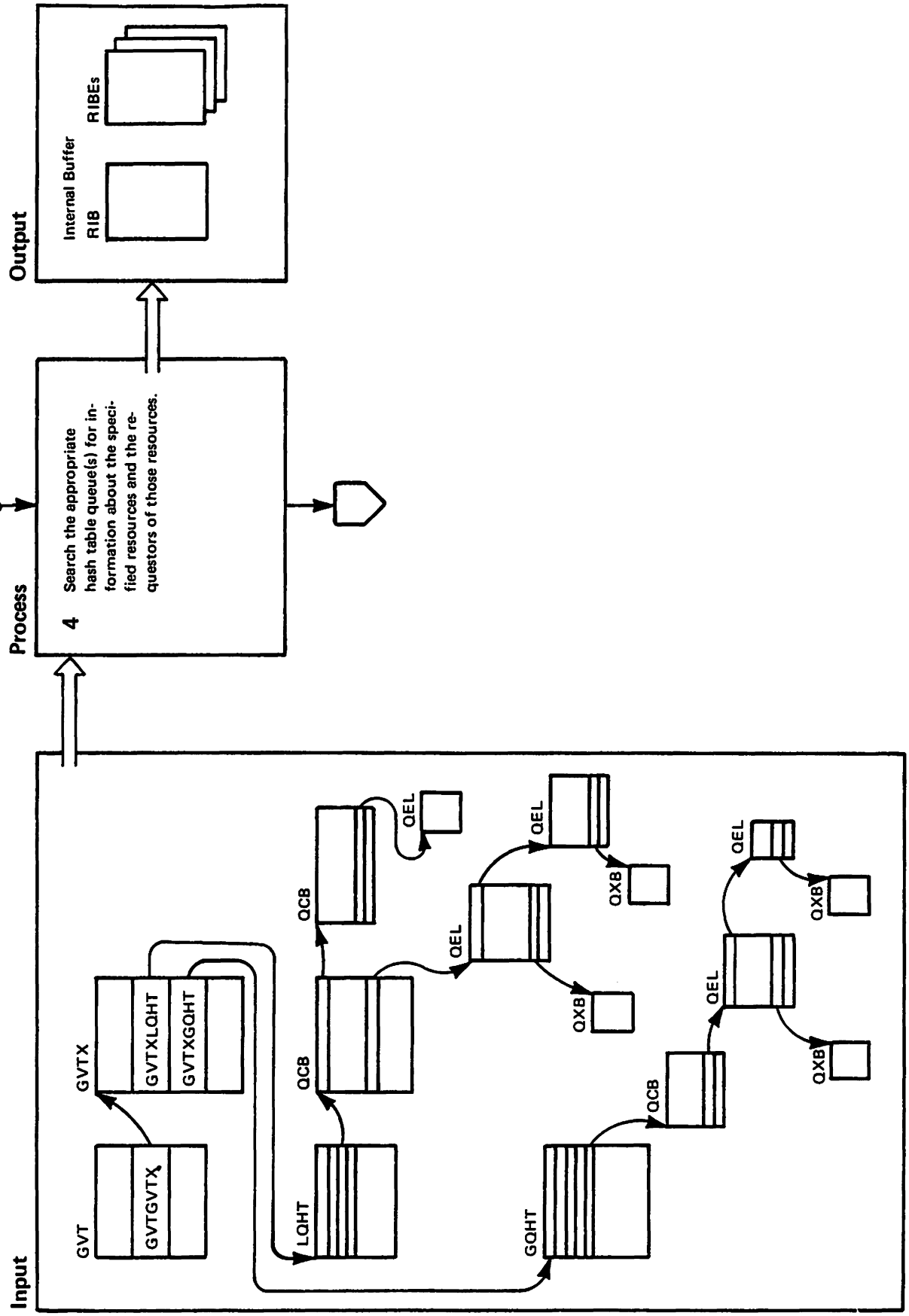


Diagram GRS-42. ISGQSCAN – Global Resource Serialization Queue Scanning Services (Part 3 of 6)

LY28-1695-0 (c) Copyright IBM Corp. 1987

LY28-1695-0 (c) Copyright IBM Corp. 1987

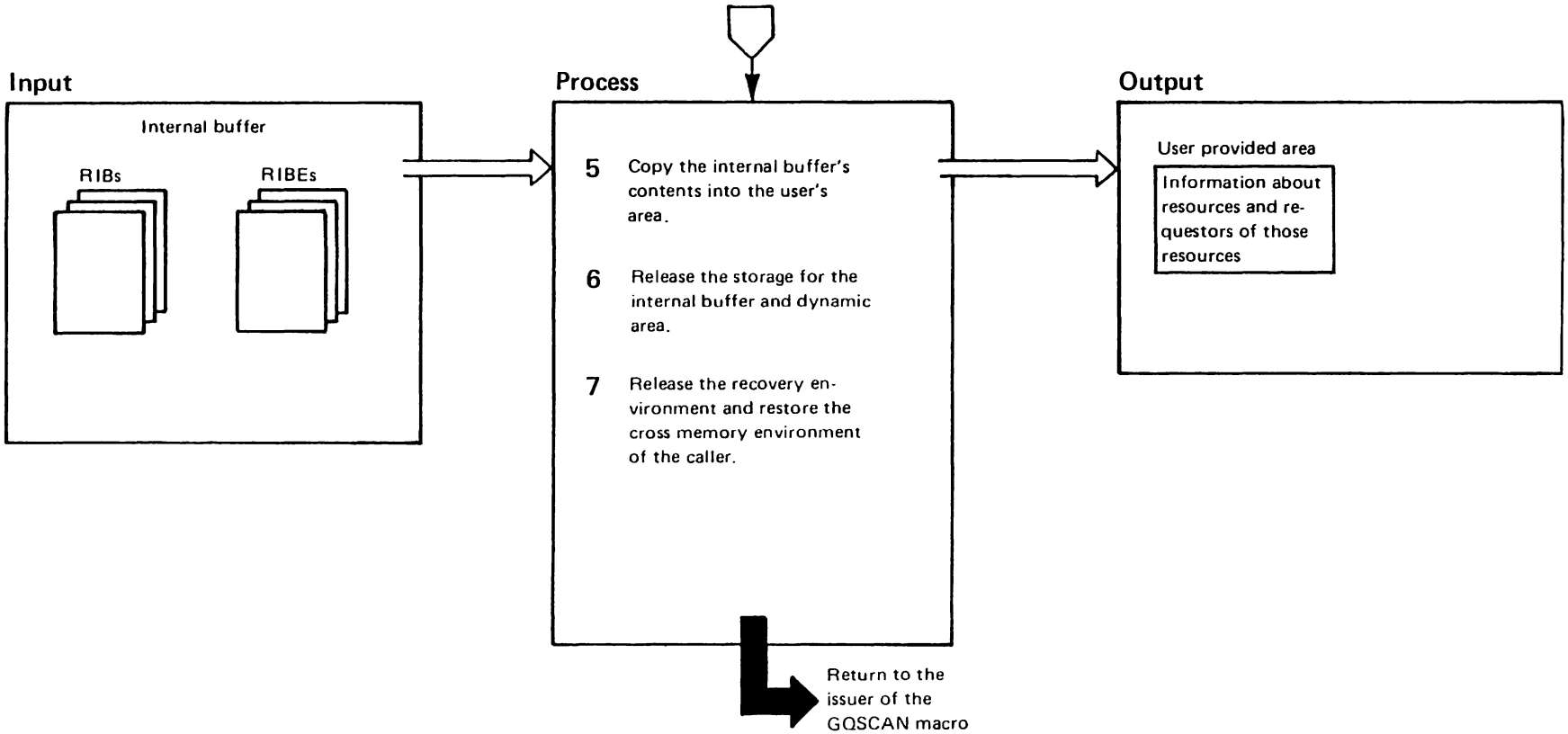
"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-42. ISGQSCAN – Global Resource Serialization Queue Scanning Services (Part 4 of 6)

Extended Description	Module	Label
4 ISGQSCAN searches either the local queue hash table (LQHT) or the global queue hash table (GQHT) for resources having the attributes specified in the parameter list. Resource attributes the caller can use to qualify the search are: Resource type – If the scope field specifies STEP, SYSTEM, SYSTEMS, LOCAL, or GLOBAL, ISGQSCAN searches for resources with that scope. Resource name – If both the QNAME and the RNAME are specified, ISGQSCAN calls the GLOBAL/LOCAL hashing routine (ISGSGLSH) to determine which entry in the queue hash table has the resource names that match the QNAME and RNAME combination. If a different valid combination of the QNAME and the RNAME is the input, ISGQSCAN searches all the queues for the resource names that match the QNAME and RNAME combination. Reserved/Unreserved – If RES=YES, ISGQSCAN looks for resources for which a RESERVE macro has been issued. If RES=NO, ISGQSCAN looks for resources for which a RESERVE macro instruction has not been issued. SYSNAME/ASID – If only the SYSNAME is specified, ISGQSCAN looks for resources requested by requestors from the specified system. If SYSNAME and ASID are both specified, ISGQSCAN looks for resources requested by requestors from the specified address space of the specified system. ISGQSCAN calls ISGBCI at entry point ISGBSRNI to convert the externally used SYSNAME to an internally used SYSID; ISGQSCAN calls ISGBCI at entry point ISGBSRIN to convert the internally used SYSID back to the externally used SYSNAME.	ISGBCI	ISGBSRNI ISGBSRIN
If no options are specified, ISGQSCAN returns information about all resources in either the local or global queue, depending on the resource type specified. ISGQSCAN can further limit the information returned by specifying the following parameters[1]: REQLIM (request limit) – when specified, limits the number of requestors of a given resource that information is returned on. REQCNT (request count) – when specified, information is returned only on those resources that have at least as many requestors as the REQCNT parameter specifies. OWNERCNT (owner count) – when specified, information is returned only on those resources that have at least as many owners as the OWNERCNT parameter specifies. WAITCNT (wait count) – when specified, information is returned only on those resources that have at least as many waiters (requestors waiting for shared or exclusive use) as the WAITCNT parameter specifies. ISGQSCAN also uses the SCOPE field to determine which queue(s) to search. Because STEP and SYSTEM requests are local requests, when either of these is specified, ISGQSCAN searches the LQHT. ISGQSCAN also searches the LQHT when SCOPE=LOCAL is specified. If SCOPE=SYSTEMS and global resource serialization is not active, the LQHT is scanned. If global resource serialization is active, the GQHT is scanned. When SCOPE=GLOBAL is specified, ISGQSCAN searches the GQHT. If SCOPE=ALL is specified, ISGQSCAN searches both the local and global queues for STEP, SYSTEM and/or SYSTEMS resources. If TOKEN=0, the request is new and ISGQSCAN starts searching at the beginning of the appropriate hash table. A nonzero TOKEN value indicates that the request is a continuation of a previous request. For example, ISGQSCAN had to interrupt the search because the user provided area is full. The nonzero TOKEN points to a placeholder queue control block (PQCB), which points to the next QCB to be searched. The PQCB is dequeued from the appropriate synoynm chain before the search continues. When ISGQSCAN finds a resource having all of the specified attributes, it places the information describing the resource into a resource information block (RIB) and it places the information describing the requestors of that resource into resource information block extents (RIBEs), one RIBE for each requestor. The internal buffer contains the RIB and RIBES.		

[1] REQCNT cannot be specified with OWNERCNT or WAITCNT.

Diagram GRS-42. ISGQSCAN – Global Resource Serialization Queue Scanning Services (Part 5 of 6)



"Restricted Materials of IBM"
Licensed Materials – Property of IBM

 (c) Copyright IBM Corp. 1987

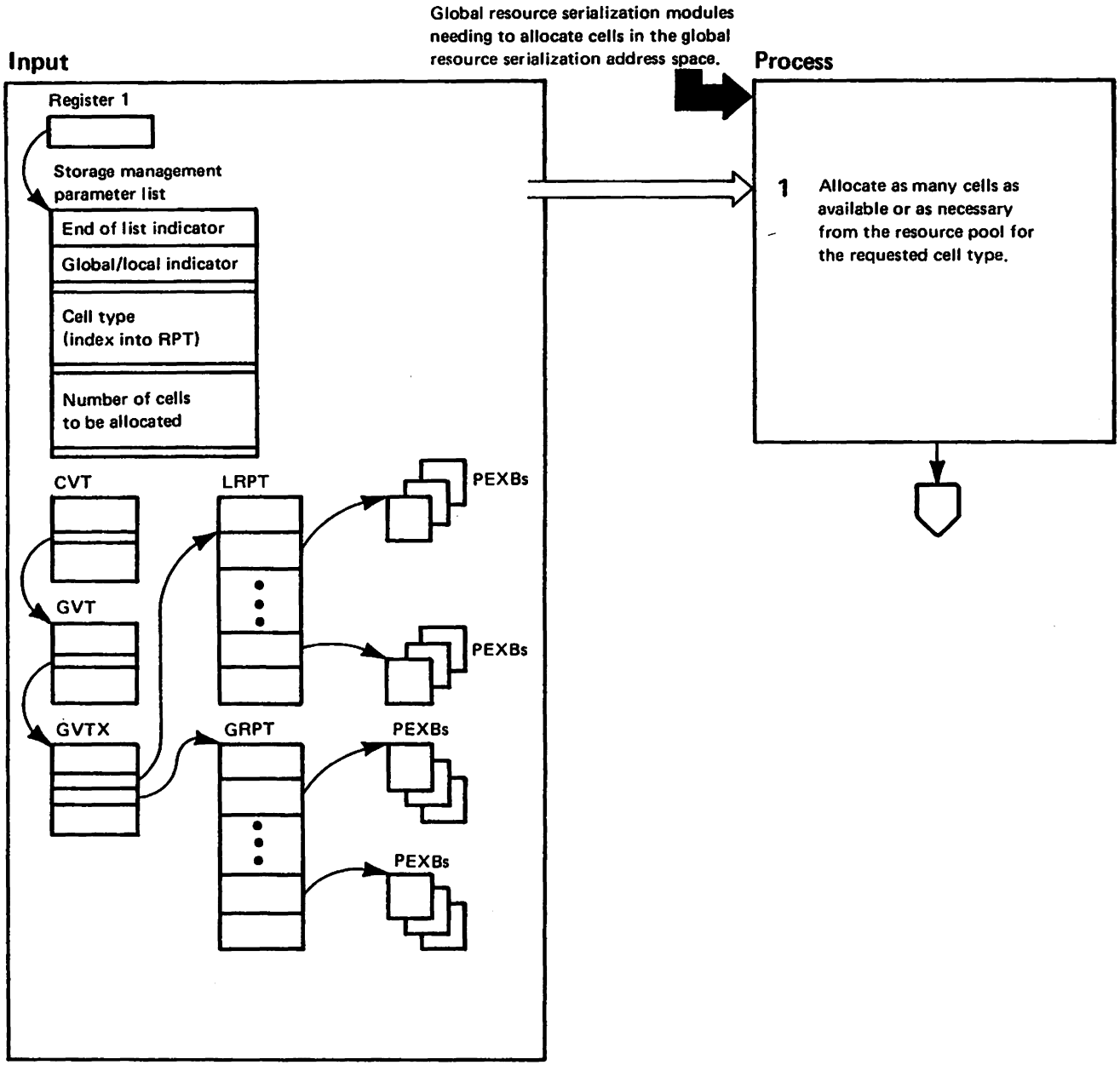
LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-42. ISGQSCAN – Global Resource Serialization Queue Scanning Services (Part 6 of 6)

Extended Description	Module	Label
5 When the internal buffer is filled to capacity or when the request has been satisfied, ISGQSCAN copies the contents of the internal buffer into the user provided area. If the user provided area is full, the request is not satisfied, the concurrent request limit has not been reached, and the user has provided an area in which the TOKEN value can be returned, ISGQSCAN obtains a PQCB from ISGSALC and places it on the appropriate QCB synonym chain. ISGQSCAN sets the TOKEN to point to where the request search will resume. Steps 5 and 6 are repeated until the user provided area is filled to capacity or the request has been satisfied.	ISGSALC	
6 ISGQSCAN calls ISGSDAL to return the storage for the dynamic area and the internal buffer to the PEXBs from which it was allocated. ISGSDAL requires the same serialization as ISGSALC in step 2. ISGQSCAN releases the serialization after the storage is returned.	ISGSDAL	
7 ISGQSCAN releases the recovery environment and issues a PCLINK UNSTACK macro to restore the cross memory environment that existed prior to the issuing of the GQSCAN macro. ISGQSCAN returns control to the caller via the PT instruction.		
Recovery Processing		
When an error occurs while ISGQSCAN is executing, RTM gives control to ISGQSCNR in key zero and supervisor state. ISGQSCNR:	ISGQSCAN	ISGQSCNR
• Converts a system completion code of X'0C4', associated with moving data to or from the user's address space, to a X'09A' ABEND with reason code X'A220', but does not record it.		
• Converts unexpected errors to a X'09A' ABEND with reason code X'A228', records them, and dumps them via SDUMP.		
• Obtains the serialization required to modify queues and to deallocate storage in the resource queue area. If ISGQSCAN or the invoker of ISGQSCAN did not hold any local lock at the time of the error, ISGQSCNR obtains the global resource serialization local lock. If ISGQSCAN or the invoker of ISGQSCAN did not hold the CMS ENQ/DEQ lock at the time of the error, ISGQSCNR also obtains the CMS ENQ/DEQ lock.		
• Frees the dynamic area and internal buffer.		
• Validates and frees any cells that ISGQSCAN obtained. ISGQSCNR calls the address verification routine (IEAVEADV) to validate each cell address. If the PQCB and the QCBs it is chained to pass validation, ISGQSCNR dequeues the PQCB from the appropriate QCB synonym chain. If a QEL and the QELs it is chained to pass validation, ISGQSCNR dequeues the QEL cell from the appropriate QEL ASCB queue.	ISGSDAL IEAVEADV	
• If the error occurred in a storage management routine, ISGQSCNR invokes ISGGFRR0 at entry point ISGGFRR1 to validate and repair the storage management control blocks.	ISGGFRR0	ISGGFRR1
• Releases the serialization: the user's local lock, the local lock of the global resource serialization address space, and/or CMS ENQ/DEQ lock.		
• Issues a PCLINK UNSTACK macro to restore the cross memory environment to what it was when the GTQSCAN macro was issued.		
• Percolates the error.		

Diagram GRS-43. ISGSALC – Global Resource Serialization Storage Management Allocation Routine (Part 1 of 6)



"Restricted Materials of IBM"
Licensed Materials - Property of IBM

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials - Property of IBM

Diagram GRS-43. ISGSALC – Global Resource Serialization Storage Management Allocation Routine (Part 2 of 6)

Extended Description

Module **Label**

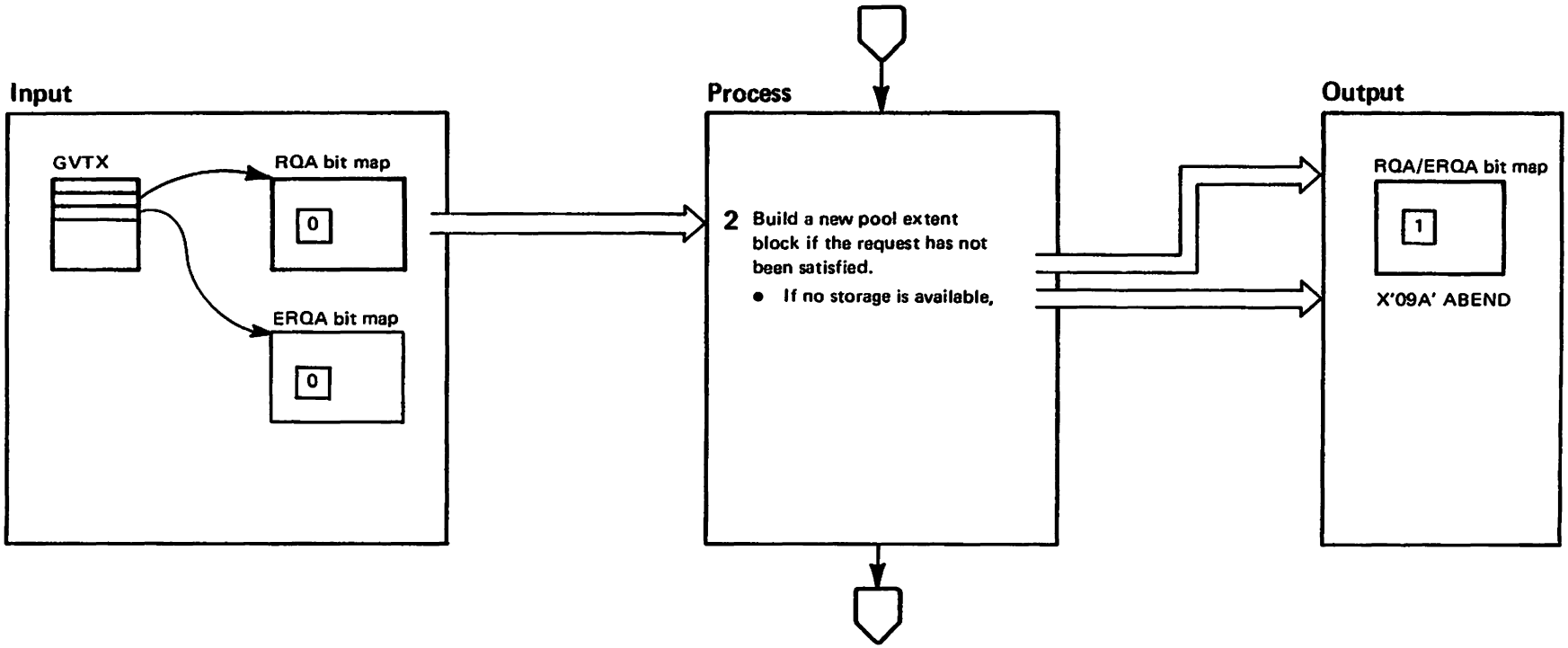
ISGSALC allocates cells from the pool extent blocks (PEXBs) in the resource queue area (RQA). The RQA is part of the private area of the global resource serialization address space.

The caller passes the address of the storage management parameter list (SMPL) in register 1 to ISGSALC. The SMPL consists of one or more requests. Each request can be for one or more cells of the same cell type. ISGSALC processes each request individually.

1 There are two resource pool tables (RPTs). One RPT for global control blocks, GRPT, and one for local control blocks, LRPT. Each table contains an entry for each cell type that can be requested. Each RPT entry contains a queue of active and inactive pool extent blocks (PEXBs) for the associated cell type. Whenever the queue of active PEXBs is empty, the first and last PEXB pointers in the RPT entry point to the beginning of that RPT entry. Each PEXB contains a count of the number of available cells in that PEXB and a queue of those available cells. ISGSALC searches the active PEXB queue for a PEXB with available cells. From each of these PEXBs (PEXBs with available cells), ISGSALC allocates cells from the beginning of the available cell queue and decreases the count of available cells within the PEXB and the count of available cells in the entire pool within the associated RPT entry for each allocated cell. If the cells from more than one PEXB are required to satisfy a request, ISGSALC chains the first available cell in the new PEXB to the last allocated cell in the previous PEXB. Thus, ISGSALC chains together all the cells allocated to satsify an individual request.

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-43. ISGSALC – Global Resource Serialization Storage Management Allocation Routine (Part 3 of 6)



"Restricted Materials of IBM"
Licensed Materials - Property of IBM

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

Diagram GRS-43. ISGSALC – Global Resource Serialization Storage Management Allocation Routine (Part 4 of 6)

Extended Description	Module	Label
2 If the request for a particular cell type has not been satisfied yet, the resource pool for the requested cell type is empty. Consequently, ISGSALC obtains a PEXB from the inactive PEXB queue or builds a new PEXB. If an inactive PEXB, a PEXB in which all cells are available, exists for the requested cell type, ISGSALC removes it from the inactive PEXB queue and places it on the active PEXB queue. Thus, ISGSALC can allocate cells from the now active PEXB. ISGSALC updates the RQA bit map if the cell type is QWB, MRB, CRB, HWKA or TWKA and ERQA bit map if the cell type is QCB, QEL, QXB or PQCB cell type, to indicate that the page containing the new active PEXB is in use. If the inactive PEXB queue for the resource pool for the requested cell type is empty, ISGSALC allocates and formats a PEXB from the RQA for QWB, MRB, CRB, TWKA, or HWKA cell type request and formats a PEXB from the ERQA for QCB, QEL, QXB, or PQCB cell type request. If a PEXB is placed on the active PEXB queue, ISGSALC increases the count of available cells in the entire pool within the associated RPT entry by the amount of cells contained in the PEXB. If no more RQA/ERQA storage is available, ISGSALC issues a X'09A' ABEND.		EXTEND

LY28-1695-0 (c) Copyright IBM Corp. 1987

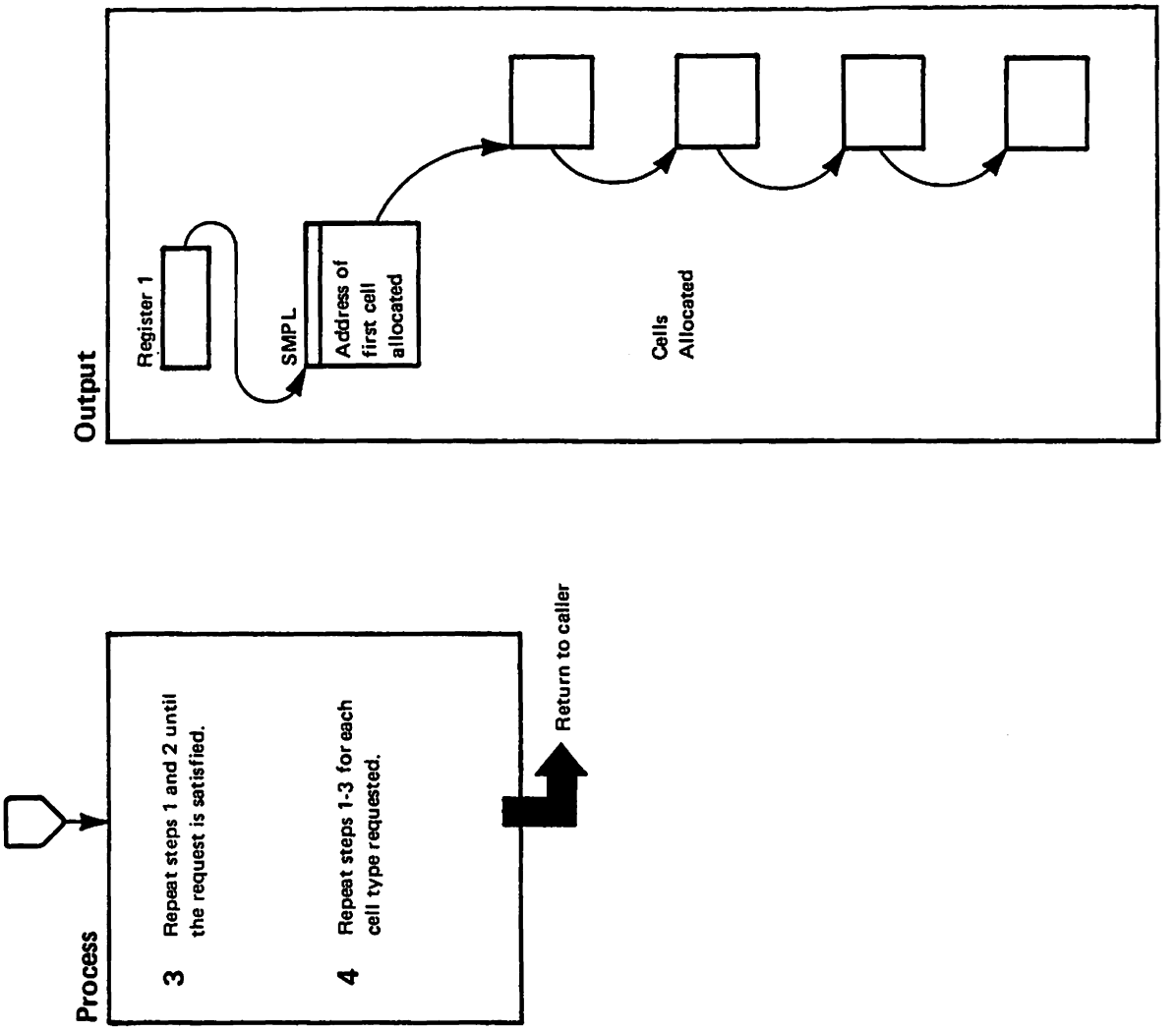


Diagram GRS-43. ISGSALC – Global Resource Serialization Storage Management Allocation Routine (Part 5 of 6)

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

Diagram GRS-43. ISGSALC – Global Resource Serialization Storage Management Allocation Routine (Part 6 of 6)

Extended Description

Module **Label**

3 If the request for a particular cell type has not been satisfied yet, ISGSALC returns to step 1 and repeats steps 1 and 2 until the requested number of cells has been allocated. If the request has been satisfied, ISGSALC places the address of the first allocated cell into the parameter list to be returned to the caller.

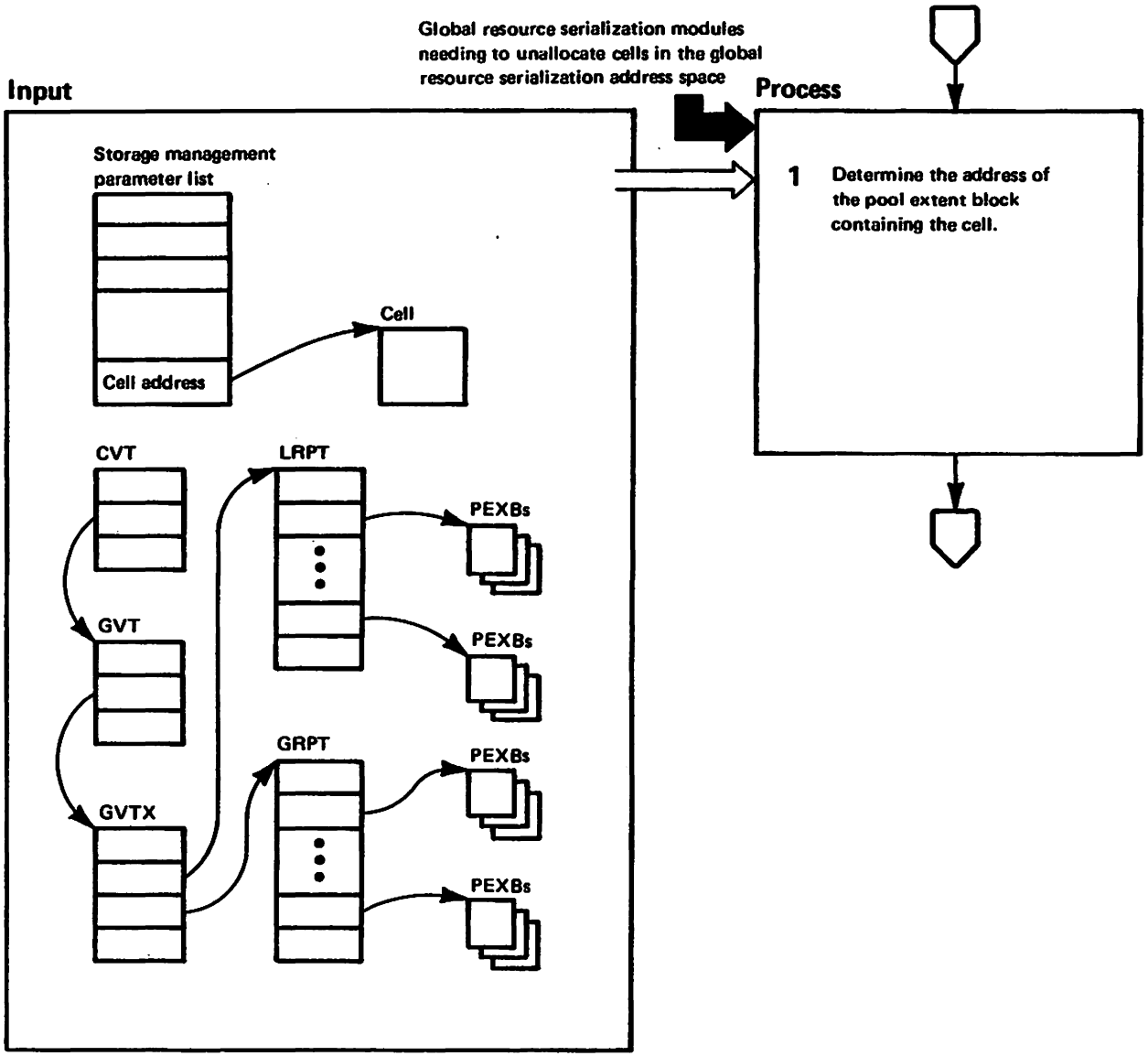
4 If more than one cell type is requested, ISGSALC returns to step 1 and repeats the entire process until all unique cell type requests have been satisfied.

Recovery Processing

The caller of ISGSALC provides recovery which calls ISGSDAL to deallocate any cells allocated to the failing request.

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-44. ISGSDAL – Global Resource Serialization Storage Management Deallocation Routine (Part 1 of 4)



"Restricted Materials of IBM"
Licensed Materials - Property of IBM

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

Diagram GRS-44. ISGSDAL – Global Resource Serialization Storage Management Deallocation Routine (Part 2 of 4)

Extended Description | **Module** | **Label**

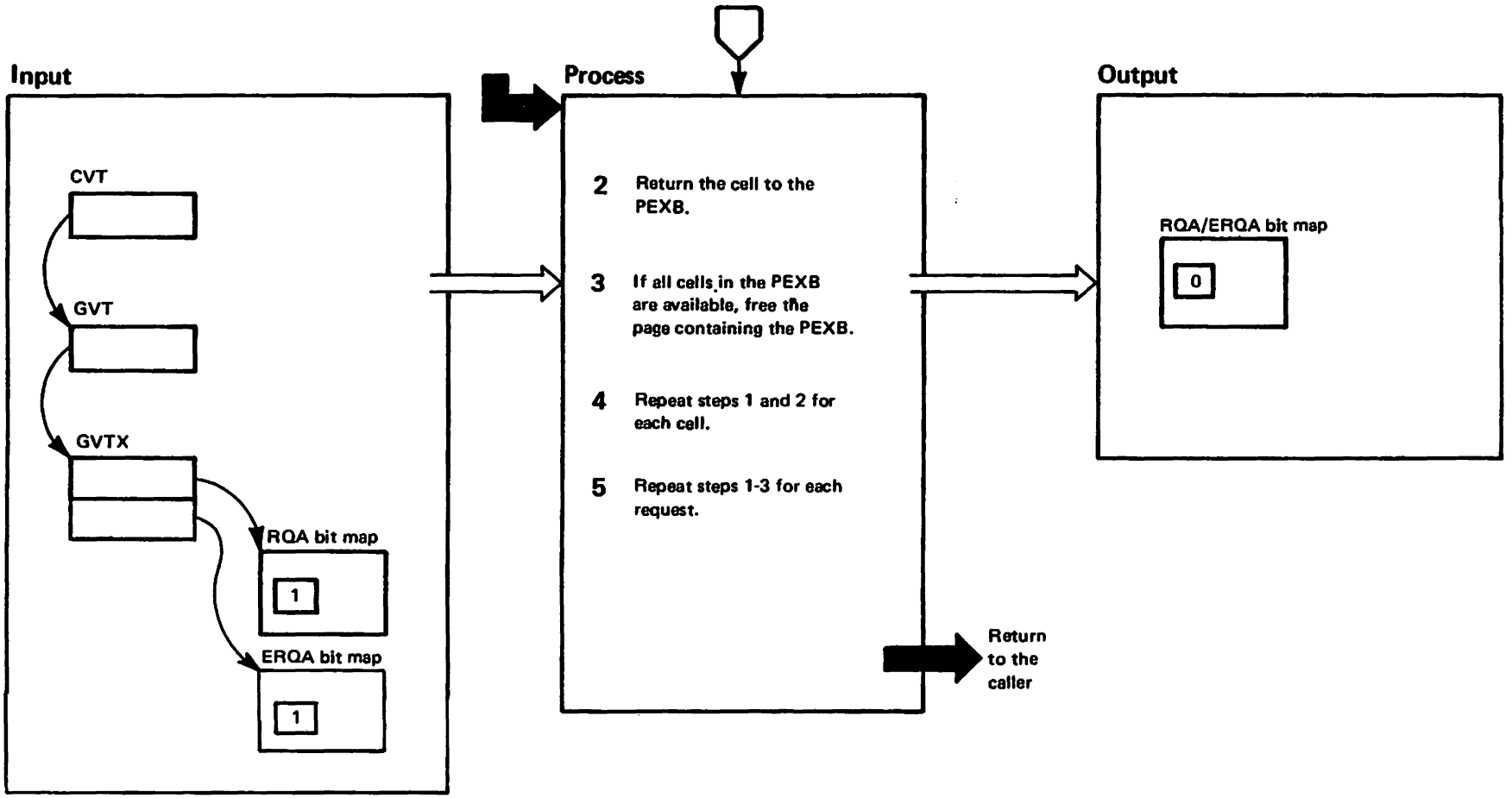
ISGSDAL deallocates QWB, MRB, CRB, TWKA or HWKA cells in the resource queue area (RQA), and QCB, QEL, QXB or PQCB cells in the extended resource queue area (ERQA), in the global resource serialization address space by returning each cell to the pool extent block (PEXB) from which it was allocated. The caller of ISGSDAL passes the address of the storage management parameter list (SMPL) in register 1. The SMPL consists of one or more requests. Each request provides the address of the first cell in the chain of cells to be deallocated. This chain of cells might consist of cells of different types, each of which is processed separately. ISGSDAL processes each request individually.

If the global resource serialization address space is initialized, ISGSDAL checks if the caller has proper lock needed to deallocate the requested cells; ISGSDAL also checks if the caller is in 24 bit amode and the request is to deallocate cells in the ERQA. ISGSDAL issues '09A' ABEND if any of the above condition is true.

1 If the global resource serialization address space has not been initialized, ISGSDAL finds the address of the PEXB header to be updated by comparing the address of the cell to be deallocated to the starting and ending address of each PEXB. If the cell address falls within a particular PEXB, that PEXB header is updated. This logic is repeated for each cell to be deallocated. Only local cells can be processed before the global resource serialization address space is initialized. If the global resource serialization address space has been initialized, ISGSDAL determines the address of the PEXB header to be updated by masking out the low order 12 bits of the address of the cell to be deallocated. This approach is possible because all PEXBs in the global resource serialization address space start on page boundaries and are a page in length (4K bytes).

LY28-1695-0 (c) Copyright IBM Corp. 1987

Diagram GRS-44. ISGSDAL – Global Resource Serialization Storage Management Deallocation Routine (Part 3 of 4)



"Restricted Materials of IBM"
Licensed Materials – Property of IBM

LY28-1695-0 (c) Copyright IBM Corp. 1987

"Restricted Materials of IBM"
Licensed Materials – Property of IBM

Diagram GRS-44. ISGSDAL – Global Resource Serialization Storage Management Deallocation Routine (Part 4 of 4)

Extended Description	Module	Label
2 If available cells exist in the PEXB, ISGSDAL chains the cell to be deallocated to the last available cell in the PEXB. If no available cells exist in the PEXB, ISGSDAL sets the first and last available cell pointer to the cell being deallocated. ISGSDAL increases the count of available cells within the PEXB and the count of available cells in the entire pool within the associated RPT entry for each cell returned.		
3 If all cells in the PEXB are available, ISGSDAL dequeues the PEXB from the active queue of PEXBs, queues it to the inactive queue of PEXBs, and decreases the count of available cells in the entire pool within the associated RPT entry by the amount of cells contained in the PEXB. If the number of inactive PEXBs exceeds the allowed number of inactive PEXBs, ISGSDAL schedules an SRB for ISGSPRLS, which is an entry point in ISGSDAL, to page-release all inactive PEXBs. ISGPRLS uses PGSER macro to release each page that contains an inactive PEXB. ISGSDAL sets each bit in the RQA bit map that corresponds to a released virtual page to zero.	IEAVPSIB	ISGSPRLS
4 ISGSDAL repeats steps 1 and 2 until each cell has been returned to its PEXB.		
5 ISGSDAL repeats steps 1-3 until each request has been satisfied.		

Recovery Processing

Recovery is provided by the caller.

LY28-1695-0 (c) Copyright IBM Corp. 1987

LY28-1695-0 (c) Copyright IBM Corp. 1987

INDEX

LY28-1695-0 (c) Copyright IBM Corp. 1987

LY28-1695-0 (c) Copyright IBM Corp. 1987

LY28-1695-0 (c) Copyright IBM Corp. 1987

S

LY28-1695-0 (c) Copyright IBM Corp. 1987

MVS/Extended Architecture
System Logic Library:
Global Resource
Serialization

READER'S
COMMENT
FORM

LY28-1695-0

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: Global Resource Serialization

"Restricted Materials of IBM"
All Rights Reserved
Licensed Materials - Property of IBM
(Except for Customer-Originated Materials)
©Copyright IBM Corp. 1987
LY28-1695-0 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 390
Poughkeepsie, New York 12602

Fold and tape Please Do Not Staple Fold and tape

Printed in U.S.A.

LY28-1695-00

MVS/Extended Architecture System Logic Library: Global Resource Serialization

"Restricted Materials of IBM"
All Rights Reserved
Licensed Materials - Property of IBM
©Copyright IBM Corp. 1987
LY28-1695-0 S370-36

Printed in U.S.A.