SY28-0762-0
File No. S370-36

Systems

OS/VS2
System Logic Library
Volume 2

VS2.03.807

IBM

This minor revision incorporates the following Selectable Unit:

Supervisor Performance #2 VS2.03.807

The selectable unit to which the information applies, is noted in the upper corner of the page.

First Edition (July, 1976)

This is a reprint of SY28-0714-0 incorporating changes released in the following Selectable Unit Newsletter:

SN28-2729 (dated May 28, 1976)

This edition applies to Release 3.7 of OS/VS2 and to all subsequent releases of OS/VS2 until otherwise indicated in new editions or Technical Newsletters. Changes are continually made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370 Bibliography,* GC20-0001, for the editions that are applicable and current.

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

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Publications Development, Department D58, Building 706-2, PO Box 390, Poughkeepsie, N.Y. 12602. Comments become the property of IBM.

© Copyright International Business Machines Corporation 1976

System Logic Library comprises seven volumes. Following is the content and order number for each volume.

OS/VS2 System Logic Library,

Volume 1 contents: SY28-0713
- MVS logic introduction
- Abbreviation list
- Index for all volumes

Volume 2 contents: SY28-0714
- Method of Operation diagrams for
- Communications Task
- Command Processing
- Region Control Task (RCT)
- Started Task Control (STC)
- LOGON Scheduling

Volume 3 contents: SY28-0715
- Method of Operation diagrams for
- System Resources Manager (SRM)
- System Activity Measurement Activity (MF/1)
- JOB Scheduling
  - —Subsystem Interface
  - —Master Subsystem
  - —Initiator/Terminator
  - —SWA Create Interface
  - —Converter/Interpreter
  - —SWA Manager
  - —Allocation/Unallocation
  - —System Management Facilities (SMF)
  - —System Log
  - —Checkpoint/Restart

Volume 4 contents: SY28-0716
- Method of Operation diagrams for
- Timer Supervision
- Supervisor Control
- Task Management
- Program Management
- Recovery/Termination Management (R/TM)

Volume 5 contents: SY28-0717
- Method of Operation diagrams for
- Real Storage Management (RSM)
- Virtual Storage Management (VSM)
- Auxiliary Storage Management (ASM)

Volume 6 contents: SY28-0718
- Program Organization

Volume 7 contents: SY28-0719
- Directory
- Data Areas
- Diagnostic Aids

Please note that if you use only one order number, you will only receive that volume. To receive all seven volumes, you must either use all seven form numbers or, simply the following number: SBOF-8210. If you use SBOF-8210, you will receive all seven volumes.

The publication is intended for persons who are debugging or modifying the system. For general information about the use of the MVS system, refer to the publication *Introduction to OS/VS Release 2,* GC28-0661.

How This Publication is Organized

This publication contains six chapters. Following, is a synopsis of the information in each section:

- *Introduction and Master Index* — an overview of each of the functions this publication documents, an abbreviation list of all acronyms used in the publication, and a complete index for all seven volumes.
- *Method of Operation* — a functional approach to each of the subcomponents, using both diagrams and text. Each subcomponent begins with an introduction; all the diagrams and text applying to that subcomponent follow.
- *Program Organization* — a description of module-to-module flow for each subcomponent; a description of each module's function, including entry and exit. The module-to-module flow is ordered by subcomponent. The module descriptions are in alphabetic order without regard to subcomponent.
- *Directory* — a cross-reference from names in the various subcomponents to their place in the source code and in the publication.
- *Data Areas* — a description of the major data areas used by the subcomponents (only those, however, that are not described in *OS/VS Data Areas,* SYB8-0606, which is on microfiche); a data area usage table, showing whether a module reads or updates a data area; a control block overview diagram for each subcomponent, showing the various pointer schemes for the control blocks applicable to each subcomponent; a table detailing data area acronyms, mapping macro instructions, common names, and symbol usage table.

- *Diagnostic Aids* — the messages issued, including the modules that issue, detect, and contain the message; register usage; return codes; wait state codes; and miscellaneous aids.

Corequisite Reading

The following publications are corequisites:

- *OS/VS2 JES2 Logic,* SY28-0622
- *OS/VS Data Areas,* SYB8-0606 (This document is on microfiche.)
- *OS/VS2 System Initialization Logic,* SY28-0623

Contents

Figures

Alphabetic List of Diagrams

Section 2: Method of Operation

This section uses diagrams and text to describe the functions performed by the scheduler, supervisor, MF/1, SRM, and ASM functions of the OS/VS2 operating system. The diagrams emphasize functions performed rather than the program logic and organization. Logic and organization is described in "Section 3: Program Organization."

The method-of-operation diagrams are arranged by subcomponent as follows:

- Communications Task.
- Command Processing (includes Reconfiguration Commands).
- Region Control Task (RCT).
- Started Task Control (STC) (includes START/LOGON/MOUNT).
- LOGON Scheduling
- System Resources Manager
- System Activity Measurement Facility (MF/1)
- Job Scheduling:
  - Subsystem Interface.
  - Master Subsystem.
  - Initiator/Terminator.
  - SWA Create Interface.
  - Converter/Interpreter.
  - SWA Manager.
  - Allocation/Unallocation.
  - System Management Facilities (SMF).
  - System Log.
  - Checkpoint/Restart.
- Timer Supervision.
- Supervisor Control.
- Task Management.
- Program Management.
- Recovery/Termination Management (R/TM).
- Real Storage Management (RSM).
- Virtual Storage Management (VSM).
- Auxiliary Storage Management (ASM).

The diagrams for each subcomponent are preceded by an introduction that summarizes the subcomponent's function. Following each introduction is a visual table of contents that displays the organization and hierarchy of the diagrams for that subcomponent.

The diagrams cross-reference each other using diagram numbers and module names. As an aid in locating the diagrams that are cross-referenced, an alphabetic list of all diagram names and their corresponding page numbers follows this introduction.

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: 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.

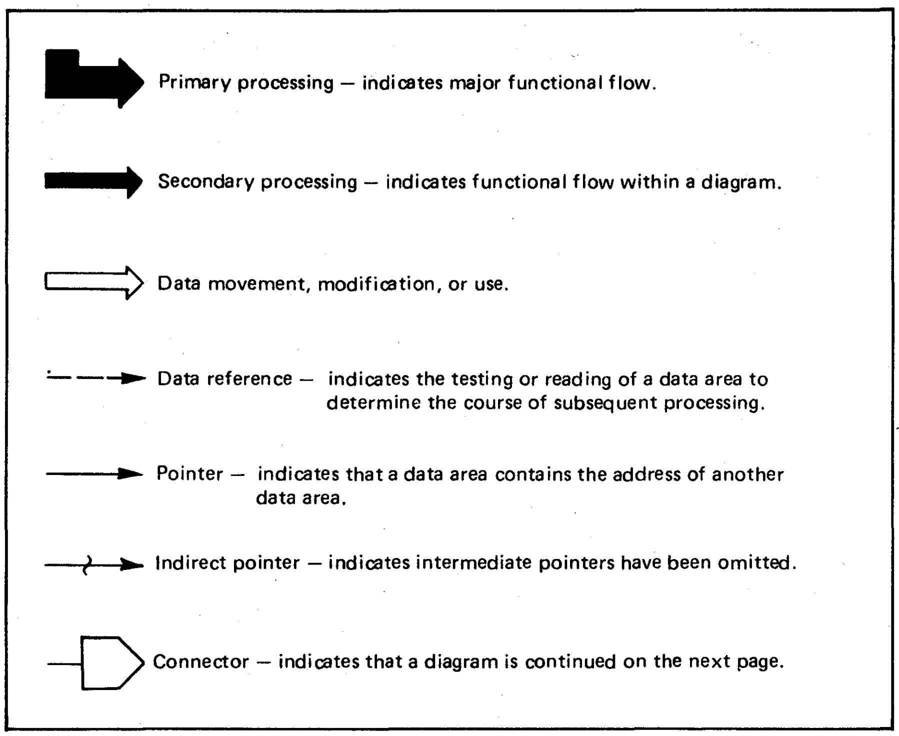


Figure 2-1. Key to Symbols Used in Method-of-Operation Diagrams

Major Function

The major function of the communication task is to transfer messages from user programs and system routines to the operators at the system consoles. This function includes the transfer of messages to TSO terminals that are operating in MONITOR mode. See Figure 2-2.

With few exceptions, three macro instructions are used to call the communication task: WTO, WTOR, and DOM.

Write To Operator (WTO) has two basic forms:

1. Each time a user or system program issues a WTO macro instruction, one message/line is transferred to one or more operator consoles.
2. A multiple line (MLWTO) permits user and system programs to transfer up to ten message lines to one or more operator consoles with one WTO macro instruction. System programs can attach an unlimited number of additional lines to the same message in 1-10 message sets per WTO macro instruction.

Write To Operator with Reply (WTOR) permits any user program or system routine to transfer one message to one or more consoles and provides a mechanism by which a console operator may respond to that message. The reply is then returned to the program or routine that issued the WTOR. (The MLWTO form is not available with the WTOR.)

Delete Operator Message DOM has two basic forms:

1. As used by all user programs and system routines, DOM deletes one to sixty WTO messages from graphic consoles. A DOM can be issued against a nongraphic console with no adverse effects.
2. User programs and system routines can issue the DOM macro instruction with the operand REPLY=YES. REPLY=YES deletes one to sixty WTOR messages from all consoles, graphic and hardcopy, for which an operator has not responded. For example, an operator could reply to a system mount message with CANCEL or he could mount the volume. Since the system can recognize that the volume has been mounted, a reply is not needed from the operator; therefore, a system routine could issue the DOM macro instruction with REPLY=YES to remove the mount message from the queue of WTOR/messages requiring an operator response.

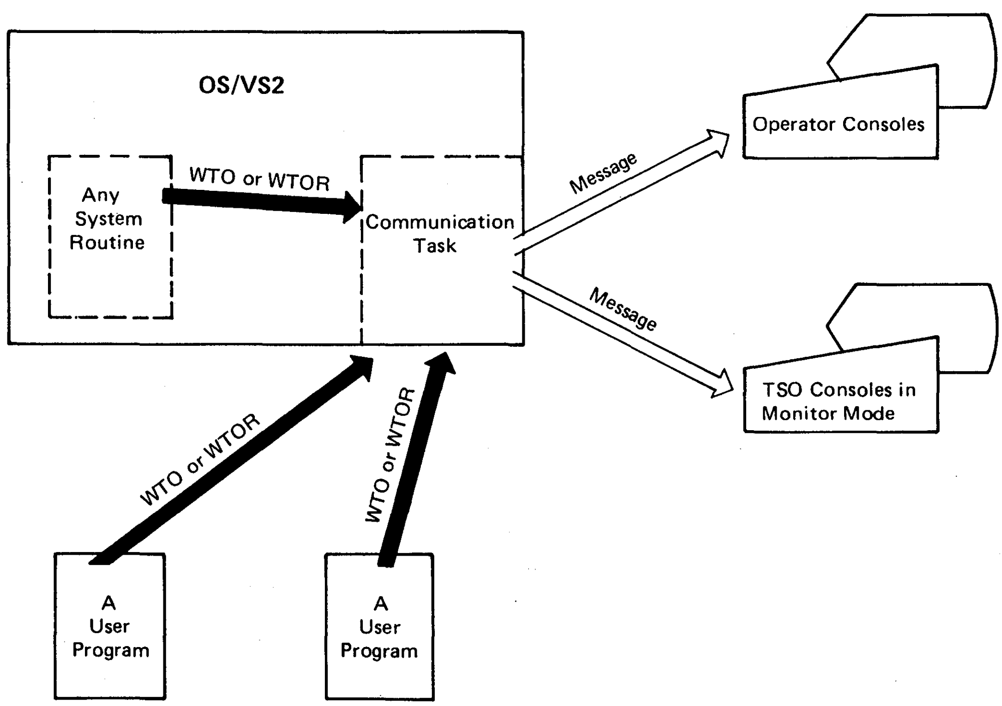


Figure 2-2. The Communication Task

Supporting Functions

In support of communications between user programs or system routines and the various consoles, the communication task also supplies either the modules that are incorporated into other system functions or performs the service itself.

- Operator console initiation (documented with NIP).
- Elimination of messages related to a terminating task (documented with task termination).
- Console attention, which permits a console operator to enter an operator command or reply to a WTOR message.
- Switching the master console functions from the current master console to an alternate console.
- Cleaning up the communication task's control block queues.
- Error recovery—both from communication task errors and system errors.
- Command processors for REPLY, DISPLAY R, and DISPLAY CONSOLES are supplied by the communication task (documented with command processors).

Console Attention

When the console operator presses the attention key, or its equivalent, an I/O interruption occurs. The IOS interrupt handler passes control to a communication task routine that posts the communication task ECB. Eventually, the communication task will become the highest priority task to be executed by the system, at which time the dispatcher will give CPU time to the communication task. The communication task checks the posted ECB to determine what needs to be done and determines the console requiring service. The communication task then calls its own SVC 72 to issue the read command to the console device from which the interrupt came.

The communication task then returns control to its wait service routine. If further communication task services have been posted to a communication task ECB, those services are performed. When all services have been performed, that is, there are no outstanding posted items in the ECB, control is returned to the dispatcher.

At this time, the console device is unlocked. The operator may enter an operator system command. That command will be processed by the operator command processor.

External Interrupt

The computer operator presses the external interrupt button on the CPU to transfer the functions of the master console to a previously defined alternate console. This feature permits a console operator to signal the system when the master console is not operating properly; however, it can be used solely to switch master console functions from one console device to another.

Note: If the master console was the only active console when the external interrupt button was pressed, the console operator can restore console operations by simply pressing the external interrupt button a second time. The system assumes that the first pressing of the button was an accident. The operator is alerted to this condition by the alarm bell ringing three times, provided the alarm bell feature is mounted on at least one of the system consoles.

Pressing the external interrupt button causes a system external interruption, which is processed by the external first level interrupt handler. Finding that the interrupt came from the CPU, control is passed to the communication task interrupt handler module supplied to the external first level interrupt handler by the communication task. This module posts the communication task ECB.

When the communication task becomes the highest priority task to be executed by the system. The dispatcher gives CPU time to the communication task who checks the posted ECB to determine what needs to be done. The communication task wait service routine calls SVC 72, the communication task console switch routine, to transfer the functions of the master console to an alternate console.

When the console switch operation is finished, control is given to the wait service routine. If further communication task services have been posted to the communication task ECB, those services are performed. When all services have been performed, that is, there are no outstanding posted ECBs, control is returned to the dispatcher.

I/O Complete Processing

The I/O completion processor handles the I/O interruption that occurs when there is an operation on a console device. The three situations that must be handled are when:

- A message is sent to a console and there is no I/O error.

- A system command was received from a console and there was no I/O error.
- An I/O error occured during data transmission.

After a message has been sent to the operator console, an I/O interruption occurs to inform the system of the status of that data transmission. If the message was received at the console without an error, the communication task flags the message (WQE) and the console's pointer to that message (CQE) for deletion at a later time. Control then returns to the wait service routine.

When an operator enters a system command, he indicates the end of the command by pressing the end of block (EOB) button, which causes an I/O interruption. When the communication task receives control, the operator command processor (SVC 34) is called to process the command. When the operator command processor returns to the communication task, control is given to the wait srevice routine.

An error causes the communication task to attempt a switch to another console device, if one is available.

Unconditional Message to Inactive Console (QREG0 Processing Routine)

During system generation, an identification is assigned to each console and placed in the unit control module entry (UCME). Any system program needing to communicate with a specific console can obtain the console identification from the UCME and place it in register 0. A message is then *unconditionally* transmitted to that console by using a WTO or WTOR macro instruction with the parameter MCSFLAGS=QREG0. Programs running under a problem program key, programs not running in supervisor state, or programs that are not authorized, are prevented from using this parameter.

When the console identified in register 0 is active, the unconditional message is processed in the same manner as any other message, and the QREG0 processing routine is not attached. QREG0 processing is only for unconditional messages to inactive consoles.

If the console is inactive, and if the QREG0 processing routine were not present, unconditional messages could start to fill the allotted write queue element (WQE) and operator reply element (ORE) space without the knowledge of the master console operator. These messages would have no way out of the system until the inactive console is made active. Once the allotted WQE and ORE space is filled, system operator message service would be slowed; thereby, slowing the operator's response to all messages. If this were permitted, the system could be left in a situation where performance might be degraded. QREG0 processing prevents this possible situation.

To prevent performance degradation, the QREG0 processing routine sends a WTOR message (IEA962A) to the master console operator. This message tells him that the system is in the process of queueing a message for an inactive console. He is given three possible responses: SEND, DELETE, and OK. SEND displays the message at the master console and deletes it from the queue of messages for the inactive console. DELETE simply deletes the message from the system. OK permits the queueing process to continue and assumes that the operator will activate the inactive console.

If the operator enters some other response to the WTOR message (probably a typographical error), the QREG0 routine issues a second WTOR message (IEA963A) informing him of his error and asking him to reenter his response. This message is repeated until the operator has entered one of the three acceptable responses.

Console Device Support

The communications task supports the following devices as consoles:

- 1052 printer-keyboard.
- 3210 console printer-keyboard.
- 3215 console printer-keyboard.
- 3213 console printer.
- 2501 card reader.
- 2520 card reader punch.
- 2540 card reader punch.
- 3505 card reader.
- 3525 card punch.
- 1403 printer.
- 1443 printer.
- 3211 printer.
- 2250 display unit.
- 2260 display station.
- 3066 system console.
- 3277 display station.
- 3284 printer.
- 3286 printer.
- 2740 communication terminal.
- System console for the Model 158.

The communications task modules that provide I/O support for these consoles are called device support processors (DSPs). The DSPs for graphics consoles are part of DIDOCS (device-independent display operator console support).

SVC 72

All the DSPs (including the DIDOCS DSPs) are part of SVC 72 (see figure 2-3). Besides the DSPs, SVC 72 contains a routing module, which passes control to the appropriate DSP, and a console switch routine, which changes the master console from the current one to an alternate.

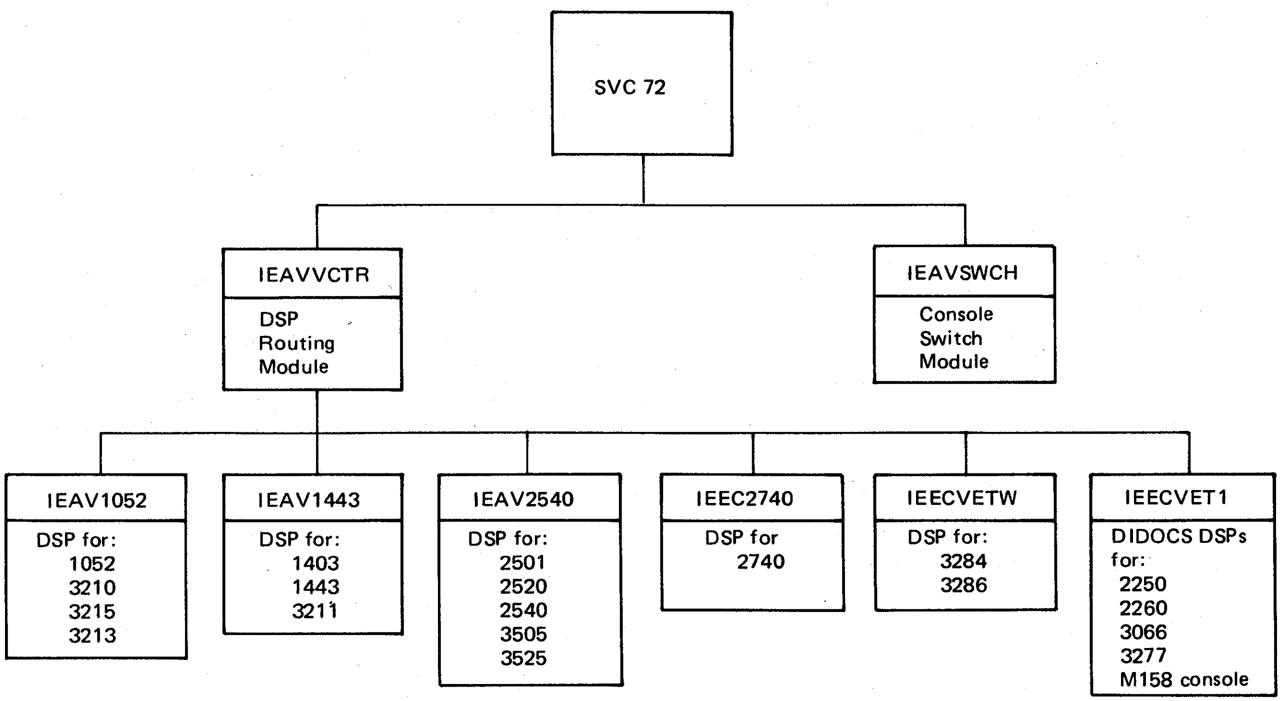


Figure 2-3. SVC 72

Method of Operation Diagrams

The method of operation diagrams are based on the specific functions being performed by the communication task. See Figure 2-4. With the exception of the communication task's SVCs and a few program modules supplied to other functions of the operating system that interface with the communication task, the function to be performed is determined by the communication task's wait service routine (IEAVMQWR).

Before the wait service routine is called by the dispatcher, at least one of several communication task event control blocks (ECBs) have been posted. From these ECBs, the wait service routine determines the function to be performed. The following is the sequence in which these ECBs are tested:

ECB or Control Bit	Function to be Performed
UCMARECB	Alternate CPU Recovery (Documented with the ACR Routine in Recovery Termination Management).
UCMXECB	External Interrupt — Switches the master console to the next available alternate console.
UCMAECB	Attention Interrupt — Prepares the interrupting console to receive a keyboard entry.
EILIOL	I/O Processing complete — Handles the I/O interruption after a message has been displayed at a console. The EILIOL is each console's unit control module entry (UCME).
UCMSYSJ or UCMPF	Console or Hardcopy Output Pending — Causes the message already queued for output to be displayed on the respective hardcopy or console device.
UCMOECB	Queue Message for Output — Prepares the message posted by a WTO macro instruction for output to the appropriate consoles.
UCMSYSI	Clean Up the WQE chain — Eliminates WQEs that are no longer needed.
UCMDECB	Delete Operator Message — Deletes the message indicated by the DOM macro instruction.
UCMNPECB	Write NIP routines.

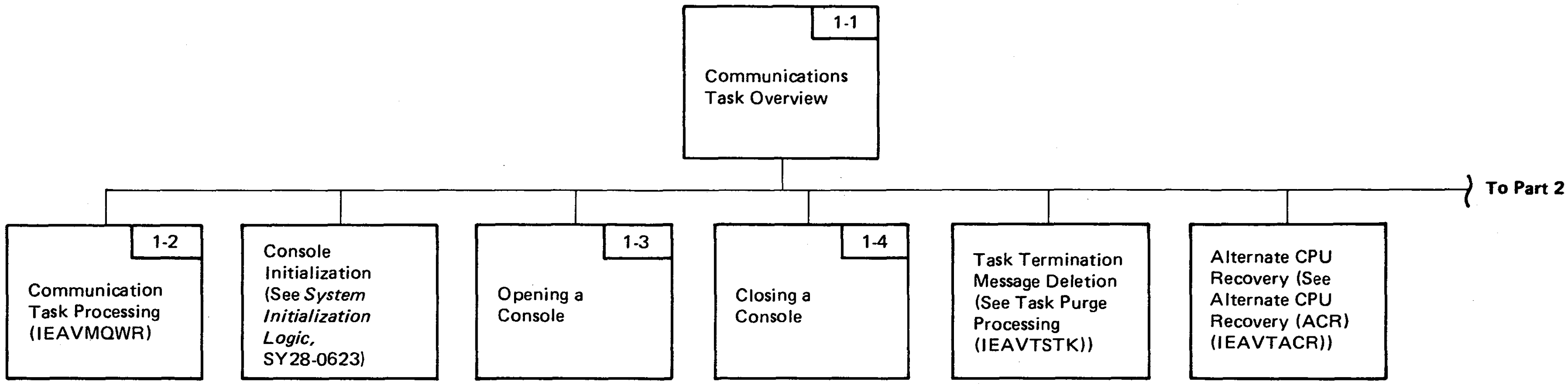


Figure 2.4. Communications Task Visual Contents (Part 1 of 3)

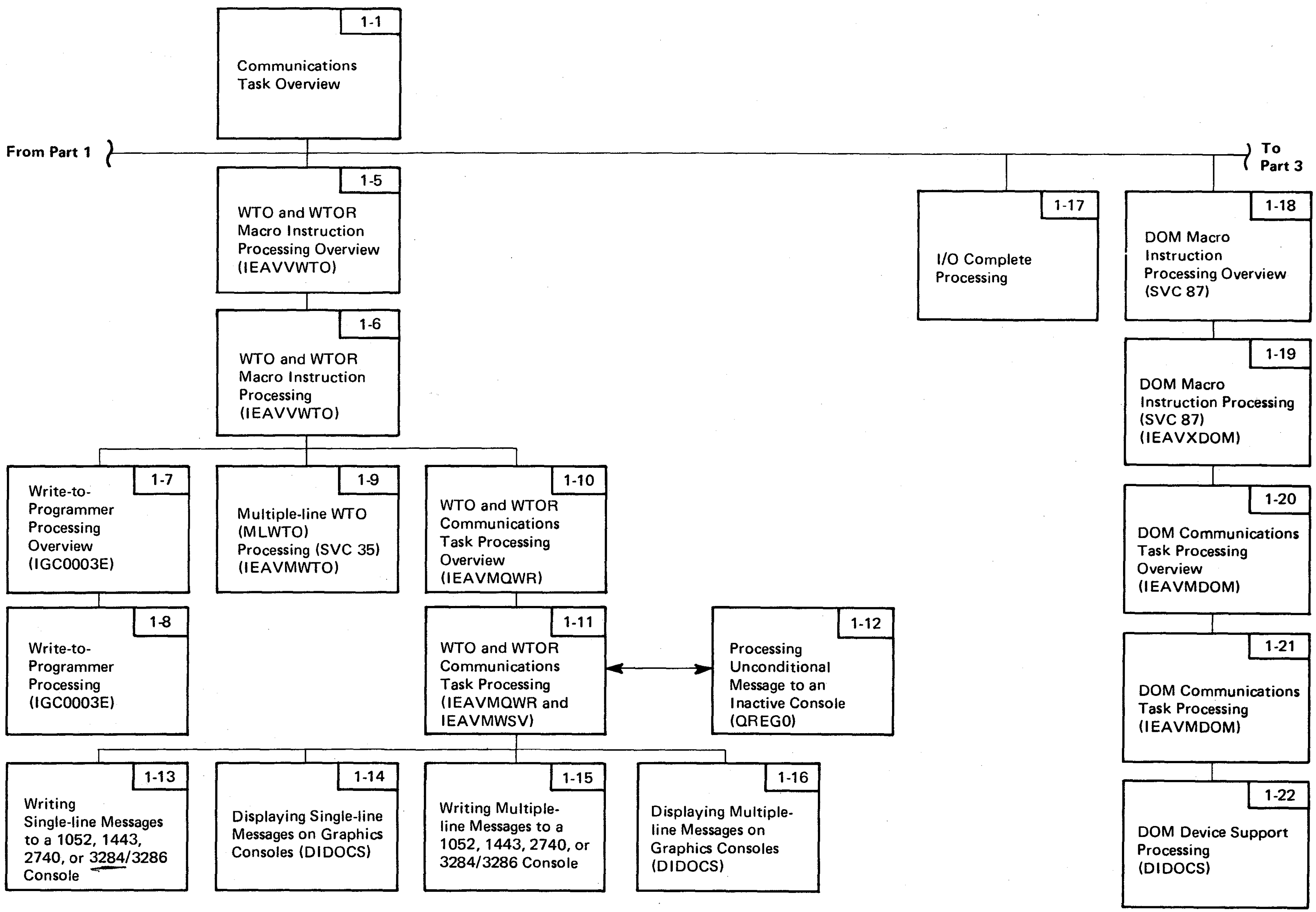


Figure 2-4. Communications Task Visual Contents (Part 2 of 3)

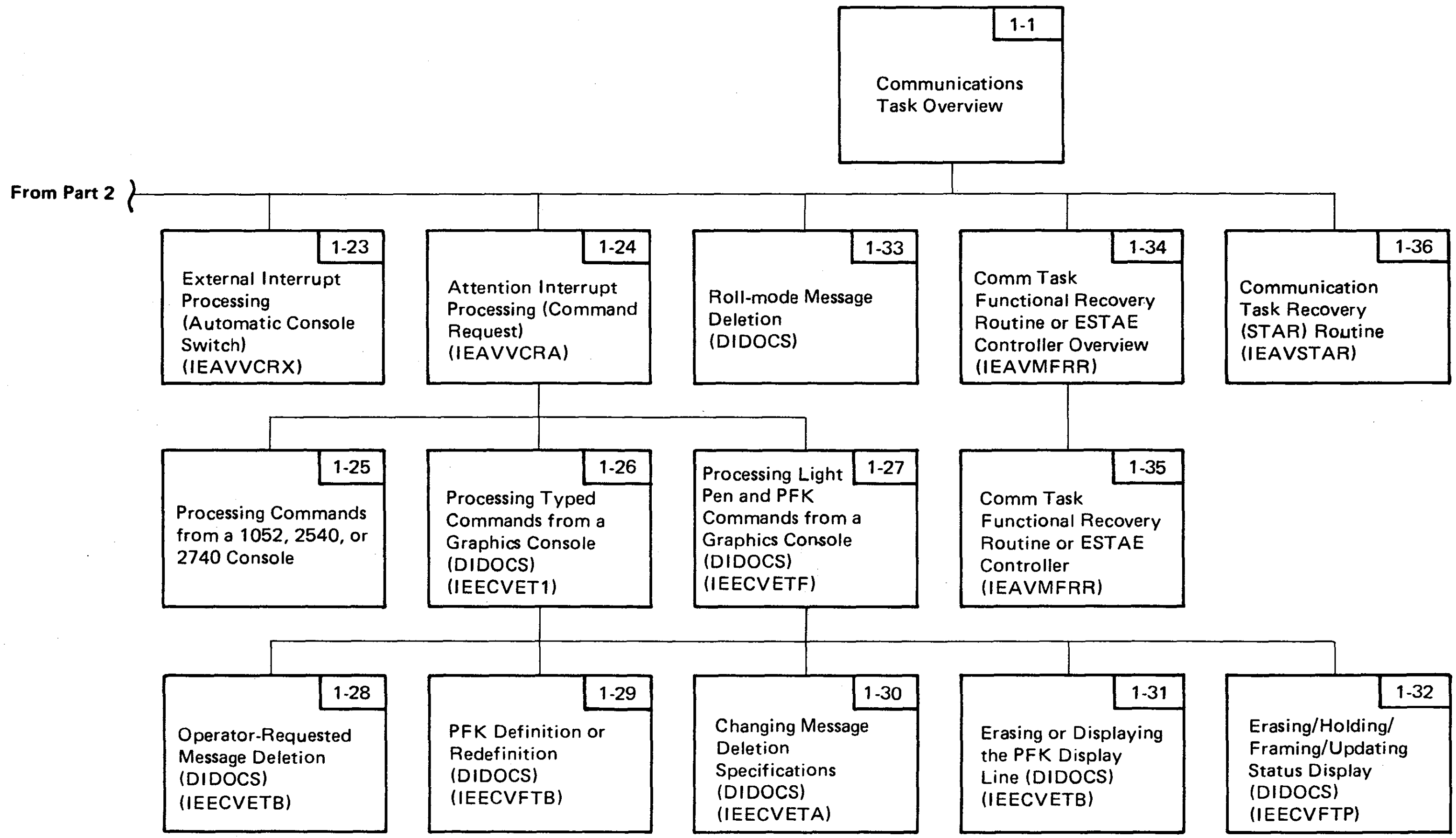


Figure 2.4. Communications Task Visual Contents (Part 3 of 3)

Diagram 1-1. Communication Task Overview (Part 1 of 2)

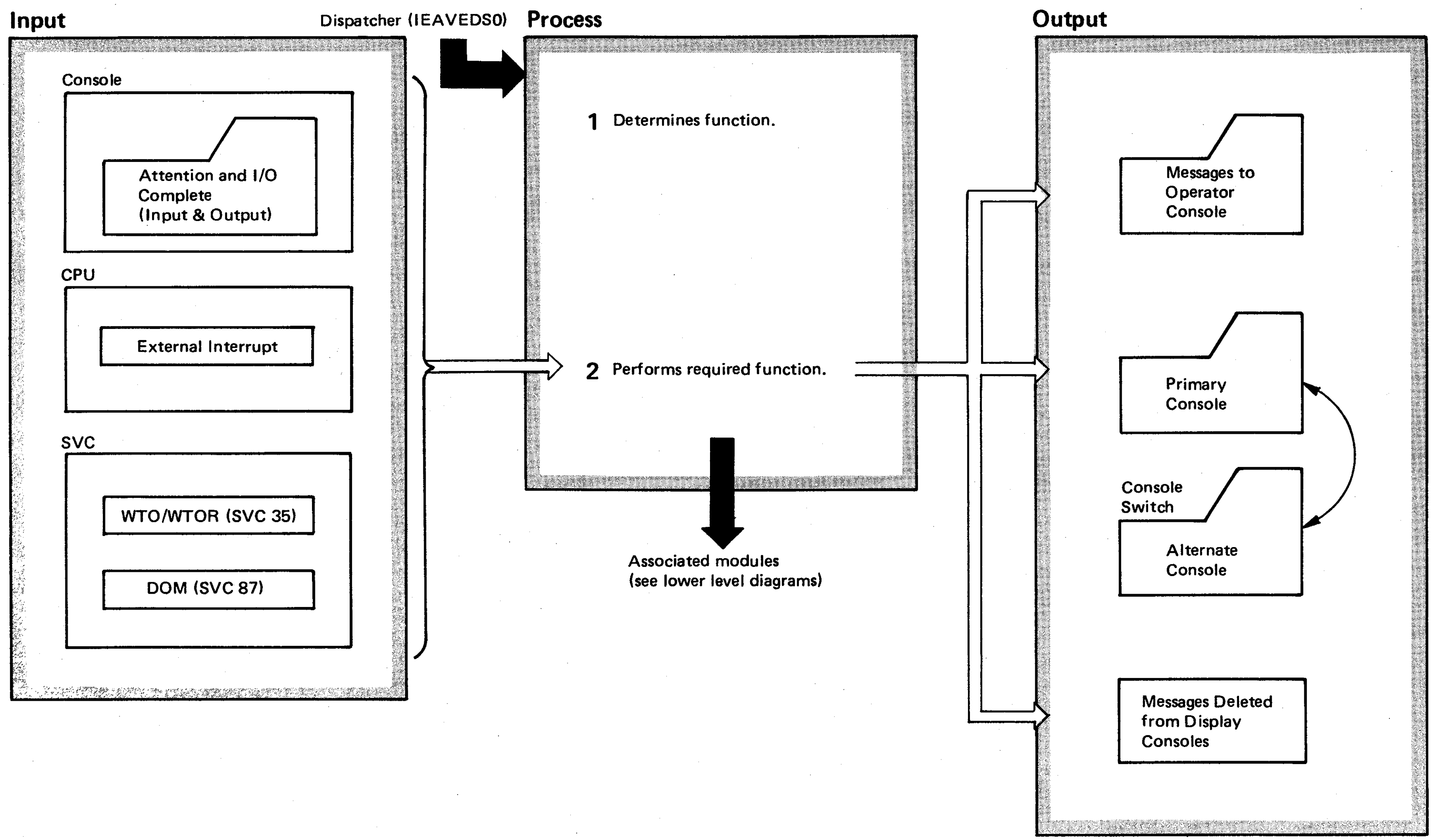


Diagram 1-1. Communication Task Overview (Part 2 of 2)

Note: COMTASK provides the modules that initialize the COMTASK and COMTASK control blocks for initialization. COMTASK also supplies the module to task termination for deleting messages associated with the terminating task. These modules are respectively documented in the *OS/VS2 System Initialization Logic,* SY28-0623, and in the Recovery/Termination Management areas of this PLM.

Extended Description **Module**

1 The Communications Task (COMTASK) handles communications between the operator(s) and the system. The types of communication that COMTASK handles are:

- Operator commands from a console.
- Output to the operator caused by the Write-To-Operator (WTO), Write-To-Operator-with-Reply (WTOR), and the Delete-Operator-Message (DOM) macro instructions.
- External interrupts, which are caused by the operator pressing the INTERRUPT key on the operator control panel. COMTASK switches the master console's functions to an alternate.
- Automatic console switching from a console to its alternate when an unrecoverable I/O error occurs on the console.
- Console switching as a result of the VARY CHANNEL, VARY CPU, or VARY MSTCONS commands.
- Console switching as a result of a CPU failure in a multiprocessing system is part of alternate CPU recovery (ACR).

Extended Description **Module**

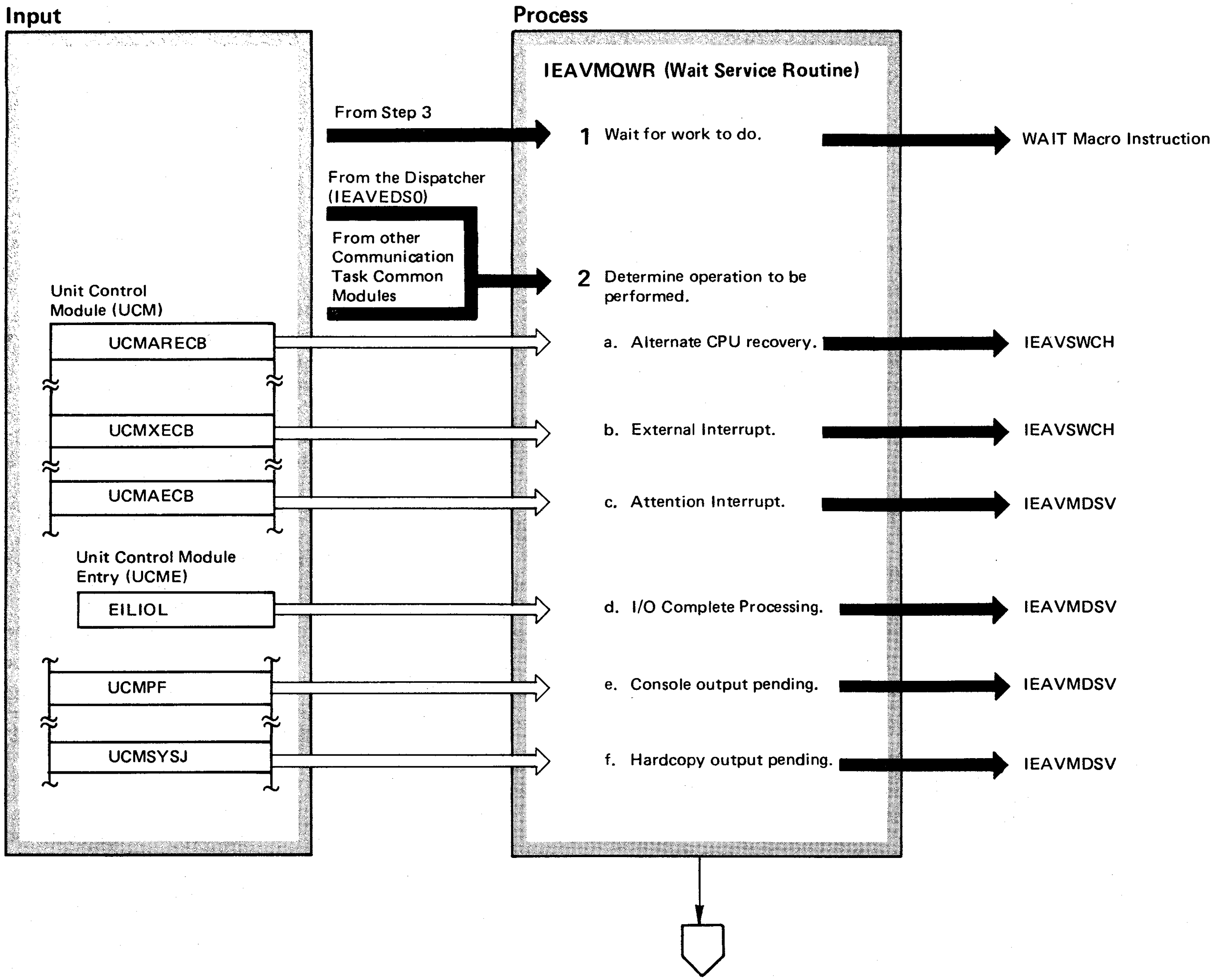
2 The COMTASK is an interrupt-driven system task. It has its own TCB, which is created at system generation time.

Multiple Console Support (MCS) is a standard feature that supports up to 32 consoles. With MCS, messages can be routed to up to 15 different functional areas, according to the type of information in the message.

Device Independent Display Operator Console Support (DIDOCS) is an option of the VS2 control program. It provides uniform operator console services for the:

- 2250 Display Unit, Models 1 and 3
- 2250 Display Station, Model 1 with 2848 Display Control or Model 3
- Model 165 II Display Console
- 3277 Display Unit, Models 1 and 2
- Model 158 Display Console

Diagram 1-2. Communication Task Processing (IEAVMQWR) (Part 1 of 4)



Extended Description	Module	Label
Wait Service Routine The communication task's wait service routine is a never ending task. It is given control by the dispatcher after one of the communication task's event control blocks (ECBs) has been posted. Upon each entry into this routine, the entire list of communication task ECBs is tested from top to bottom in priority sequence. The posted ECB determines the service that will be performed by the communication task. As each service is completed, control is returned to this routine and the entire list of ECBs is again tested for an active ECB. When no active ECBs are found, this routine issues the WAIT macro instruction. This macro instruction places this routine in the wait state until the next communication task ECB has been posted.		
1 The communication task's wait service routine issues a WAIT macro instruction when there is no further common processing to be performed by the communication task.	IEAVMQWR	
2 When the dispatcher gives control back to the communication task, control returns to this entry point. It is also the entry point for all communication task modules when returning control to the wait service routine. This step determines the function to be performed by the communication task and then branches to the communication task module that is capable of doing the work. The sequence below represents the priority in which functions are handled by the communication task.		

Extended Description	Module	Label
a. Alternate CPU recovery is the process of switching from one CPU to another in multiple CPU configurations.	IEAVMQWR	WRACR
b. External interrupt switches the master console functions from the current master console to the next available alternate console.		WREXT1
c. Attention interrupt prepares the console from which the interrupt was received to accept an operator command.		WRATTN
d. I/O processing complete is the operation performed after a message has been sent to or received from a console. The processing is the result of the interrupt an I/O device causes after performing each operation.		WRIOCOMP
e. Console output pending indicates that there is at least one message queued and ready for some console. The UCMPF bit is set if a console is busy when a WTO or WTOR message was queued for that console *or* one message was queued for several consoles.		WRDVSERV
f. Hardcopy output indicates that at least one message is queued for hardcopy output. Note: Hardcopy is strictly for messages that are placed in some data set. When a console is used for the hardcopy function, the message is queued to that console as though the message was for that console originally and the hardcopy bit is not set; however, all messages displayed at the hardcopy console are in the hardcopy format.		WROUTPUT

Diagram 1-2. Communication Task Processing (IEAVMQWR) (Part 3 of 4)

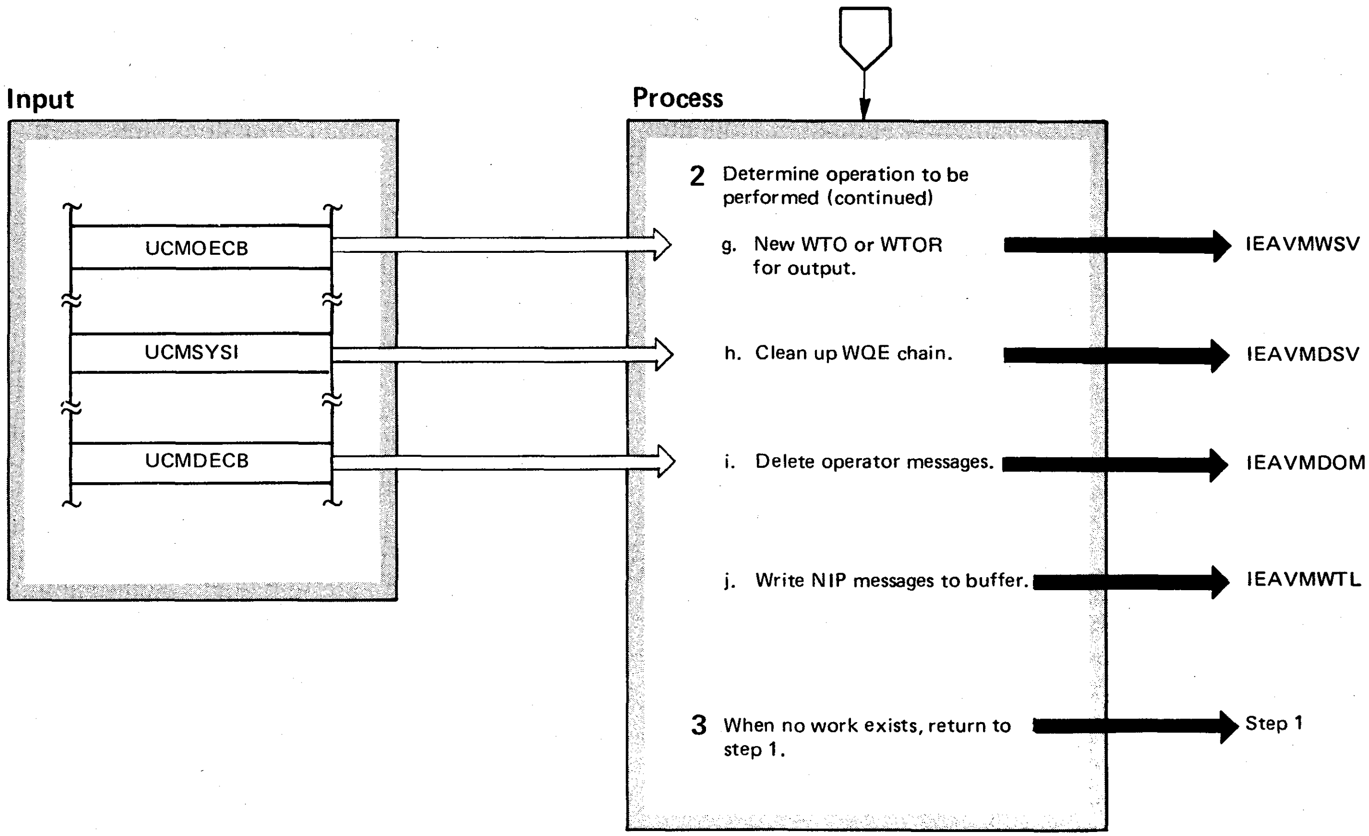


Diagram 1-2. Communication Task Processing (IEAVMQWR) (Part 4 of 4)

Extended Description	Module	Label
2 (Continued)		
g. A WTO or WTOR macro instruction has previously prepared a write queue element (WQE), possibly an operator reply element (ORE), and posted the communication task ECB (UCMOECB). As a result of this ECB being posted, a console queue element (CQE) will be built for each console that is to receive this message. A search will be made of the unit control module entry (UCME) control blocks for the first console that is to receive that message. An attempt will then be made to send the message to that console. If the attempt is successful and that is the only console to receive the message, then control is returned to step 2. If the attempt is successful and there are other consoles to receive the same message, the console output pending bit is turned on and control is returned to step 2. If the attempt was unsuccessful, for example the console is busy, the console output pending bit is turned on and control is returned to step 2; a check is not made for a second console for multiple console messages.		WRWTO
h. There are a few system functions, such a task termination, that modify communication task control blocks. If a write queue element (WQE) is marked for deletion during the execution of one of these system functions, the UCMSYSI bit is set. The communication task will eliminate these WQEs as a result of this bit being set.		WRCLNUP
i. Delete Operator message indicates that a DOM macro instruction has been issued to delete a WTOR message that the console operator has not responded to *or* to delete a WTO message from the message display of a graphic console.		WRDOM
j. Write NIP messages to buffer indicates that the NIP messages stored during NIP can now be written.		PXFOXL
3 The wait service routine can only reach this point when there is no work to be performed by the communication task. As each function is performed, control returns to step 2. Having reached this point, control is returned to step 1 where a wait macro instruction will be issued causing the communication task to go into wait state until the next communication task ECB is posted.		

Diagram 1-3. Opening a Console (Part 1 of 4)

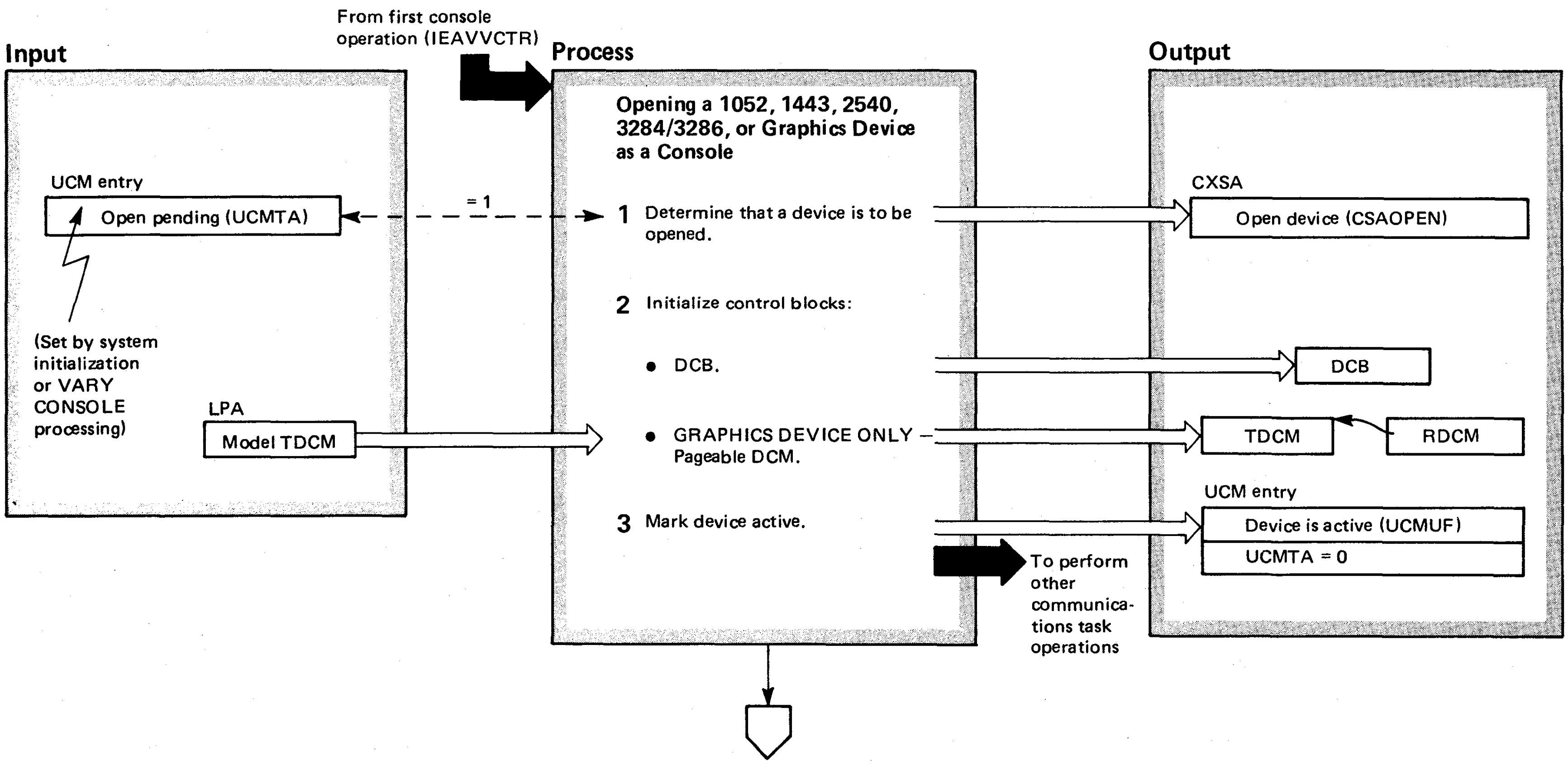


Diagram 1-3. Opening a Console (Part 2 of 4)

Extended Description	Module	Label
Before the communications task performs a console operation, it finds out if the console is open. The first time that a console operation is requested, the console will not be open. The communications task must open the console prior to performing the console operation. The communications task opens the following devices as consoles (the corresponding communications task module that performs the open processing is shown to the right):		
• 1052 printer-keyboard, 3210 console printer-keyboard, 3215 console printer-keyboard, and 3213 console printer.	IEAV1052	PJOPEN
• 1403 printer, 1443 printer, and 3211 printer.	IEAV1443	PJOPEN
• 2501 card reader, 2520 card reader punch, 2540 card reader punch, 3505 card reader, and 3525 card punch.	IEAV2540	PJOPEN
• 3284 printer and 3286 printer.	IEECVETW	
• 2740 communication terminal.	IEEC2740	OPEN
• 2250 display unit, 2260 display station, 3066 system console, 3277 display station, and Model 158 console.	IEECVETG	

Extended Description	Module	Label
Opening a 1052, 1443, 2540, 3284/3286, or Graphics Device as a Console		
1 During system initialization, NIP sets the open-pending bit (UCMTA) in a console's UCM entry to indicate that the console must be opened by the communications task. In the same manner, VARY CONSOLE processing sets the open-pending bit (UCMTA) in a console's UCM entry when a console is defined in response to a VARY CONSOLE command. When the communications task determines that the open-pending bit is on, it sets the open bit (CSAOPEN) in the CXSA.		
2 The communications task initializes control blocks for the device:		
• The communications task initializes the data control block (DCB) for the device.		
• For a graphics device that is being opened as a console, DIDOCS issues a GETMAIN macro instruction for space for the pageable display control module (TDCM). To initialize the TDCM, DIDOCS uses the model TDCM in the link pack area. DIDOCS chains the TDCM to the resident DCM (RDCM).	IEECVETG	
3 The communications task sets bit UCMUF in the device's UCM entry to indicate that the device is active.		
EXIT After the communications task opens the console, it performs the console operation for which it received control initially.		

Diagram 1-3. Opening a Console (Part 3 of 4)

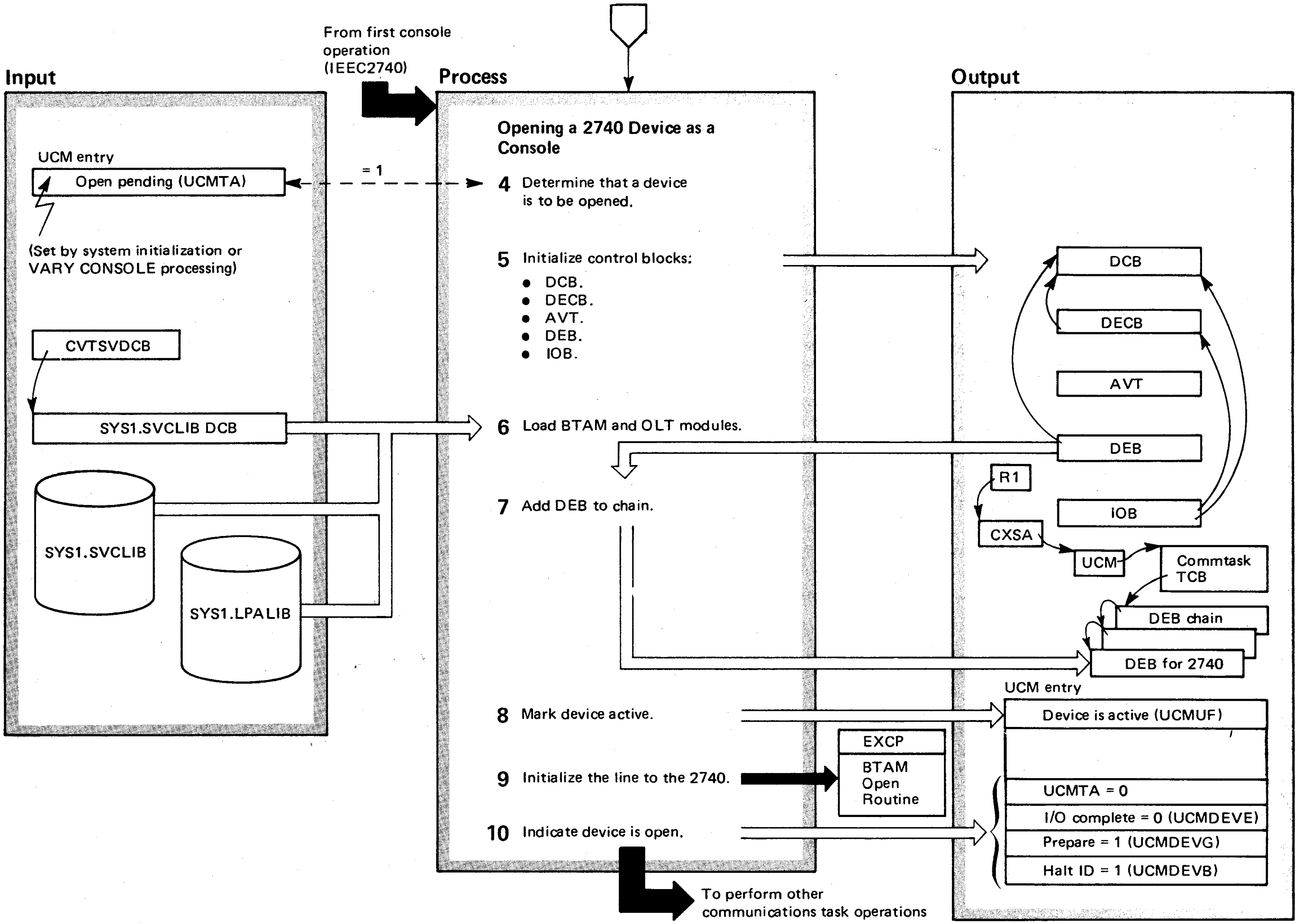


Diagram 1-3. Opening a Console (Part 4 of 4)

Extended Description	Module	Label
Opening a 2740 Device as a Console		
4 During system initialization, NIP sets the open-pending bit (UCMTA) in a console's UCM entry to indicate that the console must be opened by the communications task. In the same manner, VARY CONSOLE processing sets the open-pending bit (UCMTA) in a console's UCM entry when a console is defined in response to a VARY CONSOLE command. The 2740 device support processor (DSP) finds the open-pending bit on and determines that it must open the 2740 communications terminal as a console.	IEEC2740	IGCXX07B
5 The 2740 DSP initializes and chains control blocks for the 2740 console: • Data control block (DCB). • Data event control block (DECB). • Appendage vector table (AVT). • Data extent block (DEB). • Input/output block (IOB). The 2740 DSP contains models of the above control blocks.	IEEC2740	OPEN
6 The 2740 DSP uses the basic telecommunications access method (BTAM). The 2740 DSP loads the following BTAM modules from SYS1.SVCLIB: • IGG019MA – BTAM read/write module. • IGG019MB – BTAM channel end appendage. The 2740 DSP loads the following modules from SYS1.LPALIB: • IGG019M0 – BTAM device I/O module. • IGG019MR – OLT control module.		
7 The 2740 DSP adds the 2740 DEB to the DEB chain pointed to by the communications task TCB.	IEEC2740	OPEND
8 The 2740 DSP sets bit UCMUF in the 2740's UCM entry to indicate that the device is active.		
9 The 2740 DSP creates a channel program to initialize the line to the 2740, then issues an EXCP to execute the channel program.		
10 To indicate that the device is open, the 2740 DSP sets the following bits in the 2740's UCM entry: • Sets the open-pending bit (UCMTA) off. • Sets the I/O complete bit (UCMDEVE) off. • Sets the prepare bit (UCMDEVG) on. • Sets the HALTIO bit (UCMDEVB) on.	IEEC2740	OPENEND
EXIT After the 2740 DSP opens the console, it performs the console operation for which it received control initially.		

Diagram 1-4. Closing a Console (Part 1 of 4)

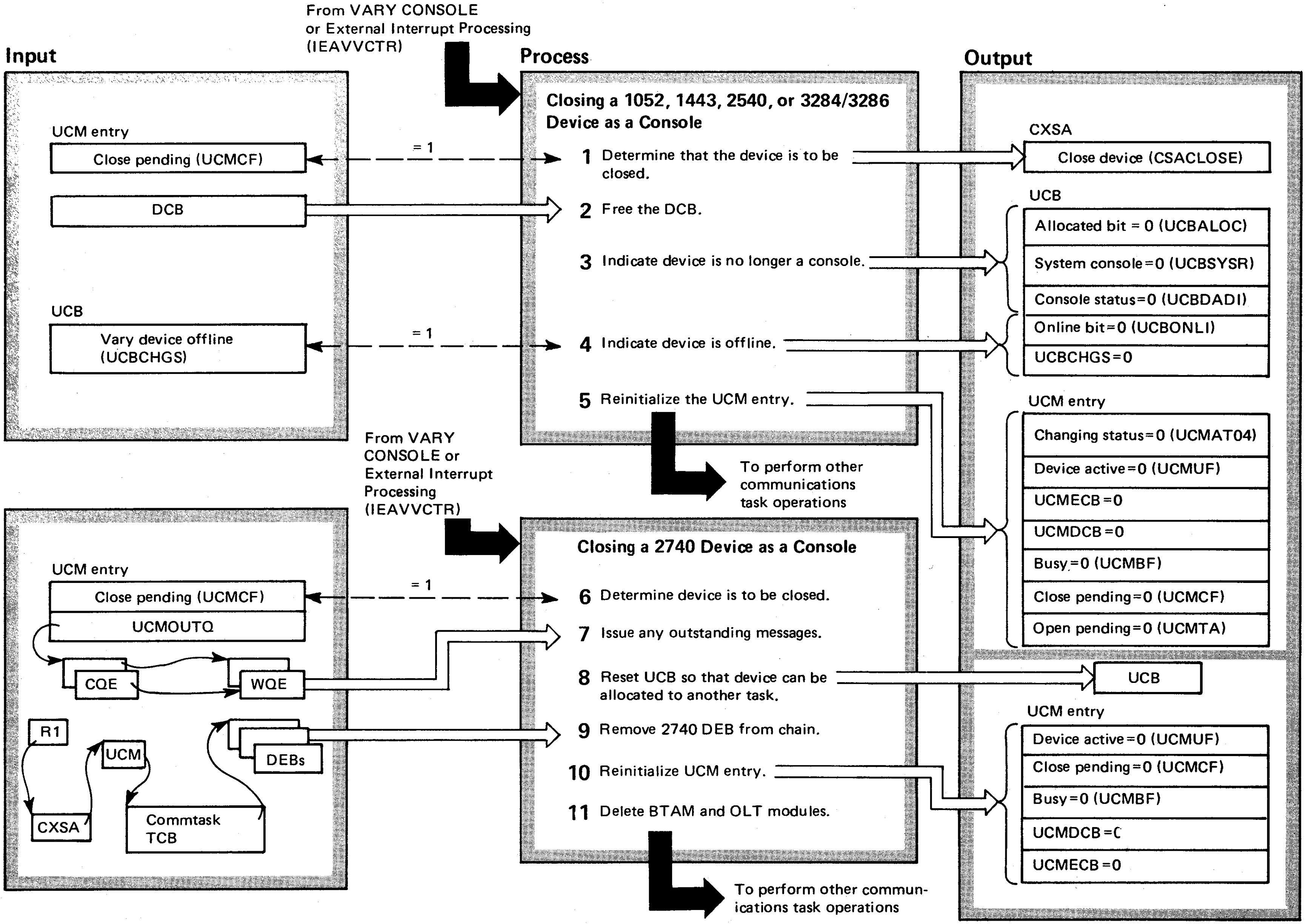


Diagram 1-4. Closing a Console (Part 2 of 4)

Extended Description	Module	Label
When a device is removed from console status in response to a VARY CONSOLE command, VARY CONSOLE processing sets the close-pending bit (UCMCF) in the console's UCM entry. In the same manner, when a device is removed from console status during console switch processing, the console switch routine sets the close-pending bit in the console's UCM entry. This bit indicates that the communications task must close the console. The communications task passes control to the appropriate device support processor to perform the close operation:		
• IEAV1052 for the 1052 printer-keyboard, 3210 console printer-keyboard, 3215 console printer-keyboard, and 3213 console printer.	IEAV1052	PJCLOSE
• IEAV1443 for the 1443 printer, 1403 printer, and 3211 printer.	IEAV1443	PJCLOSE
• IEAV2540 for the 2540 card reader punch, 2501 card reader, 2520 card reader punch, 3505 card reader, and 3525 card punch.	IEAV2540	PJCLOSE
• IEEC2740 for the 2740 communication terminal.	IEEC2740	CLOSE
• IEECVETW for the 3284/3286 printer.	IEECVETW	
• IEECVETG for a graphics console.	IEECVETG	
Closing a 1052, 1443, 2540, or 3284/3286 Device as a Console		
1 When the communications task determines that a close operation is pending, it sets the close bit (CSACLOSE) in the CXSA. Before the console is closed, all pending work is quiesced.	IEAVVCTR	
2 The appropriate device support processor (DSP) frees the device's DCB.	(See above)	
3 The DSP sets to zero the following bits in the device's UCB: • Allocated bit (UCBALOC). • System console bit (UCBSYSR). • Console status change bit (UCBDADI).		
4 The DSP checks bit UCBCHGS to determine whether the device is to be offline. If the bit is on, the DSP sets the online bit (UCBONLI) to zero to indicate that the device is offline.		

Extended Description	Module	Label
5 The DSP resets the following UCM entry fields to zero: • Changing status bit (UCMAT04). • Device active bit (UCMUF). • ECB (UCMECB). • DCB address (UCMDCB). • Busy bit (UCMBF). • Close-pending bit (UCMCF). • Open-pending bit (UCMTA).		
Closing a 2740 Device as a Console		
6 The 2740 DSP determines from bit UCMCF that the 2740 device is to be closed.	IEEC2740	IGCXX07B
7 The 2740 DSP issues any messages that are on the device's message queue (WQE queue).	IEEC2740	CLOSA
8 The 2740 DSP sets fields in the UCB so that the 2740 device can be allocated to another task.	IEEC2740	CLOSF
9 The 2740 DSP removes the DEB for the 2740 from the DEB chain pointed to by the communications task TCB. The 2740 DSP initialized and chained this DEB during device open processing (see steps 5 and 7 of "Opening a Console").	IEEC2740	CLOSJ
10 The 2740 sets the following UCM entry fields to zero: • Device active bit (UCMUF). • Close-pending bit (UCMCF). • Busy bit (UCMBF). • DCB address (UCMDCB). • ECB (UCMECB).	IEEC2740	CLOSJ
11 Finally, the 2740 DSP deletes the BTAM modules that it loaded during opening of the console (see step 6 of "Opening a Console"): • IGG019MA – BTAM read/write module. • IGG019MB – BTAM channel end appendage. • IGG019M0 – BTAM device I/O module. The 2740 DSP also deletes the OLT control module (IGG019MR).	IEEC2740	CLOSJ

Diagram 1-4. Closing a Console (Part 3 of 4)

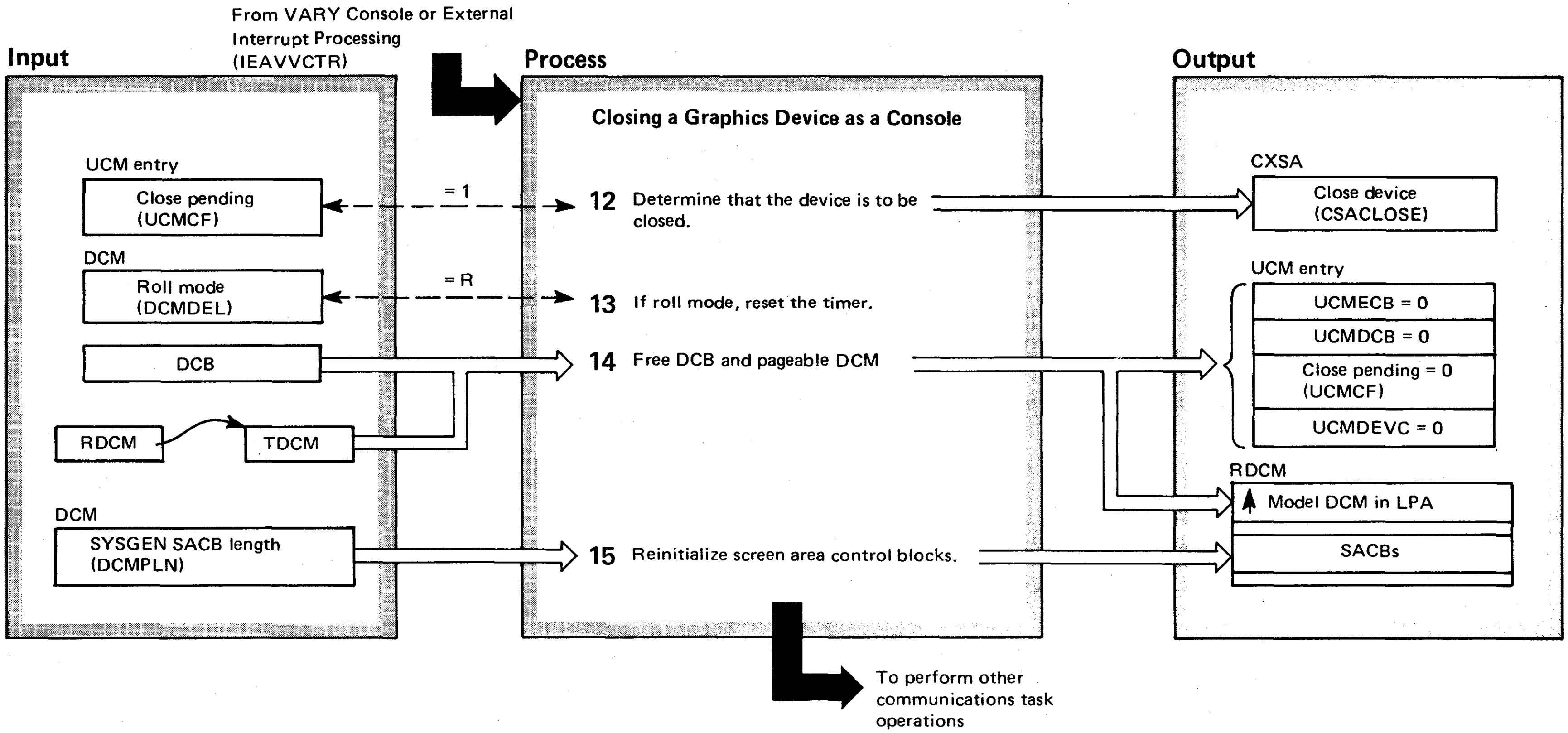


Diagram 1-4. Closing a Console (Part 4 of 4)

Extended Description	Module	Label
Closing a Graphics Console		
12 When DIDOCS determines that a close operation is pending, it sets the close bit (CSACLOSE) in the CXSA.	IEECVET1	
13 If the console is in roll mode (field DCMDEL contains the character "R"), DIDOCS resets the timer.	IEECVETK	
14 DIDOCS frees the device's DCB. Then DIDOCS frees the pageable DCM (TCDM) and places a pointer to the model TDCM in the resident DCM. DIDOCS sets the device inactive by placing zeros into the following fields of the device's UCM entry: • I/O completion ECB (UCMECB). • DCB address (UCMDCB). • Close-pending bit (UCMCF). • Control flags (UCMDEVC).	IEECVETG	
15 DIDOCS reinitializes the length of the screen area control blocks to the SYSGEN-specified length; the SYSGEN-specified length is in field DCMPLN of the resident DCM. DIDOCS issues a FREEMAIN macro instruction to free any SACBs that it obtained using GETMAIN.	IEECVFTG	

Diagram 1-5. WTO and WTOR Macro Instruction Processing Overview (SVC 35) (Part 1 of 2)

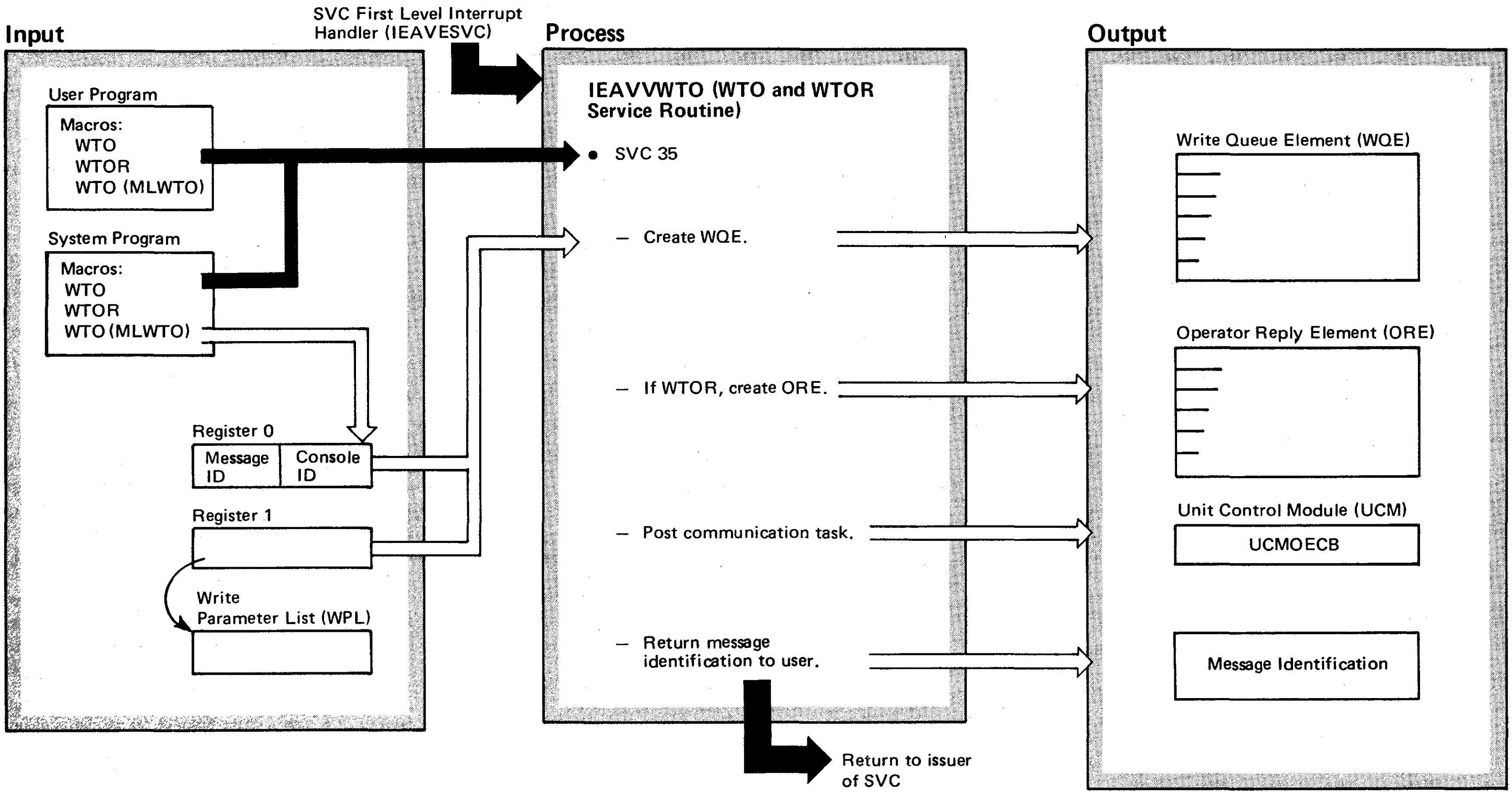


Diagram 1-5. WTO and WTOR Macro Instruction Processing Overview (SVC 35) (Part 2 of 2)

Extended Description	Module
This function is for Write-to-Operator (WTO) or Write-to-Operator with Reply (WTOR) requests. Issuing a WTO or WTOR results in a SVC 35 that builds the associated Write Queue Elements (WQEs) and Operator Reply Elements (OREs) to pass messages to the console operator.	

Diagram 1-6. WTO and WTOR Macro Instruction Processing (SVC 35) (IEAVVWTO) (Part 1 of 20)

From SVC First Level Interrupt Handler (IEAVESVC)

Input

Write Parameter List (WPL)

Process

1 The following table shows the subroutines and by whom they are called for the WTO and WTOR Service Routine:

Caller	Called Subroutines					
	SETLOCK	FREELOCK	VWTOGETB	VWTOFORE	VWTOWAIT	VWTOLREC
Main Seg.	X	X				
GETBLKS			X	X	X	
VWTOGETB						X
VWTOWAIT	X	X				
VWTOLREC					X	

Diagram 1-6. WTO and WTOR Macro Instruction Processing (SVC 35) (IEAVVWTO) (Part 2 of 20)

Extended Description **Module**

Mainline Routine: IEAVVWTO

Provides an interface with WTP, JES2 and MLWTO. It does the processing for a single line WTO (but not MLWTO). It also does the Write-To-Operator portion of a WTOR macro instruction.

Subroutines:

Main Segment (steps 1-20)
Control handling and processing of the write parameter list (WPL) and checks for an error return from any subordinate segment.

SETESTAE
Sets the ESTAE in place and initializes the audit trail.

SETUPXSA
Initializes the extended save area. This area is used for quick reference to various parts of the write parameter list. (The WPL is a variable length parameter list which makes it inconvenient to reference frequently.)

VALIDCHK
Checks the validity of the user's parameter list.

DECLARES
Defines variables and control blocks.

VWTOGETB
Allocates space for the WQE and ORE control blocks.

GETID
Obtains a reply identification and places it in the ORE control block.

Extended Description **Module**

VWTOFORE
Frees the ORE control block when the associated WQE control block can not be obtained.

BUILDORE
Fills in the ORE control block.

BUILDWQE
Fills in the WQE control block.

USEREXIT
Sets up for the calling of the WTO user exit routine (IEECVCTE) and calls that routine.

SETLOCK
Gets the local and CMS locks and sets a functional recovery routine (FRR).

FREELOCK
Frees the FRR, CMS lock, and local lock.

GETBLKS
Obtains the control blocks for the write queue element (WQE) and operator reply elements (ORE).

HASPEXIT (No Documentation)
Message alteration exit to a subsystem. In Release 2, the only applicable subsystem is JES2.

VWTOWAIT (step 15)
Wait for either WQE or ORE control block to be freed.

FREESTAE
Frees the ESTAE routine.

VWTOLREC
Signals the communication task to initialize the system log by posting the unit control module (UCMAECB) event control block.

VWTOCLNP
Handles clean up when an error has occurred during processing.

Diagram 1-6. WTO and WTOR Macro Instruction Processing (SVC 35) (IEAVVWTO) (Part 3 of 20)

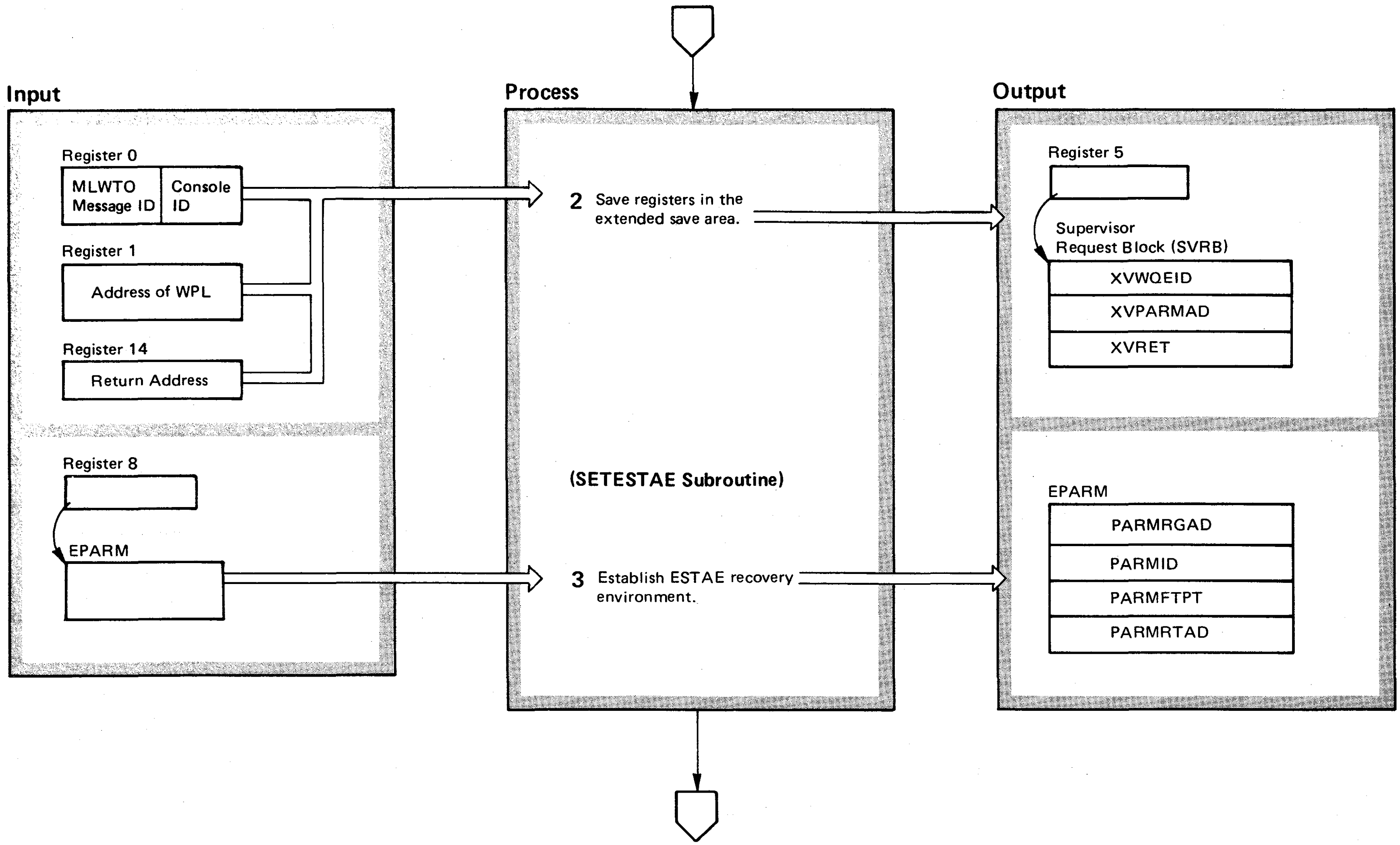


Diagram 1-6. WTO and WTOR Macro Instruction Processing (SVC 35) (IEAVVWTO) (Part 4 of 20)

Extended Description	Module	Label
2 Registers 0, 1 and 14 are saved in XVWQEID, XVPARMAD and XVRET respectively in the extended save area (XV) of the supervisor request block (SVRB). Register 0 is used when the system adds additional multiple line WTO messages to a previous string of existing messages; it contains the message identification of the original message. Register 0 also contains the UCMID for any program specifying REG0 and for privileged programs specifying QREG0.	IEAVVWTO	MAIN
3 If an error should occur, the ESTAE macro instruction ensures that queues and data areas will be cleaned up. The ESTAE parameter list is as follows:		SETESTAE

PARMRGAD	The address of the register save area.
PARMID	The four character module identifier (VWTO).
PARMFTPT	A code that identifies the subroutine that is currently processing. Should an error occur, this code indicates in which subroutine the error occurred and where the clean up needed.
PARMRTAD	A retry address. It is periodically updated to permit execution retry, and to clean up queues and data areas before returning to the caller.

Diagram 1-6. WTO and WTOR Macro Instruction Processing (SVC 35) (IEAVVWTO) (Part 5 of 20)

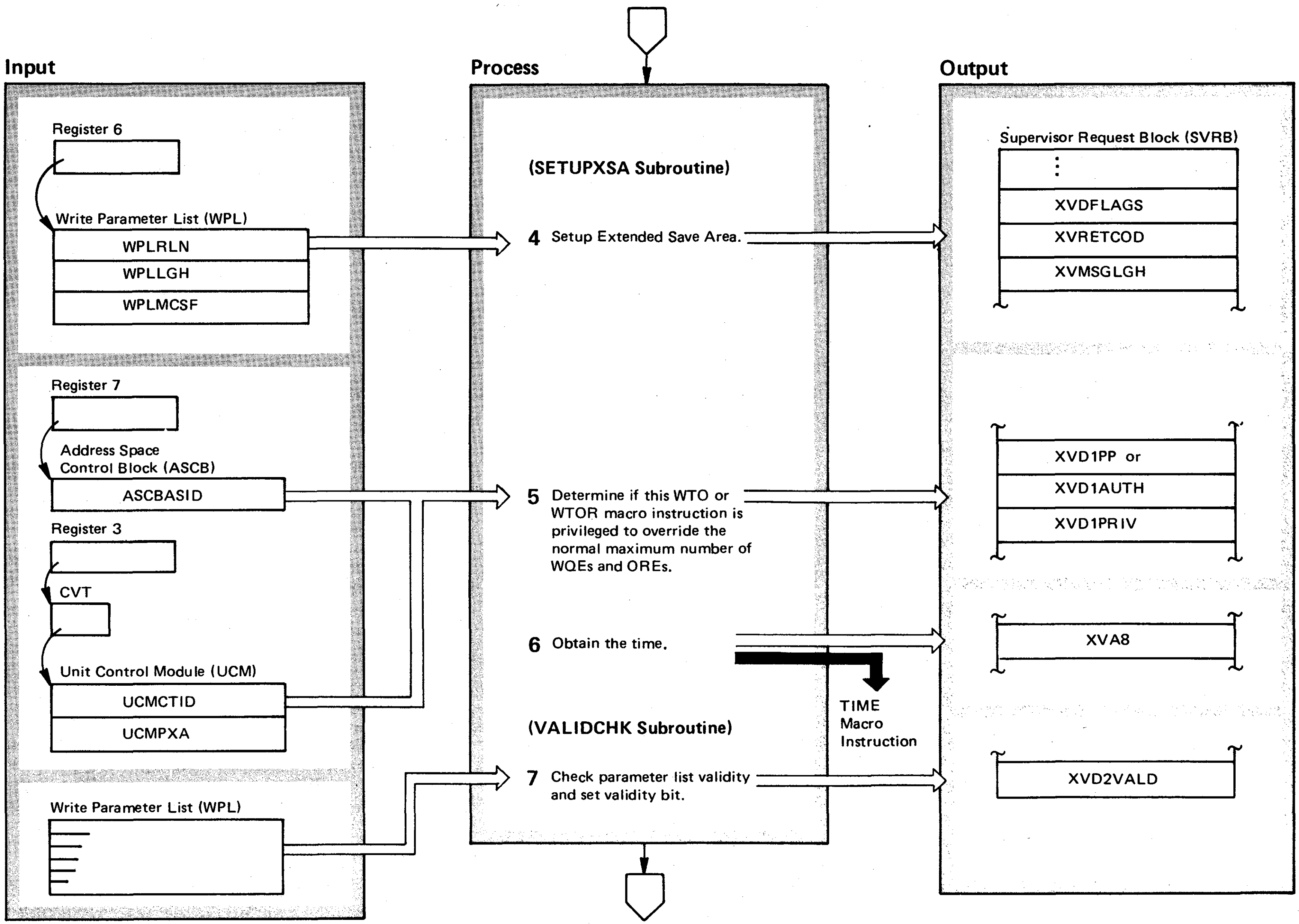
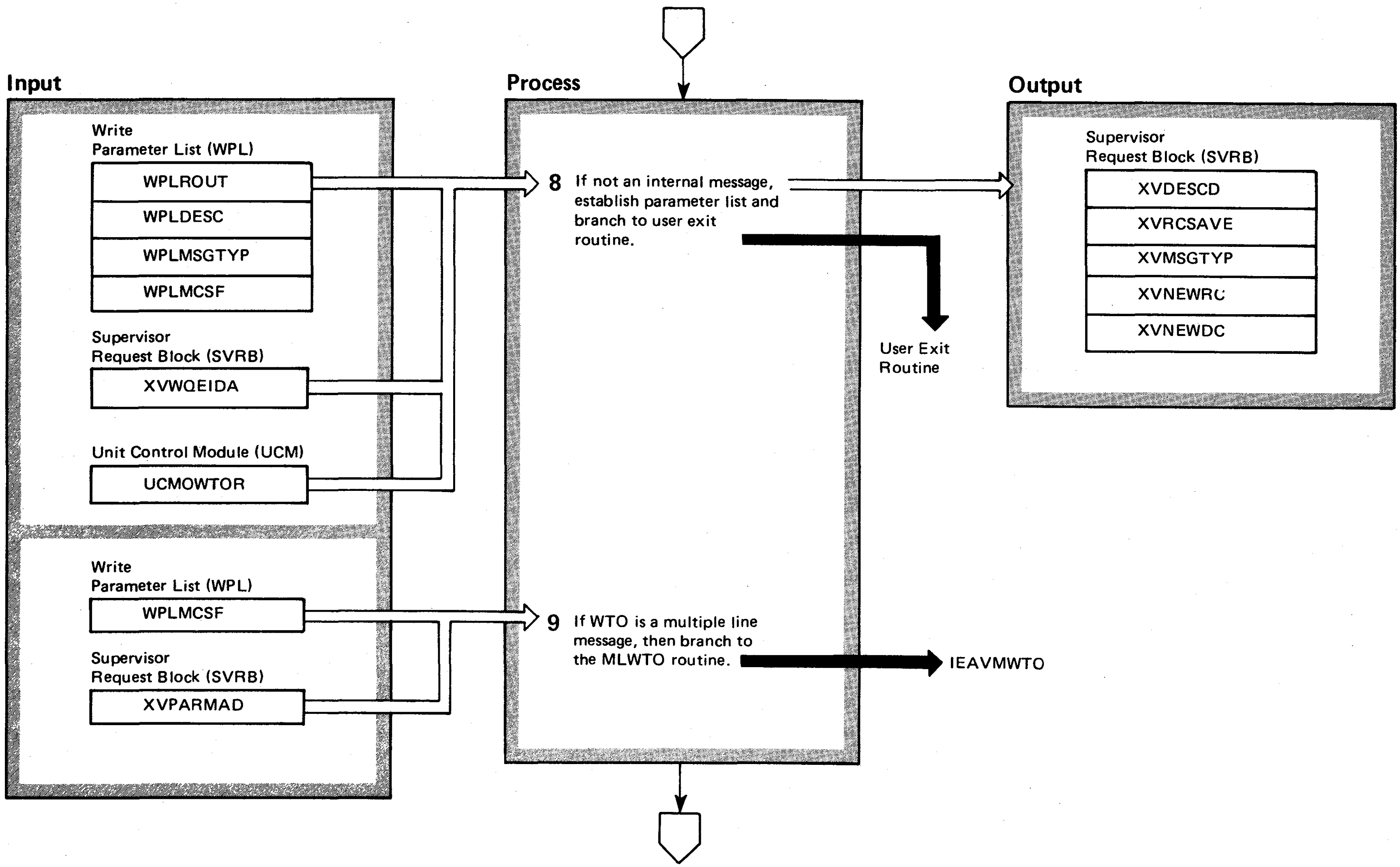


Diagram 1-6. WTO and WTOR Macro Instruction Processing (SVC 35) (IEAVVWTO) (Part 6 of 20)

Extended Description	Module	Label
4 Several fields are transferred from the user's write parameter list to the extended save area of the supervisor request block.		SETUPXSA
5 A TESTAUTH macro instruction is issued to determine if the user who issued the WTO or WTOR macro instruction is a problem program (not in protect key 0-7 or supervisor state). If the user is a problem program, a second TESTAUTH macro instruction is issued to determine if the user is authorized by the authorized program facility (APF). If the user is not authorized, then the request is marked unauthorized. A check then determines if the user is privileged. The communication task or any task running under a system interrupt request block (SIRB) is privileged to exceed the normal system limit on the number of available WQEs and OREs.		SETUPXSA
6 The time is obtained (by using the TIME macro instruction) and placed in the extended save area of the supervisor request block (SVRB).		SETUPXSA
7 The user's write parameter list (WPL) is validity checked for: • Incompatible options; for example, a multiple line (MLWTO) message as part of a WTOR macro instruction. • Being entirely within the user's addressable storage space. This check is accomplished by issuing the MODESET macro instruction to change this routine's protect key to the user's protect key, and then referencing the beginning and ending of the user's write parameter list. If either reference is outside the user's addressable storage space, an addressing error causes an abnormal termination. The abnormal termination is eventually processed by the functional recovery routine (FRR) which returns control to this routine. This routine then issues a second MODESET macro instruction to return this routine to its regular protect key. • Whether the WTOR is on a fullword boundary. Failure to pass any of these checks causes the message to be marked invalid; the user eventually receives a D23 ABEND from step 20.		VALIDCHK

Diagram 1-6. WTO and WTOR Macro Instruction Processing (SVC 35) (IEAVVWTO) (Part 7 of 20)



Extended Description	Module	Label
8 If the write parameter list is valid, if the process is not attaching additional message lines to an existing multiple line message (MLWTO), and if the message is not an internal message (authorized and any MSGFLAGS 2-8), then a copy of the message, routing codes, and descriptor codes are passed to a user exit routine. The exit routine may alter the routing and descriptor codes, flag the message for deletion, and examine the message text, but alterations to the message text are ignored. If a hardcopy log exists, deleted WTO messages are sent to the log. Deleted WTOR messages are always sent to the master console. The user exit routine branches to step 9.		USEREXIT
9 If the WTO is a multiple line message request, then this routine branches to the multiple line service module (IEAVMWTO) to process the WTO request.		MAIN

Diagram 1-6. WTO and WTOR Macro Instruction Processing (SVC 35) (IEAVVWTO) (Part 9 of 20)

Input

Supervisor Request Block (SVRB)

- XVMSGLGH
- XVDFLAGS

Supervisor Request Block (SVRB)

- XVD2RC

Write Parameter List (WPL)

- WPLROUT

Process

10 If the message length is zero, then:

a. For a WTOR message, ABEND. → D23 ABEND issuer

b. For a WTO message:

1) Set the message identifier to zero.

2) Return to caller. → Return to SVC First Interrupt Handler (IEAVVWTO)

11 If routing code 11, then:

a. Place UCM Base address in Register 10.

b. Branch to the Write to Programmer module. → WTP (IGC0203E)

Output

Supervisor Request Block (SVRB)

- XVWQEID

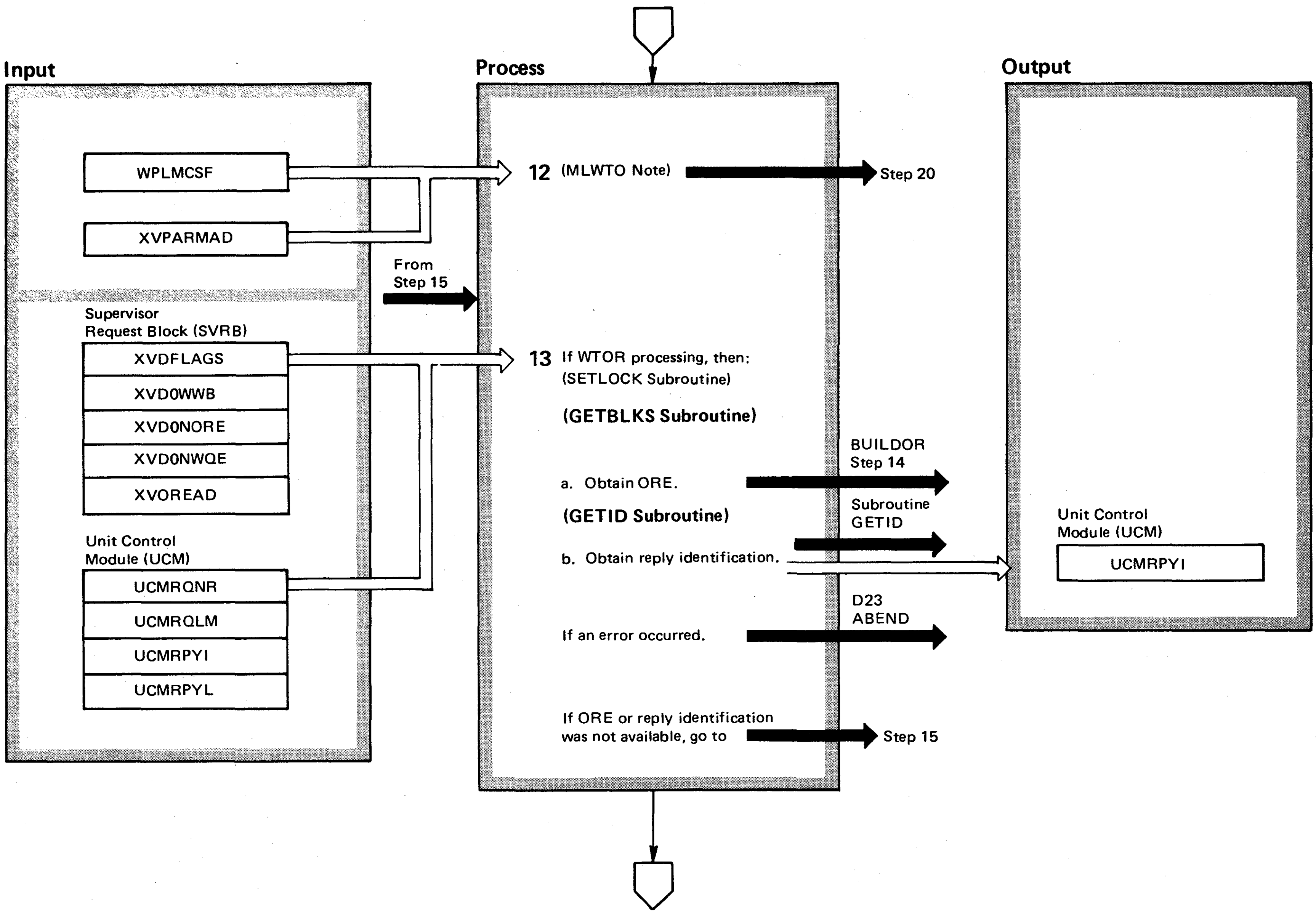
Register 10

Unit Control Module (UCM)

Diagram 1-6. WTO and WTOR Macro Instruction Processing (SVC 35) (IEAVVWTO) (Part 10 of 20)

Extended Description	Module	Label
10 The message text field is tested for a length of zero. If it is zero and this is a WTOR message, the user receives a D23 ABEND. If it is zero and this is a WTO message, the message identifier is set to zero (X'00') and control is returned to the user. The zero message identifier indicates to the user that the message was not sent to a console or hardcopy log.		MAIN
11 If routing code 11 was specified, a branch is made to the write to programmer module (IGC0203E).		MAIN

Diagram 1-6. WTO and WTOR Macro Instruction Processing (SVC 35) (IEAVVWTO) (Part 11 of 20)



Extended Description	Module	Label
12 This branch does not exist in the code. In executing steps 13-19, for one reason or another, a multiple line WTO (MLWTO) does not meet the conditions necessary to execute these steps.		MAIN
13 If this process was initiated by a WTOR macro instruction, then SVC 35 attempts to get the ORE and reply identification before getting the WQE. To serialize using the Communication Task resources (WQE, COUNT, REPLY IDs, ORE COUNT, etc.), the LOCAL and CMS locks are obtained by issuing the SETLOCK macro.		GETBLKS
a. If the number of OREs in use (UCMRQNR) is less than the number of OREs generally permitted (UCMRQLM), or if the WTOR user is privileged (as explained in step 5), the VWTOGETB subroutine is called to obtain the ORE.		GETID
b. If an ORE was obtained, UCMRPYI and UCMRPYL are used to search the 100-bit identification map for the first available reply identification. When found, the identification is placed in the ORE and the corresponding bit is turned on in the identification bit map.		
Any error causes a D23 user ABEND.		
If an ORE or reply identification was not available, control is passed to step 15.		

Diagram 1-6. WTO and WTOR Macro Instruction Processing (SVC 35) (IEAVVWTO) (Part 13 of 20)

Input

Unit Control Module (UCM)
- UCMWQNR
- UCMWQLM

Supervisor Request Block (SVRB)
- XVDFLAGS
- XVWQEAD

Supervisor Request Block (SVRB)
- XVDOWWB

Address Space Control Block (ASCB)
- ASCBASID

Process

(GETBLKS Subroutine)

14 Obtain a WQE.

Subroutine BUILDWQE Step 17

If an error occurred. — D23 ABEND

If a WQE was available, go to — Step 16

(VWTOWAIT Subroutine)

From Step 13

15 An ORE, reply identification, or WQE was unavailable for this user.

a. Free previously obtained ORE and reply identification. — Subroutine VWTOFORE

b. Wait for the next freeing of a required ORE or WQE. — Subroutine VWTOWAIT

If an error occurred. — D23 ABEND

When ORE or WQE is available, go to — Step 13

Output

Unit Control Module (UCM)
- UCMFLAGS
- UCMRPYL

Supervisor Request Block (SVRB)
- XVDOWWB
- XVWWB

WQE or ORE Write Wait Block (WWB)
- WWBASCB
- WWBTCBAD
- WWBFWDPT
- WWBBCKPT
- WWBFLAGS'
- WWBECB

Diagram 1-6. WTO and WTOR Macro Instruction Processing (SVC 35) (IEAVVWTO) (Part 14 of 20)

Extended Description	Module	Label
14 To obtain a WQE the number of WQEs in use (UCMWQNR) must be less than the number of WQEs generally permitted (UCMWQLM), or the user of the WTO or WTOR macro instruction must be privileged (as explained in step 5). The VWTOGETB subroutine is called to obtain the WQE. Any error causes a D23 user ABEND. If the WQE was obtained, control passes to step 16.	IEAVVWTO	GETBLKS
15 An ORE, a reply identification, or a WQE was not available for this WTO or WTOR macro instruction user. If this is a WTOR message and an ORE or reply identification was not available, subroutine VWTOFORE is called to free the previously obtained ORE. Subroutine VWTOWAIT is then called to create either a WQE or ORE write wait control block (WWB); this subroutine then waits for the WWBECB to be posted. Any error causes a D23 user ABEND. When the ORE or WQE is available, control is returned to step 13.		VWTOWAIT

Diagram 1-6. WTO and WTOR Macro Instruction Processing (SVC 35) (IEAVVWTO) (Part 15 of 20)

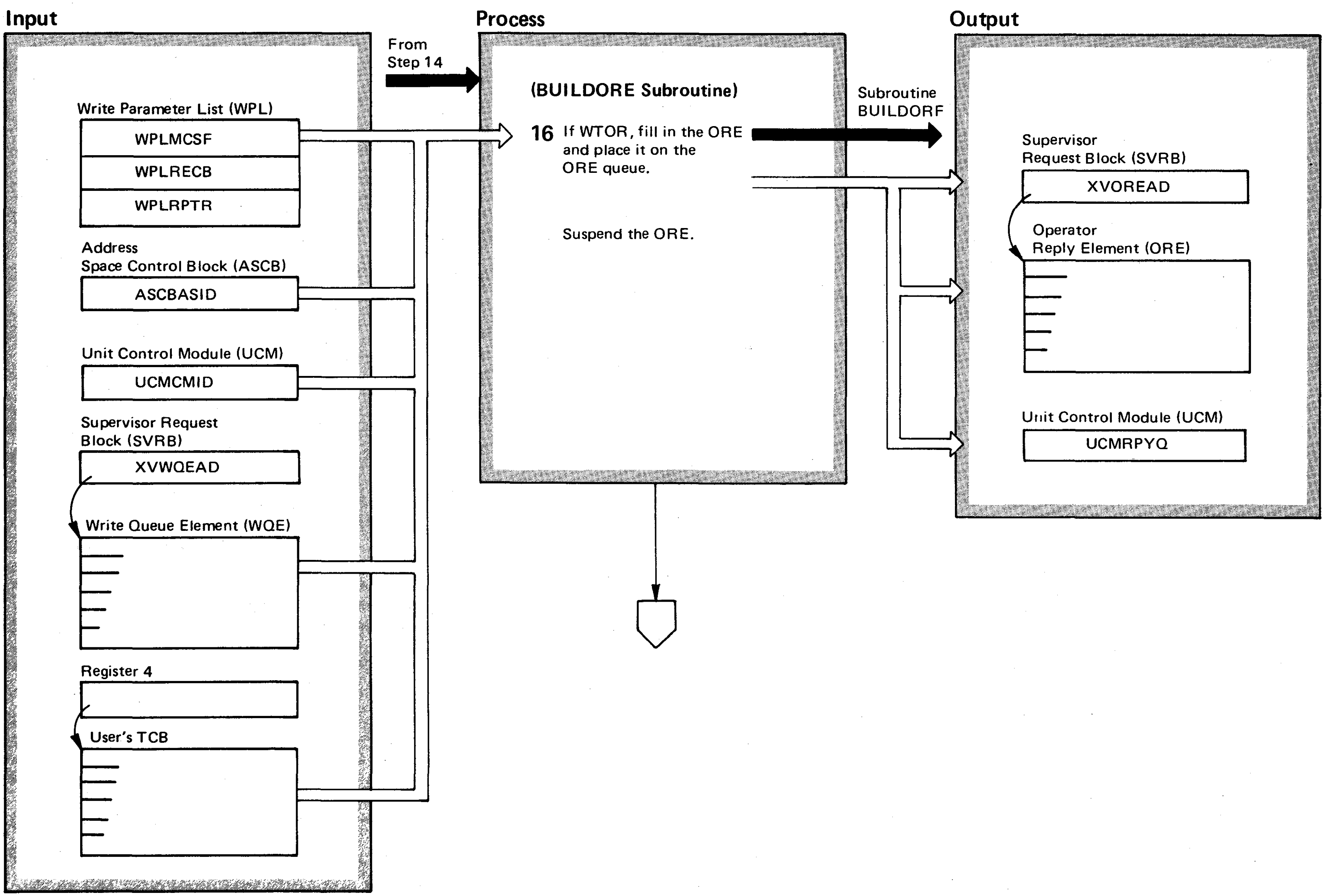


Diagram 1-6. WTO and WTOR Macro Instruction Processing (SVC 35) (IEAVVWTO) (Part 16 of 20)

Extended Description	Module	Label
16 If the processing is for a WTOR macro instruction, the ORE is always filled in before the WQE. The ORE contains the TCB address of the WTOR user, the address of the WQE associated with this ORE, and the address space identification (ASID) of the user's memory. The ORE is queued in the system ORE chain (UCMRPYQ) and marked temporarily suspended (ORESUSP). Suspend is removed after the subsystem exit routine has reviewed the message.	IEAVVWTO	BUILDORE

Diagram 1-6. WTO and WTOR Macro Instruction Processing (SVC 35) (IEAVVWTO) (Part 17 of 20)

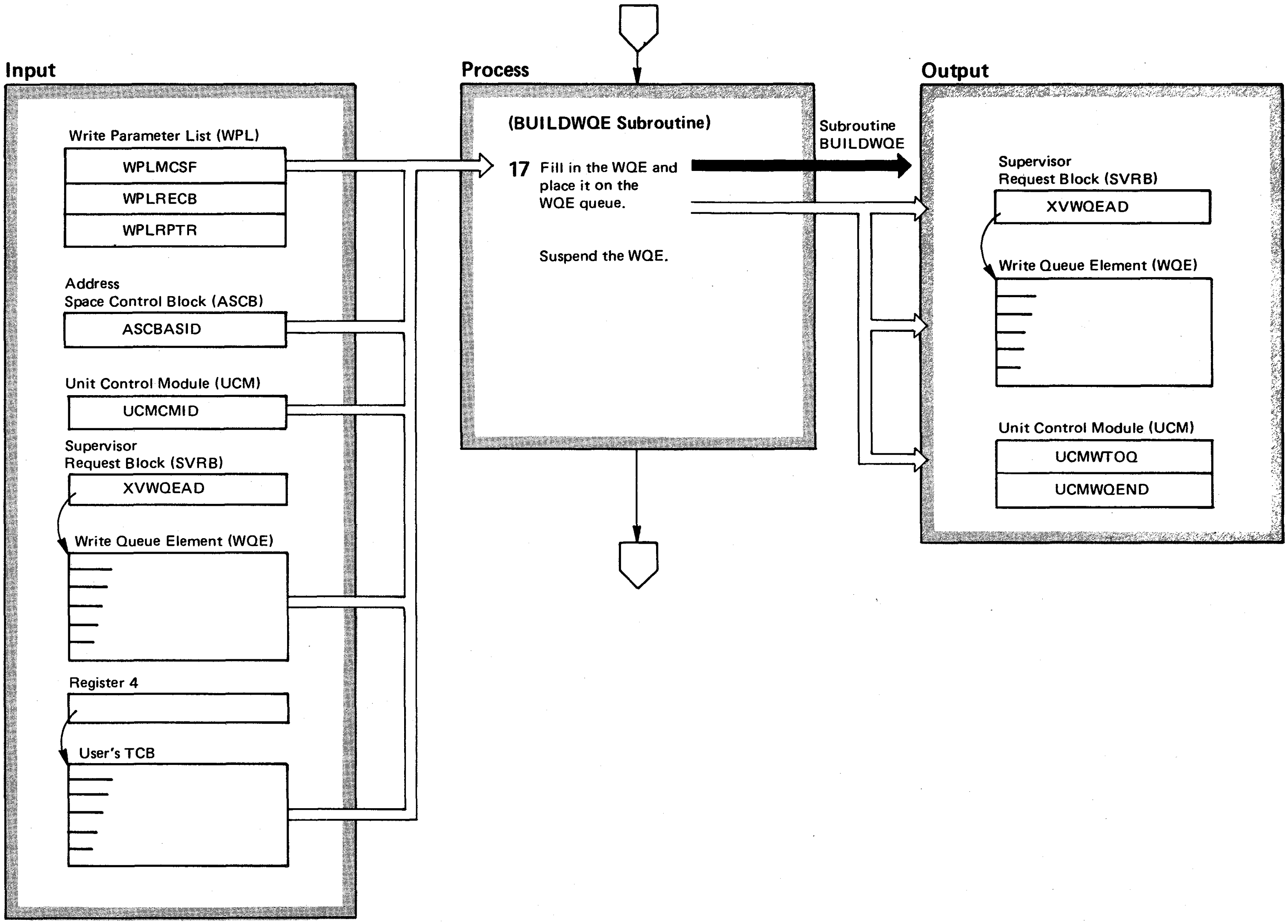
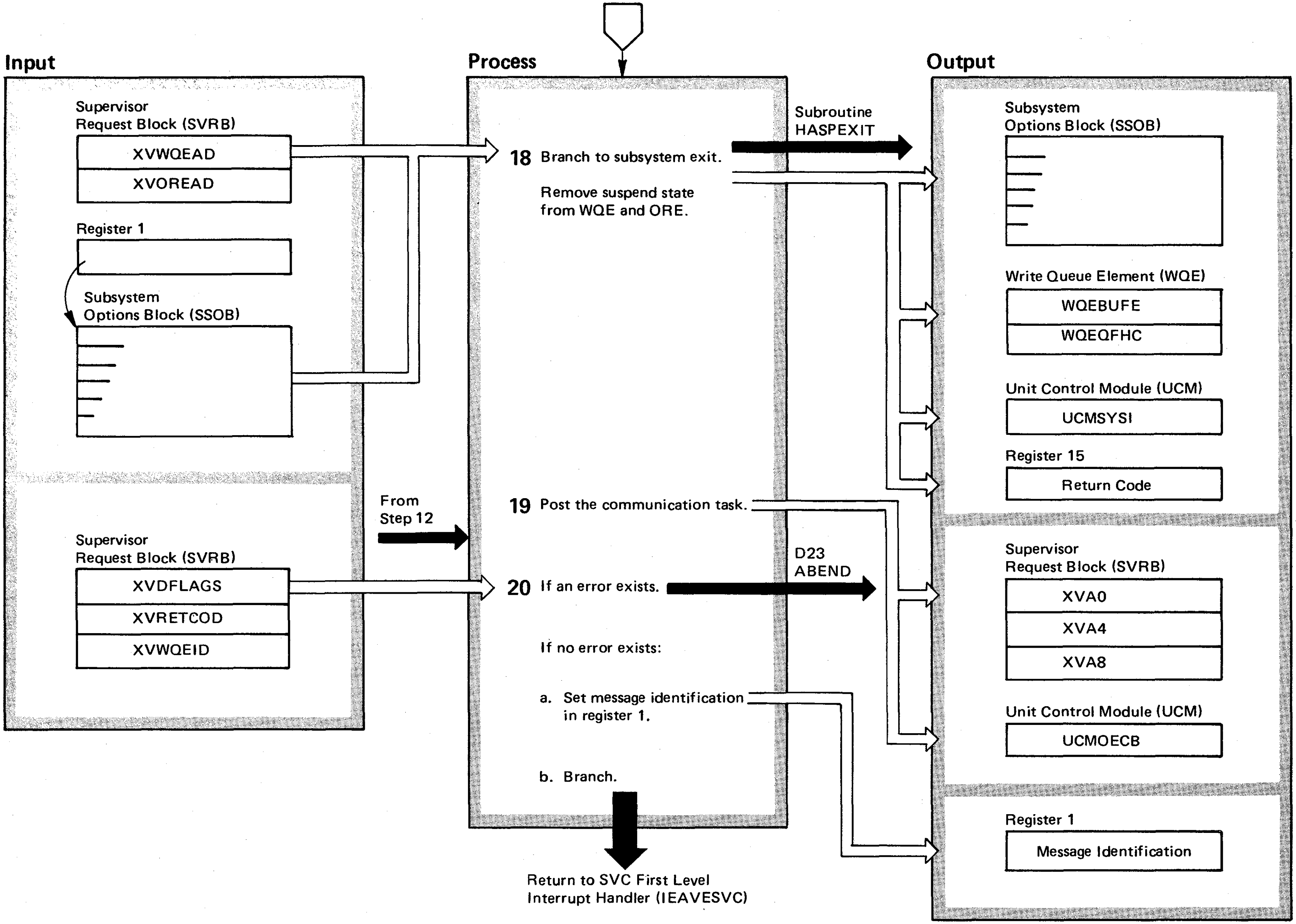


Diagram 1-6. WTO and WTOR Macro Instruction Processing (SVC 35) (IEAVVWTO) (Part 18 of 20)

Extended Description	Module
17 The WQE is filled in from the user's write parameter list (WPL) and the supervisor request block (SVRB). It is then placed on the system WQE chain, via UCMWTOQ and UCMWQEND, and marked temporarily suspended (WQESUSP). Suspend prevents the message from being displayed at the console until after the subsystem exit routine has reviewed the message.	IEAVVWTO

Diagram 1-6. WTO and WTOR Macro Instruction Processing (SVC 35) (IEAVVWTO) (Part 19 of 20)



Extended Description	Module
18 The WQE and ORE are passed to the job entry subsystem exit routine along with the subsystem options block (SSOB) and its extension (SSOBWT). The subsystem may alter the routing codes, descriptor codes and message text, or delete the message. A deleted message is sent to the hardcopy log. Upon return from the exit, the suspend states are removed from the WQE and ORE.	IEAVVWTO
19 Posting the communication task ECB (UCMOECB) indicates that the message is ready to be transmitted to a console and permits the communication task to be dispatched to transmit that message.	
20 If any error has been found in a parameter list or while processing, the user receives a D23 ABEND. Otherwise the workarea in subpool 231 is freed and this module branches back to the user with the message identification in Register 1. The ESTAE environment is cancelled.	

Diagram 1-7. Write-to-Programmer Processing Overview (IGC0203E) (Part 1 of 2)

Input

User or System Program

WTO Macro Instruction with Routing Code 11

Process

- If the WTO macro instruction was issued by a TSO terminal user:
  - a) If user wants messages sent to his terminal, then send message to terminal.
  - b) If he does not want messages sent to his TSO terminal, then set the return code.
- If the message was not sent to a TSO terminal, then send message to program's message data set.
- Check message operation and return to user.

From Recovery Termination Management

- Prepare mainline program for a retry operation.

Return to Recovery Termination Management

TPUT

PUT

Return to User

Output

Register 15

4

Extended Description

The write to programmer (WTP) message facility permits a message to be issued from a program and sent to the user of that program. For the program to send a WTP message to the user of that program, a WTO or WTOR macro instruction is issued with routing code 11. The WTP message is either placed in the system message data set defined by the user for this purpose or sent to the user's TSO terminal, provided the TSO user wants WTP messages sent to his terminal.

The WTO or WTOR message is processed by the WTO and WTOR macro instruction processing routine (IEAVVWTO). Each time a message is flagged with routing code 11, the WTO routine branches to the write to programmer routine. After the message has been processed, control is returned to the WTO routine.

Mainline Routine: IGC0203E – Write to Programmer Routine

This routine receives control from the WTO and WTOR macro instruction processing routine (IEAVVWTO) for those WTO and WTOR macro instructions that were issued with ROUTCDE=11. This routine consists of a series of subroutines that collectively perform the write to programmer (WTP) message processing.

Subroutines:

BUILDMSG
: This subroutine prepares error message IEF107I. This message is issued when the WTP routine is unable to send the WTP message to an appropriate device or TSO terminal. This error message contains all of the full words contained in the text of the first 53 bytes of the WTP message. If there are no blanks in the first 53 bytes to delimit words, the first 53 bytes are included in the error message.

Extended Description

BUILDRPL
: This subroutine obtains the location of the request parameter list (RPL) from the user's job step control block (JSCB) and fills in the necessary RPL fields.

CHECKJOB
: If this is the first WTP message for this job step, then this subroutine initializes the WTP area of the user's job step control block. Otherwise, this subroutine returns to the mainline code.

CHECKMSG
: This subroutine breaks messages that are longer than 126 bytes into multiple message lines of 126 bytes or less. An attempt is made to break the message lines between words.

CKMCSFLG
: This subroutine determines whether the WTP message will be sent to the hardcopy log or queued to a console.

CKROUTCD
: This subroutine is called when the WTP routine has failed to send the WTP message. If either the message has other routing codes or an operator console is receiving routing code 11 messages, the results of this subroutine cause the WTP routine to return to the WTO macro instruction processing routine where message processing continues. Otherwise, the results of this subroutine cause the WTO macro instruction processing routine to return to the WTO or WTOR macro instruction user with an indication that the message was not sent.

CKRETURN
: This subroutine checks the return codes upon return from the subsystem exit.

Extended Description

GETESTAE
: This subroutine builds a parameter list for ESTAE and then issues the ESTAE macro instruction.

ISSUEDEQ
: This subroutine builds the dequeue parameter list and issues a conditional dequeue macro instruction.

ISSUEENQ
: This subroutine checks the pointer to the request parameter list (RPL). If the pointer to the RPL exists, the enqueue parameter list is initialized and an unconditional enqueue macro instruction is issued to serialize the writing of the WTP message to the user's system message data set for this job. If the RPL pointer is zero, error message IEF107I will be issued to the hardcopy log with a message identification of '1'.

ISSUEMSG
: This subroutine issues error message IEF107I to the hardcopy log using a WTO macro instruction. This error message was prepared by the BUILDMSG subroutine.

ISSUTPUT
: When the WTP message is for an active TSO terminal user and that terminal user wants to receive his WTP messages at his terminal, this routine issues the TPUT macro instruction.

LOADREGS
: This subroutine is entered only during STAE retry processing. It restores the necessary registers for the mainline WTP routine before it returns control to the WTO and WTOR macro instruction processing routine.

STAE000
: This is the STAE exit subroutine; it receives control only when an ABEND situation occurs.

Diagram 1-8. Write-to-Programmer Processing (IGC0203E) (Part 1 of 22)

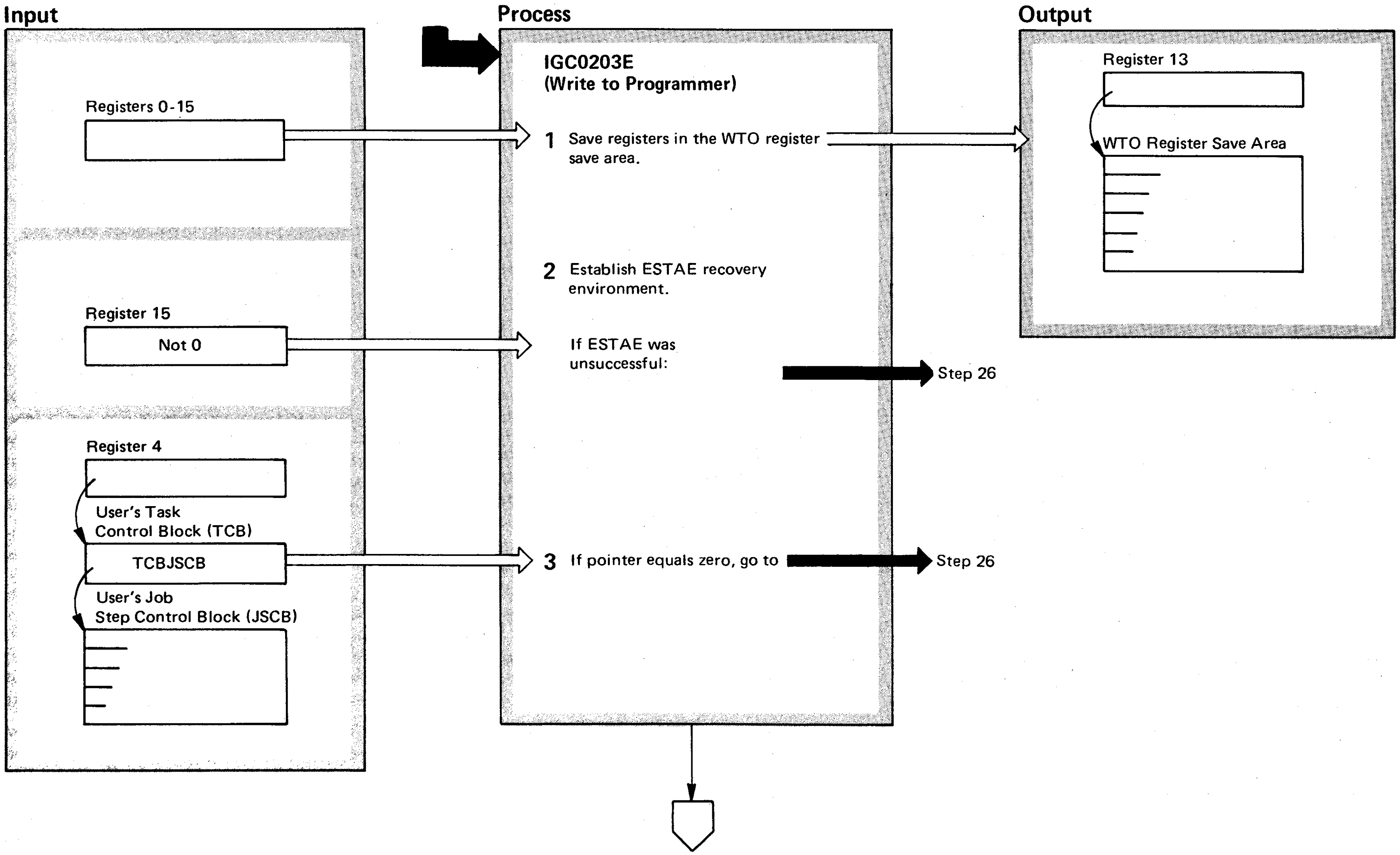


Diagram 1-8. Write-to-Programmer Processing (IGC0203E) (Part 2 of 22)

Extended Description	Module
1 All registers are saved in the save area of the WTO and WTOR macro instruction processing routine. The address of this save area was in register 13 when control was given to this routine. All of these registers are restored before control is returned to the WTO routine.	IGC0203E
2 The establishment of the ESTAE recovery environment ensures that if there is a WTP abnormal termination, the queues and data areas will be cleaned up, and an attempt will be made to restore the communication task to full operation. If the STAE exit routine is entered while a write-to-programmer (WTP) message is being processed, the STAE exit routine issues message IEF107I to the hardcopy log. This message contains the first 53 bytes of the write-to-programmer message. When control is returned to this routine from the ESTAE and if register 15 has been set to zero, the ESTAE was successful.	
3 The active job step control block (JSCBACT) contains a pointer to the user's job step control block (JSCB). The user's job step control block is needed to obtain the pointer to the request parameter list (RPL). If the pointer to the user's JSCB is zero, the write-to-programmer facility cannot be performed.	

Diagram 1-8. Write-to-Programmer Processing (IGC0203E) (Part 3 of 22)

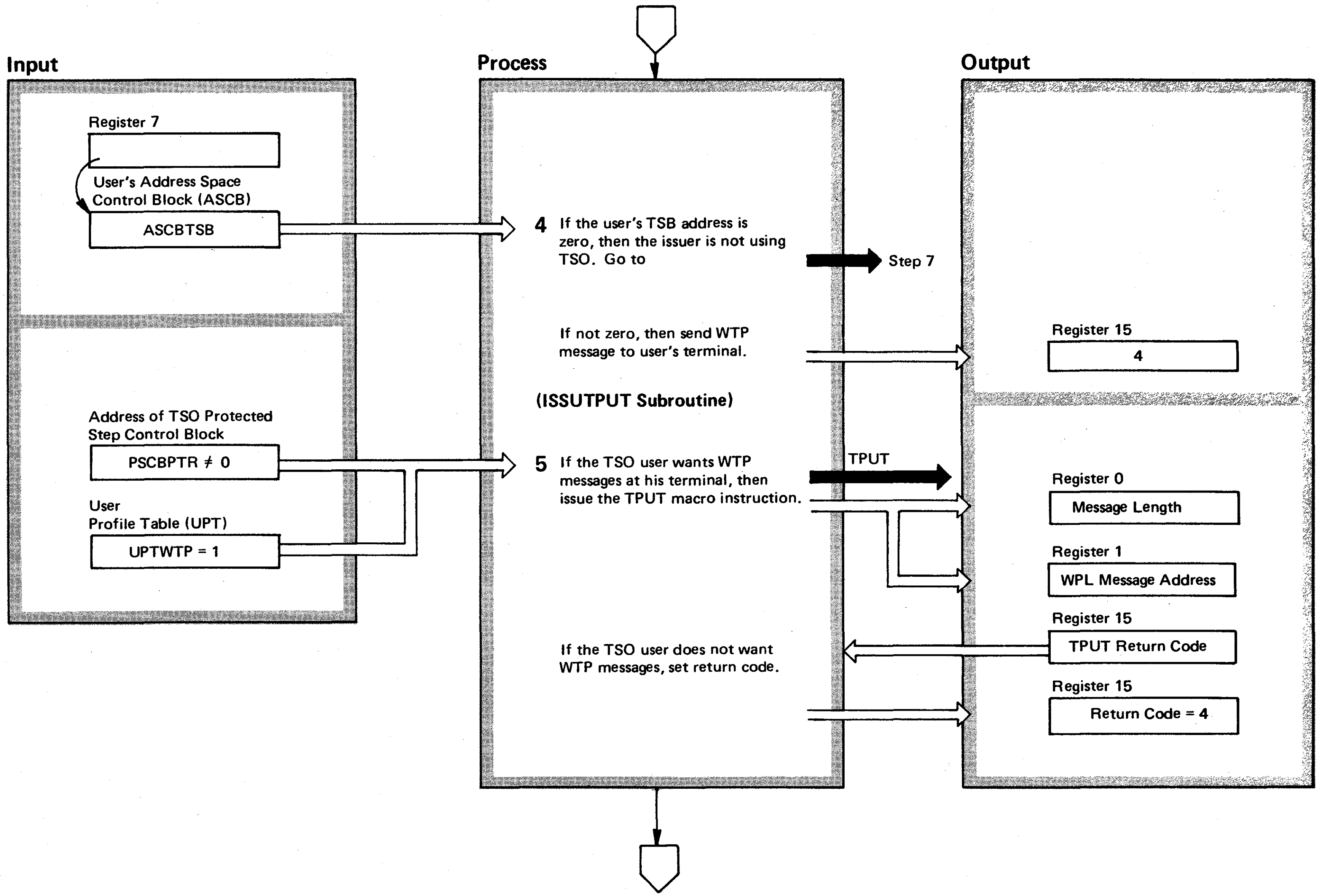
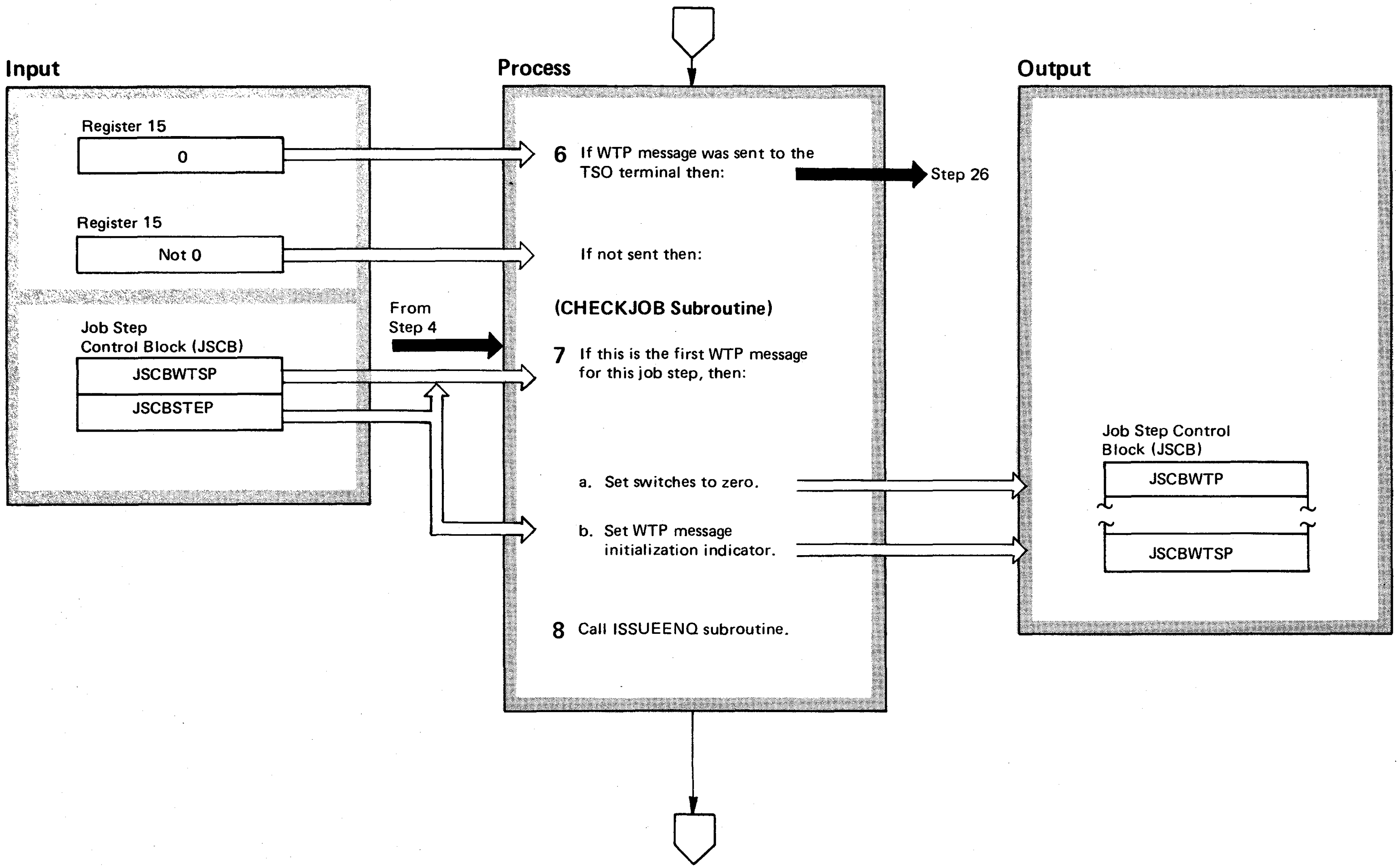


Diagram 1-8. Write-to-Programmer Processing (IGC0203E) (Part 4 of 22)

Extended Description	**Module**
4 When the recipient of the WTP message is an active TSO terminal user, the WTP message is sent to his terminal. If the recipient is not a TSO user, an attempt will be made to send the WTP message to the message data set defined for his job.	
5 Upon entry into the ISSUTPUT subroutine, a test determines if the TSO user wants WTP messages sent to his terminal. If he does, a pointer to the message text in the user's write parameter list (WPL) is placed in register 1 and the length of the WTP message is placed in register 0; this subroutine then issues the TPUT macro instruction, which will cause the WTP message to be transmitted to the user's terminal. When control is returned to this subroutine, it immediately returns to the mainline routine with the return code received from the TPUT operation. The return code is zero if the message was sent. If the TSO user does not want WTP messages sent to his terminal, a return code of 4 is placed in register 15 to indicate that the WTP message was not sent and control is returned to the mainline routine.	

Diagram 1-8. Write-to-Programmer Processing (IGC0203E) (Part 5 of 22)



Extended Description | **Module**

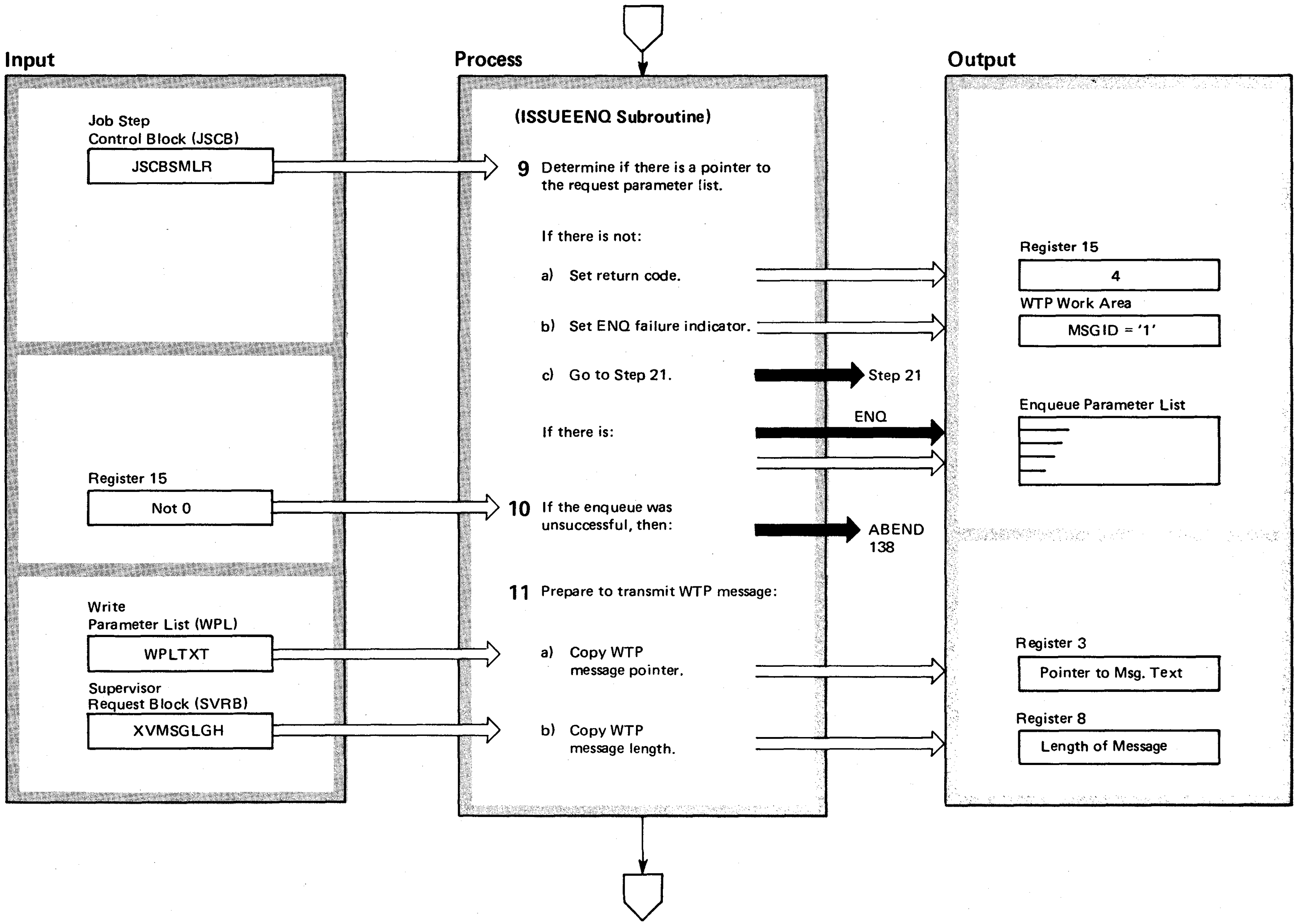
6 Upon return to the mainline routine, register 15 is checked to determine if the WTP message was sent to a TSO terminal user. If the message was sent, register 15 contains a zero; any other value indicates that the WTP message was not sent. If sent, the WTP message will not be sent to an I/O device; this routine branches to the area of this routine that returns control to the WTO and WTOR macro instruction processing routine.

If the WTP message was not sent to a TSO terminal, the CHECKJOB subroutine is called.

7 The CHECKJOB subroutine does a first pass initialization of the job step control block (JSCB) for the job step that is receiving the WTP message. To determine whether the JSCB has been initialized, the JSCB job step number (JSCBSTEP) is compared to the number of the last job step to receive a WTP message (JSCBWTSP). If they match, the JSCB has already been initialized. If they differ, the WTP flags (JSCBWTP) in the JSCB are set to zero and the job step number (JSCBSTEP) is copied into the field that indicates the last job step to have received a WTP message (JSCBWTSP). Control is returned to the mainline routine.

8 Upon return from the CHECKJOB subroutine, the mainline routine calls the ISSUEENQ subroutine.

Diagram 1-8. Write-to-Programmer Processing (IGC0203E) (Part 7 of 22)



Extended Description **Module**

9 The ISSUEENQ subroutine determines whether a pointer exists for a request parameter list (RPL).

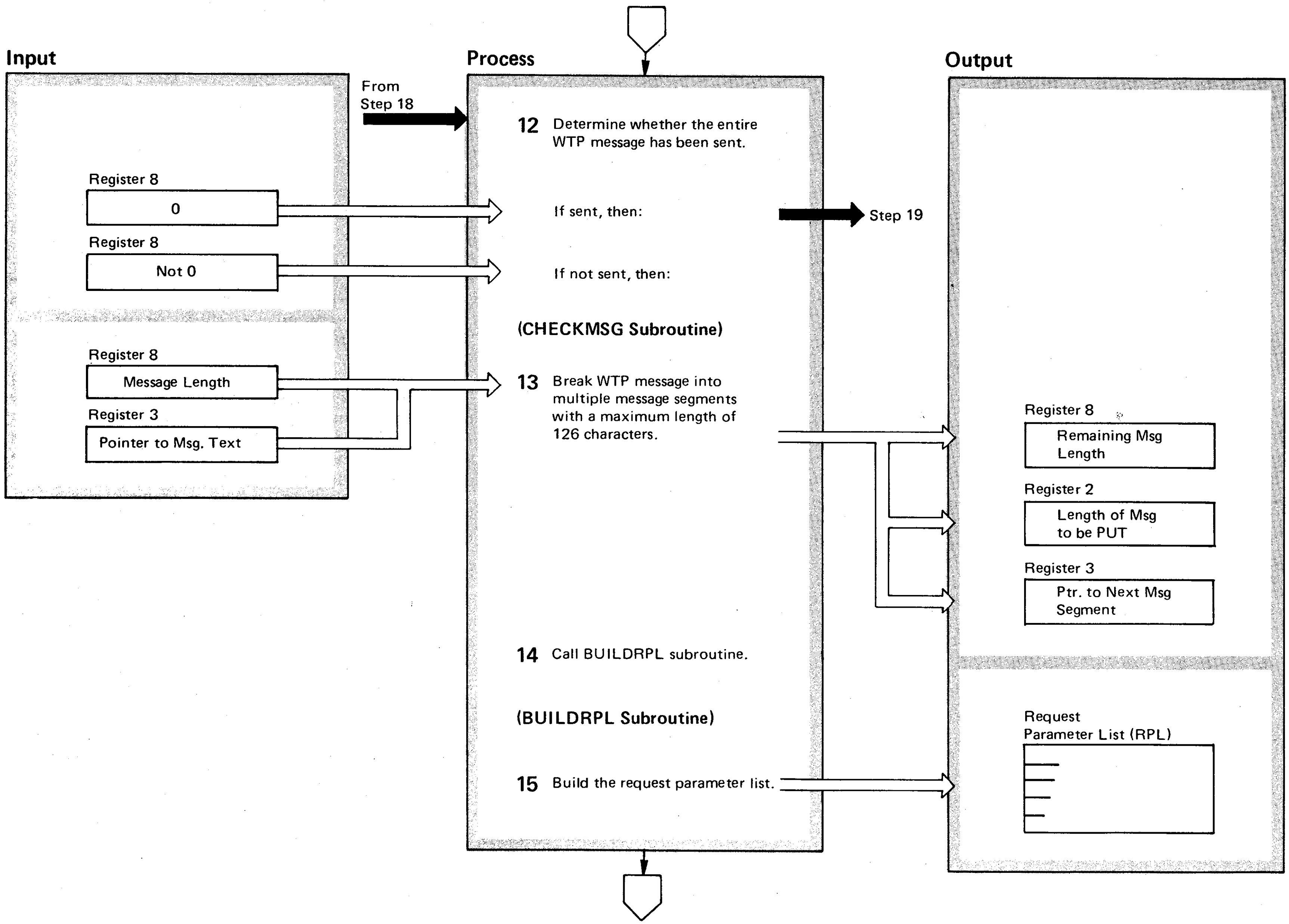
If the pointer to the RPL exists, this subroutine prepares a parameter list and issues the ENQ macro instruction. Upon return from the execution of the ENQ macro instruction, control is returned to the mainline routine; if the enqueue was successfully executed, register 15 was set to zero by the enqueue. If an abnormal termination occurs while the enqueue is executing, control will be given to the WTP STAE exit routine.

If the pointer to the RPL does not exist, a 4 is placed into register 15, the character '1' is placed in the MSGID field of the WTP work area to indicate that the enqueue was not issued. The '1' indicates the absence of the RPL pointer. Control is returned to the mainline routine.

10 Upon return from the ISSUEENQ subroutine, the mainline routine determines if the enqueue was successful. If the enqueue was successful, register 15 contains a zero; if the enqueue was unsuccessful, the mainline routine issues a 138 abnormal termination. During the processing of that abnormal termination by recovery termination management (RTM), RTM gives control to WTP STAE exit routine (STAE000) shown in step 34.

11 In preparation for sending a WTP message to a user defined data set, register 3 is initialized with a pointer to the beginning of the WTP message's text and register 8 with the length of the WTP message.

Diagram 1-8. Write-to-Programmer Processing (IGC0203E) (Part 9 of 22)



Extended Description **Module**

12-18 These steps constitute a multiple step loop for preparing and then sending the WTP message to a user defined data set.

If the WTP message is longer than 126 characters, the message will be broken into message segments of 126 characters or less; an attempt is made to break the message text between words. As each segment is prepared, the PUT macro instruction is issued to send the WTP message to the message data set for this users job. If the PUT operation was successful, the operation returns to the top of the loop to prepare the next WTP message segment.

If the WTP message is 126 characters or shorter, then only one pass is made through these steps except that the last step returns to the first to determine the end of this multiple step operation.

12 Upon the initial entry into this step, register 8 contains a count of the number of characters in the WTP message. On each subsequent entry, register 8 contains a count of the number of characters yet to be sent to the user defined data set. Eventually, the count will be zero, thus indicating that the entire WTP message has been sent. Until the entire WTP message has been sent, this routine calls the CHECKMSG subroutine. When the entire WTP message has been sent, this routine branches to the area of the routine that returns control to the WTO and WTOR macro instruction processing routine.

13 The CHECKMSG subroutine determines the WTP message segment that will be transmitted to the user defined data set. Any WTP message having more than 126 characters is divided into segment of 126 characters or less. An attempt is made to end a message segment with a blank character; thus, a WTP message segment may be less than 126 characters. Control is returned to the mainline routine.

14 Upon return from the CHECKMSG subroutine, the mainline routine calls the BUILDRPL subroutine.

15 The BUILDRPL subroutine prepares the request parameter list (RPL) that is used by the PUT macro instruction to send the WTP message to this job's message data set. Control is returned to the mainline routine.

Diagram 1-8. Write-to-Programmer Processing (IGC0203E) (Part 11 of 22)

Input

Register 15: 0

Register 15: Not 0

From Step 12

Process

16 Issue PUT macro instruction to send first WTP message segment to user defined data set. → PUT

17 Call the CKRETURN subroutine.

(CKRETURN Subroutine)

18 Determine whether PUT was successful.

If successful, then: → Step 12

If unsuccessful, then:

a. Set WTP message length to zero.

b. Set PUT failure indication.

c. Continue processing. → Step 12

19 Call the ISSUEDEQ subroutine.

Output

Register 8: Message Length

WTP Work Area: MSGID = '3'

Diagram 1-8. Write-to-Programmer Processing (IGC0203E) (Part 12 of 22)

Extended Description **Module**

16 Upon return from the BUILDRPL subroutine, the mainline routine issues the PUT macro instruction. This macro instruction moves one segment of the WTP message to the message data set. If the PUT operation is successful, a return code of zero has been placed in register 15.

17 Upon return from the PUT operation, the mainline routine calls the CKRETURN subroutine.

18 The CKRETURN subroutine determines whether the PUT operation was successful. If it was successful, then this subroutine branches back to the beginning of the loop control either to end the transmission or to send the next segment of the same WTP message.

If the PUT operation was unsuccessful, transmission of additional WTP message segments is prevented by setting to zero the remaining number of characters to be transmitted in register 8; indicating that the PUT operation failed by placing the character '3' in the WTP work area (MSGID); and branching back to the beginning of the loop where the end of transmission will be recognized.

19 After the entire WTP message has been sent to the user defined data set, the mainline routine calls the ISSUEDEQ subroutine.

Diagram 1-8. Write-to-Programmer Processing (IGC0203E) (Part 13 of 22)

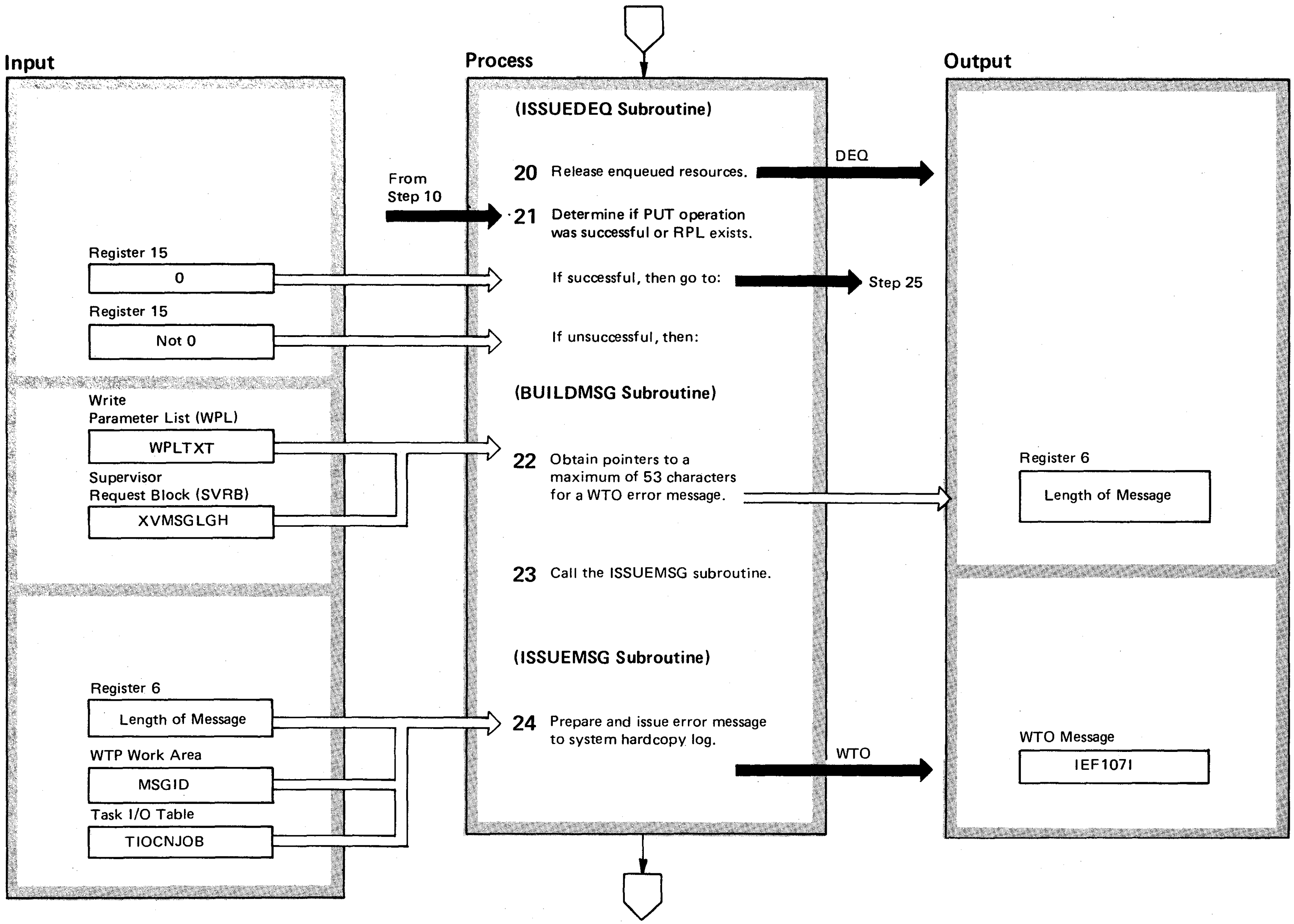
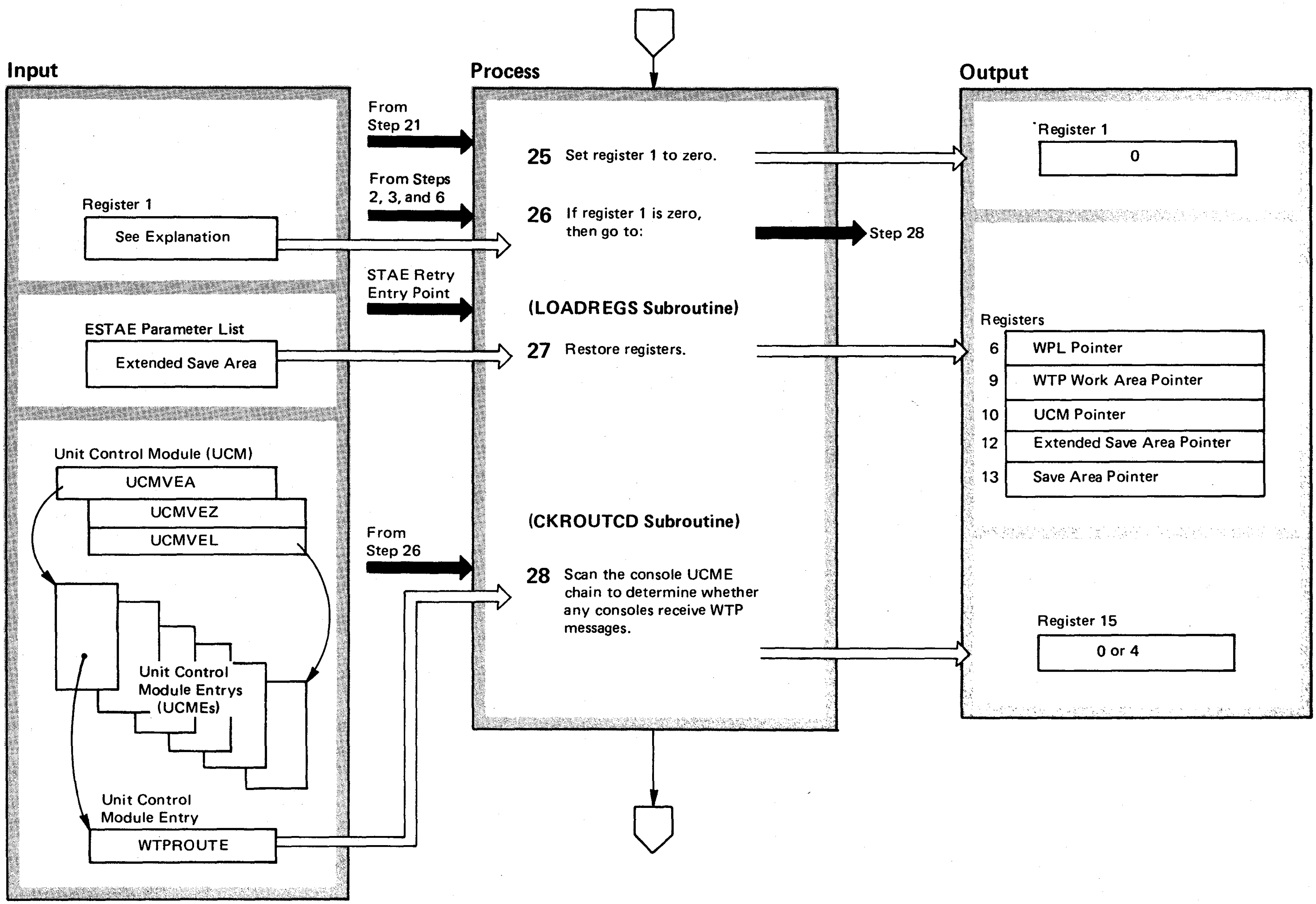


Diagram 1-8. Write-to-Programmer Processing (IGC0203E) (Part 14 of 22)

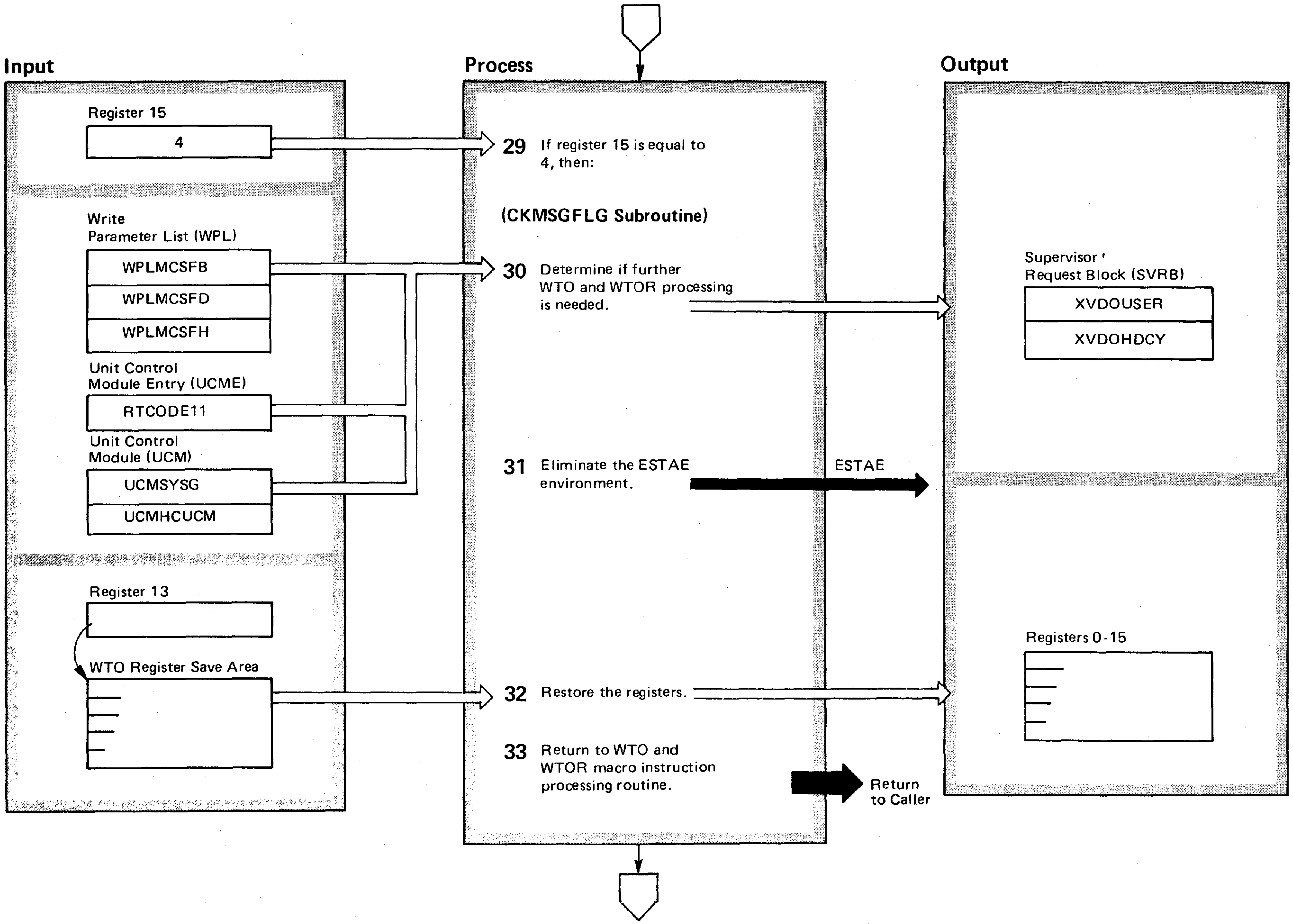
Extended Description	Module
20 The ISSUEDEQ subroutine issues the DEQ macro instruction to release the resources obtained earlier when the ENQ macro instruction was issued. Control is returned to the mainline routine.	
21 The mainline routine determines if the PUT operation was successful by testing the contents of register 15. If register 15 is zero, the PUT operation was successful. If register 15 is not zero, the PUT operation was unsuccessful; the BUILDMSG subroutine is called.	
22 The BUILDMSG subroutine locates the end of the first 53 character segment of the WTP message which was unsuccessfully sent. This 53 character segment will become part of error message IEF107I. An attempt is made to break this message segment at the blank character closest to the end of the 53 character segment to prevent a break in the middle of a word. Control is then returned to the mainline routine.	
23 Upon return from the BUILDMSG subroutine, the mainline routine calls the ISSUEMSG subroutine.	
24 The ISSUEMSG subroutine prepares the write parameter list and issues the WTO macro instruction that will cause message IEF107I to be written to the hardcopy log. Control is returned to the mainline routine.	

Diagram 1-8. Write-to-Programmer Processing (IGC0203E) (Part 15 of 22)



Extended Description	Module
25 Upon return from the ISSUEMSG subroutine or upon entry from having a successful PUT operation, register 1 is set to zero to prevent error processing.	
26 If register 1 is zero, then bypass STAE retry processing.	
27 For STAE retry only. Restore registers 6, 9, 10, 12, and 13 to the values held in these registers when the WTP routine was entered. Control is returned to the main-line routine.	
28 Determine if any of the system consoles are receiving routing code 11 messages. This determination is accomplished by scanning the unit control module entries (UCMEs) for the console. This subroutine locates a UCME for a console that is about to receive a routing code 11 message. When a routing code 11 console has been found, register 15 is set to zero. When no consoles are receiving routing code 11 messages, register 15 is set to 4. Control is returned to the mainline routine.	

Diagram 1-8. Write-to-Programmer Processing (IGC0203E) (Part 17 of 22)



Extended Description	Module

29 Upon return from the CKROUTCD subroutine, if register 15 is equal to 4, then call the CKMSGFLG subroutine.

30 The CKMCSFLG subroutine determines if the WTP message just processed is being sent to either a console or hardcopy log. If it is not, then this subroutine sets a bit (XVDOUSER) that tells the WTO and WTOR macro instruction processing routine that processing is complete and that the routine can return to the user who issued the WTO macro instruction; the communication task is not posted.

Initially, three bits in the user's write parameter list are tested. If any one of the three is set, this subroutine returns to the mainline routine without turning on the XVDOUSER bit:

WPLMCSFB	Queue message to an active console.
WPLMCSFD	Message type field exists.
WPLMCSFH	Queue message unconditionally to the identified console.

When none of these three bits are set, a further test is made to determine if the system has either an active graphic console (UCMSYSG) or an active hardcopy log (UCMCUCM). If one of these is active and the system is writing hardcopy WTP messages, the XVDOHDCY bit is set to request that this WTP message be sent to the hardcopy log, and control is returned to the mainline routine.

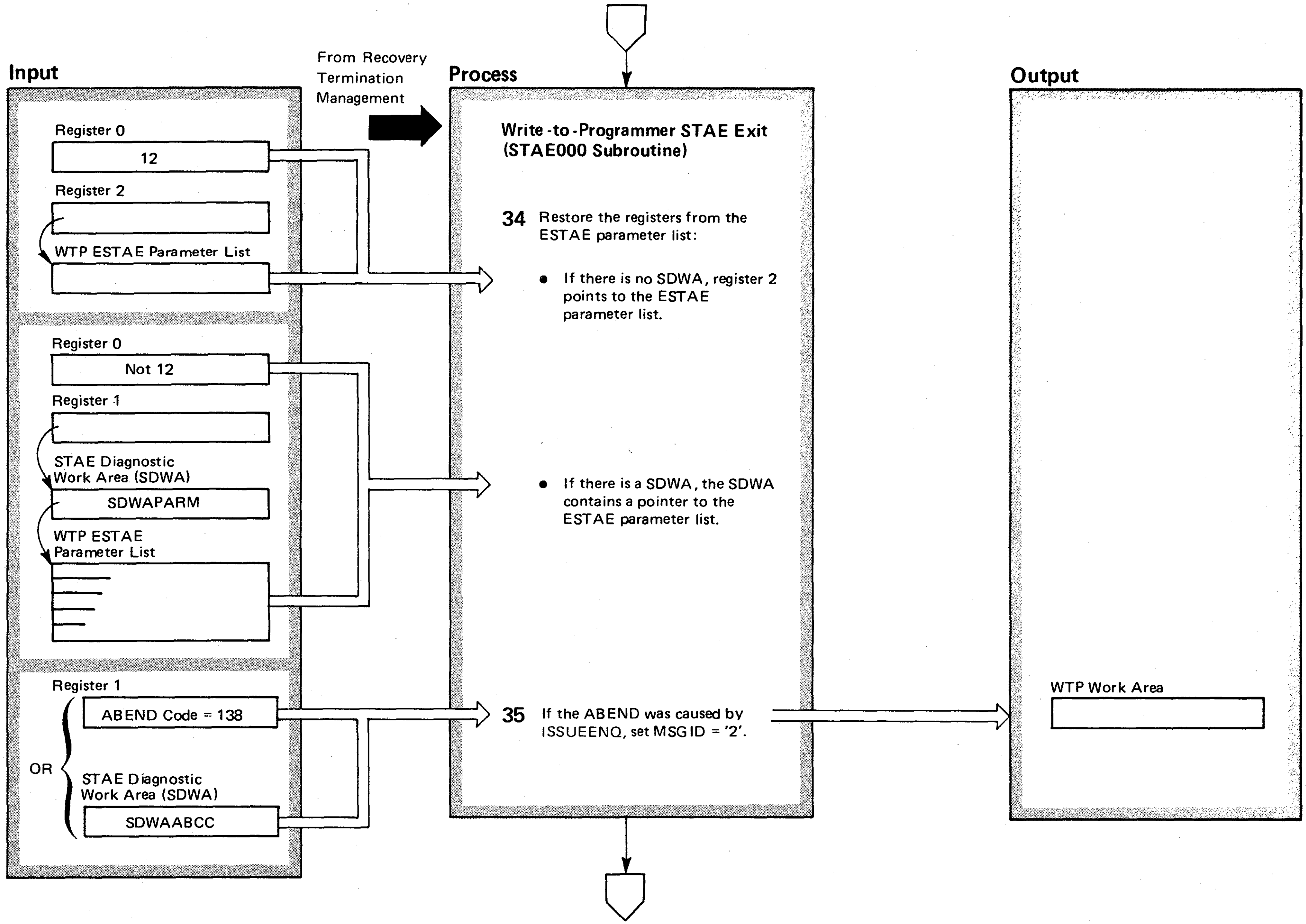
Any other combination of these bits indicates that the WTP message was the only WTO operation to be performed; therefore, the XVDOUSER bit is set. This bit tells the WTO and WTOR macro instruction processing routine that message processing is complete, and therefore, it can return to the user who issued the WTO macro instruction without posting the communication task. Control is returned to the mainline routine.

31 Upon return from the CKMCSFLG subroutine, the mainline routine issues the ESTAE macro instruction to eliminate the ESTAE environment.

32 All registers are restored to their original values from the WTO register save area.

33 Control is returned to the WTO and WTOR macro instruction processing routine.

Diagram 1-8. Write-to-Programmer Processing (IGC0203E) (Part 19 of 22)



Extended Description | **Module**

34 After the ESTAE environment has been established by the write-to-programmer routine, any abnormal termination results in control being given to the WTP STAE exit routine by recovery termination management (RTM).

Upon entry into this routine, the registers are restored from the WTP ESTAE parameter list. The means of finding the ESTAE parameter list depends on whether there is a STAE diagnostic work area (SDWA). When register 0 *is* equal to 12, register 2 contains a pointer to the ESTAE parameter list. When register 0 is *not* equal to 12, register 1 contains a pointer to the STAE diagnostic work area (SDWA). The SDWA contains a pointer to the WTP ESTAE parameter list that contains the registers.

35 If the abnormal termination occurred when the ISSUEENQ subroutine was unable to enqueue the request parameter list (RPL) in step 9, the MSGID is set to '2'. Before entering this step, the MSGID was '4' and is left at that setting for all other WTP abnormal terminations. The MSGID is printed as part of the message issued by this routine.

Diagram 1-8. Write-to-Programmer Processing (IGC0203E) (Part 21 of 22)

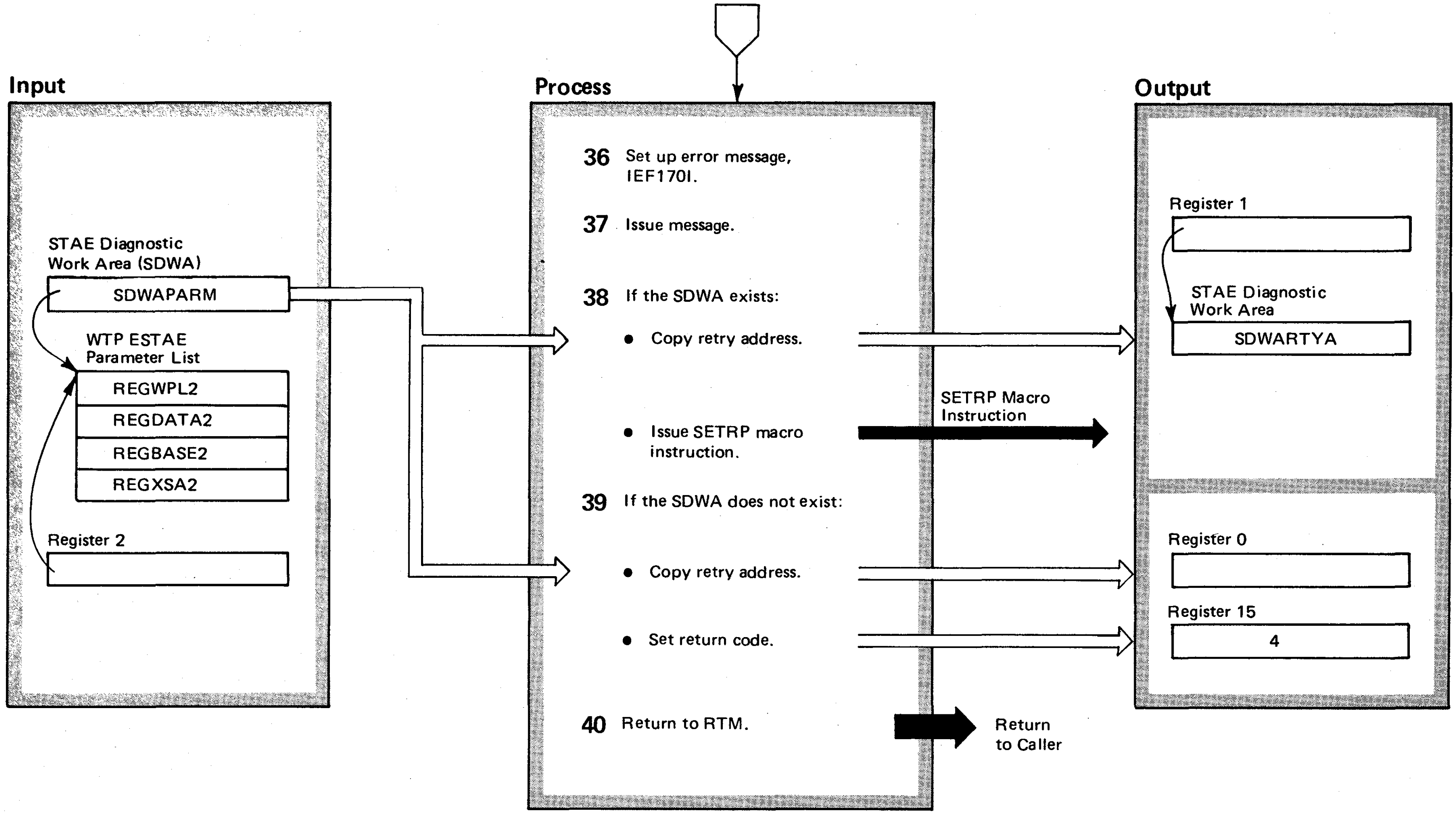


Diagram 1-8. Write-to-Programmer Processing (IGC0203E) (Part 22 of 22)

Extended Description | **Module**

36 This routine calls the BUILDMSG routine to prepare message IEF170I.

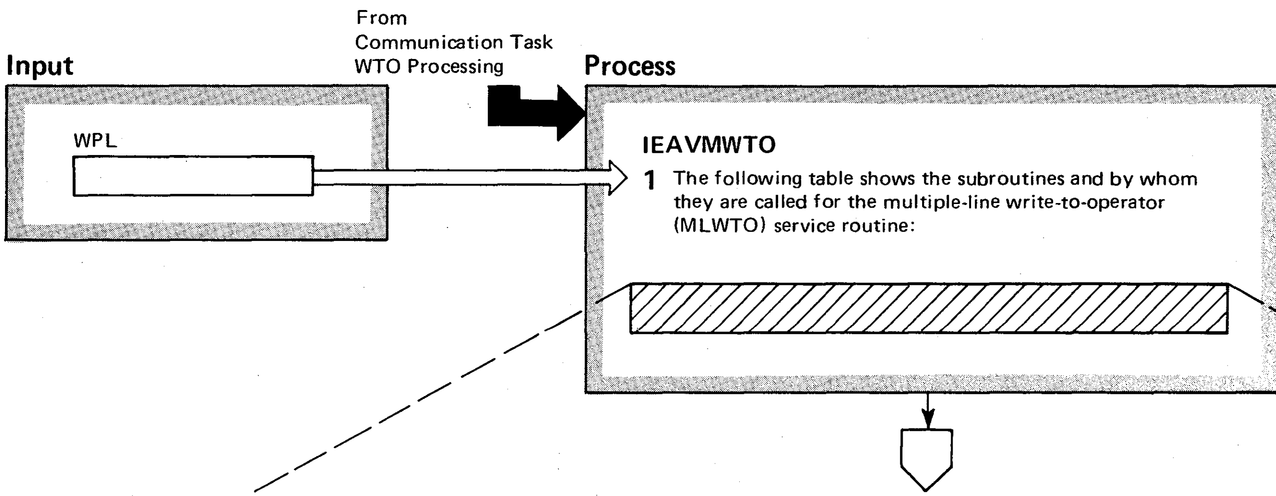
37 This routine calls the ISSUEMSG routine to send message IEF170I. This message records the error condition that caused the WTP abnormal termination.

38 If there is a STAE diagnostic work area (SDWA), this routine copies the address to be used for retrying the WTP program from the WTP ESTAE parameter list to the SDWA. The retry address is located by an offset within the code. This routine then issues the SETRP macro instruction which sets an indicator telling recovery termination management (RTM) to retry the WTP routine. The actual retry waits until this routine returns control to RTM.

39 When there is *no* SDWA, this routine copies the WTP retry address into register 0 and places a return code of 4 in register 15. Register 2 contains the pointer to the WTP ESTAE parameter list.

40 Return to caller.

Diagram 1-9. Multiple-Line WTO (MLWTO) Processing (SVC 35) (IEAVMWTO) (Part 1 of 24)



Caller:	Subroutines Called:									
	REFERLEN	SETLCKS	WAITWQE	GETWQE	GETMINOR	TEXTLINE	ENDUP	FINDID	LINEHDLR	FRELCKS
IEAVMWTO	X	X	X	X	X	X	X	X	X	X
GETMINOR				X						
LINEHDLR						X				
WAITWQE		X								X

Diagram 1-9. Multiple-Line WTO (MLWTO) Processing (SVC 35) (IEAVMWTO) (Part 2 of 24)

Extended Description | **Module**

This function builds the Major Write Queue Element (Major WQE) and one or more Minor Write Queue Elements (Minor WQE) containing messages for the console operators. Additional data lines may be connected to an existing MLWTO for key zero or supervisor mode users.

Mainline Routine: IEAVMWTO

Provides multiple line WTO support by building control blocks containing the text lines destined to go to an operator console. It also permits key 0 or supervisor mode users to add text lines to an already existing MLWTO.

Subroutines:

REFERLEN
The WTO macro instruction prepares the write parameter list (WPL). Since macro instructions execute in a privileged state, the possibility exists that part of the WPL resides outside the WTO user's address space; REFERLEN checks for this error condition.

SETLCKS
1. Obtains the local and cross memory services (CMS) locks.
2. Sets up the functional recovery routine (FRR).

WAITWQE
Waits for a WQE to be freed.

Extended Description | **Module**

GETWQE
Obtains a major WQE from the WQE cellpoll in subpool 231 and attaches it to the regular WQE chain.

GETMINOR
Obtains a minor WQE from the WQE cellpool in subpool 231 and attaches it to the minor WQE chain that is pointed to from a major WQE.

TEXTLINE
Increases a pointer to the next line in the write parameter list (WPL).

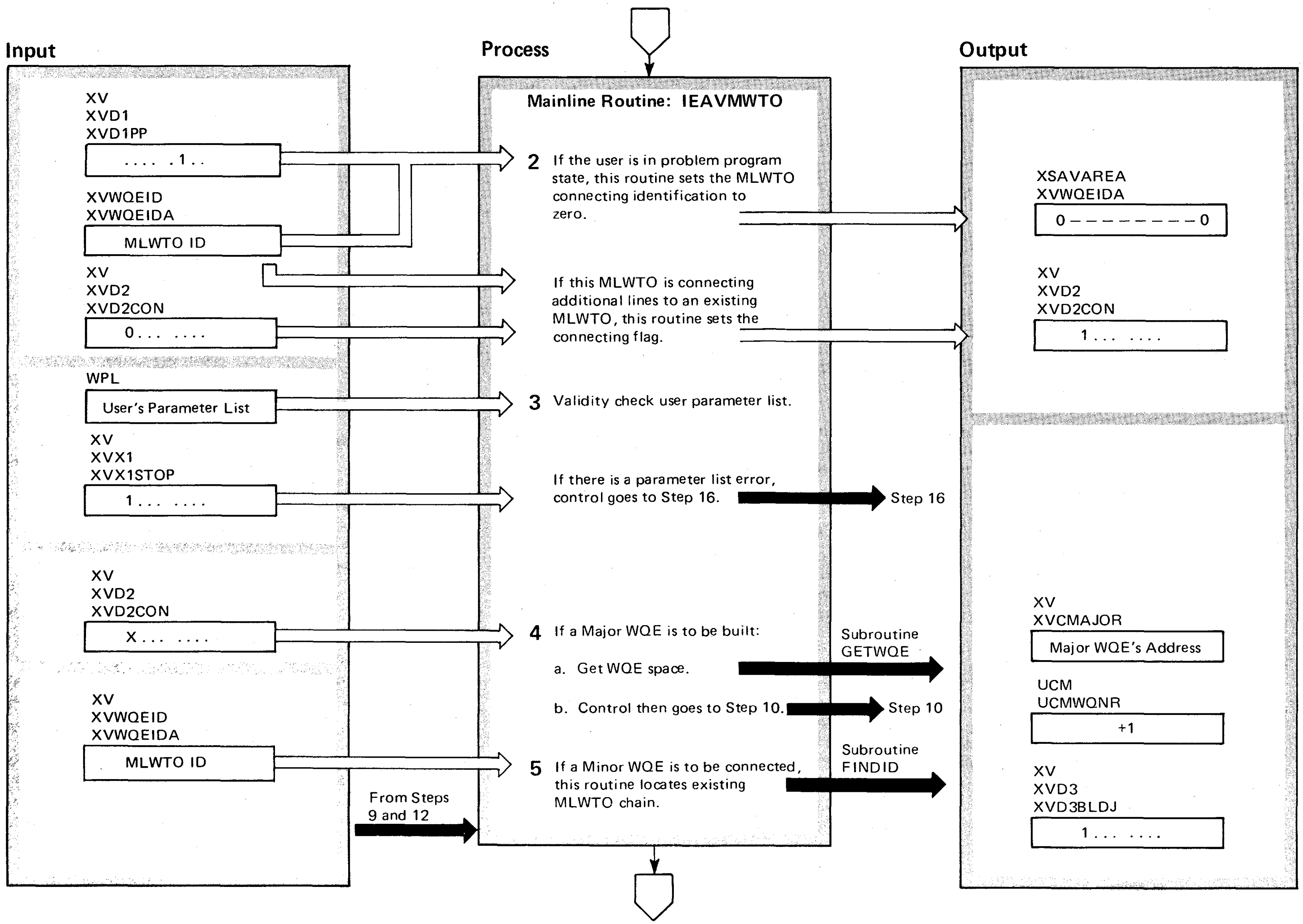
ENDUP
1. Decreases the counter containing the number of lines yet to be processed.
2. If needed, sets line type to data end.

FINDID
Locates major WQE to which a minor WQE is to be attached.

LINEHDLR
When the WTO macro instruction is issued by a problem program, this routine replaces possible control characters imbedded in the message text with blanks (X'40').

FRELCKS
1. The set up for the functional recovery routine is freed.
2. The local and cross memory services (CMS) locks are freed.

Diagram 1-9. Multiple-Line WTO (MLWTO) Processing (SVC 35) (IEAVMWTO) (Part 3 of 24)



Extended Description	Module
2 After establishing addressability and obtaining a work area from subpool 229, the 'Problem Program' flag (XVD1PP in XV) is tested to determine if the task is allowed to connect more lines. If set, connecting is not allowed and the connecting identification (XVWQEIDA in XV) is set to zero. If the 'Problem Program' flag (XVD1PP in XVD1) is not set, the connecting identification (XVWQEIDA in XV) is checked for zero. If not zero, the user is assumed to be connecting and the 'Connecting' flag (XVD2CON in XVD2) is set. For recovery protection, the ESTAE macro instruction is issued.	IEAVMWTO
3 TSTMLWTO validity checks the user's write parameter list (WPL) for: • Physical organization. • Physical location of fields. • Length of fields. • Compatibility of options. This subroutine also checks to be sure the write parameter list is in the user's address space (REFERLEN). If the parameter list has an error that will cause the MLWTO request to be ignored, the 'Stop' flag (XVX1STOP in XVX1) is tested. If it is set, control passes to the IEAMSTOP routine to exit with a return code.	
4 The 'Connecting' flag (XVD2CON in XVD2) is tested to determine if a Minor WQE is to be connected to an existing MLWTO chain. If not, subroutine GETWQE is invoked to obtain, via GETCELL from subpool 231, space for a MAJOR WQE. The count of in-use WQEs (UCMWQNR in UCM) is increased. The address returned from Subroutine GETWQE is stored (in XVCMAJOR). The 'Build Major' flag (XVD3BLDJ in XVD3) is set. Control passes to step 10 (IEAMGETN). If a write wait block (WWB) was previously obtained, it is dequeued and freed.	
5 Subroutine FINDID is invoked to locate the MLWTO chain to which lines are to be added. (Identification of the MLWTO is in XVWQEIDA of the XV).	

Diagram 1-9. Multiple-Line WTO (MLWTO) Processing (SVC 35) (IEAVMWTO) (Part 5 of 24)

Input

XV
XVX1
XVX1NOID

.1.. ....

Major WQE
WMJMMIN

First Minor WQE

XV
XVCMINOR

Last Minor Address

XVD3
XVD3BLD1

.1.. ....

XVD3BLD2

..1. ....

UCM
UCMWQNR

In Use Count

UCMWQLM

Limit

XV
XVD1
XVD1PRIV

1... ....

Process

6 If MLWTO is not found, control goes to Step 14. → Step 14

If a Minor WQE is not queued, control goes to Step 10. → Step 10

7 If Minor Line One is to be built, control goes to Step 19. → Step 19

If line two is to be built, control goes to Step 20. → Step 20

8 If a WQE is available or if the caller is a privileged user, control goes to Step 10. → Step 10

Output

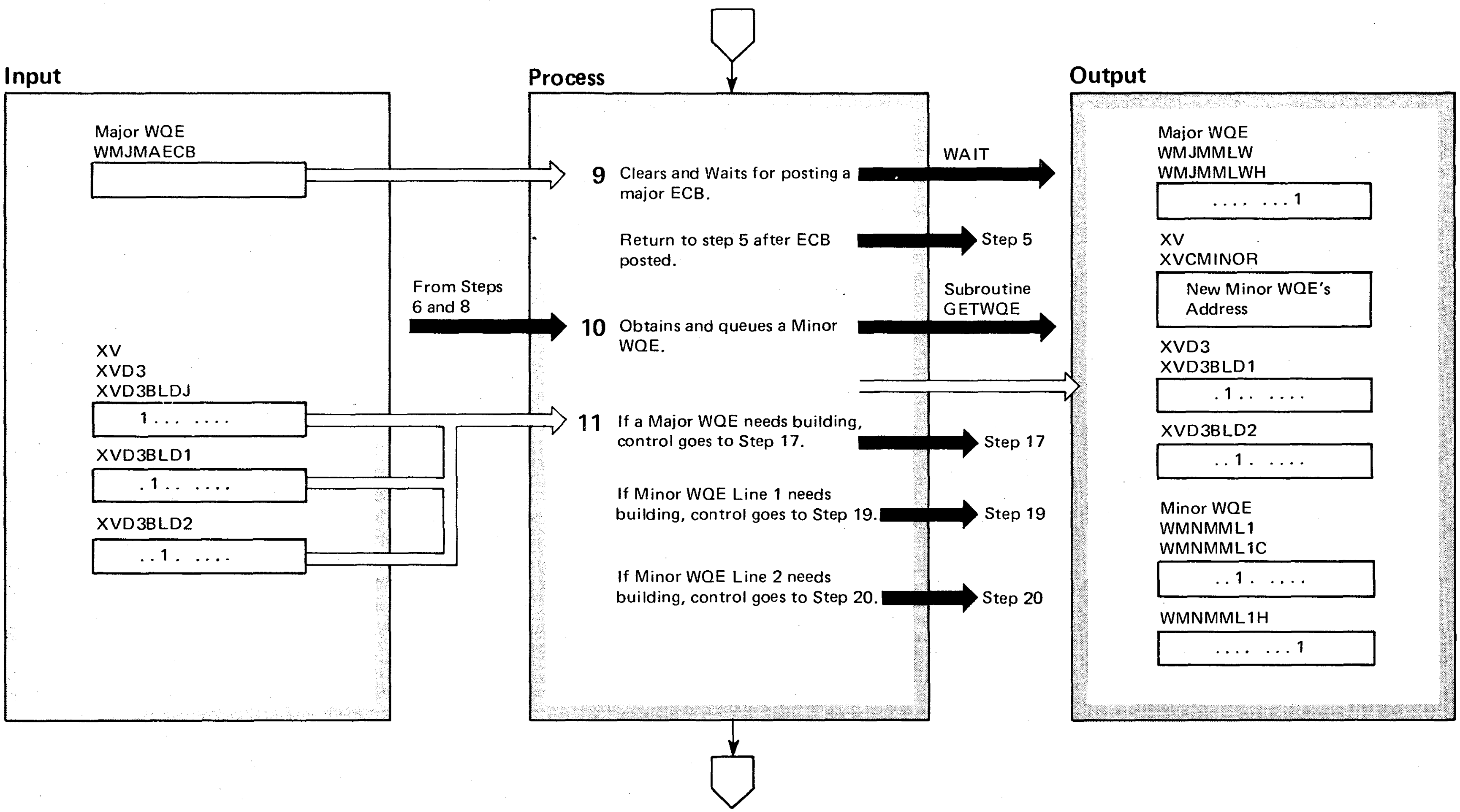
Major WQE
WMJMAECB

0 – – – – – – – – 0

Diagram 1-9. Multiple-Line WTO (MLWTO) Processing (SVC 35) (IEAVMWTO) (Part 6 of 24)

Extended Description	Module
6 On return from Subroutine FINDID, the error flag (XVX1NOID) is tested to determine if the MLWTO was found. If not, control passes to step 14. Then to step 16 to exit. If the MLWTO is found, the Major WQE is tested to determine if a Minor WQE is queued to it. If not, control passes to step 10 to get a Minor WQE.	IEAVMWTO
7 Two message lines can be stored in one minor WQE. If there are Minor WQEs queued to the Major WQE, then it may be possible to use the last queued minor. If the 'Build Line 1' (XVD3BLD1 in XVD3) and the 'Build Line 2' (XVD3BLD2) flags are on, control goes to step 19. If just the 'Build Line 2' flag (XVD3BLD2) is on, control passes to step 20.	
8 A test determines whether space exists for a minor WQE. If space is available, control passes to step 10. If the space is unavailable but the WTO was issued by a privileged user (the communication task or any task running under an SIRB), a WQE may be obtained regardless of the limit; therefore, control passes to step 10. If space is unavailable and the WTO was issued by a nonprivileged user, the user waits for WQE space to become available; control continues to the next step.	

Diagram 1-9. Multiple-Line WTO (MLWTO) Processing (SVC 35) (IEAVMWTO) (Part 7 of 24)



Extended Description	Module
9 The Major ECB is cleared (WMJMAECB). A WAIT macro instruction is issued. The Major ECB is posted when a WQE is available. Control passes to step 5 (IEA1FIND). When a WQE is freed, the communication task checks for a WTO that is waiting for a WQE. If a WTO is waiting, the waiting ECB is posted.	
10 Subroutine GETWQE is invoked to obtain space for a Minor WQE. It queues the new Minor WQE to the Major WQE if none were previously queued to it. Otherwise, it is queued to the last Minor WQE. The address of the new minor is stored (XVCMINOR in XV). If the new Minor WQE is the only Minor WQE queued to the Major WQE, the 'Dummy Minor' flag (WMJMMLWH in WMJMMLW) is set. The 'Build Line 1' and 'Build Line 2' flags (XVD3BLD1 and XVD3BLD2 in XVD3), the 'Minor WQE' and 'GETMAINed' flag (WMNMLIC and WMNMML1H in WMNMML1) are set.	IEAVMWTO
11 A test of the 'Build Major' flag (XVD3BLDJ in XVD3) determines if a Major WQE needs building. If so, control passes to step 17. If the 'Build Line 1' (XVD3BLD1 in XVD3) and the 'Build Line 2' (XVD3BLD2) flags are on, control passes to step 19. If just the 'Build Line 2' flag (XVD3BLD2) is on, control passes to step 20.	

Diagram 1-9. Multiple-Line WTO (MLWTO) Processing (SVC 35) (IEAVMWTO) (Part 9 of 24)

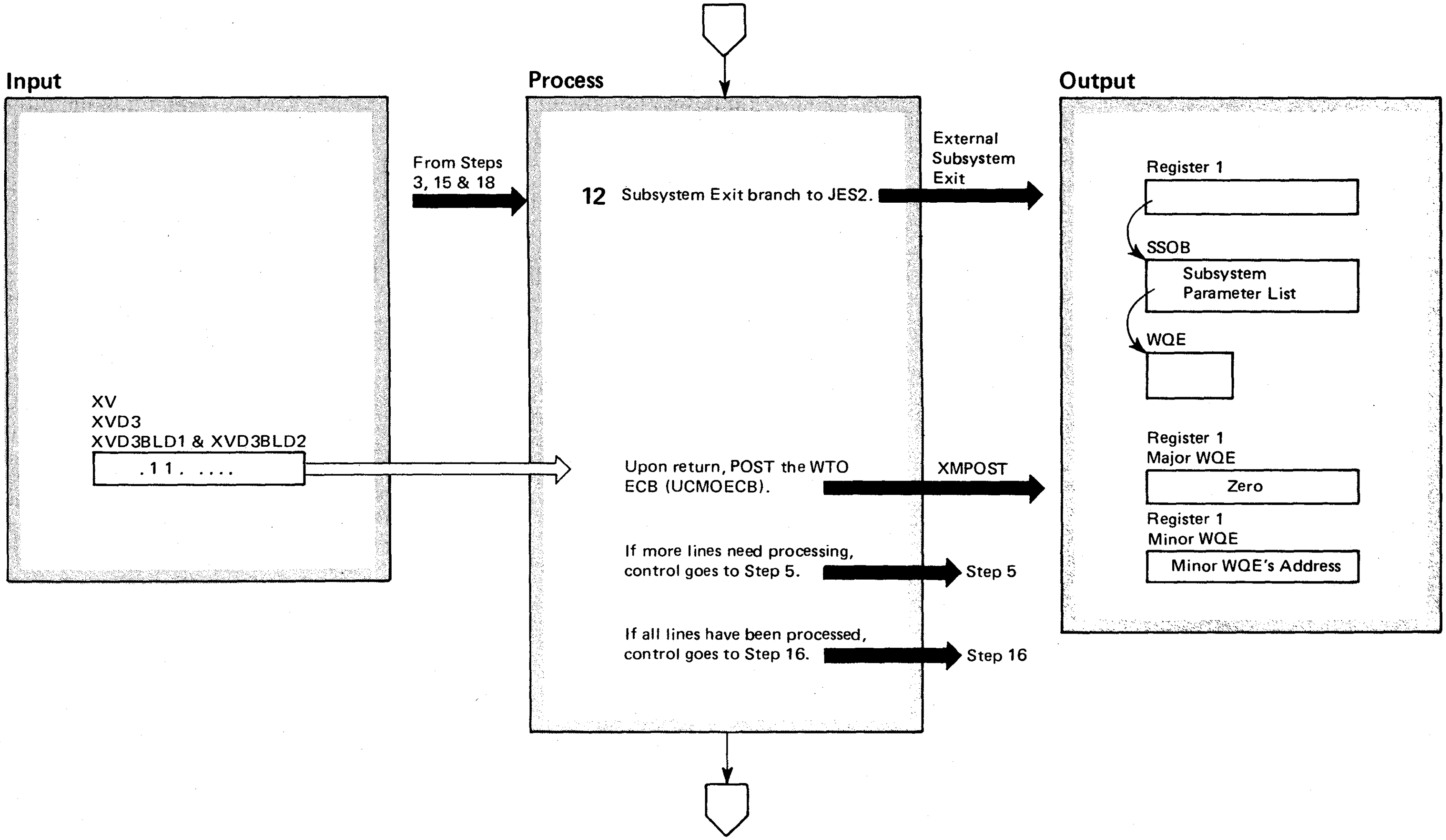


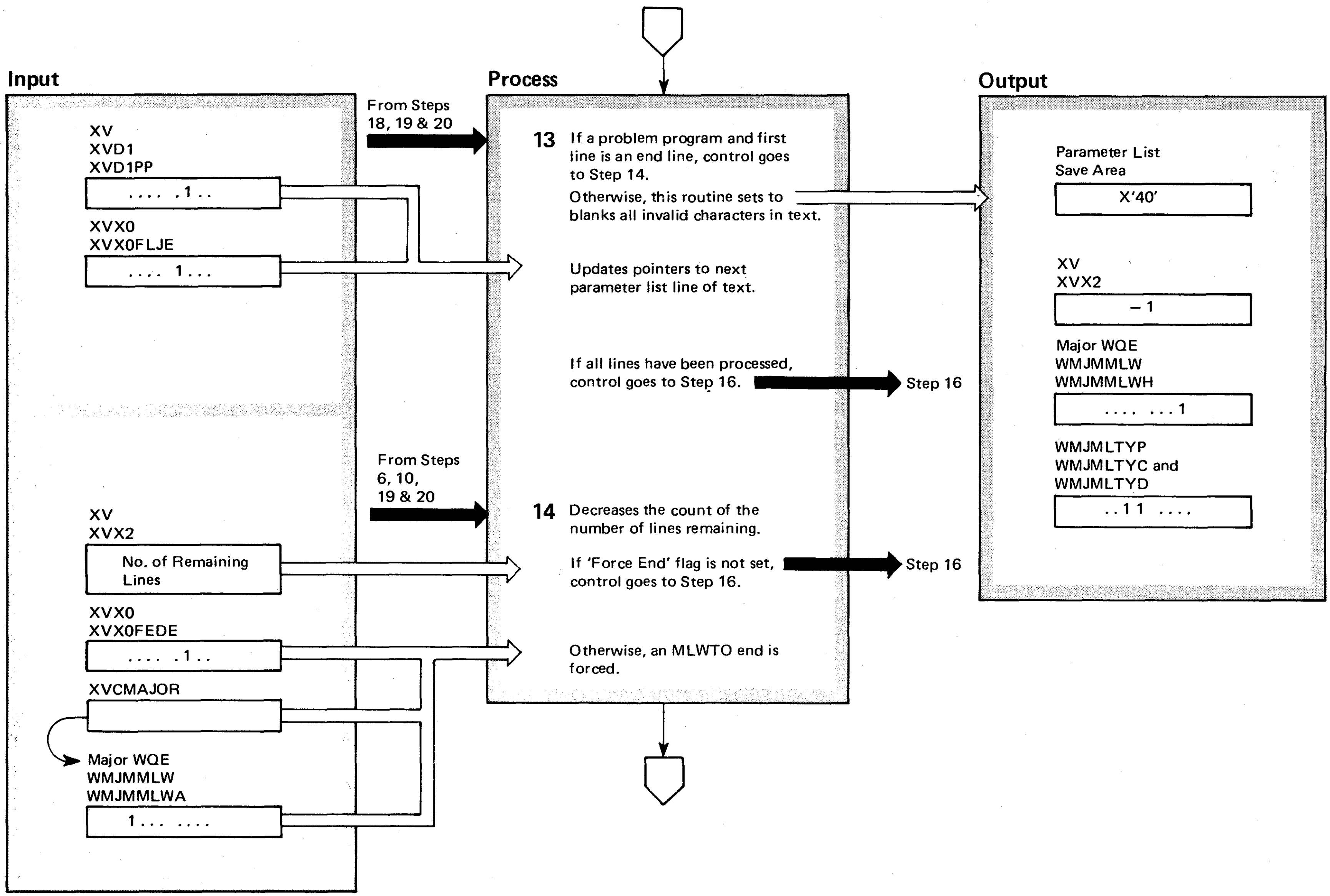
Diagram 1-9. Multiple-Line WTO (MLWTO) Processing (SVC 35) (IEAVMWTO) (Part 10 of 24)

Extended Description **Module**

12 A parameter list is built called the subsystem options block (SSOB). An extension of the SSOB called the subsystem options block for WTO (SSOBWT) points to the WQE and describes the type of WQE — major or minor. IEAVMWTO then branches to the job entry subsystem exit routine.

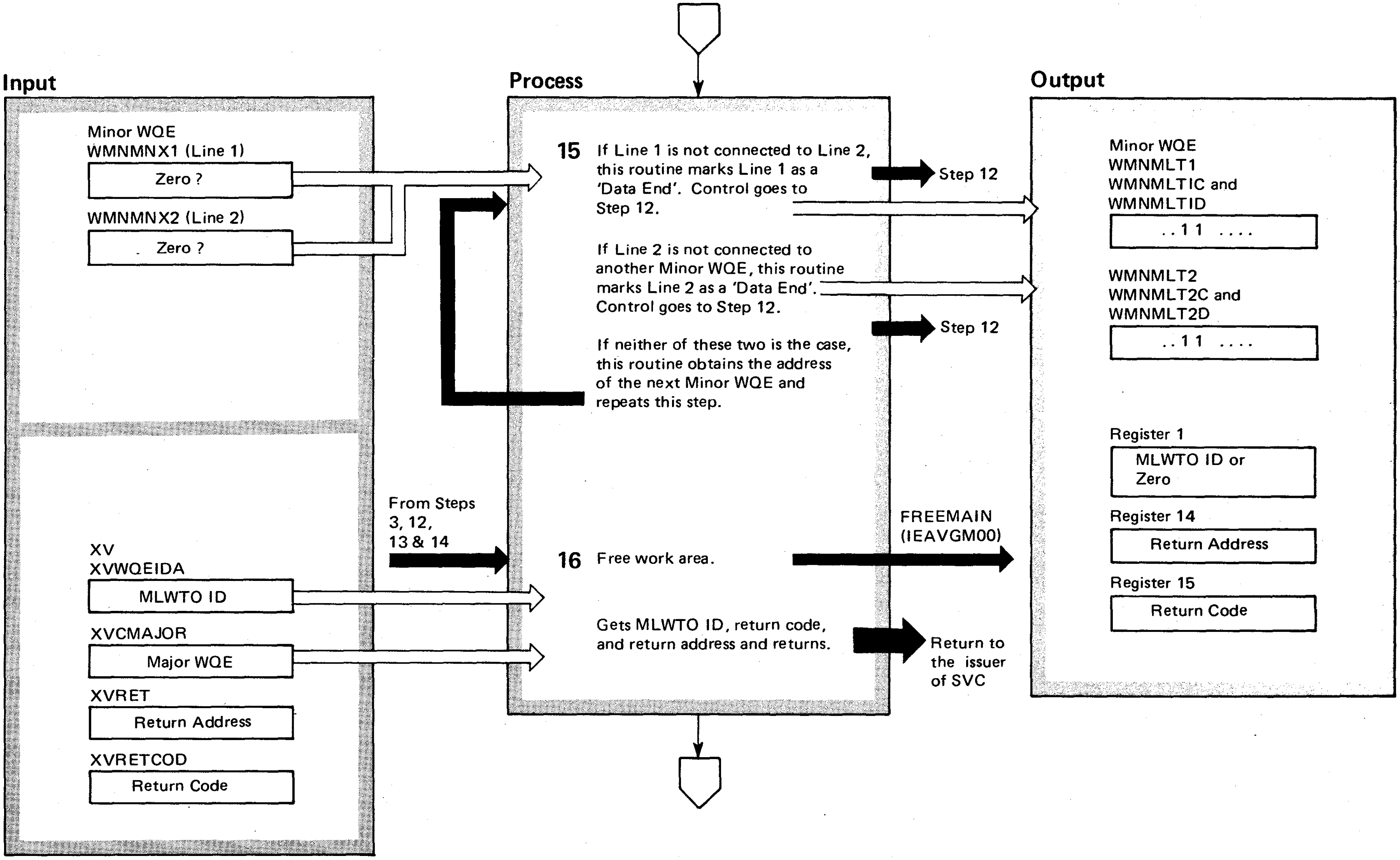
If more lines are to be processed, control passes to step 5. Otherwise, control passes to step 16.

Diagram 1-9. Multiple-Line WTO (MLWTO) Processing (SVC 35) (IEAVMWTO) (Part 11 of 24)



Extended Description	Module
13 If the problem program flag (XVD1PP in XVD1) is set and the first line is just an end line (XVX0FLJE set in XVX0), then control passes to step 14. Otherwise, the line text is scanned for invalid characters and if any are found they are set to blanks. If the problem program flag is not set, the routine does not set the new line control characters to blanks. Subroutine TEXTLINE is invoked to update the pointer to the next text line in the parameter list (WPL). Upon return, control passes to step 14.	
14 The number of lines still to do (XVX2) is decreased by one. If the 'Force End' flag (XVX0FEDE in XVX0) is not set, control passes to step 16. Otherwise, an end to the MLWTO is forced. The address of the Major WQE (XVCMAJOR) is obtained. If the 'Dummy Minor' flag (WMJMMLWH in WMJMMLW) is set, the Major WQE is flagged as a 'Data End' line (WMJMLTYC and WMJMLTYD in WMJMLTYP are set). Otherwise, the address of the Minor WQE linked to the Major WQE is also flagged as a 'Data End' line.	

Diagram 1-9. Multiple-Line WTO (MLWTO) Processing (SVC 35) (IEAVMWTO) (Part 13 of 24)



Extended Description	Module
15 The pointer to the second line is tested for zero. If zero, Line 1 of the minor is flagged as a 'Data End' line (WMNMLT1C and WMNMLT1D in WMNMLT1). Control passes to step 12. If second line pointer is not zero, the pointer to the next minor (WMNMNX2) is tested for zero. If zero, Line 2 of the minor is flagged as a 'Data End' line (WMNMLT2C and WMNMLT2D in WMNMLT2). Control passes to step 12. If not zero, the address of the next minor is obtained and this step is repeated.	
16 The previously obtained work area is freed (subpool 229). Upon return, register 15 is loaded with the return code (XVRETCOD) and register 14 with the return address (XVRET). If there was a Major WQE (XVCMAJOR), register 1 is loaded with the MLWTO ID (XVWQEIDA). Otherwise, register 1 is set to zero. Control returns via register 14.	IEAMSTOP

Diagram 1-9. Multiple-Line WTO (MLWTO) Processing (SVC 35) (IEAVMWTO) (Part 15 of 24)

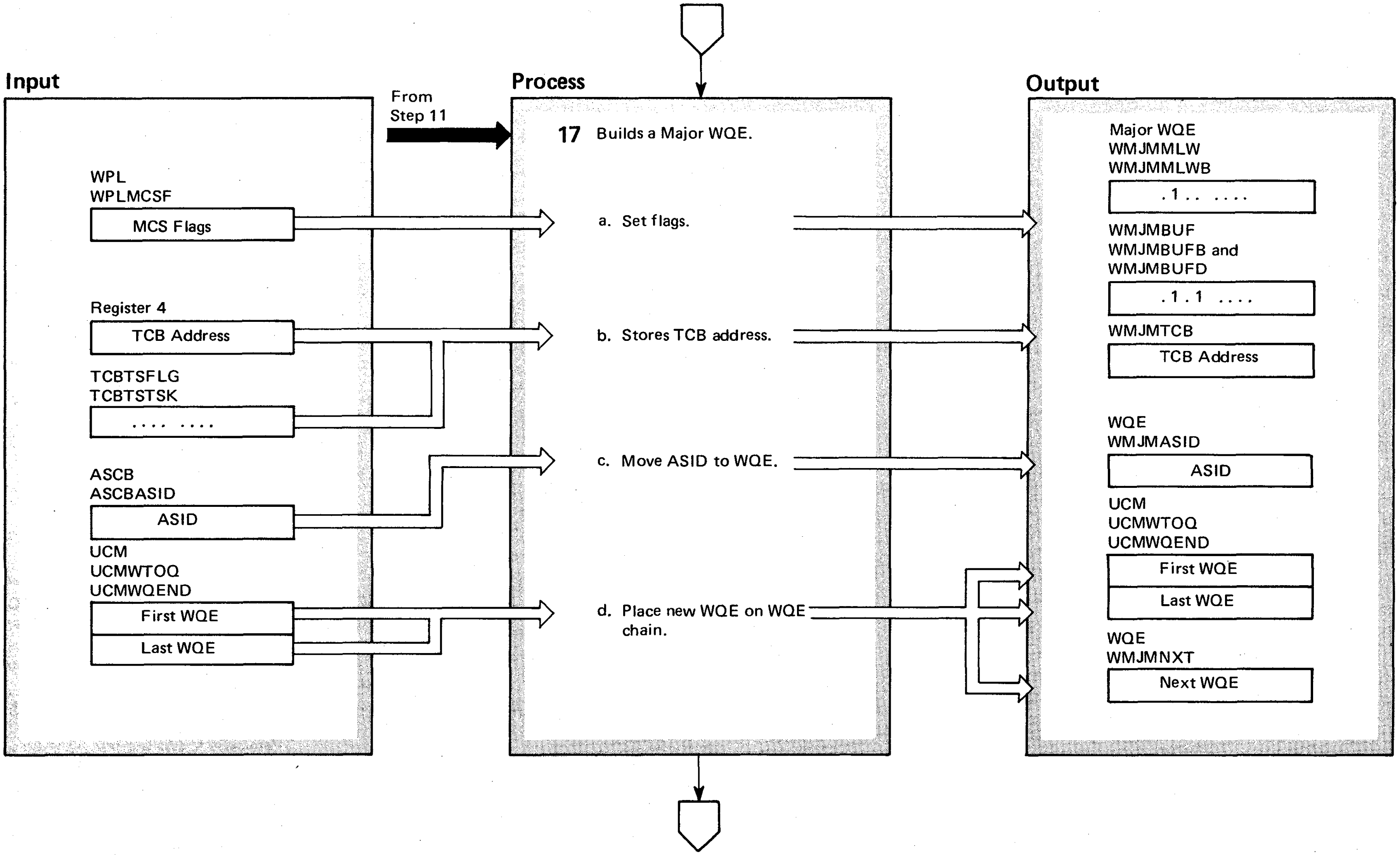


Diagram 1-9. Multiple-Line WTO (MLWTO) Processing (SVC 35) (IEAVMWTO) (Part 16 of 24)

Extended Description **Module**

17 To build the Major WQE:

a. Flags are set:

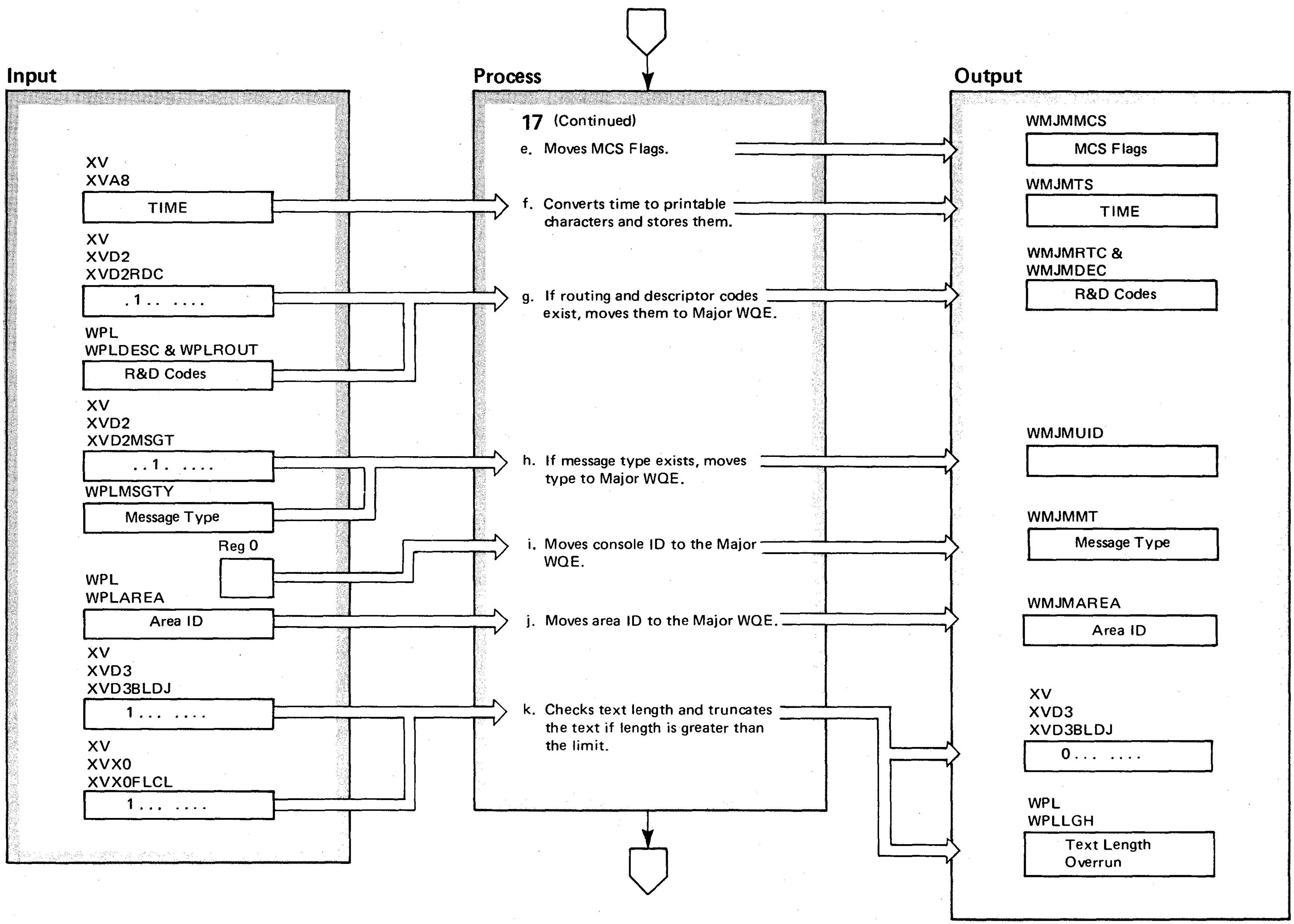
WMJMMLWB in WMJMMLW to indicate a Major WQE, WMJMBUFB and WMJMBUFD in WMJMBUF to indicate the WQE is in use and acquired by GETMAIN.

b. The TCB address (register 4) is stored in the Major WQE (WMJMTCB).

c. The ASID is moved from the ASCB to the WQE.

d. The new major WQE is added to the WQE's chained from the UCM control block.

Diagram 1-9. Multiple-Line WTO (MLWTO) Processing (SVC 35) (IEAVMWTO) (Part 17 of 24)



Extended Description	Module
e. The MCS flags are moved from the write parameter list to the Major WQE.	
f. The time XVA8 is converted to a printable form and stored (WMJMTS).	
g. If routing and descriptor codes exit (WVD2RDC), they are moved from the parameter list to the Major WQE.	
h. If message type exists (XVD2MSGT in XVD2), the message type flags are moved to the major.	
i. If the console ID was passed as input to SVC 35, that is, if WPLMCSFB or WPLMCSFH or both are on, register 0 contains the console ID. It is moved to the major (WMJMUID).	
j. If the area ID parameter (AREAID=in the WTO macro instruction) was specified, it is moved from the parameter list to the Major WQE.	
k. If the first line in the parameter list is a control line (XVXOFLCL set in XVXO), or if the 'Use Default Control Line' flag (XVXOUDCL in XVXO) is not set, the user's text length is compared to the limit for data, label or control lines. If the length exceeds the limit, the text is truncated. The text length is adjusted. If XVXOUDCL is set, the text length is set to equal that of the default control line.	

Diagram 1-9. Multiple-Line WTO (MLWTO) Processing (SVC 35) (IEAVMWTO) (Part 19 of 24)

Input

WPL
WPLTEXT
TEXT

UCM Prefix
UCMCMID
Seq. No.

WPL
WPLDESC2
WPLDESC1
1... ....

Major WQE
WMJMHCID
Seq. No.

WPL
WPLMCSF1
WPLMCSFA
1... ....

WPLROUT
Routing Codes

XV
XVD1
XVD1AUTH
.... ..1.

XVX0
XVX0UDCL
...1 ....

Process

18 Builds a Major WQE (continued)

a. Moves message text to the Major WQE.

b. Enters appropriate security message flagging.

c. Shifts text.

d. Stores text length and WQE sequence number.

e. If descriptor code is 9, moves printable MLWTO ID after text in Major WQE.

f. If routing codes are present, moves printable routing codes to Major WQE.

g. If the Major WQE is not built with a default control line, control passes to Step 13. → Step 13

Otherwise, this routine sets the 'Control Line' flag and sets a pointer to the beginning of the parameter list.

Control passes to Step 12. → Step 12

Output

Major WQE
WMJMTXT
@ OR *	+	Message
1	2	

WMJMTXT
← Message | 0 |

WMJMTXTL
Length of Message

WMJMHCID
MLWTO ID

WMJMRR
Routing Codes

WMJMPAD1
Blank

WMJMLTYP
WMJMLTYA
1... ....

Pointers

WPL

Extended Description	Module
18 The building of the Major WQE continues:	
a. The message text is moved to the Major WQE.	
b. If this is an action message (descriptor code 1 or 2) and it is authorized (XVD1AUTH set in SVD1), an asterisk (*) is inserted in the first position of the Major WQE text area. If not authorized, an at character (@) is inserted. If not an action message and not authorized, a plus (+) is inserted in the second position of the Major WQE text area. The text length is increased by two.	
c. The text is shifted one to the left, the character after the text is set to blank and the text length is increased.	
d. The updated text length is stored in the Major WQE (WMJMTXTL). The WQE sequence number is moved from the UCM Prefix (UCMCMID) to the Major WQE (WMJMSEQ and WMJMMSGN) and saved (XVWQEIDA). Then UCMCMID is increased by one. The MLWTO ID is converted to printable characters, stored (WMJMHCID in the Major WQE) and the first character is set to a blank.	
e. If descriptor code 9 (WPLDESCI set in WPLDESC) is found, the MLWTO ID (WMJMHCID) is affixed to the end of the text (WMJMTXT + WMJMTXTL).	
f. If routing codes are present (WPLMCSFA set in WPLMCSF1), the routing codes (in WPLROUT) are converted to printable characters and moved to the Major WQE (WMJMRR). WMJMPAD1 and WMJMPAD2 are set to blanks.	
g. The 'Use Default Control Line' flag (XVXOUDCL in field XVX0 of the XV) is tested to determine if the Major WQE was built with the default control line. If not, control passes to step 13.	
Otherwise, the 'Control Line' flag (WMJMLTYA in WMJMLTYP of the Major WQE) is set. Since the Major WQE contains the default control line, the text (WPLTXT) in the parameter list is used to build line 1 of the Minor WQE and pointers are therefore adjusted. Control passes to step 12.	

Diagram 1-9. Multiple-Line WTO (MLWTO) Processing (SVC 35) (IEAVMWTO) (Part 21 of 24)

Input

XV
XVCMINOR

Minor WQE Address

Minor WQE

Major WQE
WMJMHCID

Hardcopy ID

From Steps 7 & 11

Process

19 Builds Minor WQE Line 1.

a. Obtains Minor WQE address.

b. Locates, moves, and tests the line type.

If there is an end line, control passes to Step 14. → Step 14

c. Compares line length to limit. If it exceeds limit, truncates text.

d. If authorized, moves text to text area + 1.

e. If not authorized, moves text to text area + 2.

f. Moves hardcopy ID to Minor WQE.

g. Resets first line build flags.

Control goes to Step 13. → Step 13

Output

Minor WQE
WMNMLT2

Line Type

WMNMTXT2

ƀ	Text
1	2

WMNMLT2

Line Type + 1

WMNMTXT2

ƀ		Text
1	2	3

WMNMHCT2

Line Type + 2

WMNMHCT2

Hardcopy ID

WMNMML2
WMNMML2H

....1

XV
XVD3
XVD3BLD2

.0.. ....

Extended Description	Module

19 To build Line 1 of a Minor WQE, the following is done:

a. The address of the Minor WQE is obtained (XVCMINOR).

b. Line type field is located, moved to Line One (WMNMLT1) and tested for an end line. If an end line, control passes to step 14.

c. If not an end line, the line length plus 4 is decreased by 4 and compared to the label/data limit (system default is 70 characters). If more than the limit, the text is truncated. The first position of the minor line 1 text area (WMNMTXT1) is set to a blank.

d. If the authorization flag (XVD1AUTH in XVD1) is set, the text is moved from the parameter list into Minor WQE Line 1 starting at position 2, thus increasing the length of the text field by one.

e. If the authorization flag is not set, the second position of Minor Line 1 text area (WMNMTXT + 1) is blanked. The second position of the text area is set to a blank and the text is moved in starting at the third position. The length of the text is increased by two and stored (WMNMTL1 in the Minor WQE).

f. The hardcopy ID is moved to the Minor WQE (from WMJMHCID into WMNMHCT1).

g. The 'Build Line 1' flag (XVD3BLD1 in XVD3) is reset and the 'Second Line Available' flag (WMNMML2H in WMNMML2) is set. New line control characters are set to blank. Control passes to step 13.

Diagram 1-9. Multiple-Line WTO (MLWTO) Processing (SVC 35) (IEAVMWTO) (Part 23 of 24)

Input

XV
XVCMINOR

Minor WQE Address

Minor WQE

Major WQE
WMJMHCID

Hardcopy ID

From Steps 7 & 11

Process

20 Builds Minor WQE Line 2.

a. Queue Line 2 to Line 1.

b. Locates, moves, and tests the line type.

If there is an end line, control passes to Step 14. → Step 14

c. Compares line length to limit. If it exceeds limit, truncates text.

d. If authorized, moves text to text area + 1.

e. If not authorized, moves text to text area + 2.

f. Moves hardcopy ID to Minor WQE.

g. Resets second line build flags.

Control goes to Step 13. → Step 13

Output

Minor WQE
WMNMNX1

WMNMUC2

Line 2

WMNMLT2

Line Type

WMNMTXT2

| 1 | 2 Text |

WMNMLT2

Line Type + 1

WMNMTXT2

| 1 | 2 | 3 Text |

WMNMLT2

Line Type + 2

WMNMHCT2

Hardcopy ID

WMNMML2
WMNMML2H

....1

XV
XVD3
XVD3BLD2

..0. ....

Diagram 1-9. Multiple-Line WTO (MLWTO) Processing (SVC 35) (IEAVMWTO) (Part 24 of 24)

Extended Description	Module
20 To build Line 2 of a Minor WQE:	IEAVMWTO
a. The address of the Minor WQE is obtained (XVCMINOR). The second line of the Minor WQE is queued to the first (WMNMNX1 contains the address of WMNMUC2).	
b. The line type field is located, moved to Minor WQE Line 2 (WMNMLT2); then test for an end (E) line. If an end line, the MLWTO is complete and control passes to step 14.	
c. The line length plus 4 and compared to the label/data line limit (system default is 70 characters). If greater than the limit, the text is truncated. The first position of the minor line 2 text area (WMNMTXT2) is set to a blank.	
d. If the authorization flag is set, the text is moved to Minor Line 2 starting at position two. The length of the text is increased by one.	
e. If the authorization flag is not on, the second position of the text area is set to a blank and the text is moved in starting at the third position. The length of the text is increased by two and stored (WMNMTL2 in the Minor WQE).	
f. The hardcopy ID is moved to the Minor WQE (from WMJMHCID into WMNMHCT2).	
g. The 'Build Second Line' flag (XVD3BLD2 in XVD3) and the 'Second Line Available' flag (WMNMML2H in WMNMML2) are reset. Control passes to step 12.	

Diagram 1-10. WTO and WTOR Communication Task Processing Overview (IEAVMQWR and IEAVMWSV) (Part 1 of 2)

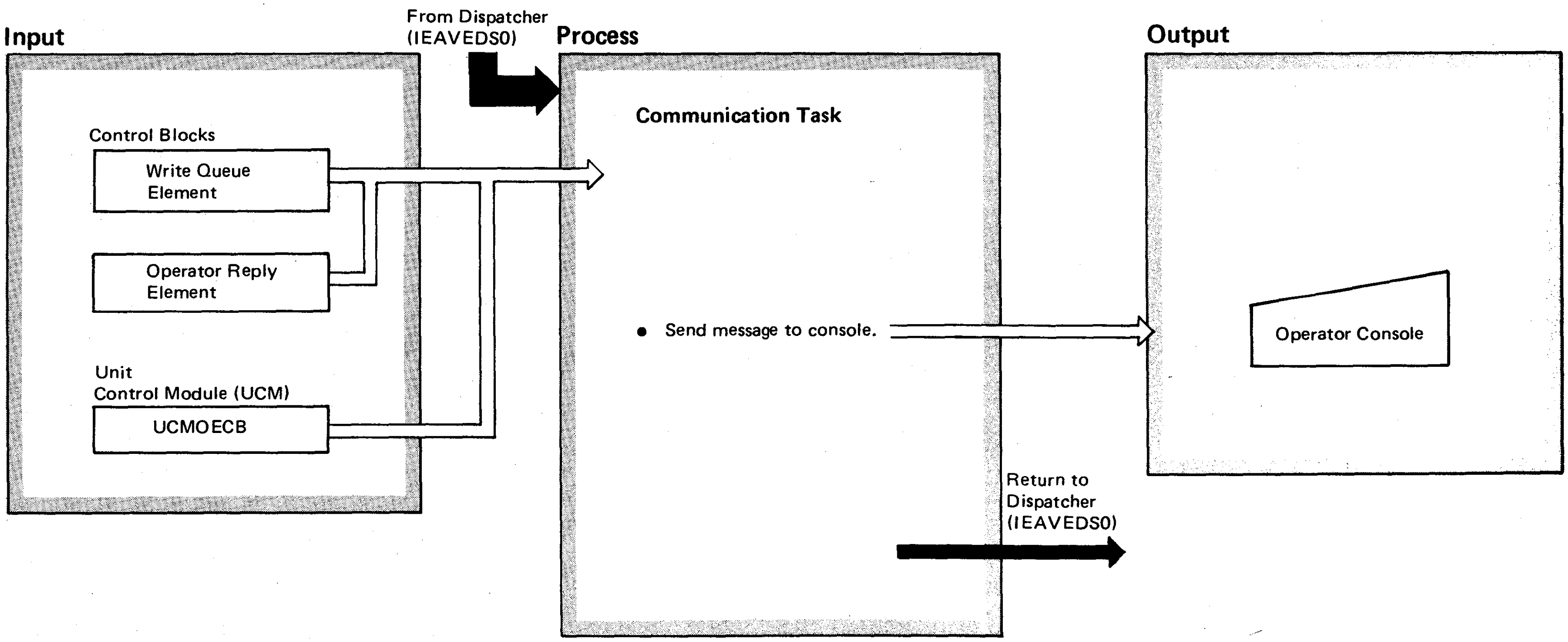


Diagram 1-10. WTO and WTOR Communication Task Processing Overview (IEAVMQWR and IEAVMWSV) (Part 2 of 2)

Extended Description	Module
The function of the WTO and WTOR communication task processing function is to accept the message associated with the WTO or WTOR macro instruction issued by the user of that macro instruction. In the process, this function prepares and chains a control block containing that message for the consoles, and then calls the device support processor (SVC 72).	IEAVMQWR
Prior to the current entry into the communication task's common processing modules, an entry was made into this module, possibly to print the NIP messages. At some point in the previous entry, the wait service routine (IEAVMQWR) determined either that no further communication task processing was needed or that no further processing could be accomplished at that time. Following that determination, the wait service routine issued a WAIT macro instruction and returned control to the dispatcher (IEAODS).	
Following the WTO or WTOR macro instruction processing, the UCMOECB event control block was set and a new write queue element (WQE) had been placed on the WQE chain. Processing as a result of the UCMOECB actually begins when the dispatcher gives control to the wait service routine in step 2.	
A diagram showing the relationship among the control blocks used by this function is shown in Figure 5-1.	

Diagram 1-11. WTO and WTOR Communication Task Processing (IEAVMQWR and IEAVMWSV) (Part 1 of 16)

Input

Unit Control Module (UCM)

- UCMPF
- UCMSYSJ
- UCMOECB

From Step 2

From Dispatcher (IEAVEDS0)

From Steps 32 & 33

Process

IEAVMQWR (Wait Service Routine)

1 Wait for work to do. → WAIT Macro Instruction

2 Determine the operation to be performed:

a) If alternate CPU recovery, see the diagram, Alternate CPU Recovery (ACR) (IEAVTACR) in Recovery Termination Management.

b) If external interrupt, see the diagram, External Interrupt Processing (IEAVVCRX).

c) If attention interrupt, see the diagram, Attention Interrupt Processing (IEAVVCRA).

d) If I/O interrupt, see the diagram, I/O Complete Processing.

e) If console output pending, then branch to → Branch to IEAVMDSV, Step 25

f) If hardcopy output pending, then branch to → Branch to IEAVMDSV, Step 25

g) If new message for output, then branch to → Branch to IEAVMWSV, Step 3

h) If WQEs are no longer needed, then branch to → Branch to IEAVMDSV, Step 25

i) If operator messages are to be deleted via DOM, see the diagram, DOM Communications Task Processing.

j) If NIP messages are to be written to hard copy, see the *System Initialization Logic* PLM for the diagram, NIP Communications Task Initialization.

When no work exists → Step 1

Extended Description **Module**

1 During some previous operation, the wait service routine issued the WAIT macro instruction after it had determined there was no further work the communication task could perform at that time. The communication task, therefore, is placed in a wait state.

2 Determine the operation to be performed and branch to the module that can perform that operation.

e) There are two conditions that turn on the console output pending bit:

1. If a single message is sent to more than one console, this bit is set during initial queueing of that message to the consoles. This bit signifies that there may be messages queued for consoles which were not written to those consoles on the last pass through the communication task.
2. If the console was busy during the first attempt to transmit the message to that console, this bit is set to permit the message to be sent to the console when it is not busy.

f) When a message is queued and ready to be sent to a data set, this bit is set. This bit is *not* set when the hardcopy output is sent to a regular console device.

g) When the UCMOECB is set, a WTO or WTOR macro instruction has just completed processing. A write queue element (WQE) containing the message to be sent to the console has been built and placed on the WQE chain. If the ECB was set by a WTOR macro instruction, an operator reply element (ORE) was also built and placed on the ORE chain.

h) When the UCMSYSI is set, it is requesting cleanup of the WQE chain, eliminating WQEs that are no longer needed.

Diagram 1-11. WTO and WTOR Communication Task Processing (IEAVMQWR and IEAVMWSV) (Part 3 of 16)

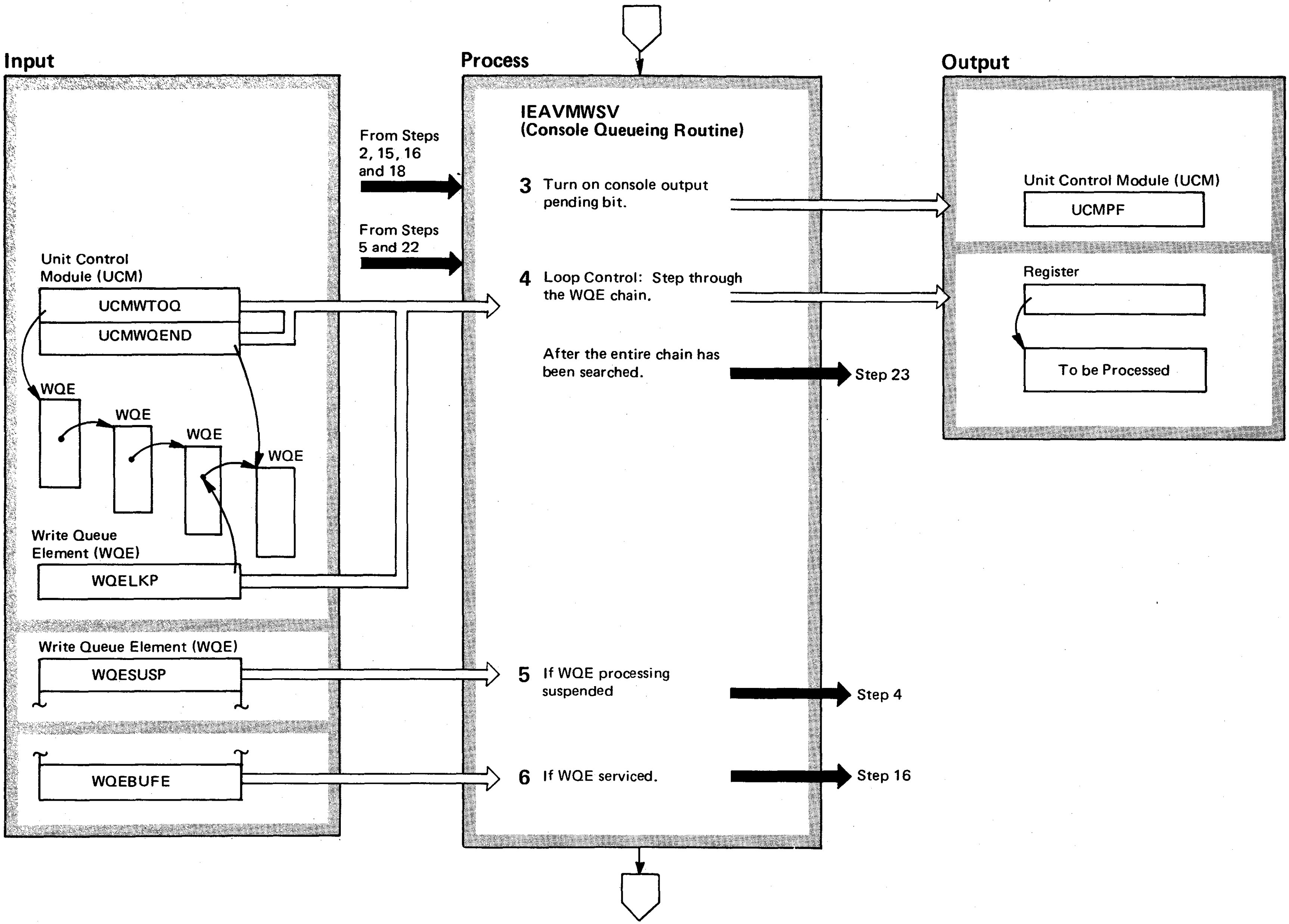
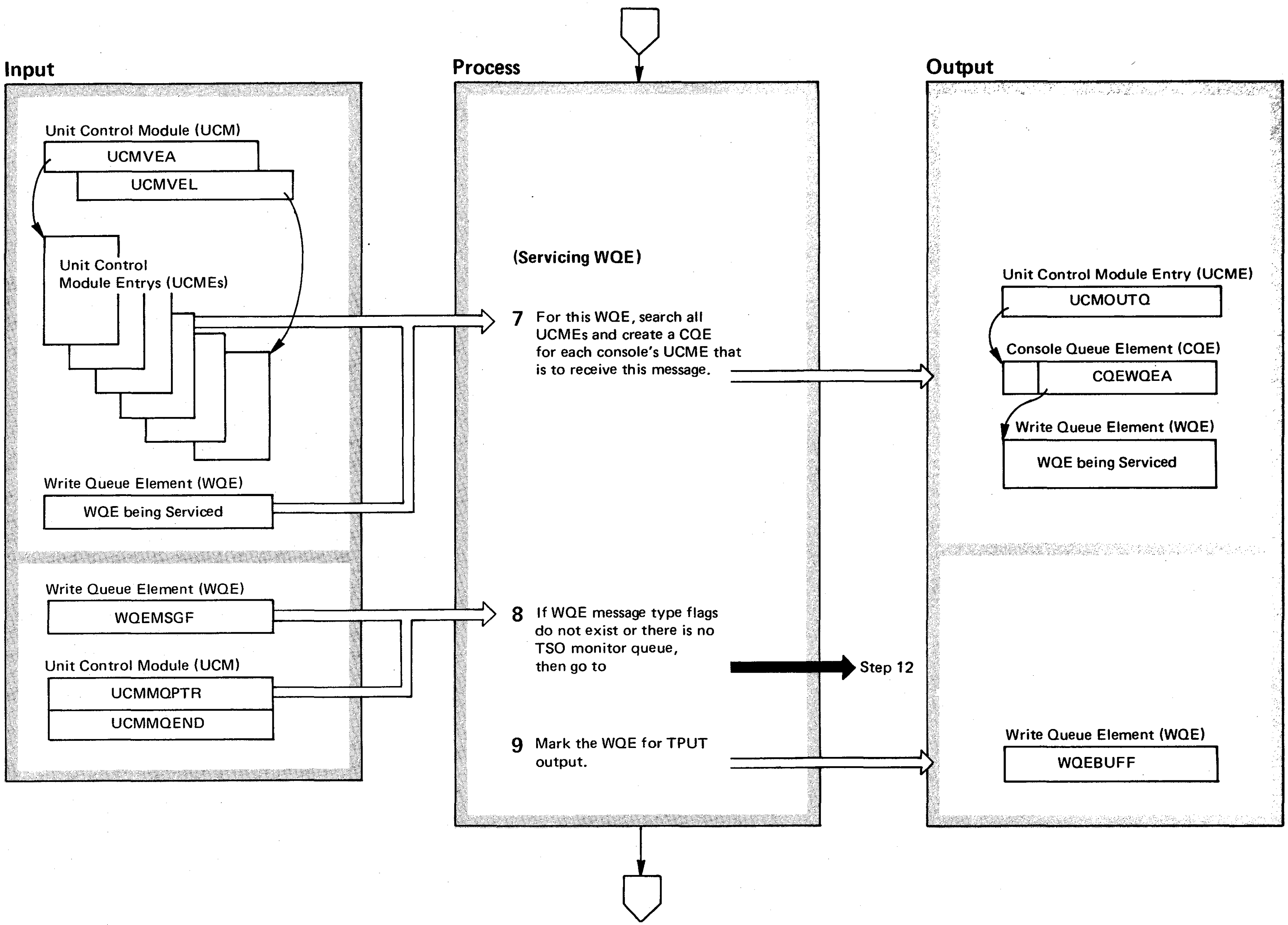


Diagram 1-11. WTO and WTOR Communication Task Processing (IEAVMQWR and IEAVMWSV) (Part 4 of 16)

Extended Description	Module
3 The console output-pending bit in the unit control module is turned on. It may be used later to transfer control within the communication task.	IEAVMWSV
4 To find the write queue elements (WQEs) needing service, the entire chain of WQEs is searched one by one.	
5 A WQE with the processing-suspended bit on is not yet ready to be queued to a console. Since the WQE can't be queued, control is given to the loop control to find the next WQE. A WQE is suspended by the WTO and WTOR macro instruction processing routine while the message in that WQE is being examined by the subsystem exit routine.	
6 If the processing-suspended bit is off, the console queueing routine determines if the WQE has been serviced. If it has, the following possibilities exist: • The WQE has already been queued to a console. • The WQE is a major multiple line WQE (MLWTO) where at least the first line of the multiple line message has been sent to the console; however, an additional line may have been added to the multiple line message. If a line has been added, the major WQE bit WMJMMLWD has been turned on. If this bit is on, turn on the output-pending bit in the CQE that was built for the major WQE. These two possibilities must be tested before control can be given back to the loop control to find the next WQE on the WQE chain to be serviced.	

Diagram 1-11. WTO and WTOR Communication Task Processing (IEAVMQWR and IEAVMWSV) (Part 5 of 16)



Extended Description	Module

7 The routing codes assigned to each console are indicated in each console's unit control module entry (UCME). These routing codes are compared against the routing codes in each write queue element (WQE). When there is a match, a console queue element (CQE) is created and chained to the UCME for that console. The CQE contains a pointer to the WQE. If REGO or QREGO issued the message, the console ID in the WQE is compared with the console ID in each unit control module element (UCME). When there is a match, a CQE is created for that console.

When a console is being used for hardcopy output, a CQE is created for that console, and the CQE points to that WQE as if the console had the routing codes that agree with those in the WQE.

Note: When a message contained in one WQE is sent to several consoles, a CQE is built and chained to the CQE chain from each console's UCME. All of the CQEs point to the one WQE.

The CQE is a one word control block. The first byte contains control bits – end of block, queued for hardcopy, pointer to next block, etc. The three remaining bytes contain the address of the WQE.

CQEs are created as a group of six contiguous CQEs The first five CQEs point to WQEs, when a WQE is queued for this console. The sixth CQE, instead of pointing to a WQE, points to the next group of six CQEs.

When a CQE is needed, this routine attempts to locate an empty WQE pointer field in an existing group of CQEs. If a CQE is found, the control bits are set and the WQE address is placed in the pointer field. If a CQE does not exist or the five WQE pointer fields in the CQE group are filled, then a new group of six CQEs is created and the WQE pointer of the sixth CQE in the original CQE group is set to point to the new CQE group.

Extended Description	Module

8 Some WTO and WTOR macro instructions are issued with the optional MSGTYPE flags. These flags cause the message to be sent to TSO terminals and operator consoles in MONITOR mode, provided these terminals and consoles are monitoring that type of message. For the WQE to be queued for a TSO terminal, the flags must exist in the WQE.

9 The WQE is marked for TPUT output.

Diagram 1-11. WTO and WTOR Communication Task Processing (IEAVMQWR and IEAVMWSV) (Part 7 of 16)

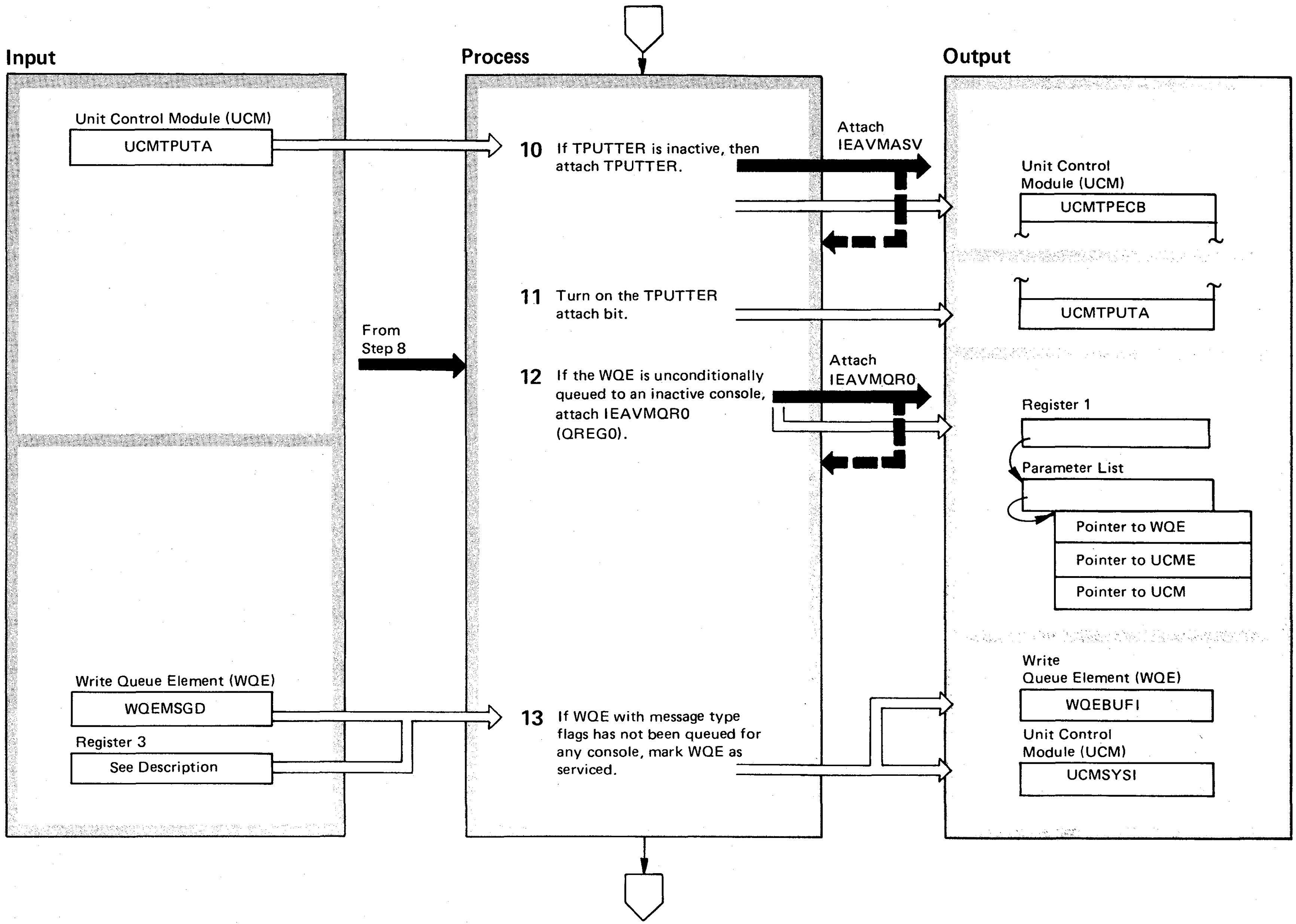


Diagram 1-11. WTO and WTOR Communication Task Processing (IEAVMQWR and IEAVMWSV) (Part 8 of 16)

Extended Description	Module
10 If the TPUTTER routine is inactive, attach it to send the message (contained in the WQE being processed) to the TSO terminals monitoring that type of message.	IEAVMASV
11 The TPUTTER attach bit is turned on to prevent attaching the TPUTTER again before it has completed its current operation. This bit is turned off by TPUTTER (IEAVMASV) when it is through sending the current messages.	IEAVMWSV
12 If the WQE is to be unconditionally queued to an inactive console, the QREGO routine is attached. QREGO prevents the possibility of the regularly allocated WQE and ORE space being filled with messages destined for an inactive console. (See QREGO — Unconditional Message to Inactive Console.) When control is returned, three possibilities exist:	IEAVMQRO

- The WQE has been requeued to the master console:
  - The CQE pointing to this WQE when control was passed to the QREGO routine has been deleted.
  - A new CQE on the master console's CQE chain now points to this WQE.
- The WQE is to be deleted.
- The WQE remains queued to the inactive console.

The first two conditions mark the WQE queued for hardcopy before control is returned.

Extended Description	Module
13 There are a number of operating system messages that are issued specifically for TSO terminals that might be in MONITOR mode; however, the system may be without a TSO terminal or operator console in monitor mode, or a TSO terminal monitoring the type of message contained in this WQE. When this condition occurs, the WQE is marked as serviced. Later, the WQEs that have been serviced are deleted.	IEAVMWSV

Register 3 contains a count of the number of consoles (CQEs) for which this WQE has been queued.

Diagram 1-11. WTO and WTOR Communication Task Processing (IEAVMQWR and IEAVMWSV) (Part 9 of 16)

Input

Unit Control
Module Entry (UCME)

WMJMMLWB
WMJMMLWD

From
Step 6

Process

14 Mark WQE serviced.

15 Service next WQE. → Step 3

16 If a minor WQE has not been added to a major WQE. → Step 3

(MLWTO Minor WQE Processing)

17 The output pending bit is turned on for every console that is to receive the message contained in the minor WQE.

Output

Write
Queue Element (WQE)

WQEBUFC

Unit Control
Module Entry (UCME)

UCMPF

Diagram 1-11. WTO and WTOR Communication Task Processing (IEAVMQWR and IEAVMWSV) (Part 10 of 16)

Extended Description | **Module**

14 The processing to be performed by the console queueing routine for this WQE is complete.

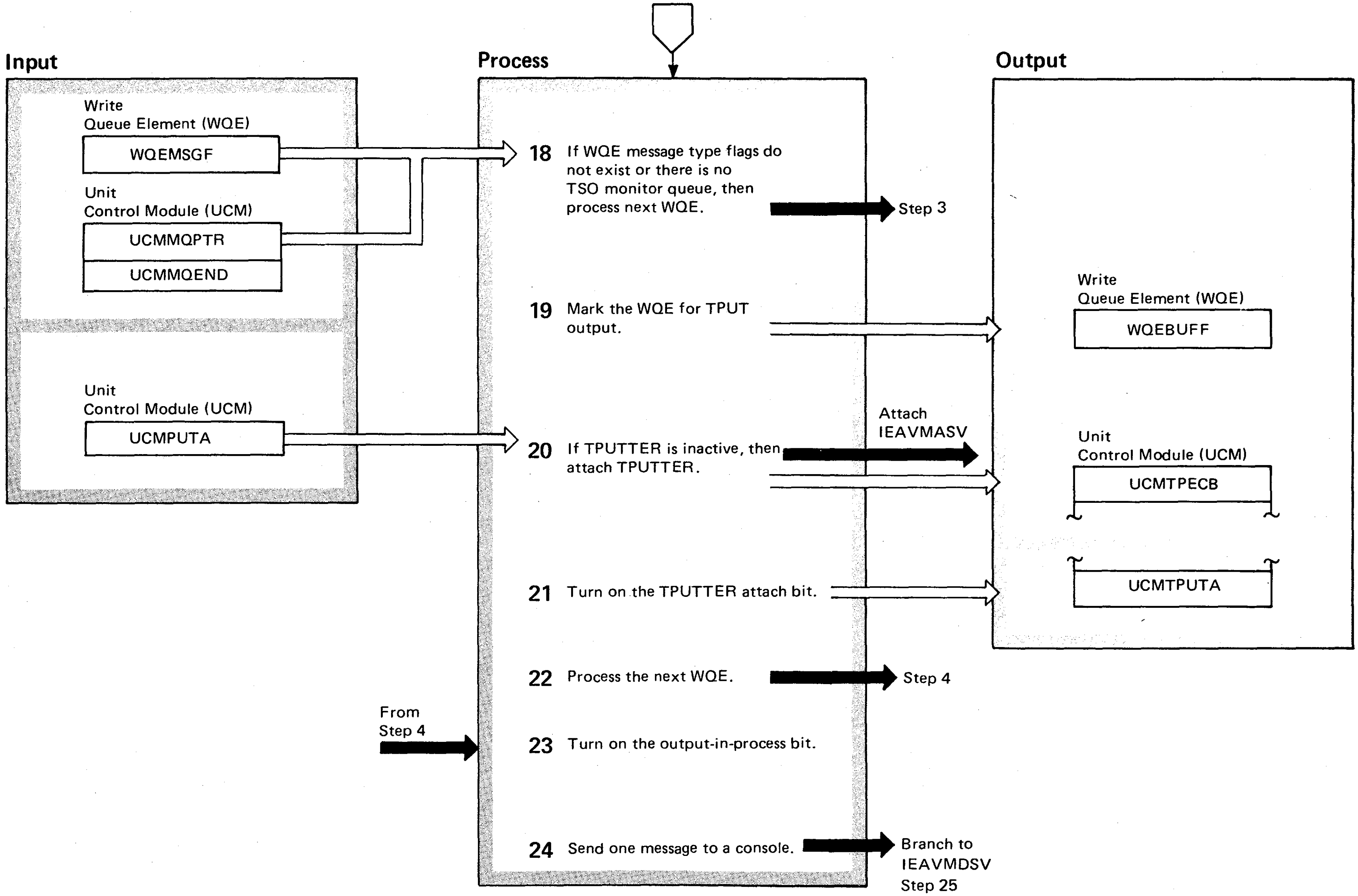
15 Service the next WQE.

16 If the WQE being processed is a major WQE, then this WQE has been previously processed by the console queueing routine; therefore, service the next WQE.

If the WQE chain altered flag (WMJMMLWD) is on, at least one line has been added to a minor WQE chained to a major multiple line WQE that has been previously serviced. The output-pending bit in the UCME for each console that receives this message is turned on. If either this WQE is not a major WQE or the WQE chain altered flag of a major WQE is off, then the next WQE on the WQE chain is serviced.

17 The output-pending bit is turned on so that the message contained in this WQE will be sent to a console.

Diagram 1-11. WTO and WTOR Communication Task Processing (IEAVMQWR and IEAVMWSV) (Part 11 of 16)



Extended Description | **Module**

18 Some WTO and WTOR macro instructions are issued with the optional MSGTYPE flags. These flags cause the message to be sent to TSO terminals and operator consoles in MONITOR mode, provided these terminals and consoles are monitoring that type of message. For the WQE to be queued for a TSO terminal, the flags must exist in the WQE.

19 The WQE is marked for TPUT output.

20 If the TPUTTER routine is inactive, attach it to send the message (contained in the WQE being processed) to the TSO terminals monitoring that type of message.

21 The TPUTTER attach bit is turned on to prevent attaching the TPUTTER before it has completed its current operation. This bit is turned off by TPUTTER (IEAVMASV) when it is through sending the current messages.

22 Process the next WQE.

23 The output-in-process bit is turned on for use by the wait service routine. This bit indicates that output processing has been started to at least one console but has not been started to all consoles.

24 At this point, the WQE that was being processed has been queued to all of the consoles to which the message contained in that WQE will be sent. This one message will be sent to one of the consoles to which it is queued. Control will be passed to the device service routine. Normally, when this routine is called, it attempts to send all of the messages queued for all consoles. Only one message will be sent each time the console queueing routine is called.

Diagram 1-11. WTO and WTOR Communication Task Processing (IEAVMQWR and IEAVMWSV) (Part 13 of 16)

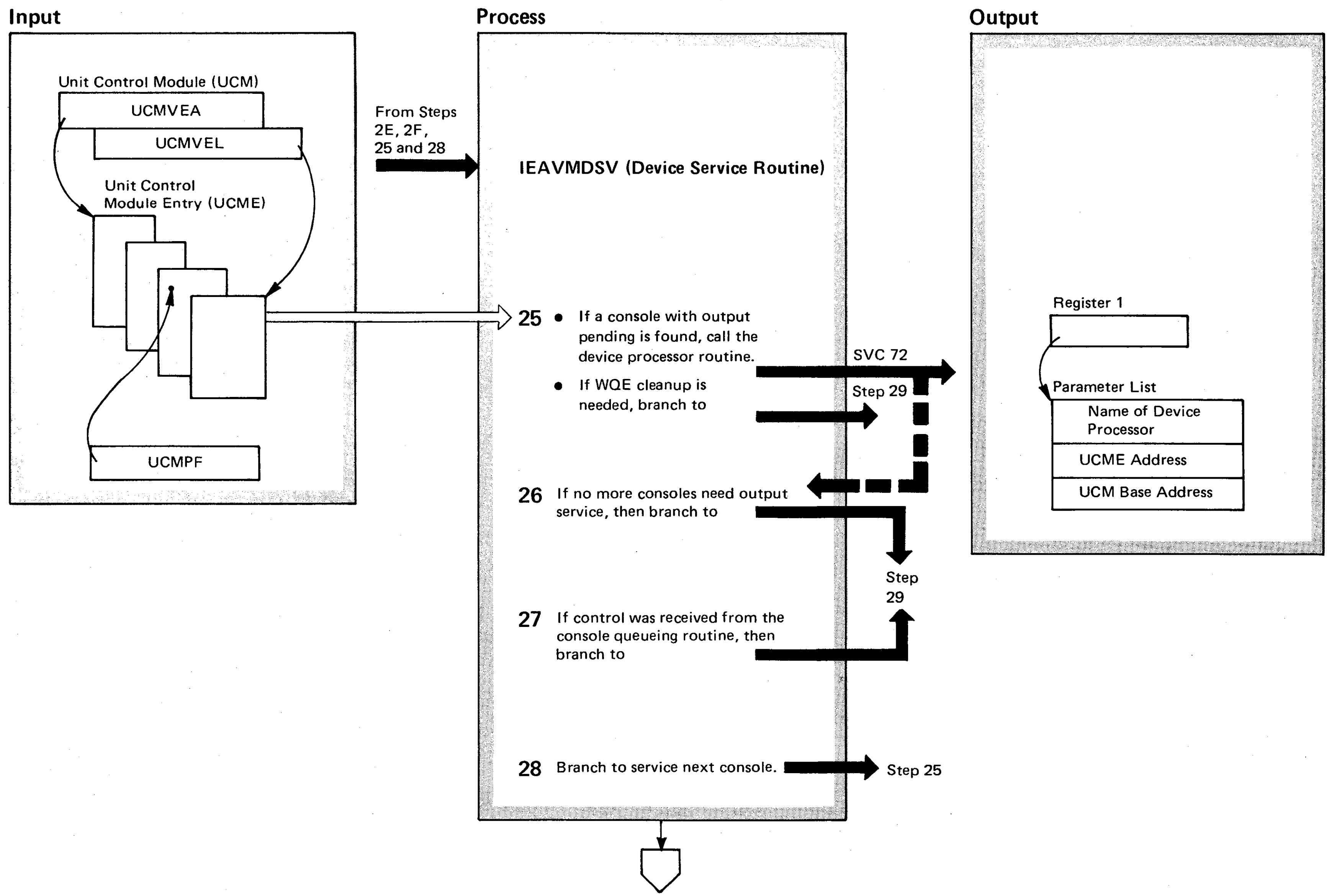


Diagram 1-11. WTO and WTOR Communication Task Processing (IEAVMQWR and IEAVMWSV) (Part 14 of 16)

Extended Description | **Module**

25 A search is made for a console that has output pending. When found, SVC 72 is issued to pass control to the device controller routine which eventually passes control to a particular device support routine that sends the message to the console device. SVC 72 is called once for each message sent to each console, see Writing Messages to a Console diagram.

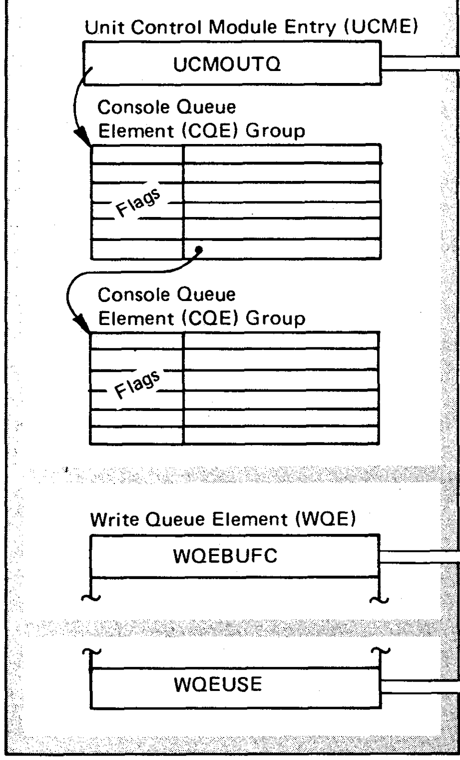
26 After all of the consoles have been serviced that could be serviced, control branches to the queue cleanup and hardcopy control section of the device service routine.

27 If control was received from the console queueing routine, the purpose of branching to the device service routine was to display the message just queued to one console; therefore, having called SVC 72 once, the purpose of this function has been fulfilled.

28 If neither of the previous two conditions exist, there may be other output messages pending for display at the console. The loop is, therefore, repeated until all messages that can be displayed have been displayed.

Diagram 1-11. WTO and WTOR Communication Task Processing (IEAVMQWR and IEAVMWSV) (Part 15 of 16)

Input



Process

From Steps 26 and 27

29 Consolidate CQE entries.

30 If message is to be hardcopied, issue SVC 36

31 Delete WQE when message has been sent to all consoles.

32 Return to caller.

a. If called by the console queuing routine. — Branch to IEAVMWSV Step 33

b. If called by the wait service routine. — Branch to IEAVMQWR Step 2

IEAVMWSV (Console Queueing Routine)

From Step 32

33 Return to caller. — Branch to IEAVMQWR Step 2

Output

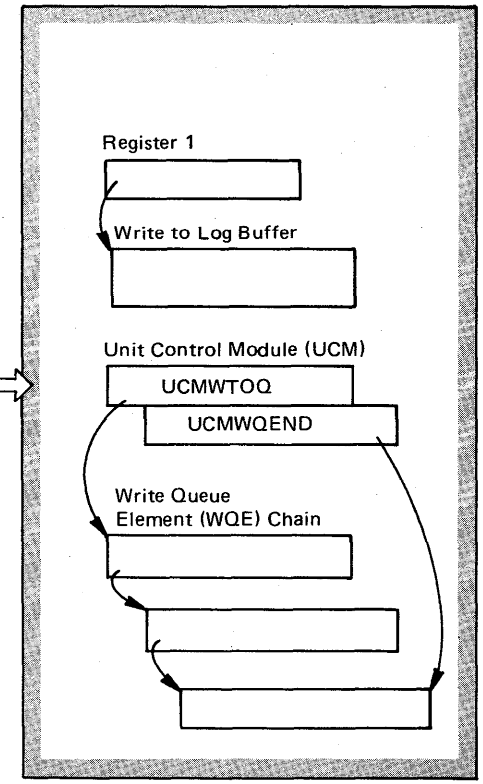


Diagram 1-11. WTO and WTOR Communication Task Processing (IEAVMQWR and IEAVMWSV) (Part 16 of 16)

Extended Description	Module

29 Having displayed at least one message, there is at least one console queue element (CQE) that is no longer needed. The CQE flags are checked to determine which ones are no longer needed. CQEs, however, are created in groups of six. The first five CQEs point to WQEs that are to be displayed by that console. The sixth CQE points to the next group of CQEs. CQEs are not freed until the first five CQEs in the group are no longer needed. At that time, the necessary pointers are changed and a FREEMAIN macro instruction is issued to relinquish the CQE group.

30 A test is made of the WQE chain to determine those messages that are to be hardcopied. Each time such a message is found, SVC 36 is issued.

31 The WQE chain is searched for those WQEs that can be deleted. The appropriate pointers are changed and the WQE is deleted via a FREEMAIN macro instruction.

32 Return to caller. The return is always to the routine that called the device service routine.

33 If the device service routine was called by the console queueing routine, control is returned to the wait service routine via the console queueing routine. There is no further processing in the console queueing routine.

Diagram 1-12. Unconditional Message to Inactive Console – QREG0 (IEAVMQR0) (Part 1 of 6)

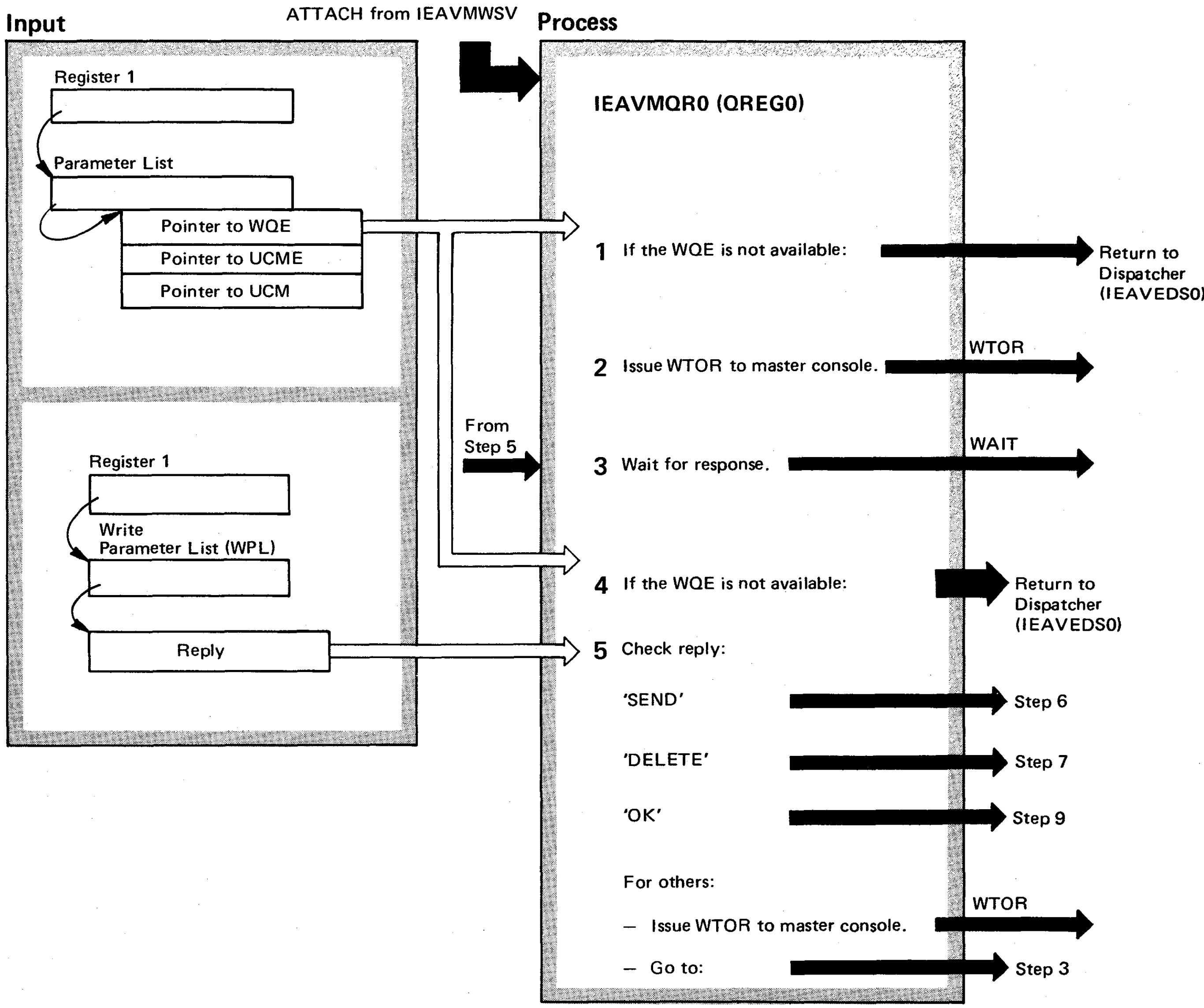


Diagram 1-12. Unconditional Message to Inactive Console – QREG0 (IEAVMQR0) (Part 2 of 6)

Extended Description	Module
Control is passed to this routine only when all of the following conditions exist: • A WTO or WTOR macro instruction has been issued with a QREG0 parameter. • When the WTO or WTOR macro instruction was issued, the console identified in register 0 exists but is *in*active. This routine issues WTOR messages IEA962A and IEA963A to inform the master console operator that an unconditional message is being processed for an inactive console. The operator responses are: SEND — Causes the unconditional message to be rerouted to the master console. DELETE — Causes the unconditional message to be deleted from the system. If the unconditional message is a WTOR message, the user eventually receives a D23 ABEND. OK — Causes queueing of the unconditional message to continue. The SEND and DELETE responses also cause the unconditional message to be sent to the hardcopy log, when the hardcopy log is active.	IEAVMQR0

Extended Description	Module
1 Upon entry and after each wait, this routine determines whether the WQE passed to it initially is still active. The WQE could have been removed by: • The inactive console being activated causing the message to be displayed at the specified console. • The issuing task could have been terminated whereby the task termination routine would eliminate the WQE. • A DOM macro instruction for a graphic terminal could have been issued to eliminate the WQE.	IEAVMQR0
2 WTOR message IEA962A is sent to the master console.	
3 Wait for the operator's response to the WTOR message.	
4 Following the wait, repeat the same determination that was made in step 1.	
5 Test the reply from the WTOR message for one of the three acceptable responses. If an unacceptable response was received, this routine sends WTOR message IEA963A to the master console. Go back to step 3 and wait for the operator's response. Message IEA963A will be repeated until an acceptable response is received from the console operator.	

Diagram 1-12. Unconditional Message to Inactive Console – QREG0 (IEAVMQR0) (Part 3 of 6)

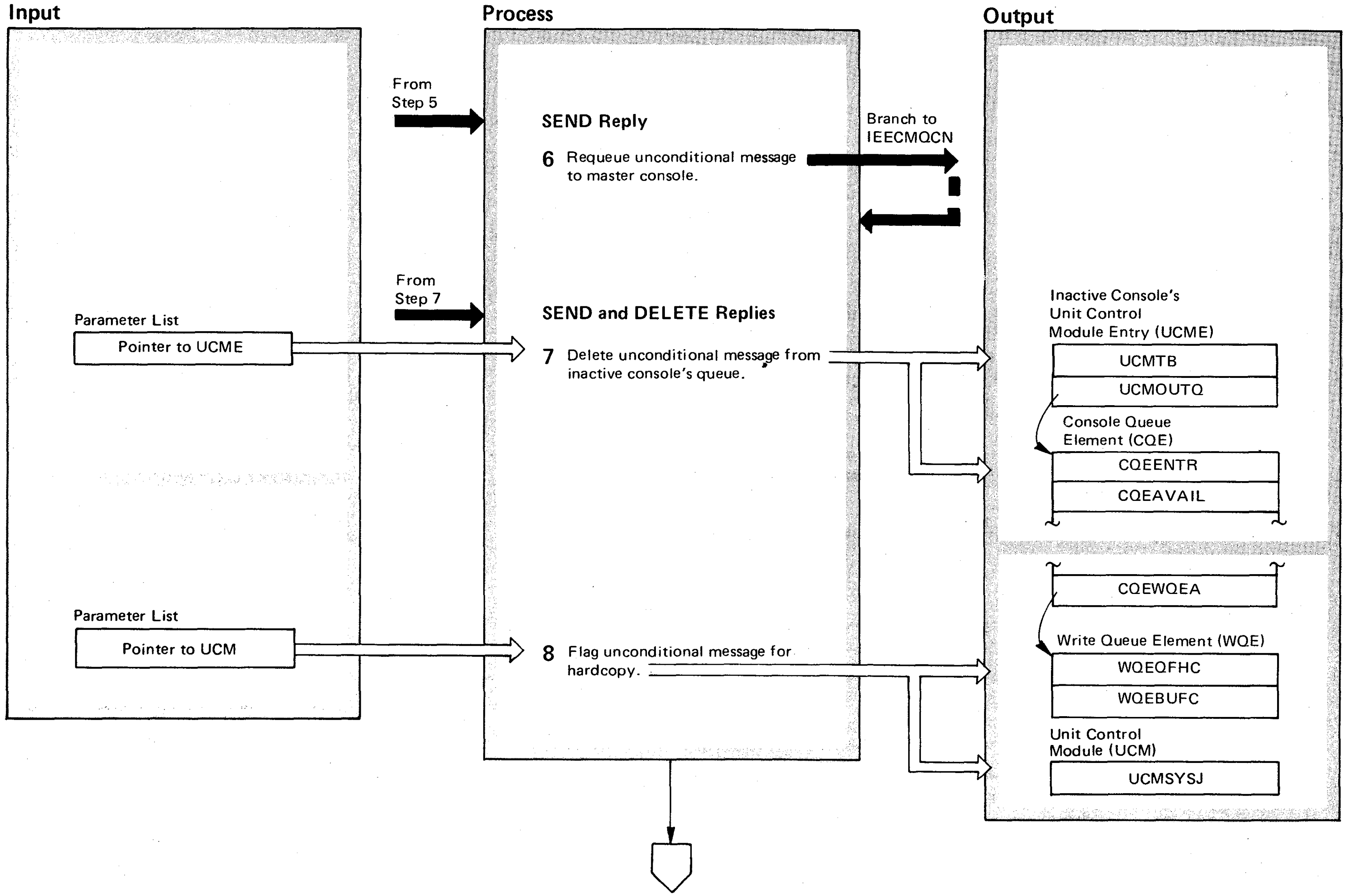


Diagram 1-12. Unconditional Message to Inactive Console – QREG0 (IEAVMQR0) (Part 4 of 6)

Extended Description	Module
6 As a result of the SEND response from the master console operator, this routine branches to IEECMQCN. IEECMQCN builds a console queue element (CQE) entry on the master console's CQE chain for the unconditional message.	
7 As a result of either a SEND or DELETE response from the master console operator, this routine removes the CQE entry for the unconditional message from the inactive console's CQE chain.	
8 As a result of the SEND or DELETE response from the master console operator, the WQE for the unconditional message is flagged for the hardcopy log.	

Diagram 1-12. Unconditional Message to Inactive Console – QREG0 (IEAVMQR0) (Part 5 of 6)

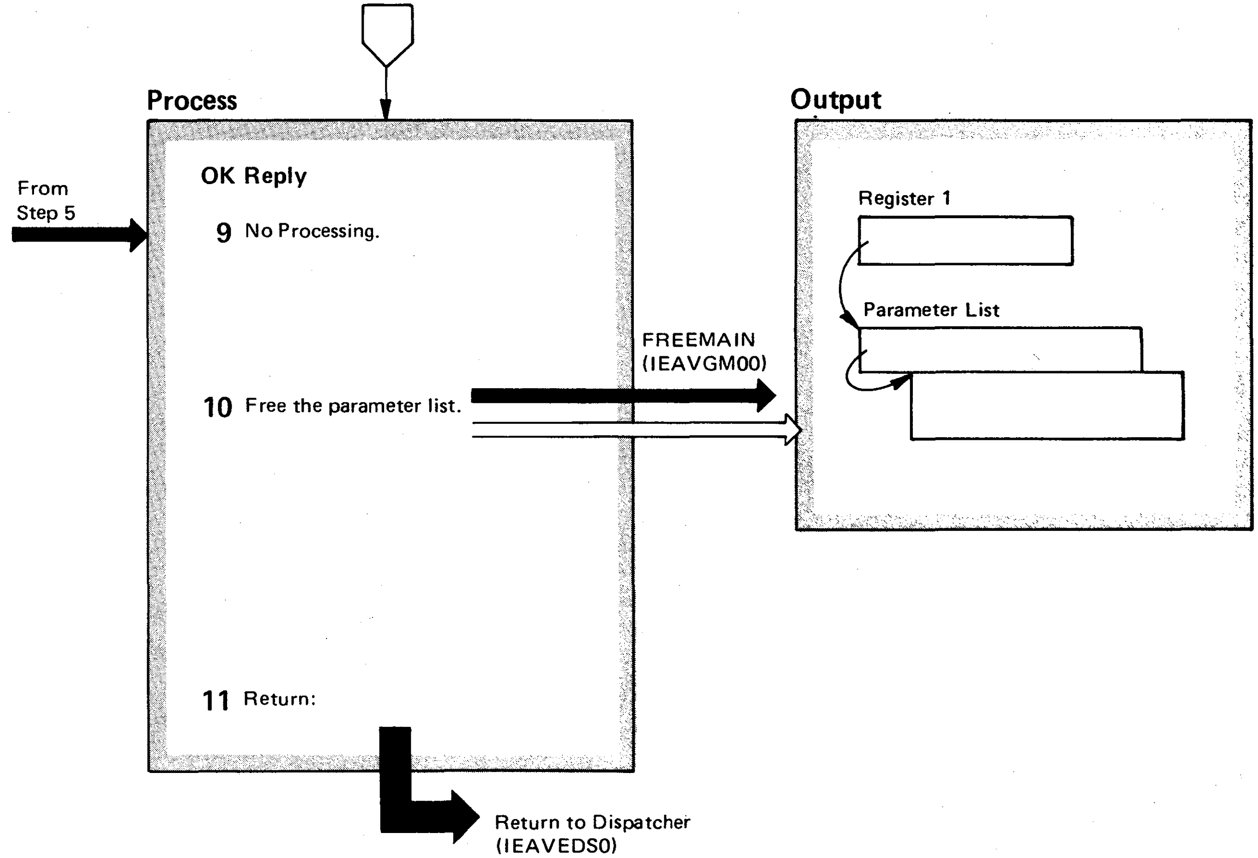


Diagram 1-12. Unconditional Message to Inactive Console – QREG0 (IEAVMQR0) (Part 6 of 6)

Extended Description	Module
9 As a result of the OK response from the master console operator, no processing is done by this routine.	
10 Before returning to the dispatcher, the parameter list passed to this routine is freed.	
11 Return to the dispatcher.	

Diagram 1-13. Writing Single-line Messages to a 1052, 1443, 2740, or 3284/3286 Console (Part 1 of 2)

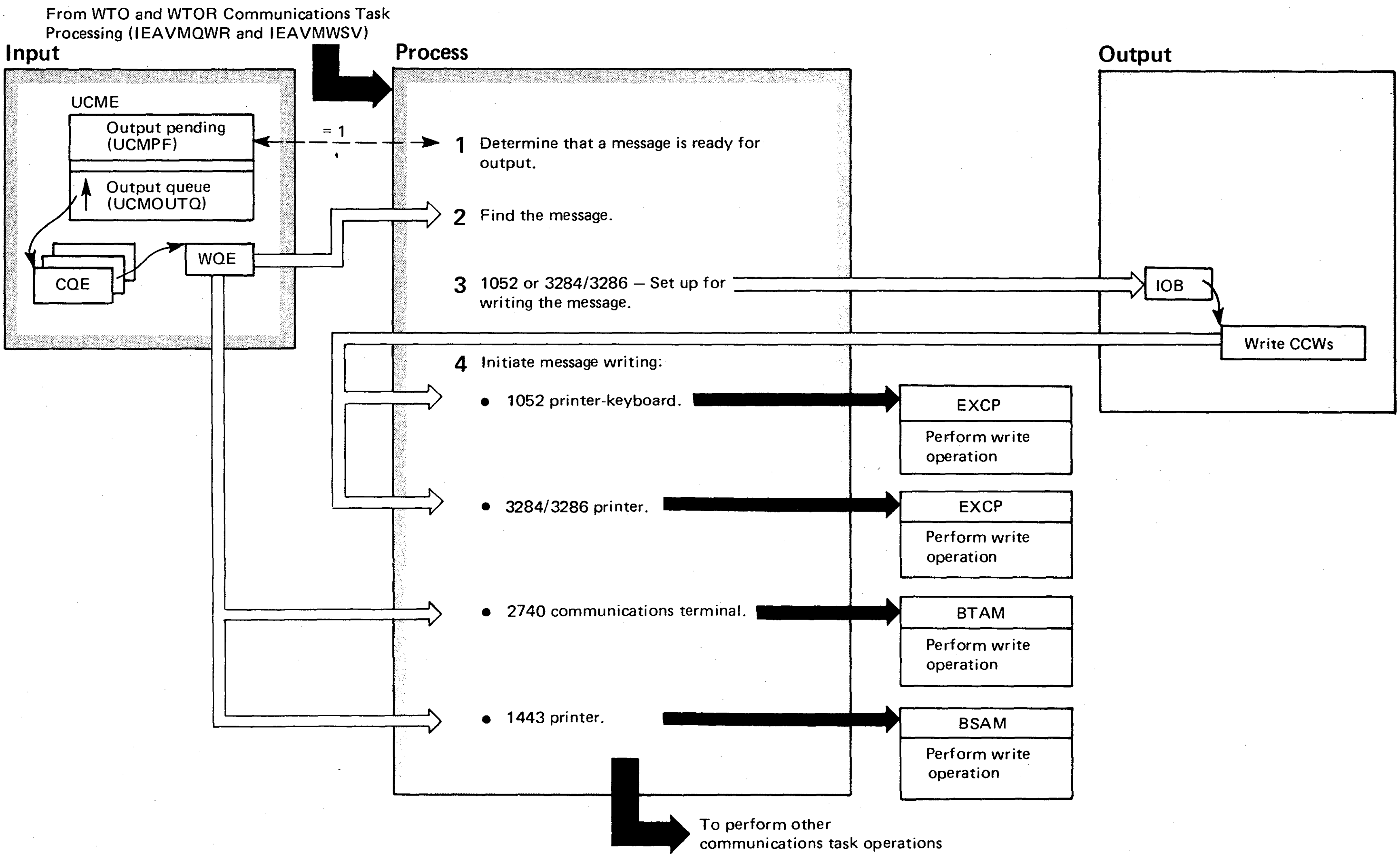


Diagram 1-13. Writing Single-line Messages to a 1052, 1443, 2740, or 3284/3286 Console (Part 2 of 2)

Extended Description	Module	Label
User programs and system routines issue a WTO or WTOR macro instruction to send messages to the operator's consoles. The following communications task device support processors (DSPs) write single-line messages to consoles:		
• For a 1052 printer-keyboard, 3210 console printer-keyboard, 3215 console printer-keyboard, and 3213 console printer, the DSP is IEAV1052.	IEAV1052	
• For a 1443 printer, 1403 printer, and 3211 printer, the DSP is IEAV1443.	IEAV1443	
• For a 2740 communications terminal, IEEC2740.	IEEC2740	
• For a 3284/3286 printer, IEECVETW.	IEECVETW	
1 The appropriate DSP checks the output-pending bit (UCMPF) to determine that a message is to be written to the console.	(See above)	
2 The DSP searches the CQEs for a pointer to a WQE that contains a message for processing.		
3 For a 1052 or 3284/3286, the appropriate DSP obtains an IOB for the write operation and places the addresses of the write CCWs into the IOB.		
4 The DSPs initiate message writing:		
• For the 1052 printer-keyboard, the 1052 DSP issues an EXCP macro instruction to execute the write channel program.	IEAV1052	PMEXCP
• For the 1443 printer, the 1443 DSP issues a WRITE macro instruction to pass control to BSAM, which writes the message.	IEAV1443	PMEXCP
• For the 2740 communications terminal, the 2740 DSP issues a WRITE macro instruction to pass control to BTAM, which writes the message.	IEEC2740	
• For the 3284/3286 printer, the 3284/3286 DSP issues an EXCP macro instruction to execute the write channel program.	IEECVETW	

Diagram 1-14. Displaying Single-line Messages on Graphics Consoles (DIDOCS) (Part 1 of 2)

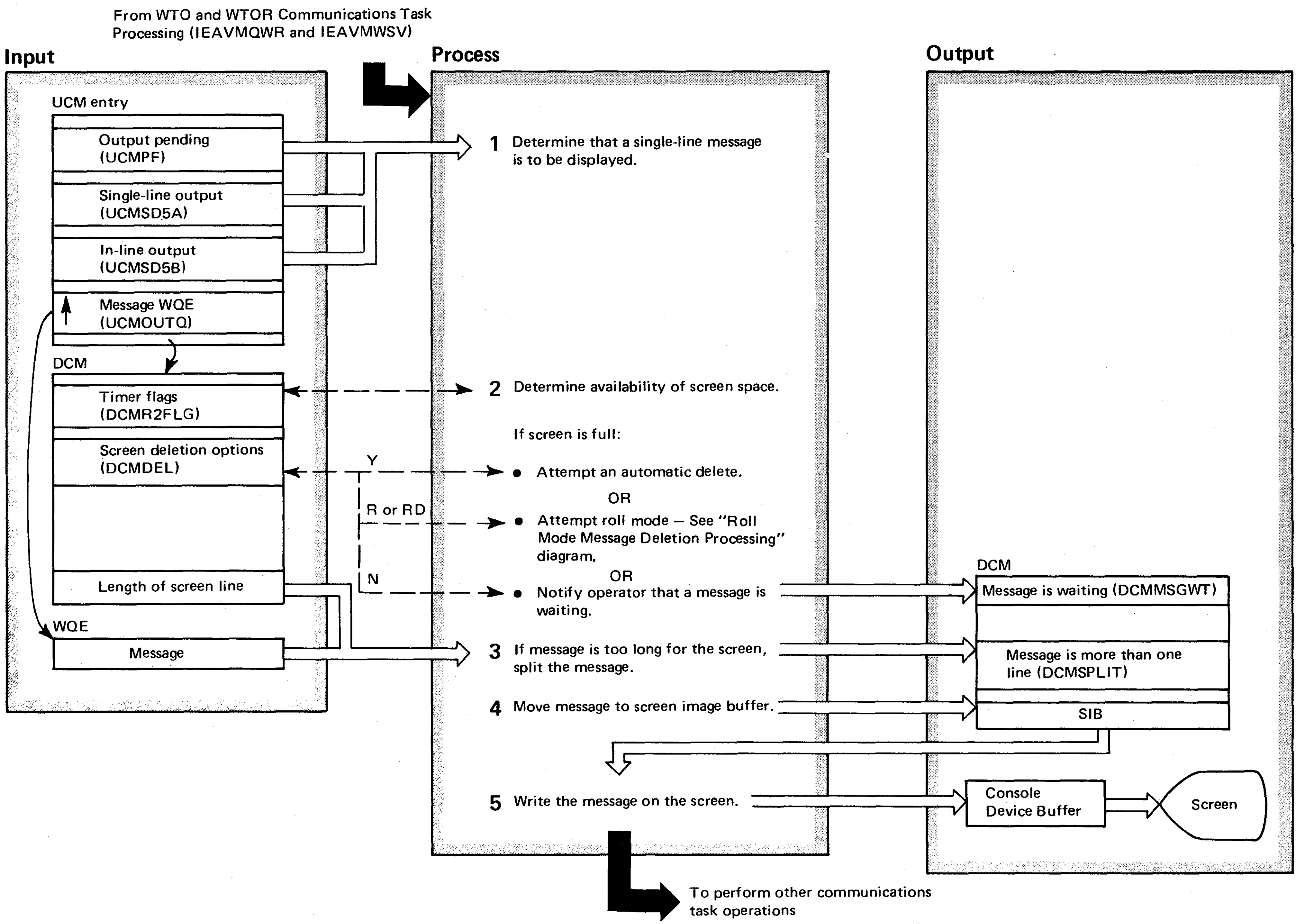


Diagram 1-14. Displaying Single-line Messages on Graphics Consoles (DIDOCS) (Part 2 of 2)

Extended Description	Module	Label
User programs and system routines issue a WTO or WTOR macro instruction to send messages to the operator's consoles. DIDOCS displays single-line messages on a graphics console as follows:		
1 DIDOCS checks the output-pending bit (UCMPF) and the in-line output bit (UCMSDS5B); if both bits are on and UCMSDS5A is off, DIDOCS has received control to display an in-line, single-line message.	IEECVET1	
2 DIDOCS checks the DCMR2FLGS flags to determine whether sufficient screen space is available. If no space is available, DIDOCS attempts to clear the screen in accordance with the screen deletion options (DCMDEL):	IEECVET2	
• If automatic deletion is in effect, DIDOCS attempts to remove deletable (flagged) messages from the screen. If no deletable messages exist, DIDOCS sets the message-waiting bit (DCMMSGWT) to indicate that the MESSAGE WAITING message should be issued to the operator.	IEECVET9	
• If roll mode is in effect, DIDOCS attempts to roll the screen, as described in diagram "Roll Mode Message Deletion Processing (DIDOCS)."	IEECVETJ	
• If automatic deletion is not in effect, DIDOCS sets bit DCMMSGWT to issue MESSAGE WAITING to the operator. The operator must then use the CONTROL command or a light pen to delete messages.	IEECVETD	
3 DIDOCS compares the message length with the length of the screen line. If the message is longer, DIDOCS sets bit DCMSPLIT, then splits the message.	IEECVFT2	
4 DIDOCS moves the message into the screen image buffer (SIB).	IEECVFT2	
5 DIDOCS writes the SIB to the console device buffer using EXCP.	IEECVETH/P/R/U	

Diagram 1-15. Writing Multiple-line Messages to a 1052, 1443, 2740, or 3284/3286 Console (Part 1 of 4)

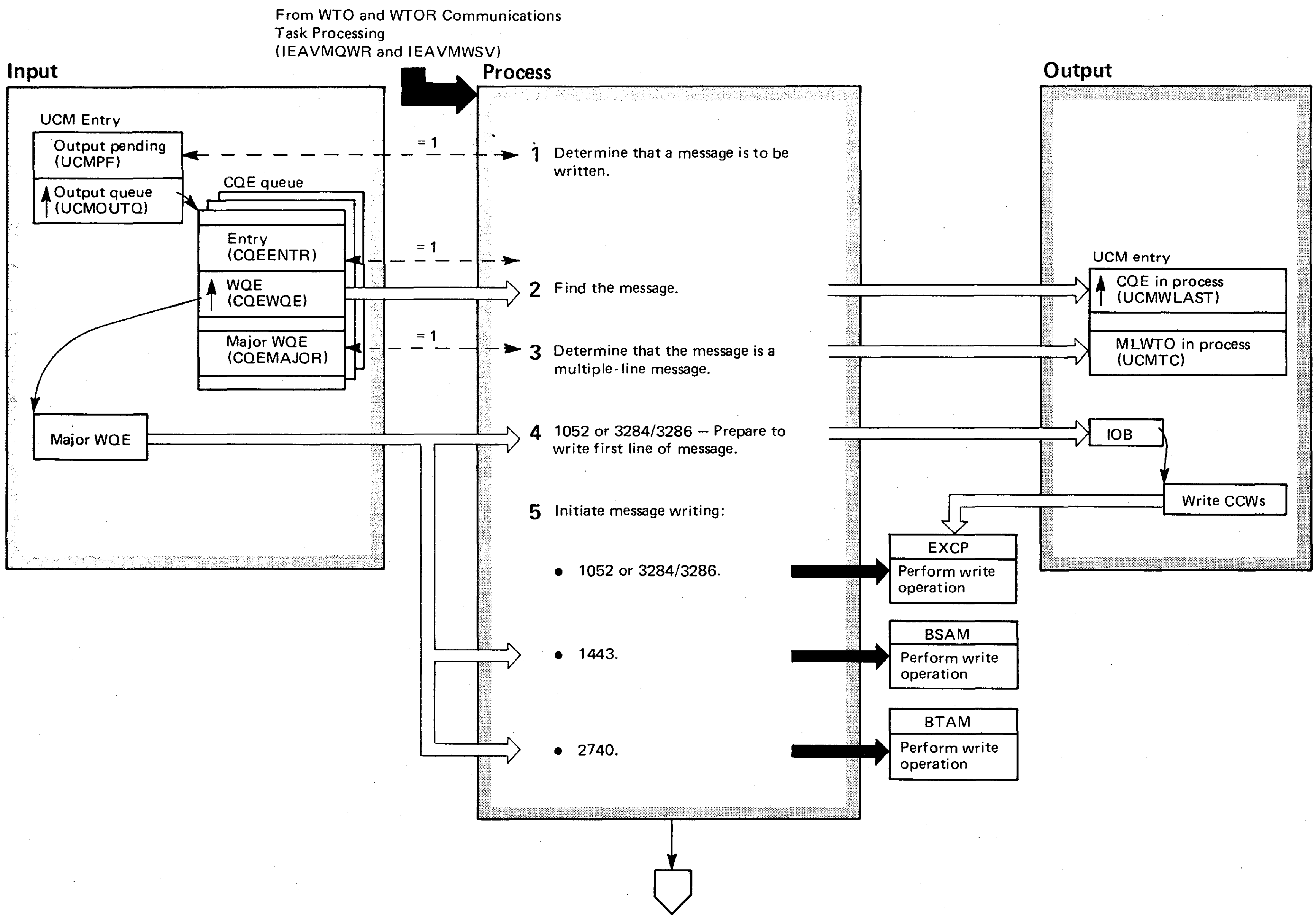


Diagram 1-15. Writing Multiple-line Messages to a 1052, 1443, 2740, or 3284/3286 Console (Part 2 of 4)

Extended Description	Module	Label
User programs and system routines issue a WTO or WTOR macro instruction to send messages to the operator's consoles. The following communications task device support processors (DSPs) write multiple-line messages to consoles:		
• For a 1052 printer-keyboard, 3210 console printer-keyboard, 3215 console printer-keyboard, and 3213 console printer, the DSP is IEAV1052.	IEAV1052	
• For a 1443 printer, 1403 printer, and 3211 printer, the DSP is IEAV1443.	IEAV1443	
• For a 2740 communications terminal, IEEC2740.	IEEC2740	
• For a 3284/3286 printer, IEECVETW.	IEECVETW	
1 The appropriate DSP checks the output-pending bit (UCMPF) to determine that a message is to be written.	(See above)	
2 The DSP searches the CQE chain (pointed to by UCMOUTQ) for a CQE pointing to a message (flag CQEENTR is on). When the DSP finds such a CQE, it saves the address of the CQE in field UCMWLAST. Field CQEWQE points to the WQE that contains the message.		
3 The DSP checks bit CQEMAJOR to determine whether the message is a multiple-line message (MLWTO). If CQEMAJOR is on, the DSP sets bit UCMTC to indicate that a multiple-line message is being processed. The CQE points to a major WQE; the major WQE contains the first line of the multiple-line message.		
4 For a 1052 or 3284/3286, the appropriate DSP obtains an IOB for the write operation and places the addresses of the write CCWs into the IOB.		
5 The DSPs initiate the writing of the first line of the message:		
• For a 1052 or 3284/3286, the appropriate DSP issues an EXCP macro instruction to execute the write channel program.	IEAV1052 or IEECVETW	
• For a 1443 printer, the 1443 DSP issues a WRITE macro instruction to pass control to BSAM, which writes the message.	IEAV1443	
• For a 2740 communications terminal, the 2740 DSP issues a WRITE macro instruction to pass control to BTAM, which writes the message.	IEEC2740	

Diagram 1-15. Writing Multiple-line Messages to a 1052, 1443, 2740, or 3284/3286 Console (Part 3 of 4)

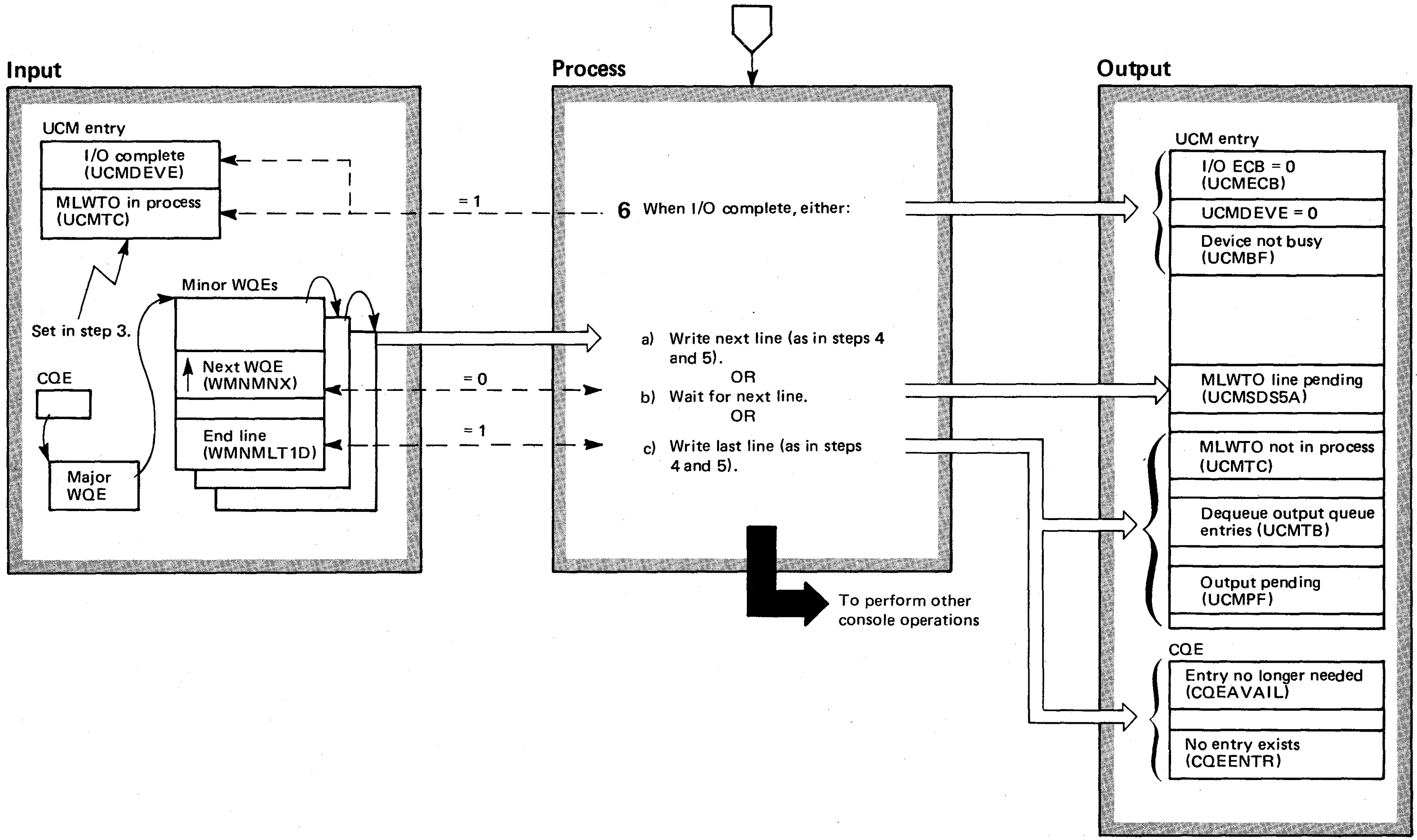


Diagram 1-15. Writing Multiple-line Messages to a 1052, 1443, 2740, or 3284/3286 Console (Part 4 of 4)

Extended Description	Module	Label
6 When the write operation that was initiated in step 5 is complete (bit UCMDEVE is on), the DSP clears the I/O ECB (UCMECB), the I/O-complete bit (UCMDEVE), and the device-busy bit (UCMBF). Then the DSP checks bit UCMTC (set in step 3) to determine that a multiple-line message is being processed. The DSP continues processing the multiple-line message as follows:		
a) If a minor WQE containing the next message line exists, the DSP initiates the writing of the next line as in steps 4 and 5.		
b) If no minor WQE with the next message line exists, the DSP sets bit UCMSDS5A and waits for the next line to be available.		
c) If the WQE indicates that it contains the last message line (WMNMLT1D is on), the DSP initiates writing of the last line as in steps 4 and 5. The DSP also turns off the MLWTO-in-process bit (UCMTC), turns on bit UCMTB to indicate that the appropriate output queue entries must be dequeued, and turns on bit UCMPF to indicate that output is pending. Finally, the DSP marks the CQE available (CQEAVAIL) and indicates that no entry is associated with the CQE (CQEENTR).		

Diagram 1-16. Displaying Multiple-line Messages on a Graphics Console (DIDOCS) (Part 1 of 2)

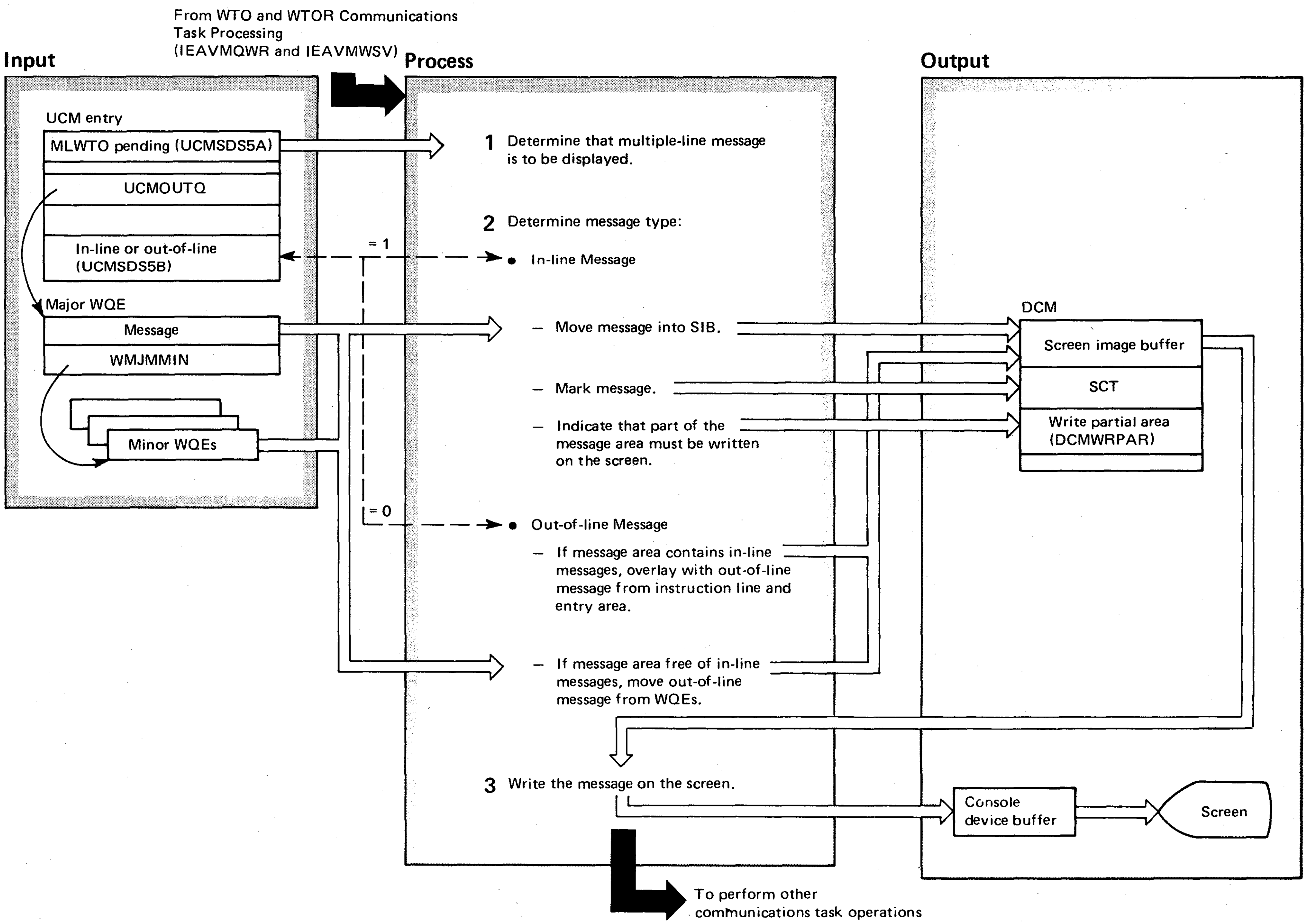


Diagram 1-16. Displaying Multiple-line Messages on a Graphics Console (DIDOCS) (Part 2 of 2)

Extended Description	Module	Label
User programs and system routines use the WTO or WTOR macro instruction to send multiple-line messages (as well as single-line messages) to the operator's consoles. DIDOCS displays a multiple-line message on a graphics console as follows:		
1 DIDOCS checks bit UCMSDS5A to determine that a multiple-line message is to be displayed.	IEECVET1	
2 DIDOCS checks bit UCMSDS5B to determine whether the multiple-line messages is in-line or out-of-line.	IEECVFT2	
• If the message is in-line (UCMSDS5B is on), DIDOCS continues to move the lines of the message from the major and minor WQEs into the SIB until the screen is full. DIDOCS marks each message with the appropriate message indicator. Finally, DIDOCS sets bit DCMWRPAR to indicate that part of the message area containing the new message must be written to the screen.	IEECVFTL	
• If the message is out-of-line (UCMSDS5B is off), DIDOCS searches the console queue for a major WQE with a valid target area ID (WMJMAREA). If the area is occupied by in-line messages, DIDOCS writes the out-of-line message, three lines at a time, from the instruction line and entry area of the screen image buffer to the area. If the area is free of in-line messages, DIDOCS moves the out-of-line message lines from the major and minor WQEs to the area until the area is full.	IEECVFTM IEECVFTQ IEECVFTO	
3 DIDOCS writes the message from the screen image buffer to the console device buffer.	IEECVETH/P/R/U	

Diagram 1-17. I/O Complete Processing (Part 1 of 8)

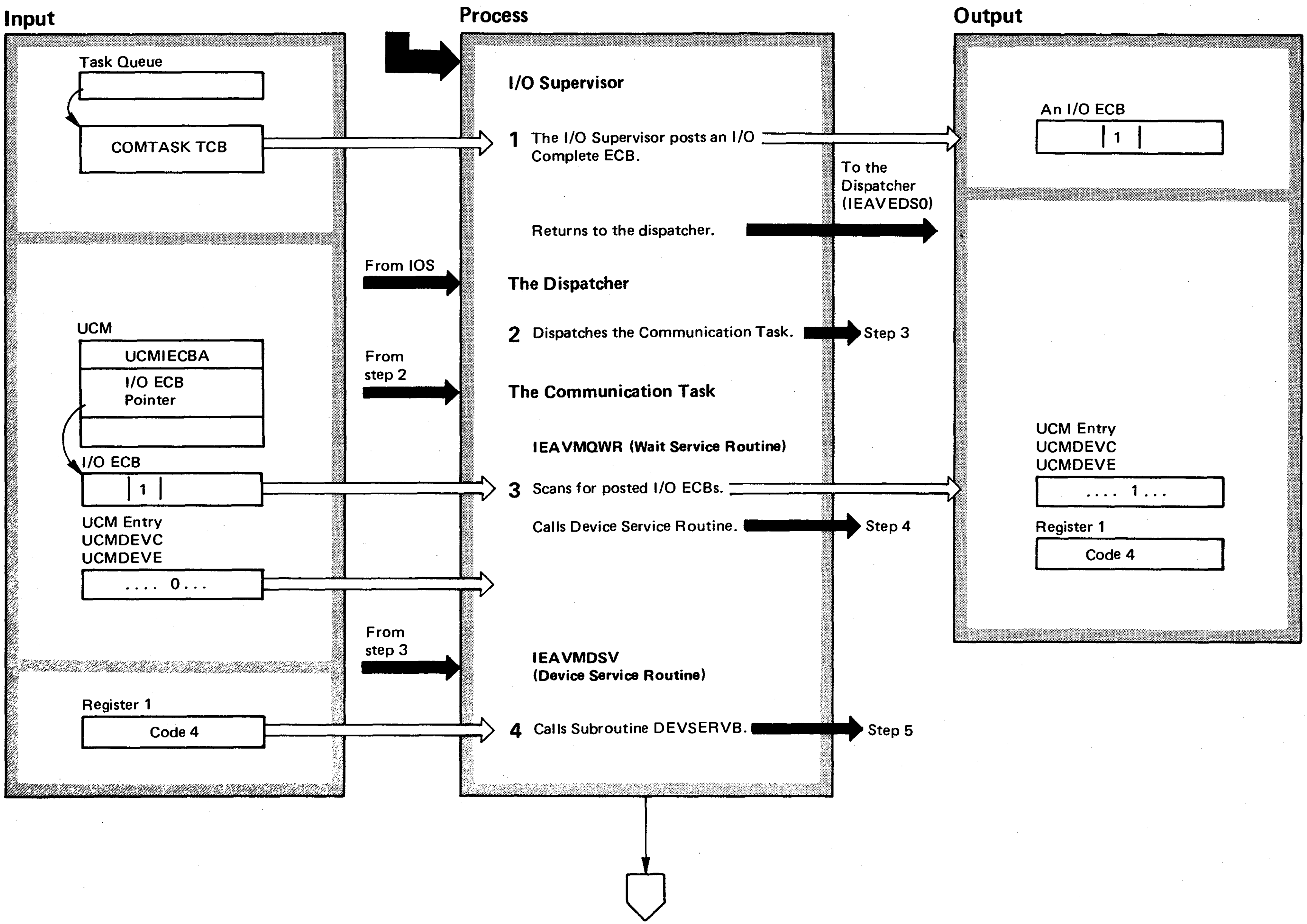
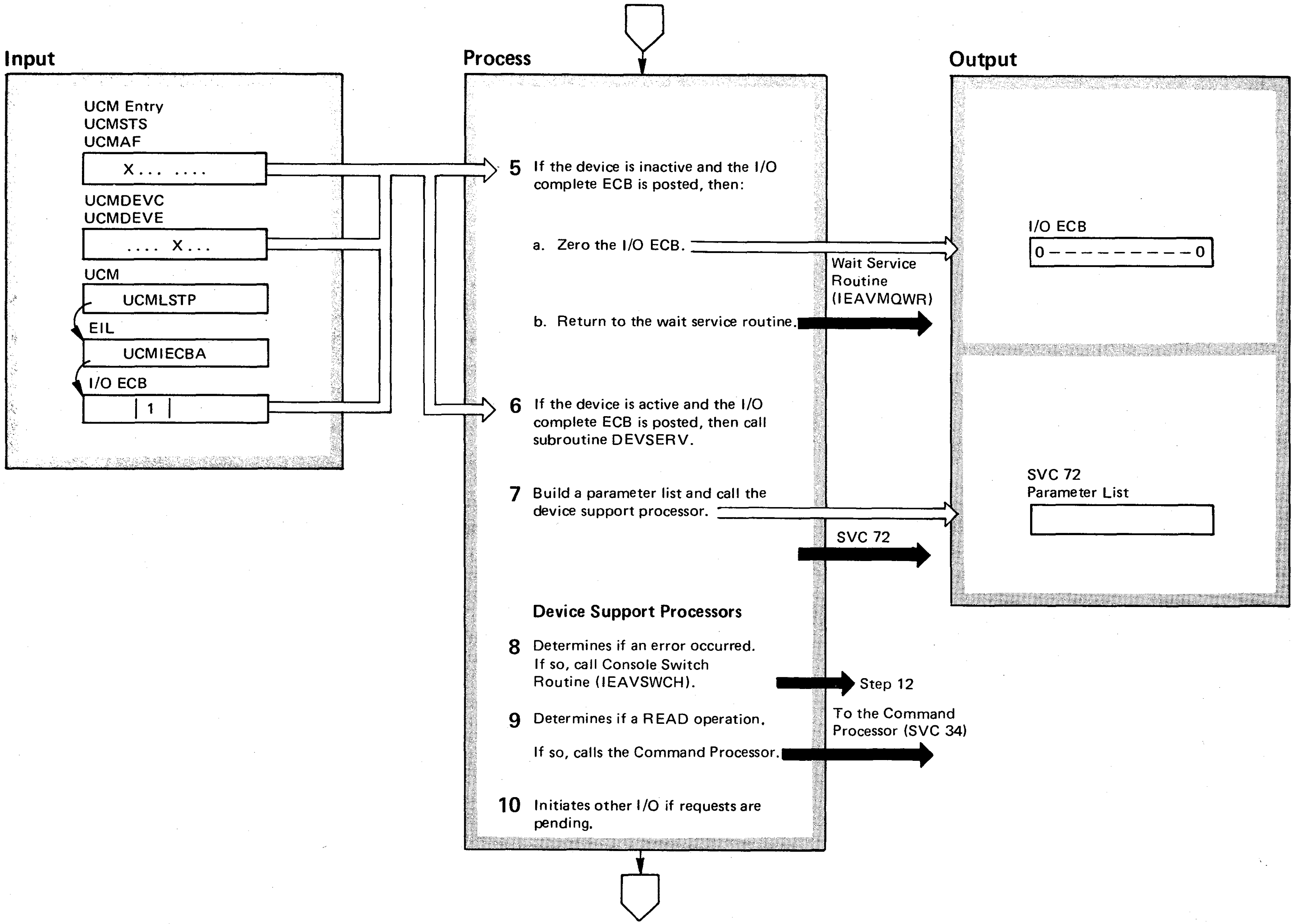


Diagram 1-17. I/O Complete Processing (Part 2 of 8)

Extended Description	**Module**
This procedure handles the completion of a requested input/output operation.	
I/O Supervisor	
1 The I/O Supervisor (IOS) will post the appropriate I/O Complete ECB. The Communications Task TCB is marked ready.	
Dispatcher	
2 The Dispatcher passes control to the Communication Task when its TCB is the highest priority ready TCB on the queue.	IEAVEDS0
The Wait Service Routine	
3 The list of I/O ECB pointers (EIL) is used to check I/O ECB postings. If a posted I/O ECB is found, the 'I/O Complete' flag (UCMDEVE in UCMDEVC of the UCM Entry) is set. Control passes to the Device Services Manager (IEAVMDSV) with a code of four in register one. Upon return, control passes to the beginning of the ECB check loop (WREXT).	IEAVMQWR
Device Service Routine	
4 The code passed in register one by the Wait Service Routine (IEAVMQWR) is checked. Control passes to Subroutine DEVSERVB if the code is four.	IEAVMDSV

Diagram 1-17. I/O Complete Processing (Part 3 of 8)

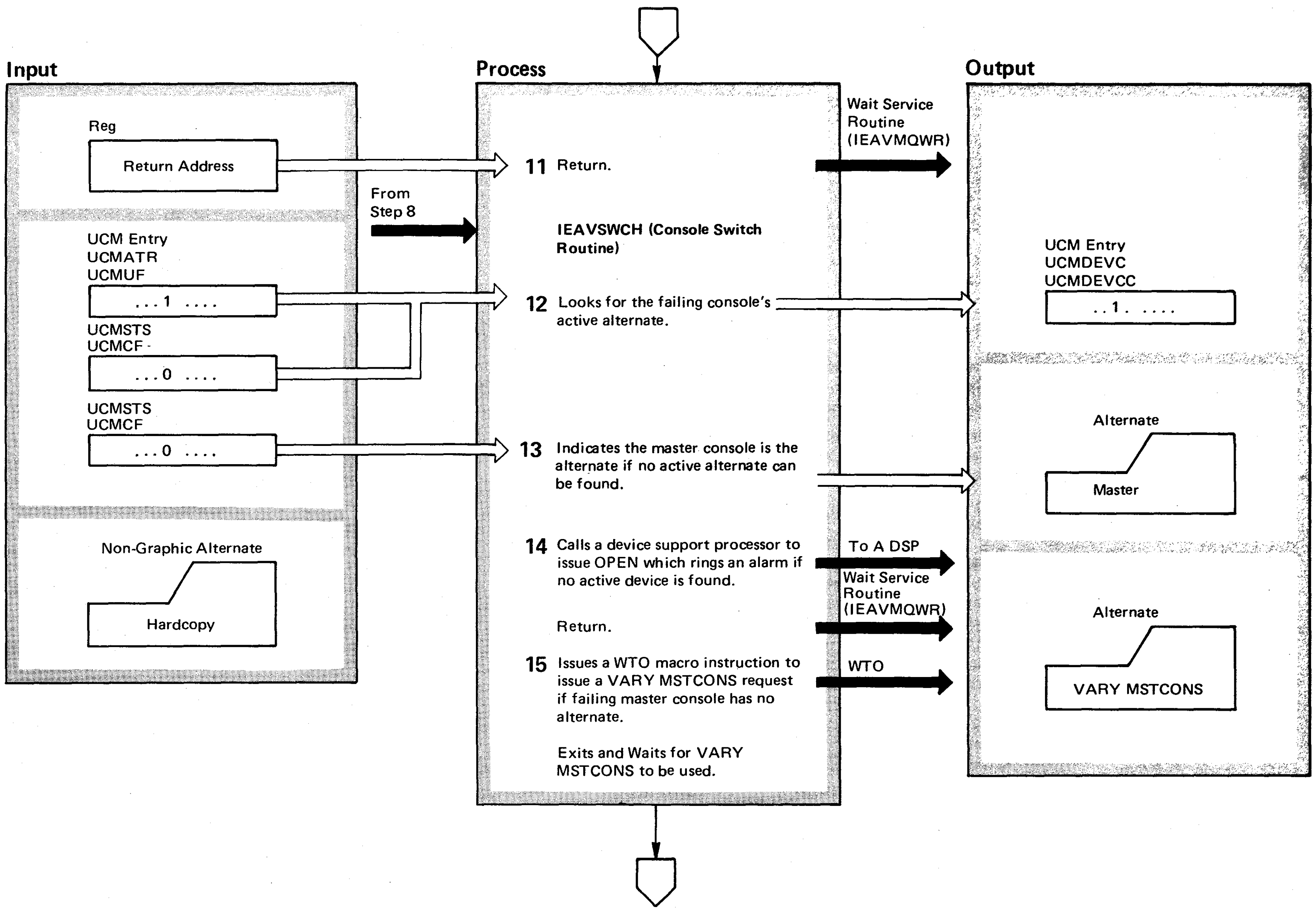


Extended Description	Module
5 In Subroutine DEVSERVB, a check is made to determine if the device is active (UCMAF on in UCMSTS). If so, control passes to step 6. If not, a check is made for an I/O completion (UCMDEVE on in UCMDEVC). If so, the pointer to the EIL (Event Indicator List) is obtained (from UCMLSTP in the UCM). The index in register 6 is used to point to the I/O Complete ECB for the console (UCMIECBA points to the list of ECBs). The ECB is then zeroed. Control returns to the wait service routine.	
6 If I/O has completed (UCMDEVE off in UCMDEVC), a branch is taken to Subroutine DEVSERV. Otherwise, Control returns to IEAVMQWR at entry point WRABXLE.	
7 Subroutine DEVSERV constructs the SVC 72 parameter list and a SVC 72 calls the appropriate non-resident support processor. For additional information on SVC 72, see either Writing Messages to a Console *or* Processing Commands From a Console diagrams.	
8 The device support processor determines if the device is busy, if the I/O is complete and if the I/O operation was successful. If an error has occurred, branch to the Console Switch Routine (IEAVSWCH). See step 12.	IEAVVCTR
9 If a READ I/O operation is successful, the SVC 34 Command Processor is called to analyze the input message.	Note 1
10 When a WRITE operation is successful, the associated CQE is marked completed. (CQEENTR set to off in CQEFLAG.) A check is then made for other communication task work to be performed. If an attention is pending (UCMAF in UCMSTS), a read is issued. If an output message is available for this console, the next message is selected and sent to the console using either a WRITE or EXCP macro.	Note 2

Notes:

1. The appropriate console device processor with input and output capabilities is given control (see "Console Device Support" in the introduction to this section).
2. The appropriate console device processor with input and output or output only capabilities is given control (see "Console Device Support" in the introduction to this section).

Diagram 1-17. I/O Complete Processing (Part 5 of 8)



Extended Description	Module
11 Control returns to the wait service routine.	
Console Switch Routine	
12 Upon entry, determines if request is for failing console. A scan of the failing console's alternate chain is done for an active console (UCMUF on in UCMATR) which does not have a CLOSE pending (UCMCF off in UCMSTS). Each UCM Entry is marked 'Device Tested' (UCMDEVCC in UCMDEVC) when not found to be acceptable as an alternate. This flag is reset in each UCM Entry by Subroutine CLEARDVC when the alternate has been selected.	IEAVSWCH
13 If the failing console has no eligible alternate in its chain and it is not the master console, the master console is selected to replace the failing console.	IEAVSWCH

Extended Description	Module
14 If no active input/output device is found, another scan of the UCM Entries is made for an output-only display device which has I/O capability, which is active and does not have a CLOSE pending. If found, the SWFULCAP subroutine is called to switch the console to full I/O capability. If no eligible alternate can be found, another scan is performed for a console with an alarm (e.g., a 1052 or 2150 console). The routine branches to the appropriate device support processor to issue the OPEN macro instruction. On return an EXCP macro instruction is issued to ring the alarm.	
15 If the failing console is the master console, a scan of the UCM Entries is done for a console which is not output-only, which is not the master console, which is active and which does not have a CLOSE pending. If found, a message (IEE141A) is broadcasted requesting a 'VARY MSTCONS' command from an operator.	IEAVSWCH

Diagram 1-17. I/O Complete Processing (Part 7 of 8)

Process

16 Adds the failing console's attributes to the alternate console's attributes.

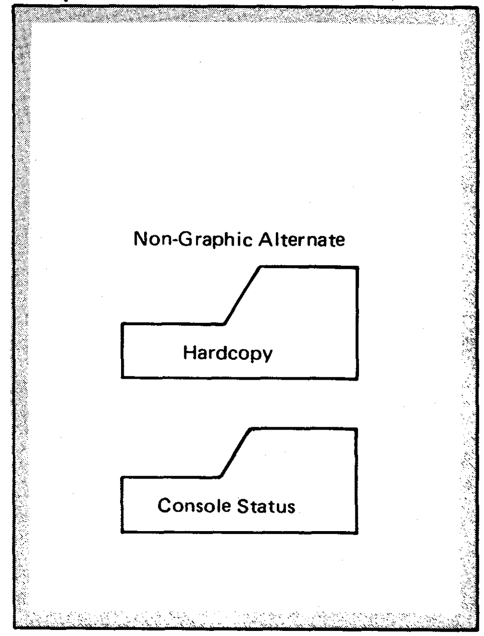
17 Constructs WQE with console status message. → SVC 35 → Output

18 Queues the failing console's messages and unanswered reply requests to the alternate console's queue.

19 Switches the hard copy function to a non-graphic alternate console if needed.

20 Return. → Wait Service Routine (IEAVMQWR)

Output



Extended Description	Module
16 The authorization codes and routing codes of the failing console are added to those of its eligible alternate console.	
17 The WQE is constructed.	
18 If it is determined that the WTOR (a message and a requested reply) was queued to the failing console, the message is requeued to the alternate console's output queue. In Subroutine ML1, if the multiple-line message is being written, or the end of the message is not indicated, the message is purged. Otherwise, the message is queued to the new console's output queue. The new console's output queue is scanned for duplicate entries. If any are found, they are marked to prevent them from being written.	
19 If the failing console was the hardcopy device (UCMDISPE on in UCMDISP), and the switch was not a result of a 'VARY MSTCONS' command or a system requested 'VARY', the hardcopy function is switched to the alternate console provided it is not a display console. Otherwise, another alternate is selected that can support the hardcopy function. A message (IEE142I) is issued to inform the operator of the console that is performing the hardcopy function. When a hardcopy console is unavailable, a WTO macro is issued to send message IEA964I to the master console and the hardcopy facility is suspended.	IEAVSWCH
20 The failing console is marked 'Close Pending' (UCMCF in UCMSTS) and not 'Busy' (UCMBF off in UCMSTS). The failing console's output queue pointer (UCMOUTQ) and pointer to the last CQE entry serviced (UCMWLAST) are zeroed. The console switch flags (UCMSYSK and UCMSYSM in UCMFLG2, UCMSYSD in UCMSFLG1) are reset. A FREEMAIN macro instruction is issued to free the save area. A POST macro instruction is issued to post the WTO ECB (UCMOECB). Register 14 is restored from CSAXA and control passes via register 14 to the Console Switch Routine (IEAVSWCH).	

Diagram 1-18. DOM Macro Instruction Processing Overview (SVC 87) (IEAVXDOM) (Part 1 of 2)

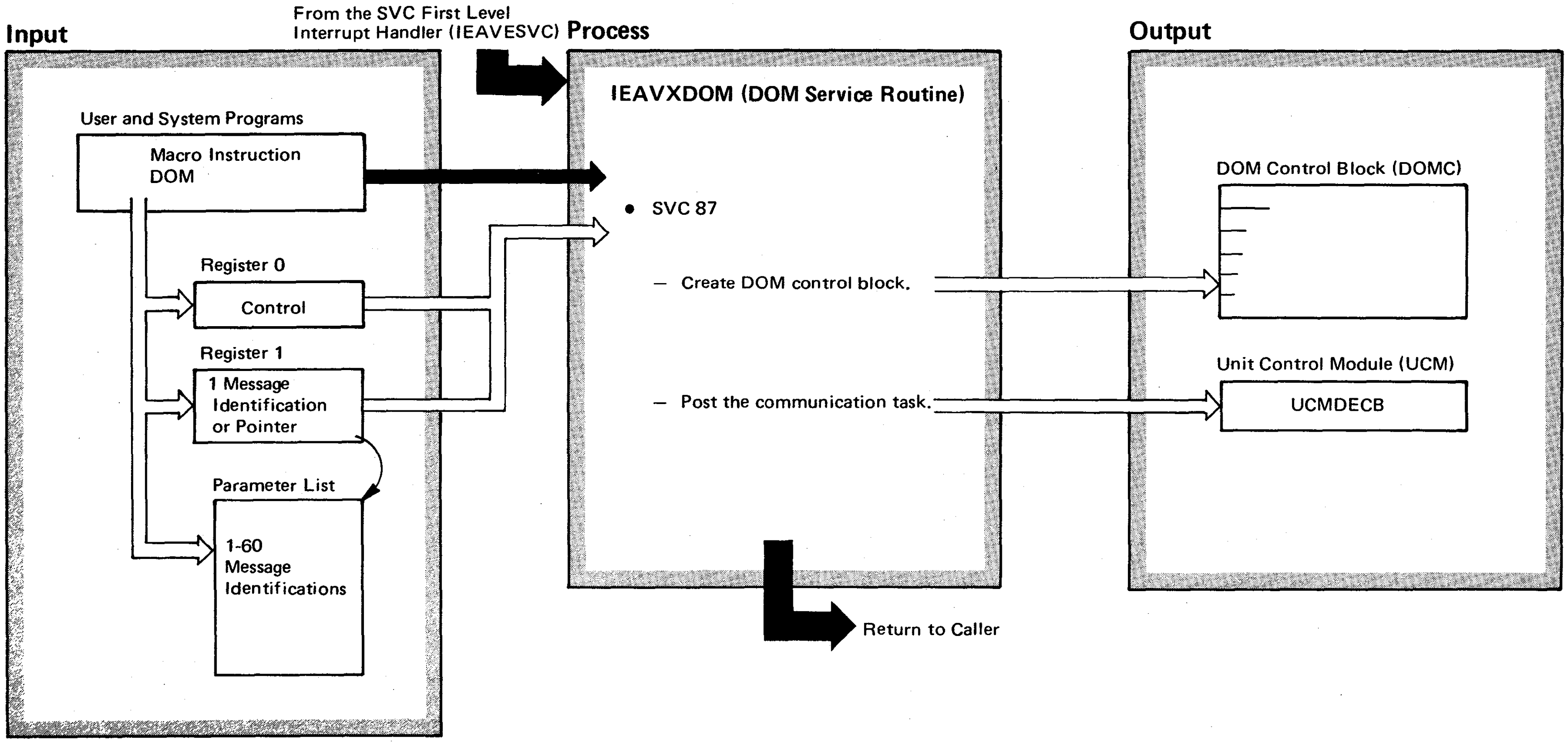


Diagram 1-18. DOM Macro Instruction Processing Overview (SVC 87) (IEAVXDOM) (Part 2 of 2)

Extended Description	Module
To delete a message, that message must already reside in a write queue element (WQE). A message identification was assigned to each WQE when it was created by a WTO or WTOR macro instruction. This message identification was returned to the WTO or WTOR macro instruction user in register 1.	IEAVXDOM
The DOM macro instruction service routine builds the delete operator message control block (DOMC) to pass from one to sixty message identifications to the communication task for deletion. This routine then posts the communication task (UCMDECB), which removes the WQEs and their associated operator reply elements (OREs).	

Diagram 1-19. DOM Macro Instruction Processing (SVC 87) (IEAVXDOM) (Part 1 of 12)

From the SVC First Level Interrupt Handler (IEAVESVC)

Process

IEAVXDOM

1 The following table shows the subroutines called for the DOM macro instruction service routine:

Called Subroutines are: SETLCKS FRELCKS

Extended Description	Module
Mainline Routine: IEAVXDOM	
1 This routine services the delete operator message (DOM) macro instruction and SVC 87. The one to sixty message identifications passed to this routine are placed in a delete operator message control block (DOMC) and the communication task is posted. A copy of the identifications is given to the subsystem exit routine; no response is accepted from this exit. Control is then returned to the SVC first level interrupt handler.	IEAVXDOM

Extended Description	Module
Subroutines:	
SETLIST This subroutine validates the contents of register 0 and prepares the routine to process the list of message identification passed by the user of the DOM macro instruction.	
AUTHCHK Determines if the DOM user may delete messages that were issued by other jobsteps or outside the user's address space.	
SCANIDS Checks the message identifications passed by the user and moves the valid message identifications into the dummy delete operator message control block (DUMDOMCB).	
GETDOMCB Computes the size required for the variable length delete operator message control block (DOMC) from the dummy delete operator message control block (DUMDOMCB) and obtains the space for the DOMC from subpool 231.	
FILLDOM Moves the information needed into the delete operator message control block (DOMC) and places this DOMC on the DOMC chain for processing by the communication task.	
SUBEXIT Passes the dummy delete operator message control block to the subsystem. In Release 2, the only subsystem is JES2.	
SETLCKS Obtains the local and CMS locks, and sets up the functional recovery routine (FRR). The locks serialize the use of communication task resources. The FRR helps clean up the communication task should a processing error occur.	
FRELCKS Releases the functional recovery routine, and frees the CMS and local locks.	

Diagram 1-19. DOM Macro Instruction Processing (SVC 87) (IEAVXDOM) (Part 3 of 12)

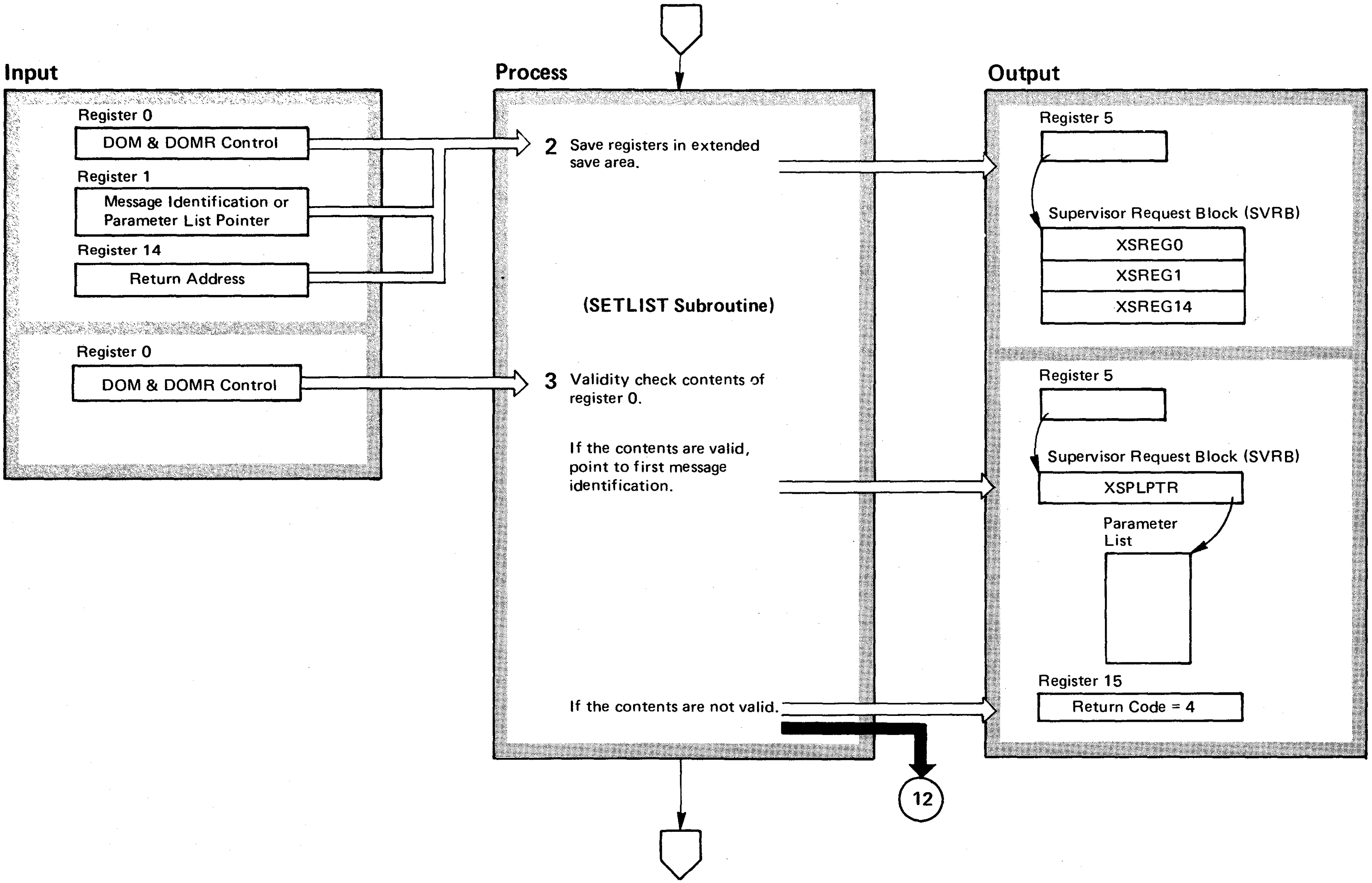


Diagram 1-19. DOM Macro Instruction Processing (SVC 87) (IEAVXDOM) (Part 4 of 12)

Extended Description	Module	Label
2 Registers 0, 1, and 14, are saved in the extended save area (XS) of the supervisor request block (SVRB).	IEAVXDOM	
3 Checks the validity of the contents of register 0. Only four values are acceptable:		SETLIST

0 — Register 1 contains the message identification of one WTO message. (This DOM will attempt to delete one WQE.)

4 — Register 1 contains the message identification of one WTOR message. (This DOM will attempt to delete one WQE and one ORE.)

12 — Register 1 contains a pointer to a user supplied parameter list having several WTO and WTOR message identifications. (This DOM will attempt to delete each WQE and, when an ORE exists, delete the OREs associated with the messages contained in the WQEs.)

Negative Number — Register 1 contains a pointer to a user supplied parameter list having several WTO message identifications. (This DOM will attempt to delete each WQE but no OREs.)

Any other value in register 0 causes the user to eventually receive a 157 ABEND with a return code of 4. When the contents of register 0 are valid, XSPLPTR is set to point to the first message identification. If the message identification is in register 0, XSPLPTR is set to point to XSREG1 and the high order bit in XSREG1 is turned on to indicate that this is the last message identification. If register 1 contains a pointer, XSPLPTR points to the user supplied parameter list.

Diagram 1-19. DOM Macro Instruction Processing (SVC 87) (IEAVXDOM) (Part 5 of 12)

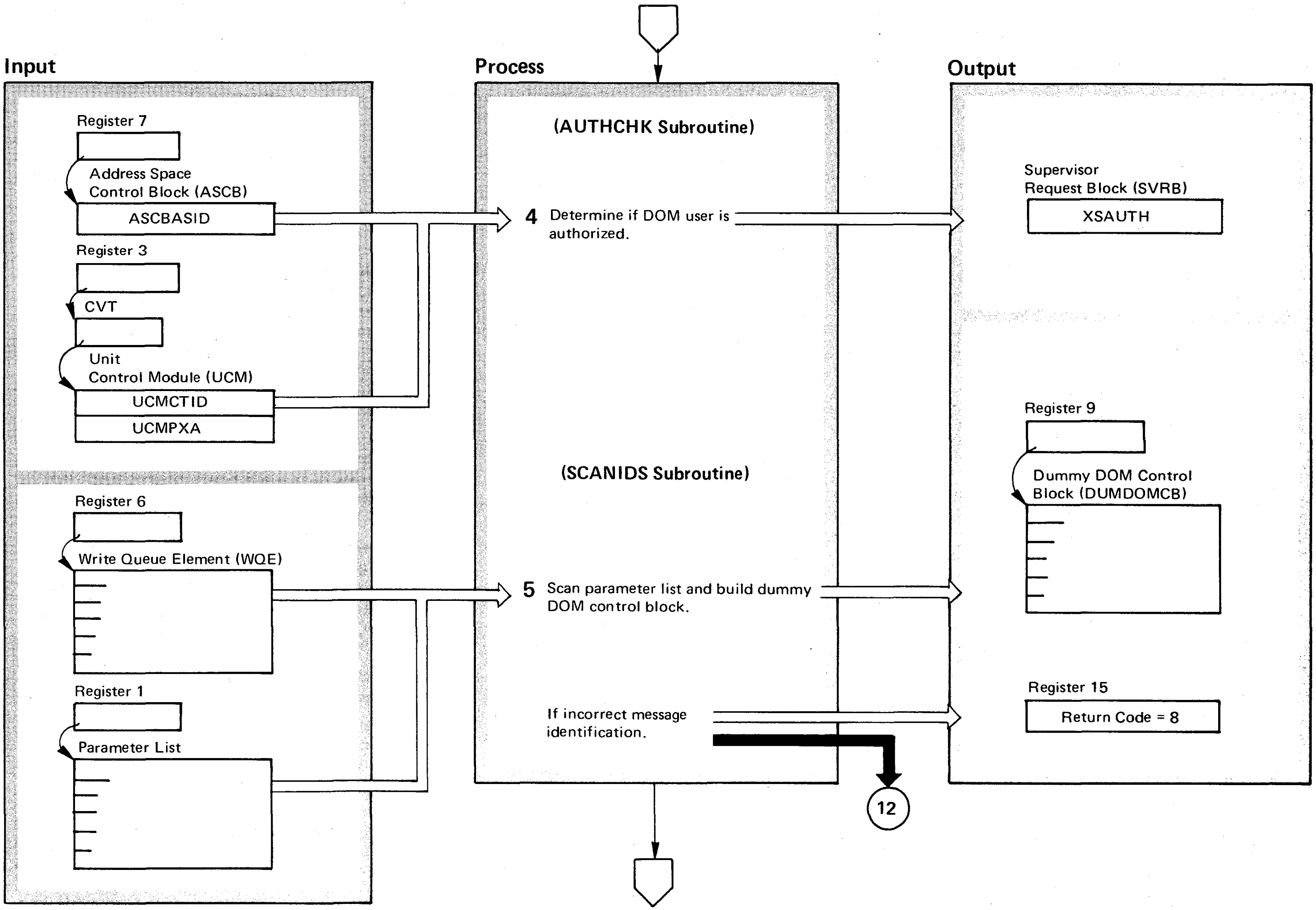


Diagram 1-19. DOM Macro Instruction Processing (SVC 87) (IEAVXDOM) (Part 6 of 12)

Extended Description	Module	Label
4 In this instance, an authorized program is a program running in supervisor state, protect keys 0-7, or a problem program sanctioned by the authorized program facility (APF).	IEAVXDOM	AUTHCHK

The TESTAUTH macro instruction is issued the first time to determine if the DOM user is running in supervisor state or using keys 0-7; if so, the XSAUTH bit is turned on to indicate that the user is authorized.

If the first TESTAUTH macro instruction indicated that the DOM user is a problem program, a second TESTAUTH macro instruction determines if the DOM user is authorized by the authorized program facility; if so, the XSAUTH bit is turned on to indicate that the user is authorized.

If both TESTAUTH macro instructions fail to indicate an authorized DOM user, the XSAUTH bit is turned off.

Extended Description	Module	Label
5 Each of the message identifications in the user parameter list is tested against the message identifications in each WQE on the chain of WQEs. During this test, the message identifications from the parameter list that are not rejected are copied into the dummy DOM control block. The reasons for rejecting a message identification are:		SCANIDS

- The WQE with that message identification is suspended. A suspended WQE is not yet in the system.
- The WQE with that message identification is for a WTOR and the DOM user specified a WTO, not a WTOR. The user receives a 157 ABEND; the return code is ignored.
- The DOM user is not authorized to issue a DOM macro instruction against the WQE with that message identification. For example, the user is not authorized, or is a unauthorized problem program with a different address space identifier (ASID) or different jobstep from the program that issued the WTO. The user receives a 157 ABEND with return code 8.
- The WQE with that message identification is for a WTOR and the ORE has already been replied to by REPLY processing.

Note: When a message identification from the user-supplied parameter list fails to match any of the message identifications in the chain of WQEs, that message identification is copied into the dummy DOM control block. This message identification may exist for a message being displayed by a graphic console.

Diagram 1-19. DOM Macro Instruction Processing (SVC 87) (IEAVXDOM) (Part 7 of 12)

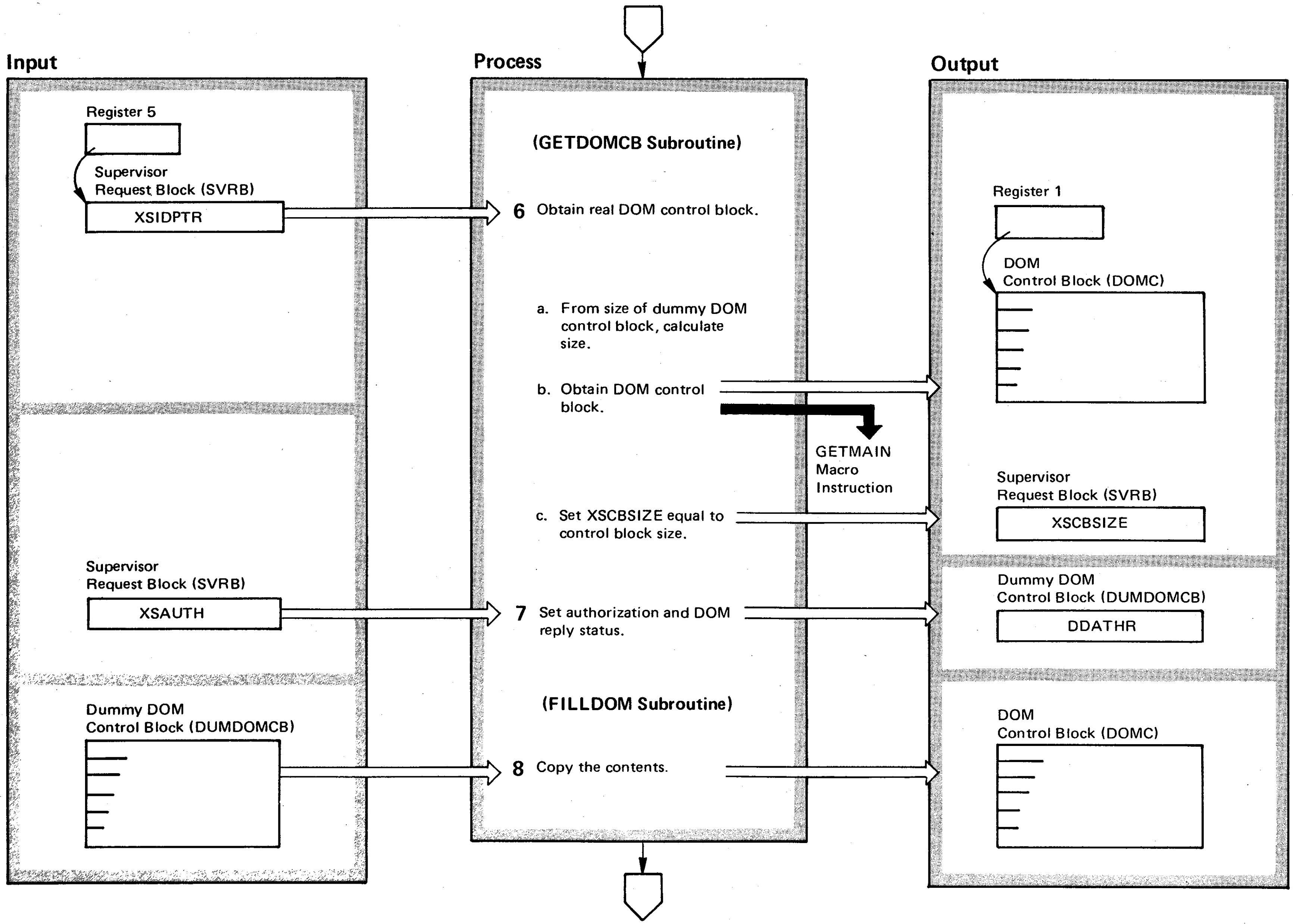


Diagram 1-19. DOM Macro Instruction Processing (SVC 87) (IEAVXDOM) (Part 8 of 12)

Extended Description	Module	Label
6 Having built the dummy DOM control block (DUMDOMCB), this routine:	IEAVXDOM	GETDOMCB
a) Calculates the size needed for the real DOM control block (DOMC).		
b) Issues a GETMAIN macro instruction to obtain space for the DOMC from subpool 231.		
c) Fills the newly obtained DOMC with X'00'.		
• Places the size of the DOMC in the supervisor request block (SVRB).		
Upon return from executing the GETMAIN macro instruction, register 1 points to the newly obtained DOMC.		
7 The bits in the dummy DOM control block (DUMDOMCB) representing whether the DOM user is authorized and whether this DOM is permitted to remove WTORs are set.		
8 The contents of the dummy DOM control block (DUMDOMCB) are copied into the DOM control block (DOMC).		FILLDOM

Diagram 1-19. DOM Macro Instruction Processing (SVC 87) (IEAVXDOM) (Part 9 of 12)

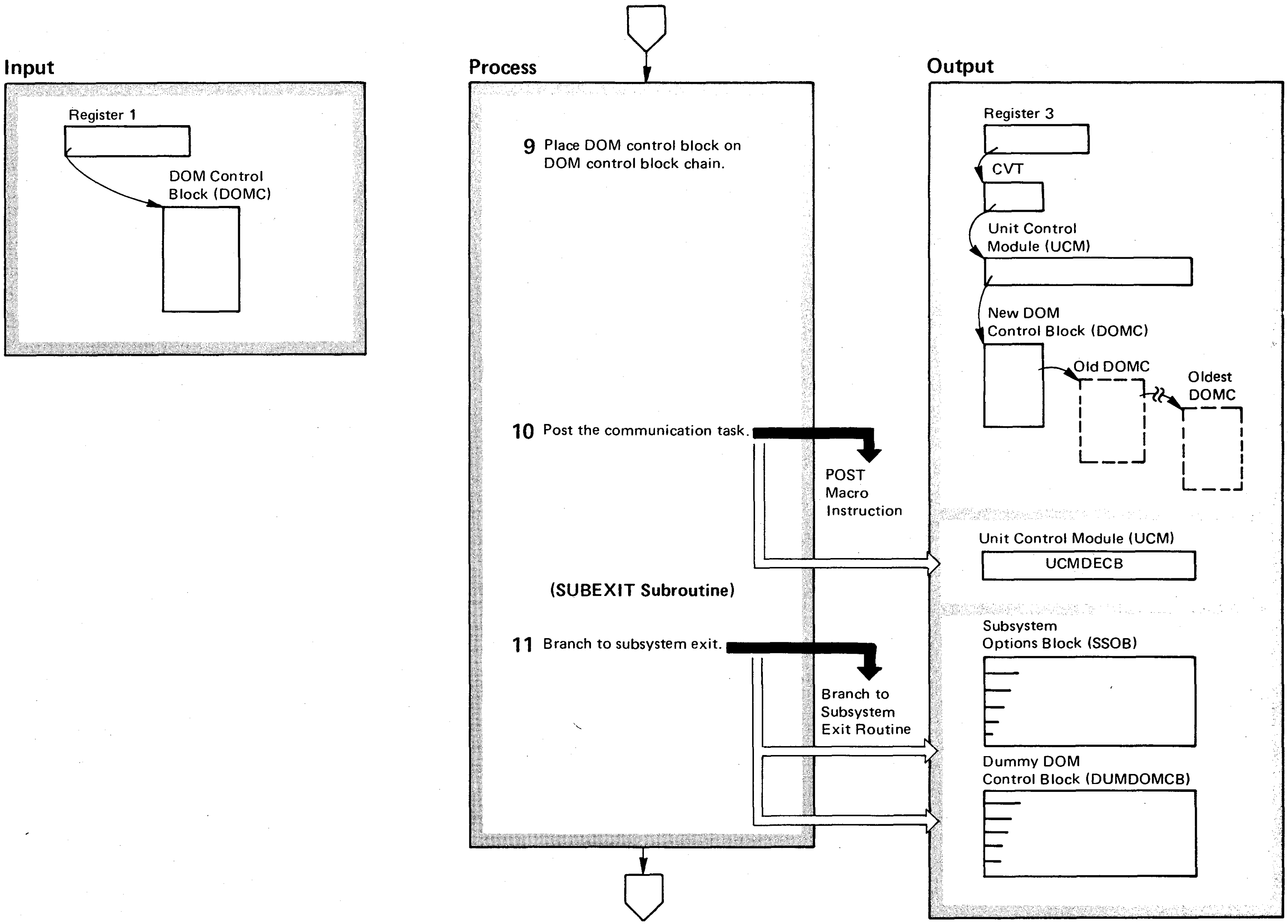


Diagram 1-19. DOM Macro Instruction Processing (SVC 87) (IEAVXDOM) (Part 10 of 12)

Extended Description	Module	Label
9 The DOM control block (DOMC) built by this routine is placed at the top of the DOMC queue. The pointer from the unit control module (UCM) to the DOMC is in the prefix area of the UCM.	IEAVXDOM	
10 With the DOM control block (DOMC) on the DOMC queue, the message identification in the DOMC are ready to be processed by the communication task. The communication task is posted to perform this service by turning on the UCMDECB event control block in the unit control module (UCM).		
11 A subsystem options block (SSOB) is created and passed to the job entry subsystem exit routine along with the dummy DOM control block (DUMDOMCB). No response is expected from the exit routine other than the return of control.		SUBEXIT

Diagram 1-19. DOM Macro Instruction Processing (SVC 87) (IEAVXDOM) (Part 11 of 12)

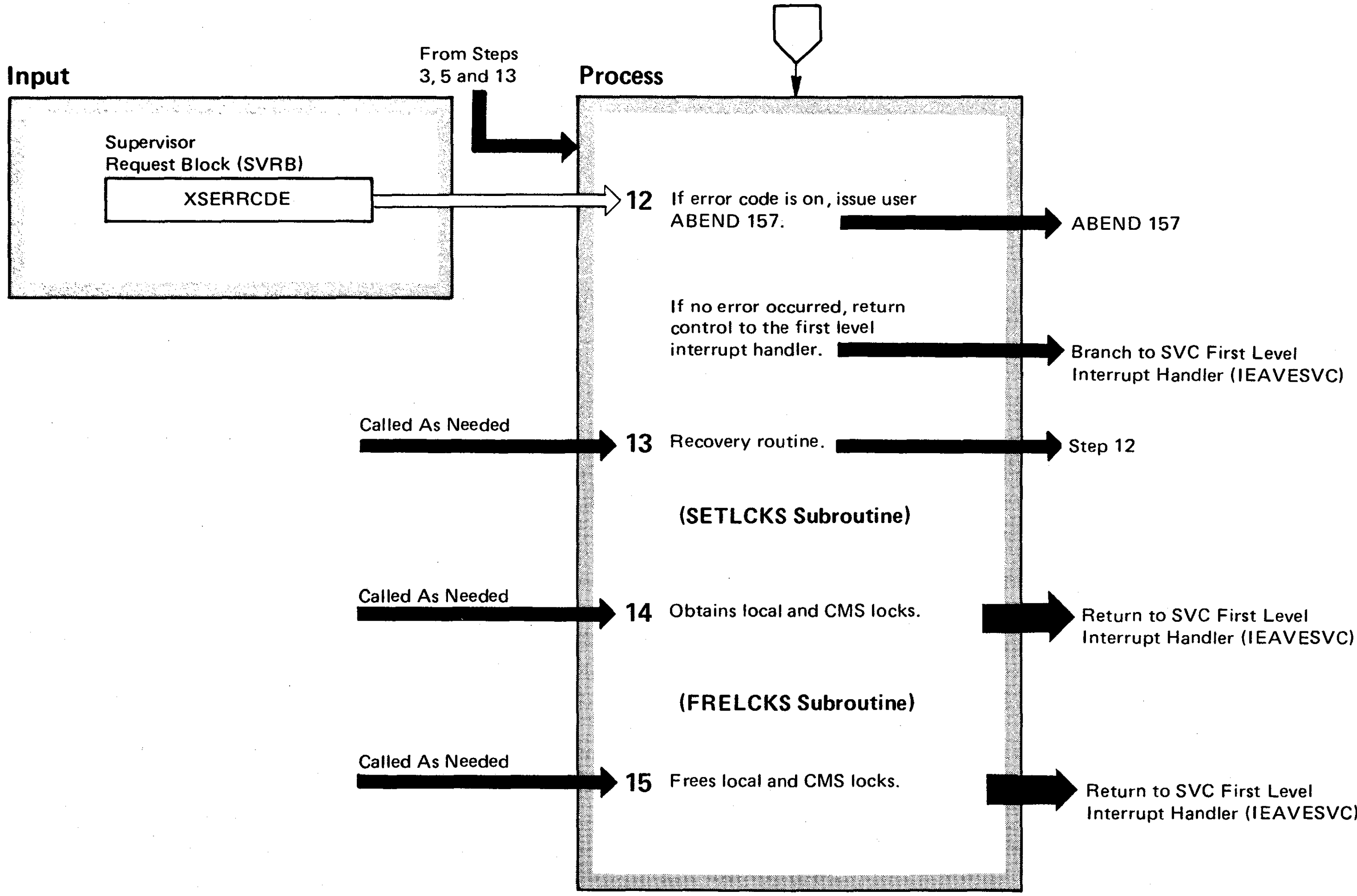


Diagram 1-19. DOM Macro Instruction Processing (SVC 87) (IEAVXDOM) (Part 12 of 12)

Extended Description	Module	Label
12 In preparation for returning control, all work areas are freed using the FREEMAIN macro instruction. If an error occurred, the user receives a 157 ABEND. If no error occurred, return to the user via the SVC first level interrupt handler.	IEAVXDOM	
13 Something caused an abnormal termination of this routine. The system eventually gives control to this step, which sets an error indicator and branches to end normal processing. The user will receive a 157 ABEND.		
14 The SETLCKS subroutine serializes the use of the unit control module (UCM), the write queue elements (WQEs), and the operator reply elements (OREs). To serialize their use, this subroutine obtains the local and CMS locks, and sets the functional recovery routine (FRR) for recovery processing should an unexpected abnormal termination occur during the process.		SETLCKS
15 The FRELCKS subroutine frees the functional recovery routine (FRR), and releases the CMS and local locks obtained by the SETLCKS subroutine.		FRELCKS

Diagram 1-20. DOM Communication Task Processing Overview (IEAVMDOM) (Part 1 of 2)

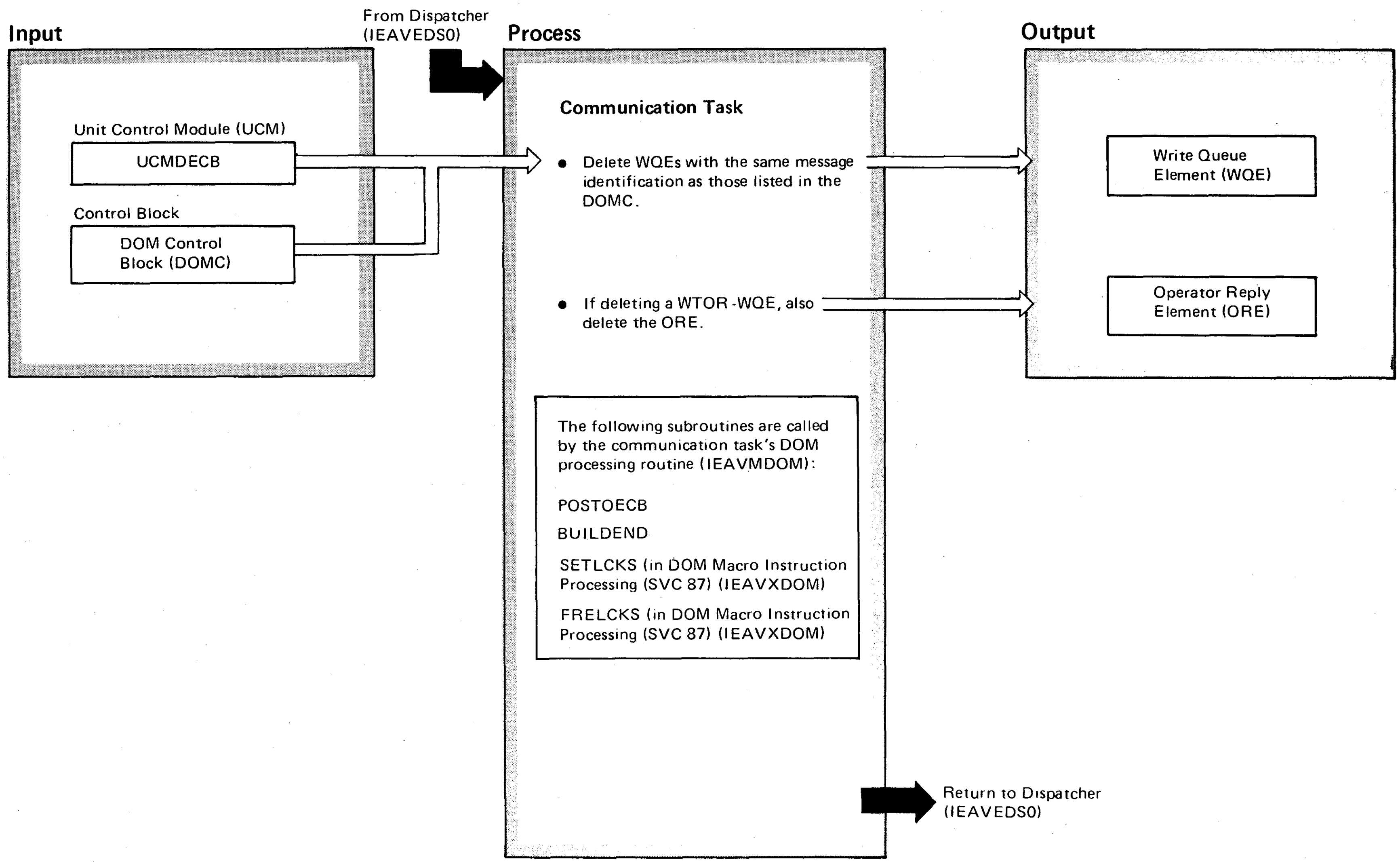


Diagram 1-20. DOM Communication Task Processing Overview (IEAVMDOM) (Part 2 of 2)

Extended Description	Module
Prior to entry into this module, some system or user program issued a delete operator message (DOM) macro instruction. The DOM service routine, SVC 87 (IEAVXDOM), prepared a DOM control block (DOMC) containing one to sixty message identifications of messages to be deleted from the system. It then posted the event control block UCMDECB for the communication task to actually delete these messages.	IEAVMDOM

The communication task searches the appropriate write queue element (WQE) and operator reply element (ORE) chains for the message identifications listed in the DOM control block (DOMC). Assuming the DOM macro instruction user has the proper authority and correctly indicated the deletion of WTO-WQEs and WTOR-WQEs, the appropriate WQEs and OREs are deleted.

Mainline Routine: IEAVMDOM – Communication Task DOM Processor

This routine processes the delete operator message control block (DOMC). For each message identification listed in the DOM control block, it scans the WQE chain. When a message is found with that identification, this routine also:

- Insures that the message is terminated if it is a multiple line WTO message (MLWTO).
- Frees the operator reply element (ORE) if the message is a WTOR message.
- Marks the WQE for deletion.

If there is an active graphic console in the system, SVC 72 is issued to permit messages to be deleted from the graphic console's storage area.

Subroutines:

AVAILID (Step 12)
When an operator reply element (ORE) is deleted, this subroutine places the reply identification for that ORE back into the reply identification bit map.

OREREMV (Step 11B)
When an operator reply element (ORE) is deleted, this subroutine unchains and frees the ORE.

FREEBUF (Step 10)
When an operator reply element (ORE) is deleted, this subroutine frees the temporary reply buffer pointed to by the ORE being deleted.

Extended Description | **Module**

Subroutines (continued):

GRAPHICS (Step 17)
This subroutine searches the unit control module entries (UCMEs) for active graphic consoles. For each active graphic console found, this subroutine calls the device support processor (SVC 72).

FREEDOMS (Step 19)
After all of the message identifications in the DOM control block (DOMC) have been processed, this subroutine is called to unchain and free the DOM control block.

POSTOECB
When a TSO terminal is in MONITOR mode and has received a WTOR message, an operator reply element-write wait block (ORE-WWB) is built. This subroutine posts the UCMOECB event control block that will permit the ORE-WWB to be freed.

BUILDEND (Step 7)
This subroutine builds the end-line message that is necessary to end a multiple line WTO (MLWTO) message.

MSGPROC (Step 13)
When WTOR messages are deleted from the system, operator responses to those messages are no longer needed. This subroutine prepares a message containing the message identifications of the deleted messages. The prepared message is then sent to the system operators informing them that these messages are no longer outstanding.

GETWPL
This subroutine obtains a write parameter list to issue a message.

SETLCKS (Step 18)
This subroutine obtains the local and CMS locks.

FRELCKS (Step 16)
This subroutine frees the CMS and local locks.

SUBEXIT (Steps 5b-c)
This routine builds the subsystem interface control blocks and passes the DOM control block to the subsystem.

SETESTAE (Steps 5f-i)
This routine creates the ESTAE recovery environment for this module.

SETFRRIN (Step 5k)
This subroutine creates the functional recovery routine for this module.

RELFRRIN
This subroutine removes the last recovery environment created for this module.

Also refer to the control block chaining diagram, Figure 5-1.

Diagram 1-21. DOM Communication Task Processing (IEAVMDOM) (Part 1 of 11)

Input

CVT

Unit Control Module (UCM)

UCMDECB

Unit Control Module (UCM)

UCMDOME

Process

From Step 2

From the Dispatcher (IEAVEDS0)

From Steps 3 & 21

IEAVMQWR (Wait Service Routine)

1 Wait for work to do. → WAIT Macro Instruction

2 Determine operation to be performed:

⋮

I. If delete operator message, branch to → Step 3

⋮

When there is no work to be done, branch to → Step 1

IEAVMDOM (DOM Processor)

From Step 2

3 If the DOM control block pointer is zero:

a. Set the DOM ECB to zero.

b. Check for other work for the communication task. → Step 2

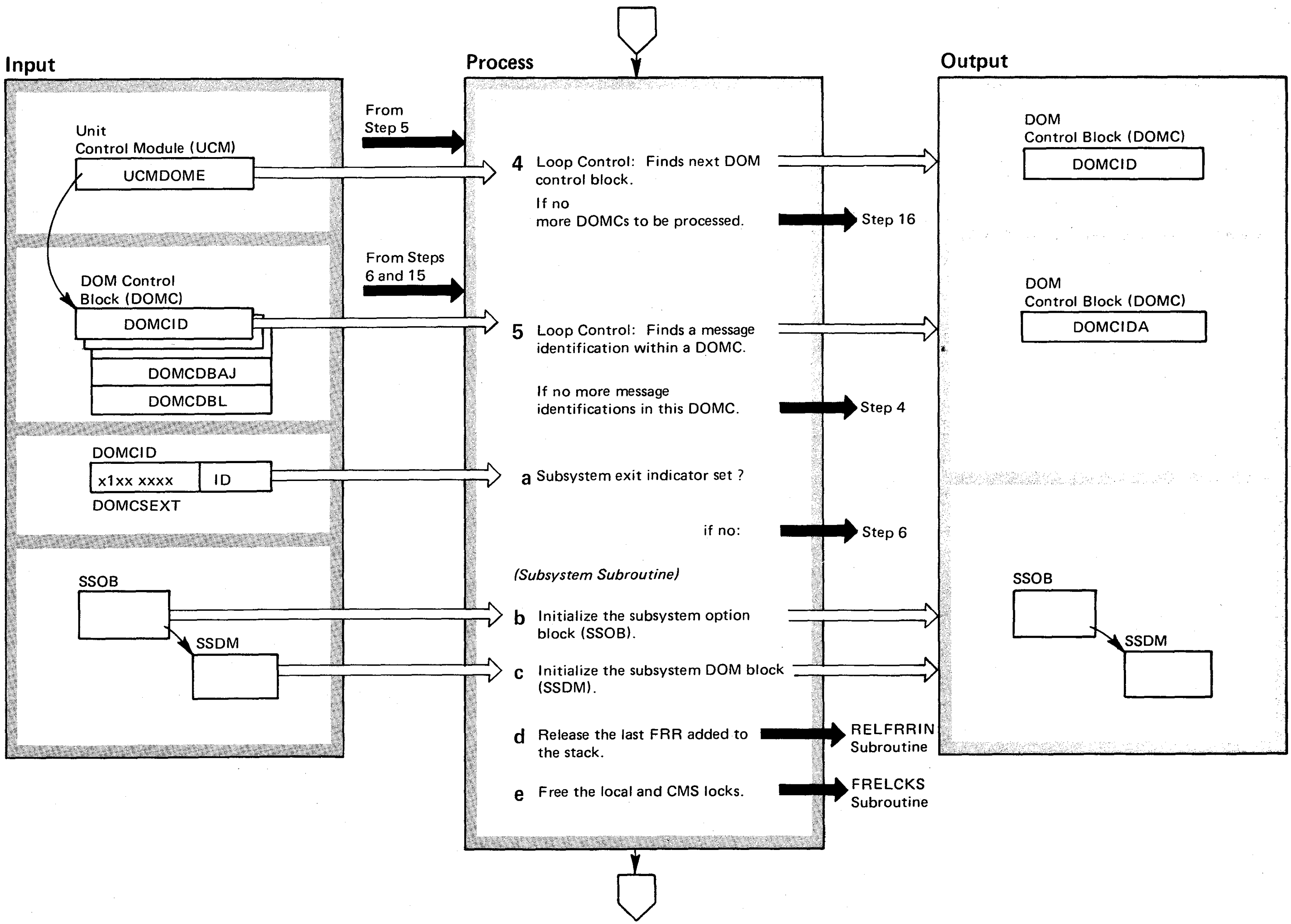
Output

Unit Control Module (UCM)

UCMDECB

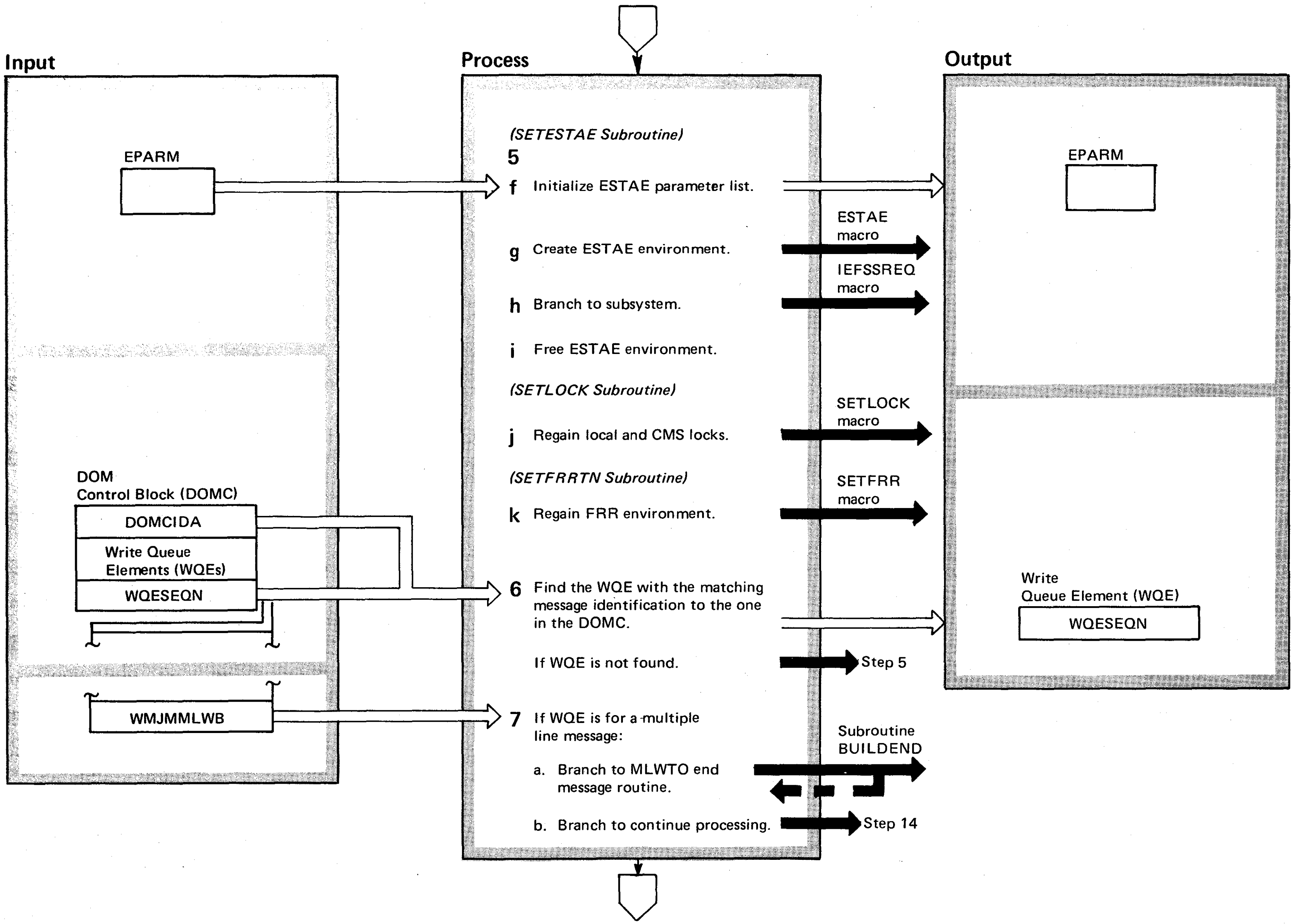
Extended Description	Module
1 During some previous operation, the wait service routine issued the WAIT macro instruction after it had determined there was no further work the communication task could perform at that time.	IEAVMQWR
2 Determine the operation to be performed and branch to the module that can perform that operation.	
I. For this particular set of method-of-operation diagrams, the DOM event control block (UCMDECB) was turned on by the DOM macro instruction processing routine.	
3 The pointer (UCMDOME) to the first control block on the DOM control block (DOMC) chain is tested for zero. If it is zero, there are no DOMCs to be processed; therefore, the event control block that started the DOM processing operation is turned off and control is returned to the wait service routine.	IEAVMDOM

Diagram 1-21. DOM Communication Task Processing (IEAVMDOM) (Part 3 of 11)



Extended Description	Module
4 This control determines the next DOM control block (DOMC) to have its message identifications processed. When there are no more DOMCs to be processed, this routine branches to an area that cleans up the queues and returns control to the dispatcher.	IEAVMDOM
5 This control determines the next message identification to be processed against the write-queue-element (WQE) chain. The WQEs contain the messages to be deleted.	
5a If this DOM control block has not been examined by the subsystem, the "exit-to-be-taken" bit will be on in the DOM control block (ID portion).	
5b A subsystem option block is created indicating DOM function.	
5c A subsystem DOM block is created to contain a pointer to the DOM control block.	
5d A SETFRR macro, specifying the delete option, is issued to delete the last created FRR on the stack for this module.	
5e All locks currently held are released. (These locks were set by the SETLCKS routine (step 18).	

Diagram 1-21. DOM Communication Task Processing (IEAVMDOM) (Part 5 of 11)



DOM Communication Task Processing (Part 5.2 of 11)

Extended Description **Module**

5f and 5g An ESTAE recovery routine is created to protect this module while the subsystem is in control.

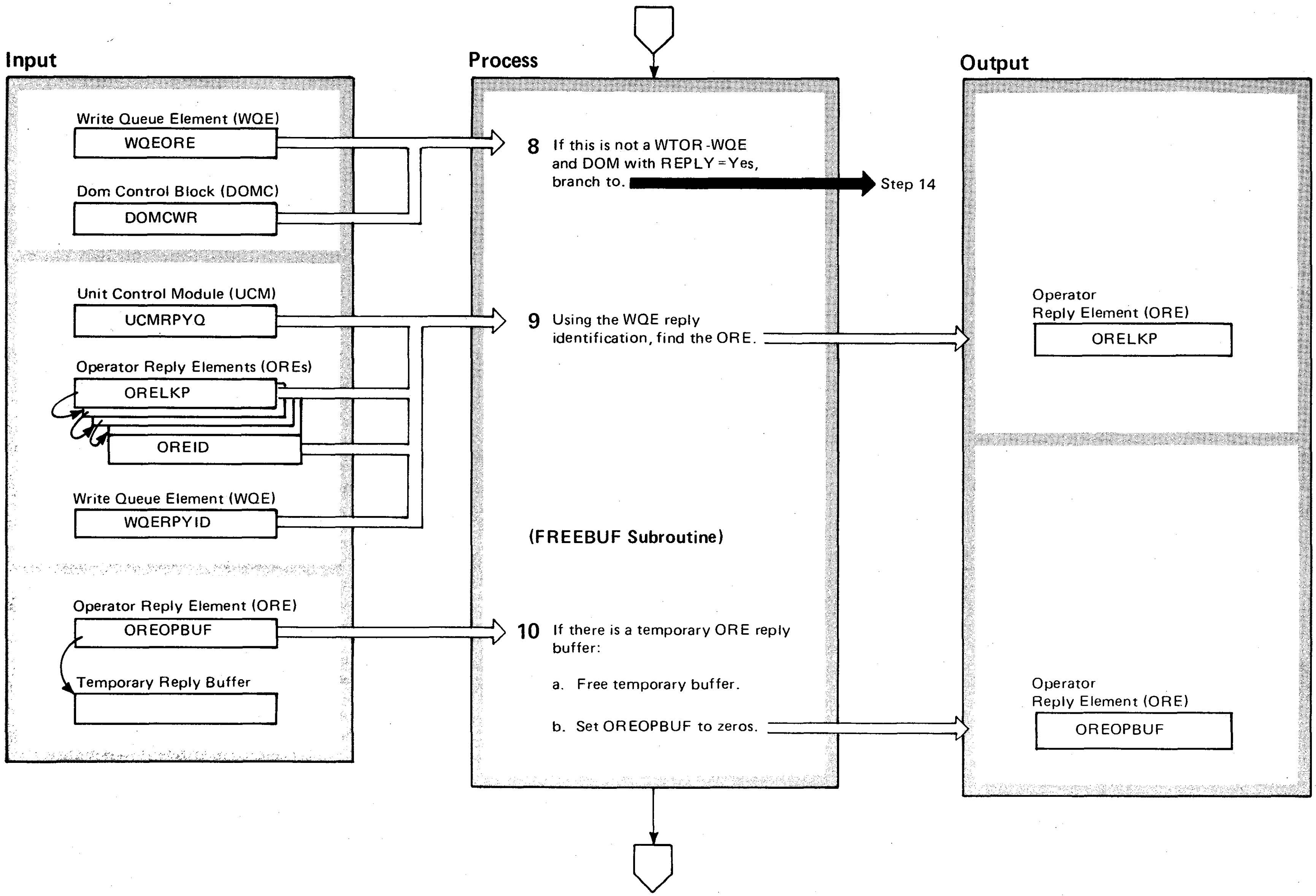
5h The created SSOB and SSDM are passed to the job entry subsystem exit routine, along with a pointer to the DOM control block. No response is expected from the exit routine other than the return of control.

5i – 5k Upon return from subsystem the ESTAE environment is freed and all locks and recovery exits are regained.

6 The DOM control block message identification is then compared against the message identification in each of the WQEs on the WQE chain. If no match is found, then the next DOM control block (DOMC) message identification is processed.

7 Having found a WQE-DOMC message identification match, this routine determines if the WQE is for a multiple line message (MLWTO). If it is, the BUILDEND routine is called to end the message.

Diagram 1-21. DOM Communication Task Processing (IEAVMDOM) (Part 6 of 11)



Extended Description **Module**

8 When an operator queue element (ORE) exists and is associated with a write queue element (WQE), that WQE and ORE were created by a single WTOR macro instruction. Before the WQE can be deleted, the ORE must be deleted. Two tests are made:

- If this WQE has an associated ORE, the WQE bit WQEORE has been turned on.
- If the WQEORE bit is on, the user who issued the DOM macro instruction must have included the REPLY=YES parameter in that macro instruction. If included, the bit DOMCWR is on.

If both conditions are met, the ORE will be deleted; proceed to the next step.

If either or both conditions are not met, the WQE will be deleted; bypass the ORE deletion steps.

9 To locate the operator queue element (ORE) associated with the WTOR created write queue element (WQE), start with the pointer (UCMRPYQ) in the unit control module (UCM) and search through the ORE chain for the first ORE having the same reply identification as the WQE (OREID versus WQERPYID). A match indicates which ORE is to be deleted.

10 Before the ORE can be deleted, a possibility exists that the console operator may have started to enter a reply, in which case a temporary buffer has been assigned to the ORE. When the pointer field (OREOPBUF) in the ORE is not zero, then a temporary buffer exists. When it exists, the buffer is freed and the ORE pointer field (OREOPBUF) is set to zero.

Diagram 1-21. DOM Communication Task Processing (IEAVMDOM) (Part 8 of 11)

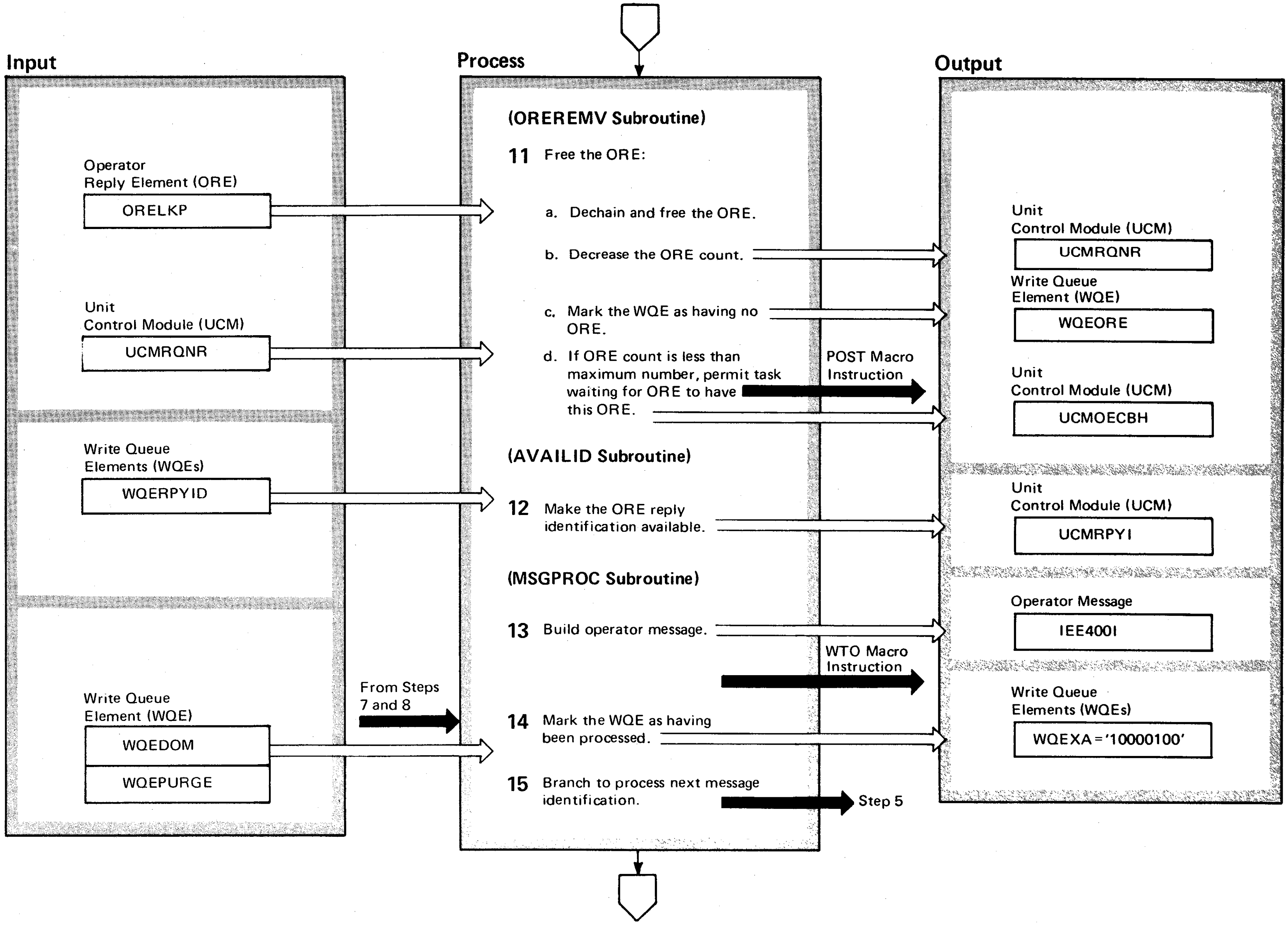


Diagram 1-21. DOM Communication Task Processing (IEAVMDOM) (Part 9 of 11)

Extended Description	Module
11 The operator reply element (ORE) associated with the write queue element (WQE) is unchained and freed. As a result of this action, this routine also decreases the ORE count, marks the WQE as having no ORE, and if the ORE count is below the normal system number of permissible OREs, the POST macro instruction is issued to an event control block that will eventually allow another user who is waiting for an ORE to obtain it.	
12 The ORE's reply identification is returned to the reply identification bit map. This particular reply identification can now be reassigned to another ORE.	
13 When an ORE is deleted, an operator message is prepared to inform the console operator that he no longer needs to respond to the message being deleted.	
14 The WQE is marked as DOM processing complete to prevent further ORE processing against this WQE.	
15 Branch to process the next message identification.	

Diagram 1-21. DOM Communication Task Processing (IEAVMDOM) (Part 10 of 11)

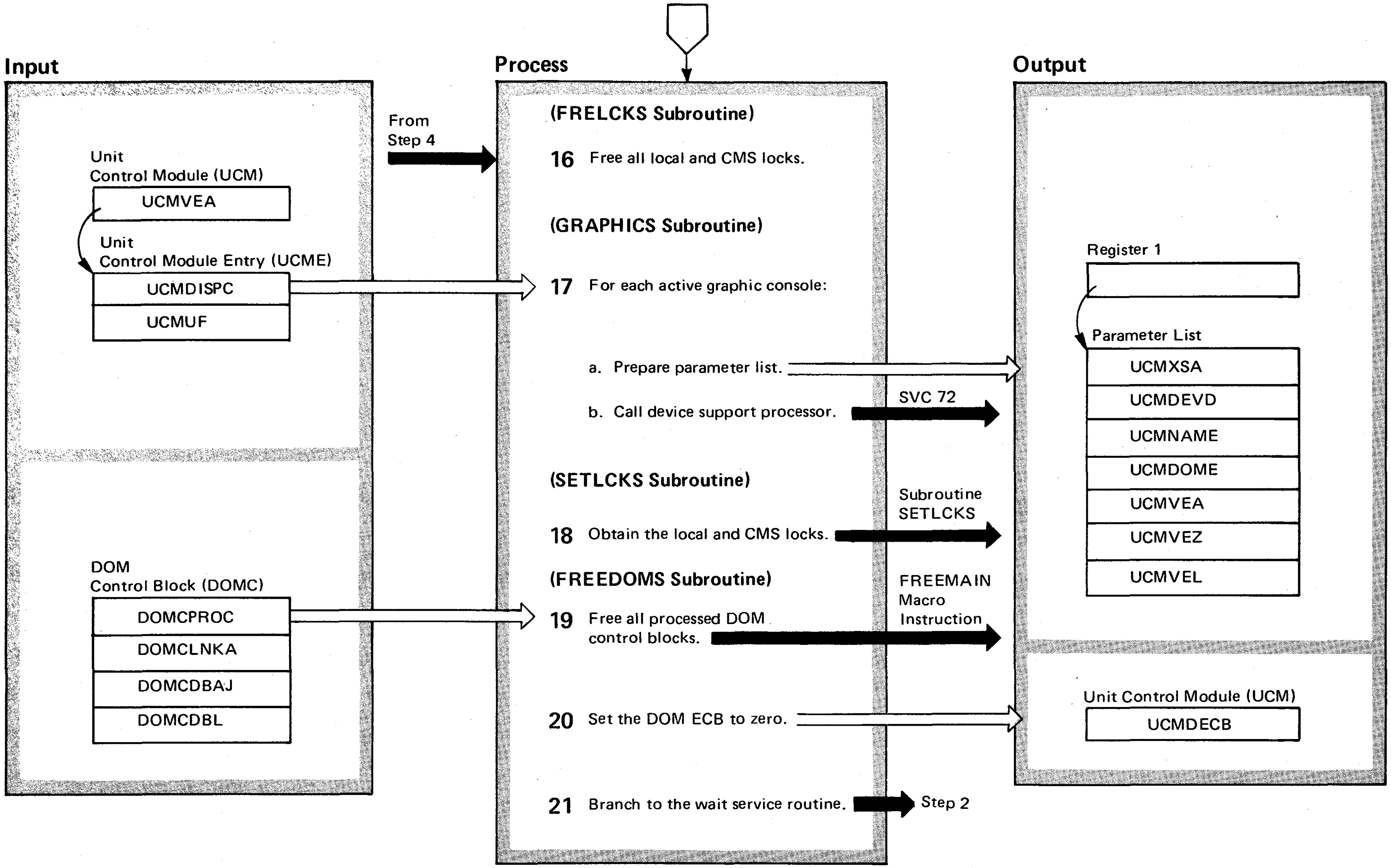


Diagram 1-21. DOM Communication Task Processing (IEAVMDOM) (Part 11 of 11)

Extended Description	Module
16 The current functional recovery routine (FRR) on the stack for this routine is released, and the local and CMS locks are freed.	
17 The unit control module entry (UCME) control blocks are tested for active graphic consoles. For each active graphic console, SVC 72 is issued. SVC 72 receives a parameter list pointed to by register 1. For SVC 72, see DOM Device Support Processing diagram.	
18 The local and CMS locks are obtained, and a functional recovery routine (FRR) for this routine is placed back on the stack.	
19 A FREEMAIN macro instruction is issued to release all of the DOM control blocks (DOMC) that have been processed.	
20 The DOM event control block (UCMDECB) is set to zero.	
21 Control is returned to the wait service routine.	

Diagram 1-22. DOM Device Support Processing (DIDOCS) (Part 1 of 2)

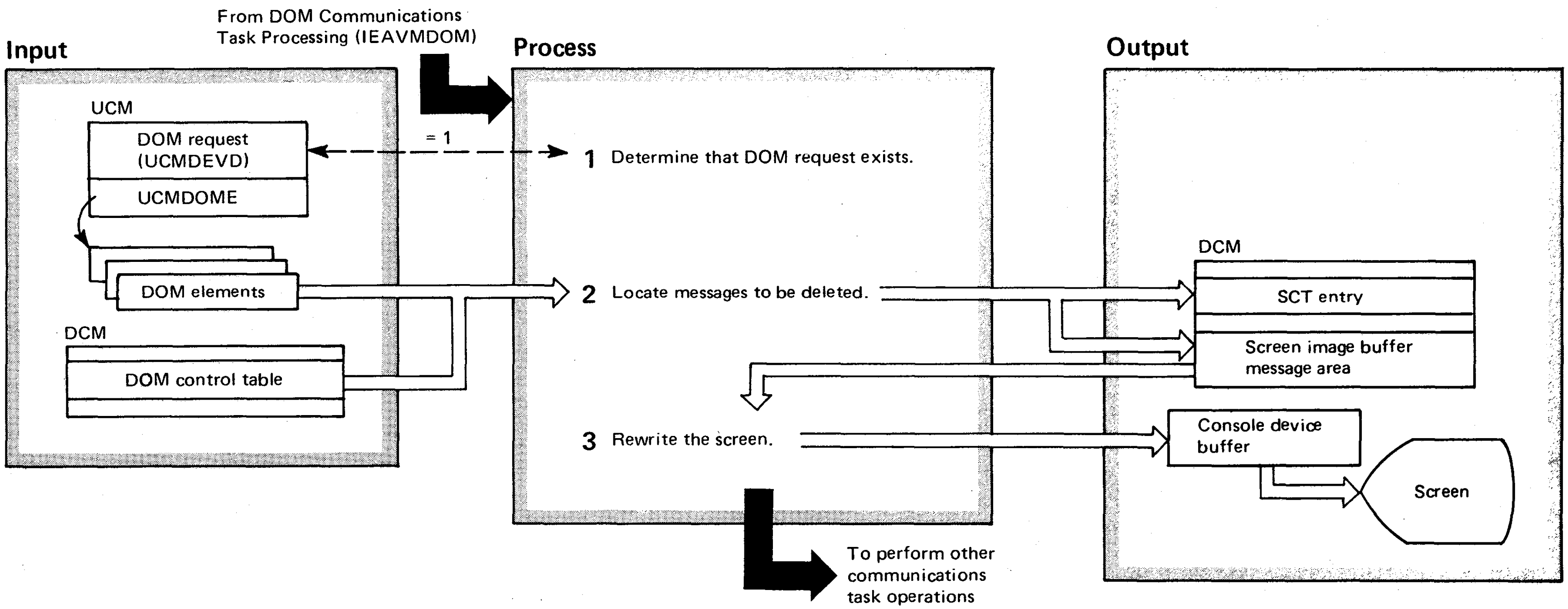


Diagram 1-22. DOM Device Support Processing (DIDOCS) (Part 2 of 2)

Extended Description	Module	Label
User programs and system routines issue a DOM macro instruction to remove messages from a graphics console screen. DIDOCS processes DOM requests as follows:		
1 DIDOCS checks bit UCMDEVD for an indication that DOM was issued.	IEECVET1	
2 DIDOCS locates the DOM element table through field UCMDOME. The DOM element table contains the protect keys or IDs of messages to be deleted. The DOM control table in the DCM contains the protect keys or IDs of messages that are displayed on the screen. DIDOCS compares the DOM elements with the DOM control table entries; if DIDOCS finds a match, it marks the message (in the SIB) with a vertical bar and marks the message's SCT entry to indicate that the message is deletable.	IEECVET7	
3 DOM rewrites the screen from the screen image buffer. The messages with vertical bars will not appear on the screen.	IEECVETH/P/R/U	

Diagram 1-23. External Interrupt Processing (Automatic Console Switch) (IEAVVCRX) (Part 1 of 6)

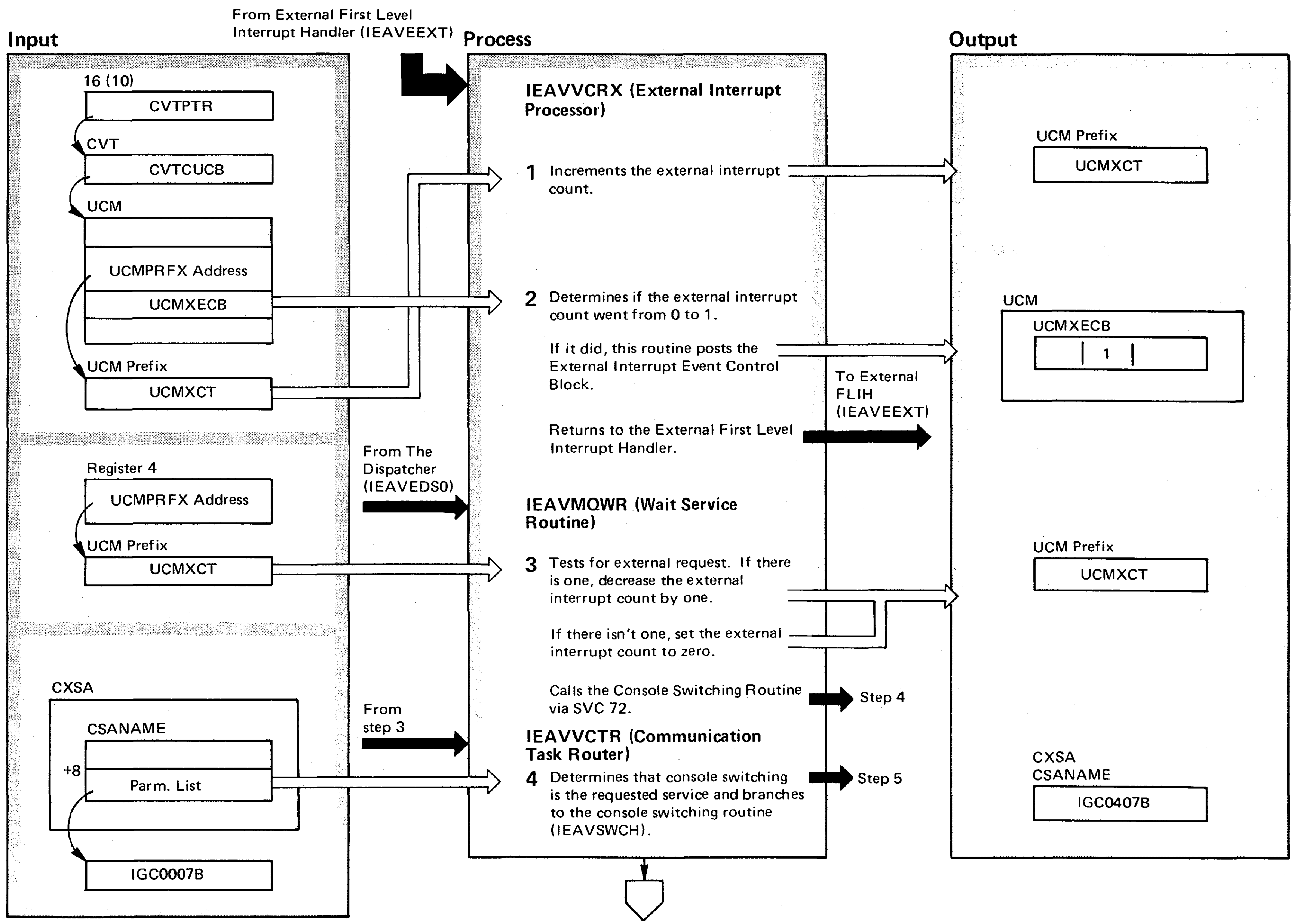


Diagram 1-23. External Interrupt Processing (Automatic Console Switch) (IEAVVCRX) (Part 2 of 6)

Extended Description	Module
Pressing the interrupt key on the operator console control panel switches the functions of the master console to an alternate console. The console switch routine performs that function.	
1 The count of external interrupts (field UCMXCT, in the UCMPRFX) is incremented by one using compare and swap.	IEAVVCRX
2 If no previous external interrupt was still queued (updated count equal 1), a branch (BALR 14, 15) is taken to the POST routine. Control returns to the External Interrupt Handler via a branch on register 2.	
3 The External Request Count (UCMXCT) is checked for zero. If not zero, the count is decreased by one using compare and swap. The Console Switching Routine (IEAVSWCH) is invoked via SVC 72. A code of 4 is passed indicating an external interrupt. If the count is zero, the External Event Control Block (UCMXECB) is zeroed and other event control blocks are processed.	IEAVMQWR
4 The address of the Extended Save Area is obtained. The parameter list is stored in the Extended Save Area and the first two words are compared to the entry point names of the various device support processors. Upon finding a match, a branch is taken to the corresponding routine. For IEAVSWCH, the entry point address is contained in UCMSWCH of the UCM. If no match is found, an XCTL macro instruction is issued for the indicated processor.	

Diagram 1-23. External Interrupt Processing (Automatic Console Switch) (IEAVVCRX) (Part 3 of 6)

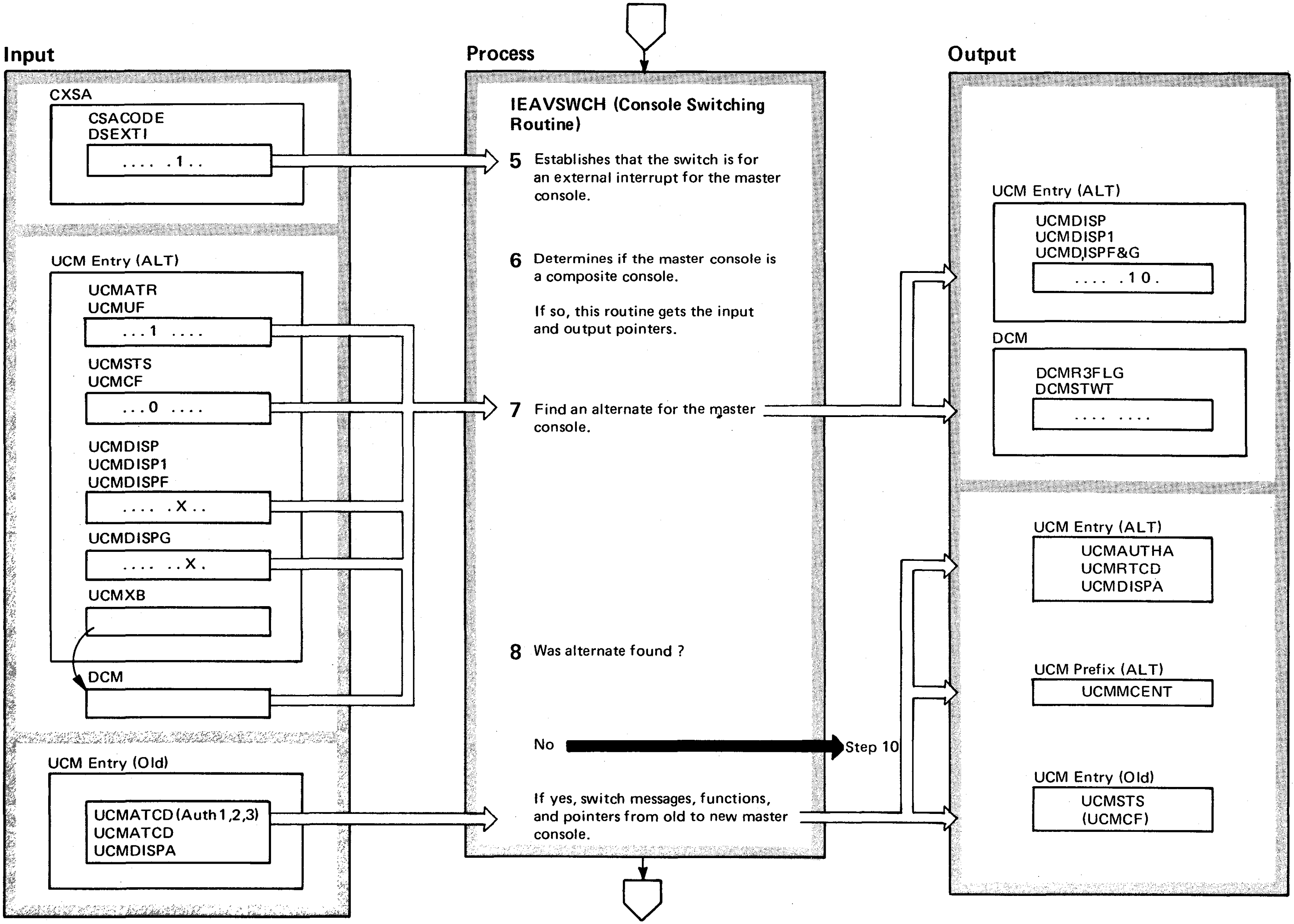


Diagram 1-23. External Interrupt Processing (Automatic Console Switch) (IEAVVCRX) (Part 4 of 6)

Extended Description	Module
5 The type of console switching is established (CSAEXTI on in CSACODE). A check determines if there are active consoles (UCMSYSE on in UCMSFLG1).	IEAVSWCH
6 Determines if the failing console is a composite. If so, mark output-half as tested (UCMDEVCC on in UCMDEVC). Since the master is being switched, UCMSYSD is set on in UCMSFLG1. Mark the failing console as tested.	
7 Search failing console's alternate chain for an active console (UCMUF on in UCMATE) without a CLOSE pending (UCMCF off in UCMSTS). If one is found and if fully capable of handling both input and output but is marked for output only, the console is switched to full capability (UCMDISPF on and UCMDISPG off in UCMDISP). Load address of resident DCM from UCMXB and turn off DCMSTWT in DCMR3FLG.	IEAVSWCH
8 Tests whether the search for an alternate master console was successful. If unsuccessful but there are other active consoles, this routine issues a message to all active consoles requesting the operator to enter a VARY MSTCONS command from any of those active consoles. (The master console is no longer an active console.) If unsuccessful and the master console had been the only active console, the system eventually hangs waiting for the console operator to restore the master console to the active. He does this by pressing the external interrupt key a second time. For those consoles having the alarm bell special feature, this routine rings the alarm bell three times. If the search was successful, this routine adds the authority and routing codes of the old master console to the found alternate master console. The messages are requeued from the old to the alternate master console.	IEAVSWCH

Diagram 1-23. External Interrupt Processing (Automatic Console Switch) (IEAVVCRX) (Part 5 of 6)

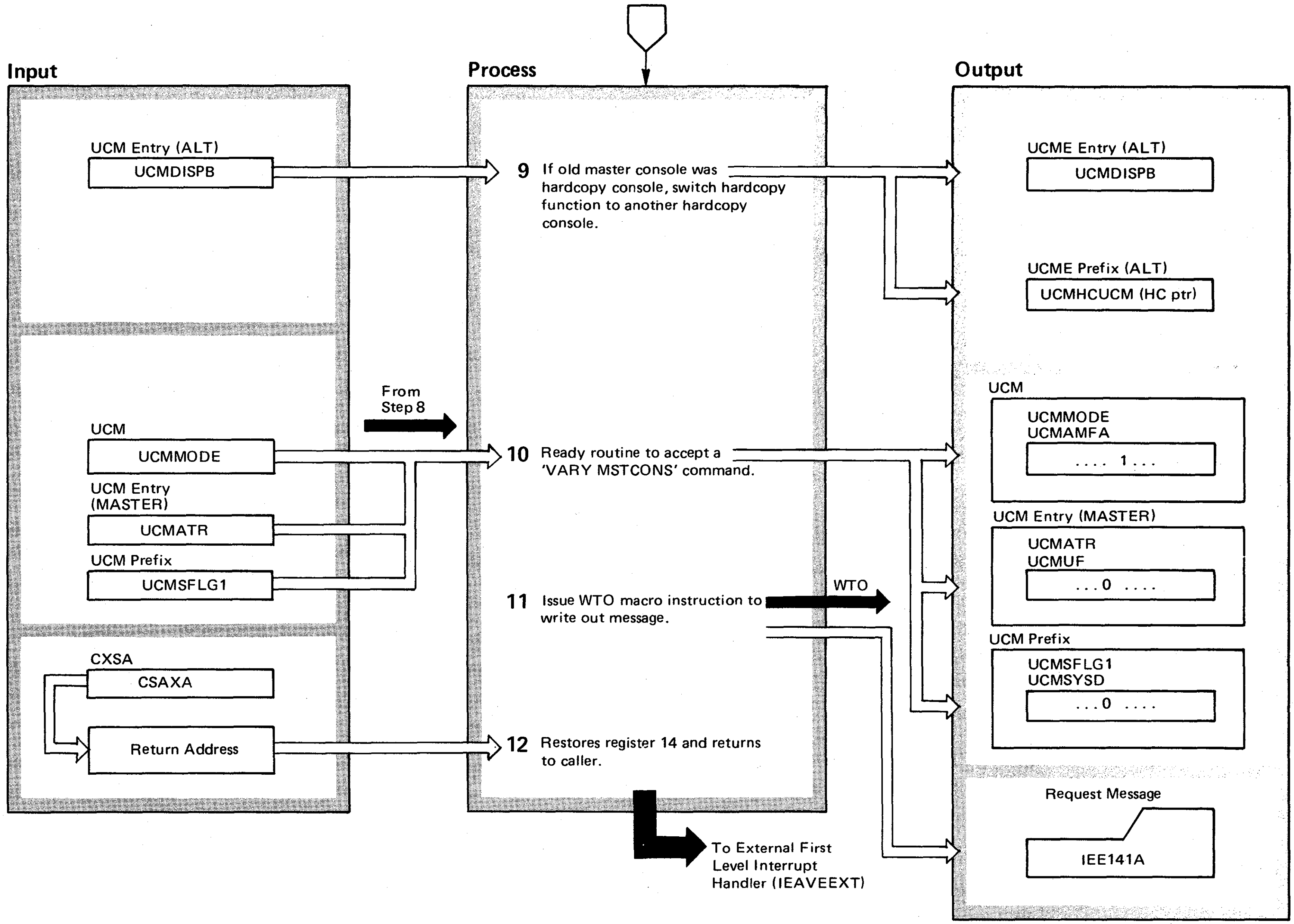


Diagram 1-23. External Interrupt Processing (Automatic Console Switch) (IEAVVCRX) (Part 6 of 6)

Extended Description | **Module**

9 If the old master console was also the hardcopy console, this routine switches the hardcopy function to a suitable console. If another hardcopy console is unavailable, message IEA964I is issued and hardcopy is suspended.

10 Indicate that a 'VARY MSTCONS' command will be accepted from any console (UCMAMFA set in UCMMODE). Mark master 'Not Active' (UCMUF off in UCMATR) and 'Failing Console is Master' (UCMSYSD off in UCMSFLG1).

11 Issue a WTO macro instruction to broadcast the message IEE141A.

12 Restore register 14 from CSAXA and return to caller via a branch register 14.

Diagram 1-24. Attention Interrupt Processing (Command Request) (IEAVVCRA) (Part 1 of 8)

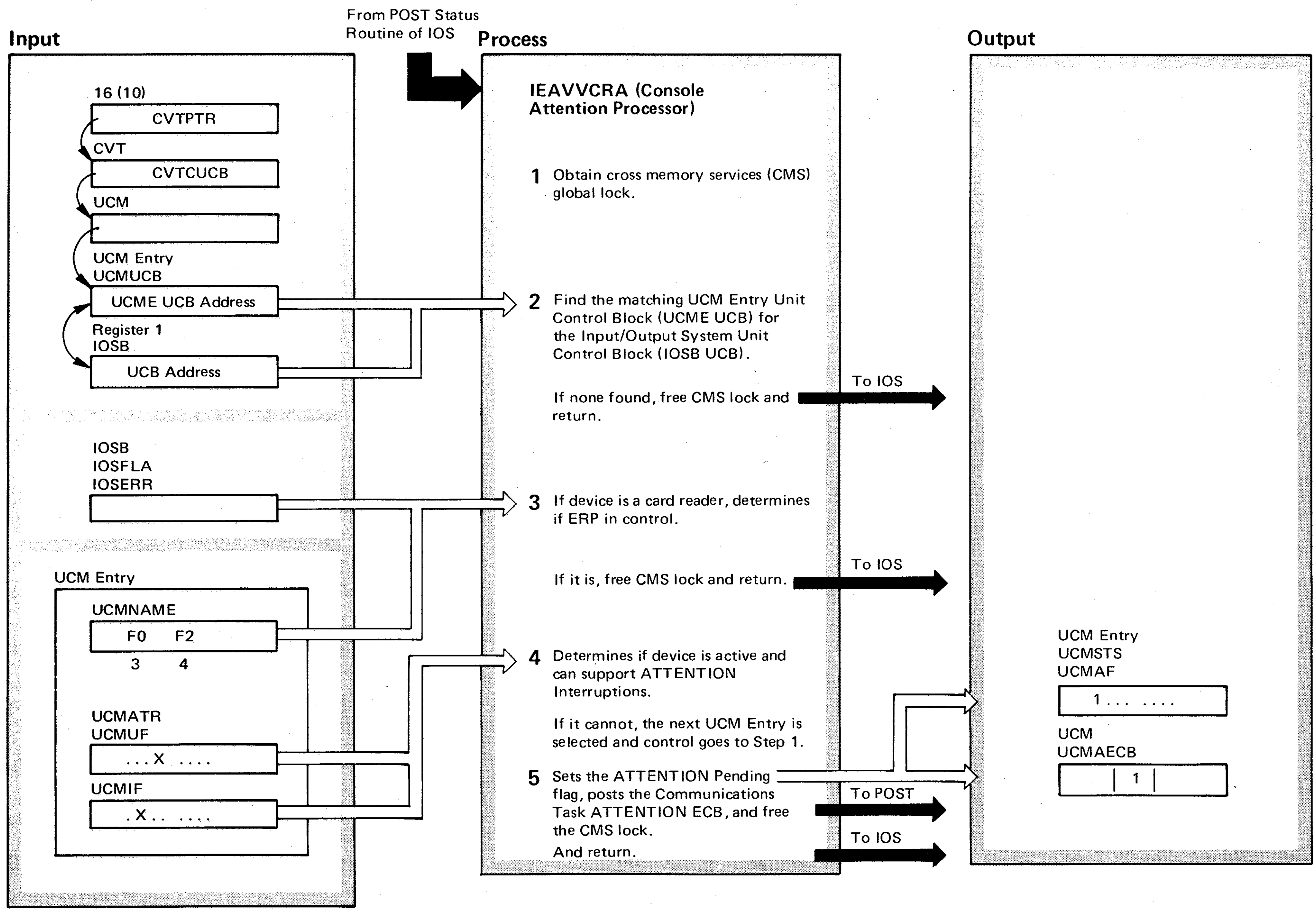
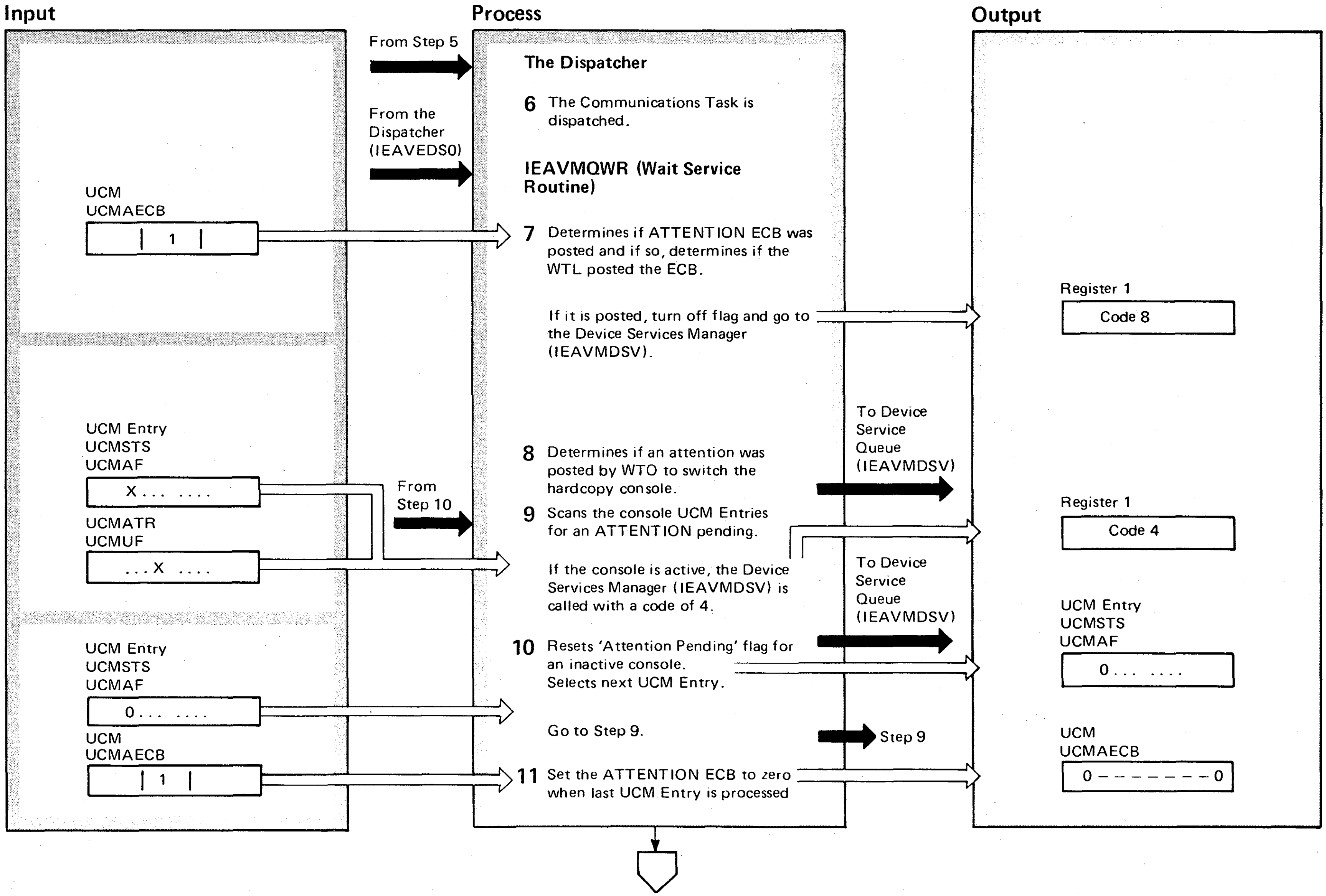


Diagram 1-24. Attention Interrupt Processing (Command Request) (IEAVVCRA) (Part 2 of 8)

Extended Description	Module
This procedure handles input from operator consoles signaled by the ATTENTION interruption.	
1 Obtain the CMS global lock.	IEAVVCRA
2 The UCB address from the IOSB is compared to the UCB address in the UCM Entry (UCMUCB). If they are not equal, the next UCM Entry is selected and the same comparison is repeated. If, however, the UCM Entry is the last, the CMS lock is freed, the registers are restored and control returns to IOS.	
3 If this is not the last UCM entry, a test determines if the ERP is in control (IOSERR set in IOSFLA of the IOSB) and if the device is a card reader (UCMNAME+3 and UCMNAME+4 in the UCME are X'F0' and X'F4' respectively). If both tests are valid, the CMS lock is freed, the registers are restored, and control returns to IOS.	
4 If the device is inactive, a CLOSE is pending for the device (UCMCF set in UCMSTS), or the device does not support ATTENTION interruptions, the next UCM Entry is selected and control returns to step 1.	
5 If the device is active (UCMUF on in UCMATR) and the device supports ATTENTION interruptions (UCMIF on in UCMATR), the 'Attention Pending' flag is set (UCMAF in UCMSTS). A branch to the POST processor is taken to post the ATTENTION ECB (UCMAECB). Upon return, the CMS lock is freed, the registers are restored and control returns to IOS.	

Diagram 1-24. Attention Interrupt Processing (Command Request) (IEAVVCRA) (Part 3 of 8)



Extended Description	Module
The Dispatcher	
6 The Dispatcher passes control to the Communications Task when its TCB (Task Control Block) is the highest priority ready TCB on the queue.	IEAODS
Communications Task Wait Service Routine	
7 The ATTENTION ECB is checked (UCMAECB in UCM) and, if posted, a test determines if Write-To-Log (WTL) posted the ECB (UCMSYSO on in UCMSFLG2). If so, a code of eight is loaded into register one to indicate that cleanup is needed.	IEAVMQWR
8 If the attention ECB was posted with an X'23' code, SVC 72 is called to switch the hardcopy SYSLOG to a console.	
9 The console UCM Entries are scanned for ATTENTION interruptions pending (UCMAF on in UCMSTS). If one is found and it is an active console (UCMUF on in UCMATR), register one is loaded with a code of four to indicate processing to be done by subroutine DEVSERVB of the Device Services routine (IEAVMDSV) which is called.	
10 If a UCM Entry is flagged for ATTENTION interruptions pending but the indicated device is not active, the ATTENTION interruptions pending flag (UCMAF in UCMSTS) is reset and the scan continues with the next UCM Entry.	
11 If the ATTENTION ECB (UCMAECB) is posted but no UCM Entry is found with an ATTENTION interruption pending, then all pending attention interruptions have been serviced and the ATTENTION ECB is set to zero. Processing continues for other types of ECBs.	

Diagram 1-24. Attention Interrupt Processing (Command Request) (IEAVVCRA) (Part 5 of 8)

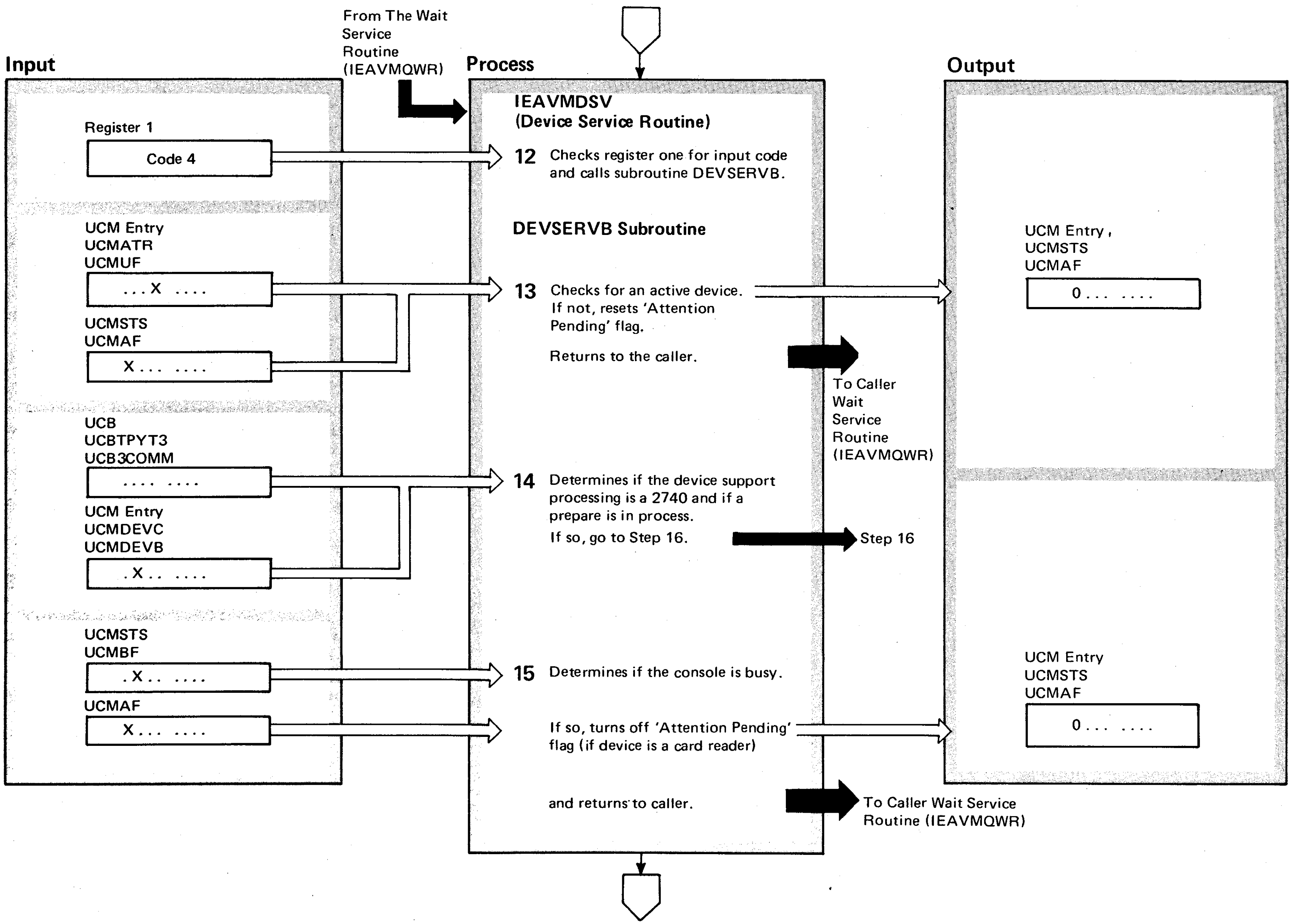


Diagram 1-24. Attention Interrupt Processing (Command Request) (IEAVVCRA) (Part 6 of 8)

Extended Description	Module
12 The input code in register one is checked. Subroutine DEVSERVB is called, if the code is a four, to process the ATTENTION interruption.	IEAVMDSV
13 A check is made to determine if the device is active (UCMUF on in UCMATR). If not, the 'Attention Pending' flag (UCMAF in UCMSTS) is reset and further checking is for I/O Completion. Then control returns to the caller.	
14 If the console is active, the console is a 2740 type (UCB3COMM on in UCBTPYT3) and a prepare command was issued (UCMDEVB on in UCMDEVC), then Subroutine DEVSERV is called to call the 2740 Device Support Processor.	
15 If the active console is not a 2740, a test is made for an 'Attention Pending' (UCMAF on in UCMSTS) on a console that is not busy (UCMBF off in UCMSTS). If so, Subroutine DEVSERV is called to call the appropriate device support processor. Otherwise, a check is made for a card reader. If not, checking continues for I/O Completion. If the device is a card reader, the 'Attention Pending' flag is reset. Control returns to the Wait Service Routine (EP=WRABXLE).	

Diagram 1-24. Attention Interrupt Processing (Command Request) (IEAVVCRA) (Part 7 of 8)

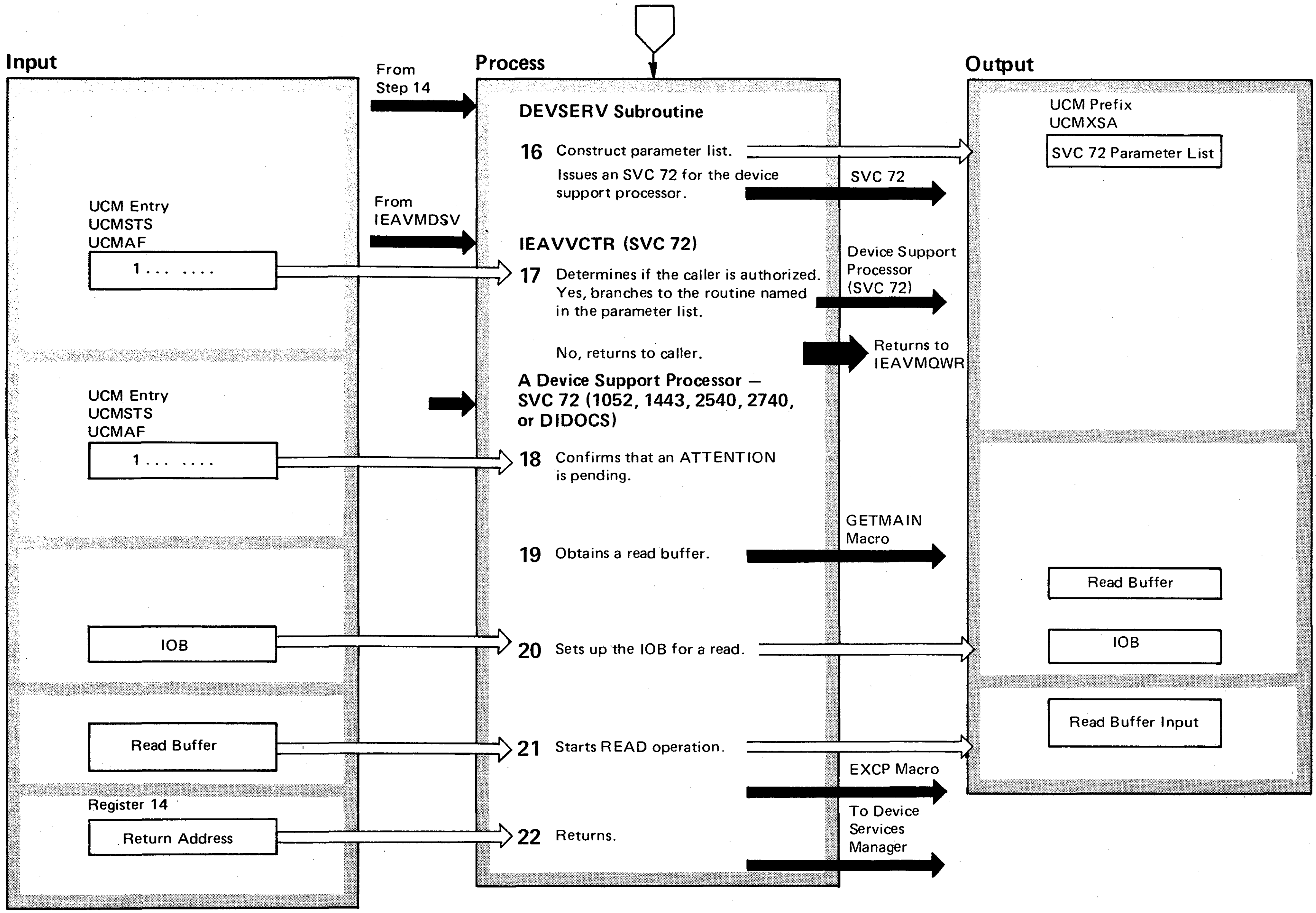


Diagram 1-24. Attention Interrupt Processing (Command Request) (IEAVVCRA) (Part 8 of 8)

Extended Description	Module
16 Subroutine DEVSERV prepares the parameter list and SVC 72 is issued. For SVC 72, see Processing Commands from a Console diagrams.	
17 Determines if the caller is in supervisor state (key 0). If so, the location of the routine supporting the console device type is located in the name list and a branch is made to that routine. If not found, an XCTL macro instruction is issued to the module named in the parameter list. If the caller is not in supervisor state (key 0), return to caller.	
18 The device support processor confirms the ATTENTION interruption pending.	IEACVET1, IEAV1052, IEAV1443, IEAV2540, IEEC2740, or IEECVETW
19 A GETMAIN macro instruction is issued to obtain a read buffer.	
20 The IOB is set up for the READ operation.	
21 An EXCP or READ macro instruction starts the READ operation.	
22 Control then returns to the Device Services Manager (IEAVMDSV).	

Diagram 1-25. Processing Commands From a 1052, 2540, or 2740 Console (Part 1 of 2)

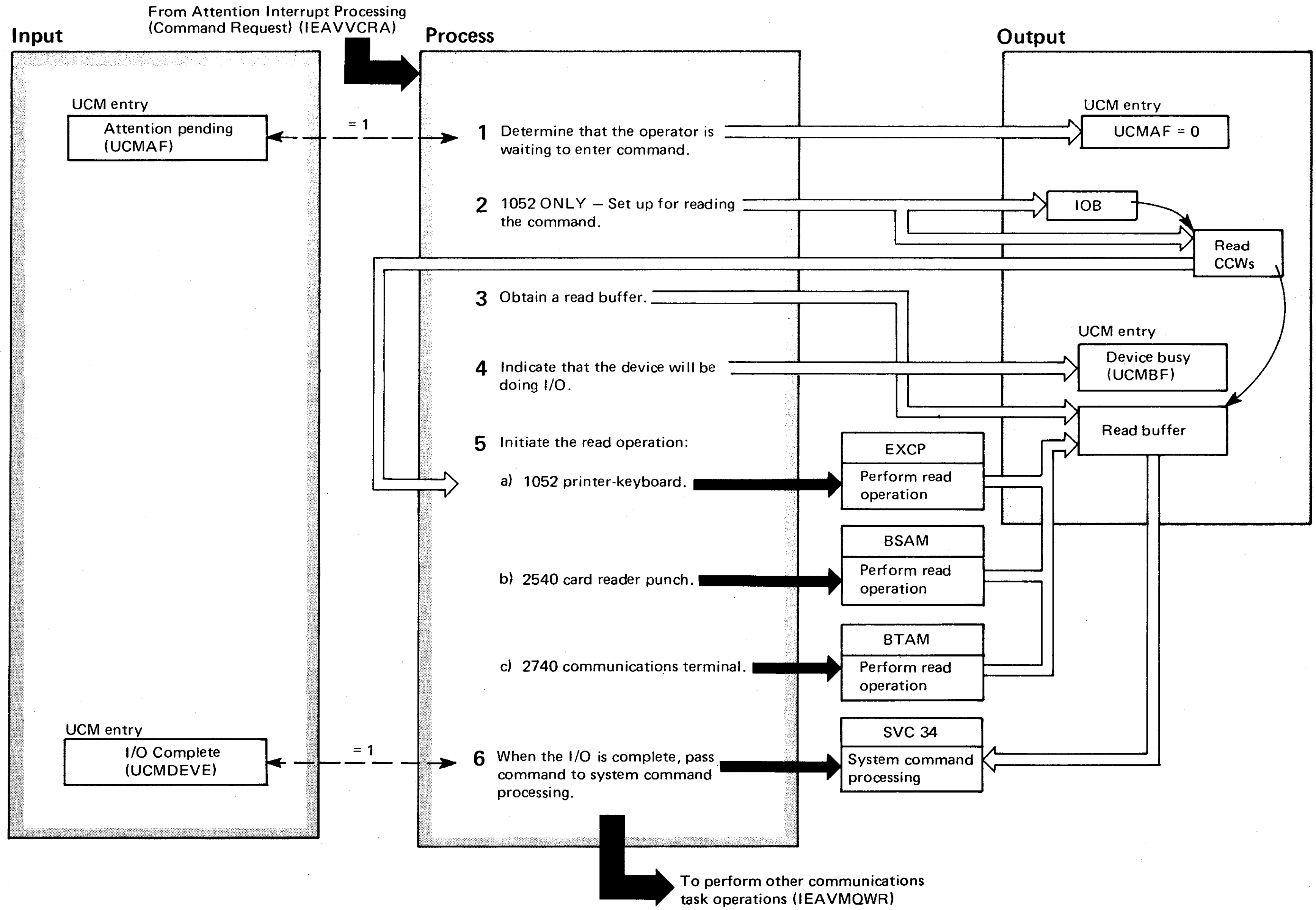


Diagram 1-25. Processing Commands From a 1052, 2540, or 2740 Console (Part 2 of 2)

Extended Description	Module	Label
Three console devices that enable the operator to communicate with the system are the printer-keyboard, the card reader, and the 2740 communications terminal. The operator uses these devices to enter commands into the system. The communications task device support processors (DSPs) are:		
a) For the 1052 printer-keyboard, 3210 console printer-keyboard, and 3215 console printer-keyboard, the DSP is IEAV1052.	IEAV1052	
b) For the 2540 card reader punch, 2501 card reader, 2520 card reader punch, and 3505 card reader, the DSP is IEAV2540.	IEAV2540	
c) For the 2740, IEEC2740.	IEEC2740	
These DSPs process operator-entered commands as follows:		
1 If the attention-pending bit (UCMAF) in the console's UCM entry is on, the operator is waiting to enter a command. The DSP turns off the attention-pending bit.		
2 If the console is a 1052 printer-keyboard, the 1052 DSP obtains an IOB for the read operation and places the address of the read CCWs into the IOB.		
3 The DSPs obtain storage for the input buffer and blank it. For the 1052 printer-keyboard console, the 1052 DSP initializes the read CCWs with the buffer address and buffer length.		
4 The DSPs set the device-busy bit (UCMBF) in the console's UCM entry to indicate that an I/O operation is taking place on the device.		
5 The DSPs initiate the read operation:		
a) For a 1052 printer-keyboard, the 1052 DSP issues an EXCP macro instruction to execute the channel program that reads the command into the buffer.	IEAV1052	PMEXCP
b) For a 2540 card reader punch, the 2540 DSP issues a READ macro instruction to pass control to BSAM; BSAM reads the command into the read buffer.	IEAV2540	PMEXCP
c) For a 2740 communications terminal, the 2740 DSP issues a READ macro instruction to pass control to BTAM; BTAM reads the command into the read buffer.	IEEC2740	PREPC
6 When the read operation that was initiated in step 5 is complete (bit UCMDEVE is on), the DSP passes the command to the system command processing routine (SVC 34).		

Diagram 1-26. Processing Typed Commands From a Graphics Console (DIDOCS) (IEECVET1) (Part 1 of 2)

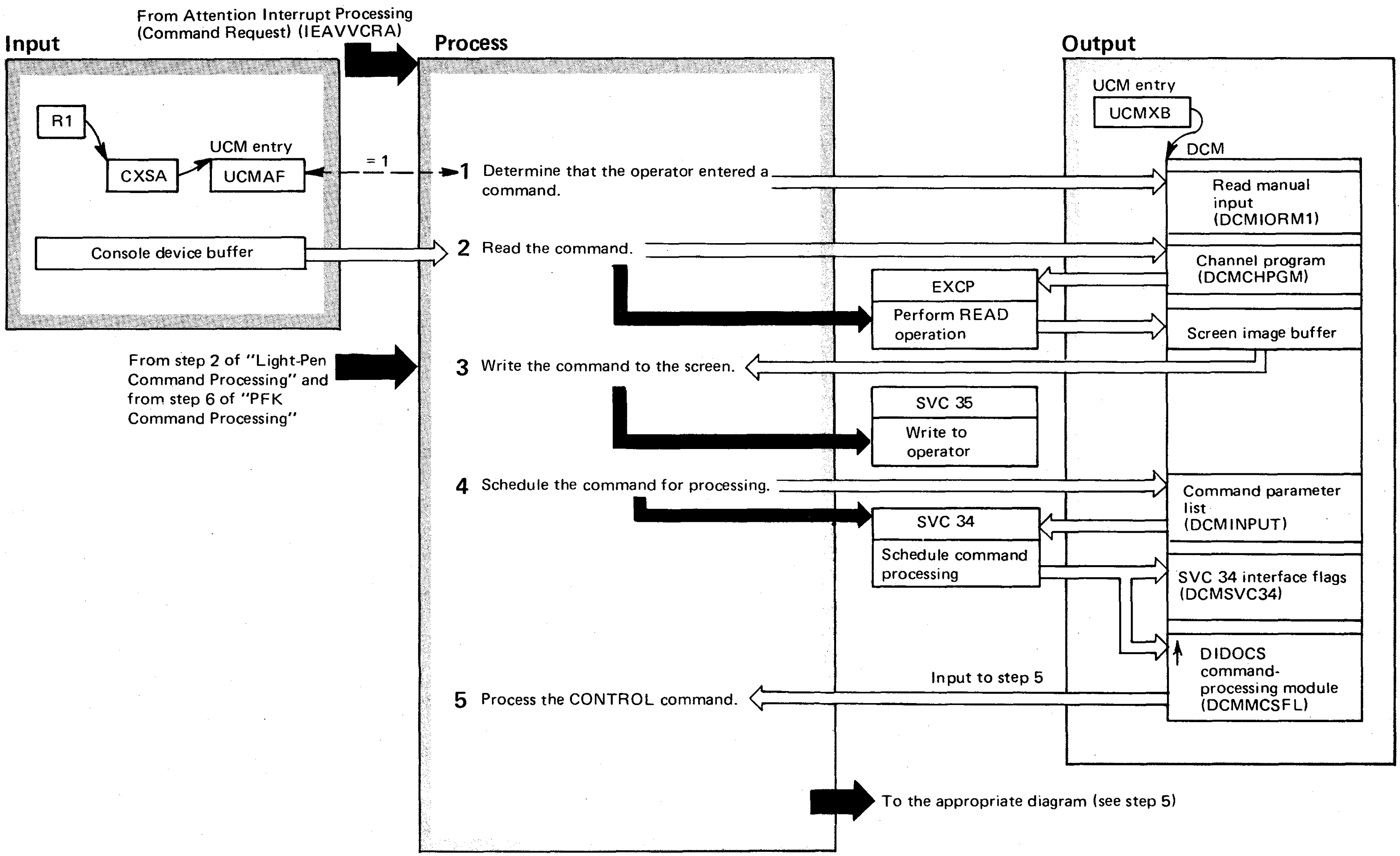


Diagram 1-26. Processing Typed Commands From a Graphics Console (DIDOCS) (IEECVET1) (Part 2 of 2)

Extended Description	Module	Label
Most display consoles include a device that enables the operator to communicate with the system. One such device is a typewriter keyboard. DIDOCS processes commands from a typewriter-keyboard console as follows:		
1 If the attention-pending bit (UCMAF) in the UCM entry for the console is on, the operator has entered a command. DIDOCS sets the read-manual-input bit (DCMIORM1) in the console's DCM; this bit notifies the device's I/O routines to read the typed command.	IEECVET1	
2 DIDOCS builds a channel program in field DCMCHPGM of the DCM and issues an EXCP to cause a read operation. The read operation reads the command from the console device buffer into the DCM screen image buffer (SIB).	IEECVET4 IEECVETH/P/R/U	
3 After the command is in the SIB, DIDOCS checks the command syntax. If a syntax error exists, DIDOCS issues a message to the operator. If the command syntax is correct and the command is not a CONTROL[1] command, DIDOCS issues a WTO macro instruction specifying that the command is to be the message text. The communications task WTO routine (SVC 35) writes the command to the screen.	IEECVET4	

[1] DIDOCS does not write CONTROL commands to the screen.

Extended Description	Module	Label
4 DIDOCS builds a command parameter list in field DCMINPUT, then passes the command to the system command processing routine (SVC 34). The command processing routine schedules the command for processing; for CONTROL commands requiring further DIDOCS processing, the command processing routine passes in field DCMMCSFL the address of the DIDOCS module that performs the processing.		
5 The following list indicates which CONTROL commands require more DIDOCS processing and which diagram describes the processing:		

• K • K E • K E,SEG • K E,nn • K E,F	See "Operator-Requested Message Deletion"
• K N,PFK	– See "PFK Definition or Redefinition."
• K S	– See "Changing Message Deletion Specifications."
• K E,PFK • K D,PFK	See "Erasing or Displaying the PFK Display Line."
• K E,D	– See "Erasing Status Displays."
• K D,H	– See "Holding Status Displays."
• K D,F	– See "Framing Status Displays."
• K D,U	– See "Updating Status Displays."

Diagram 1-27. Processing Light-Pen and PFK Commands From a Graphics Console (DIDOCS) (IEECVETF) (Part 1 of 2)

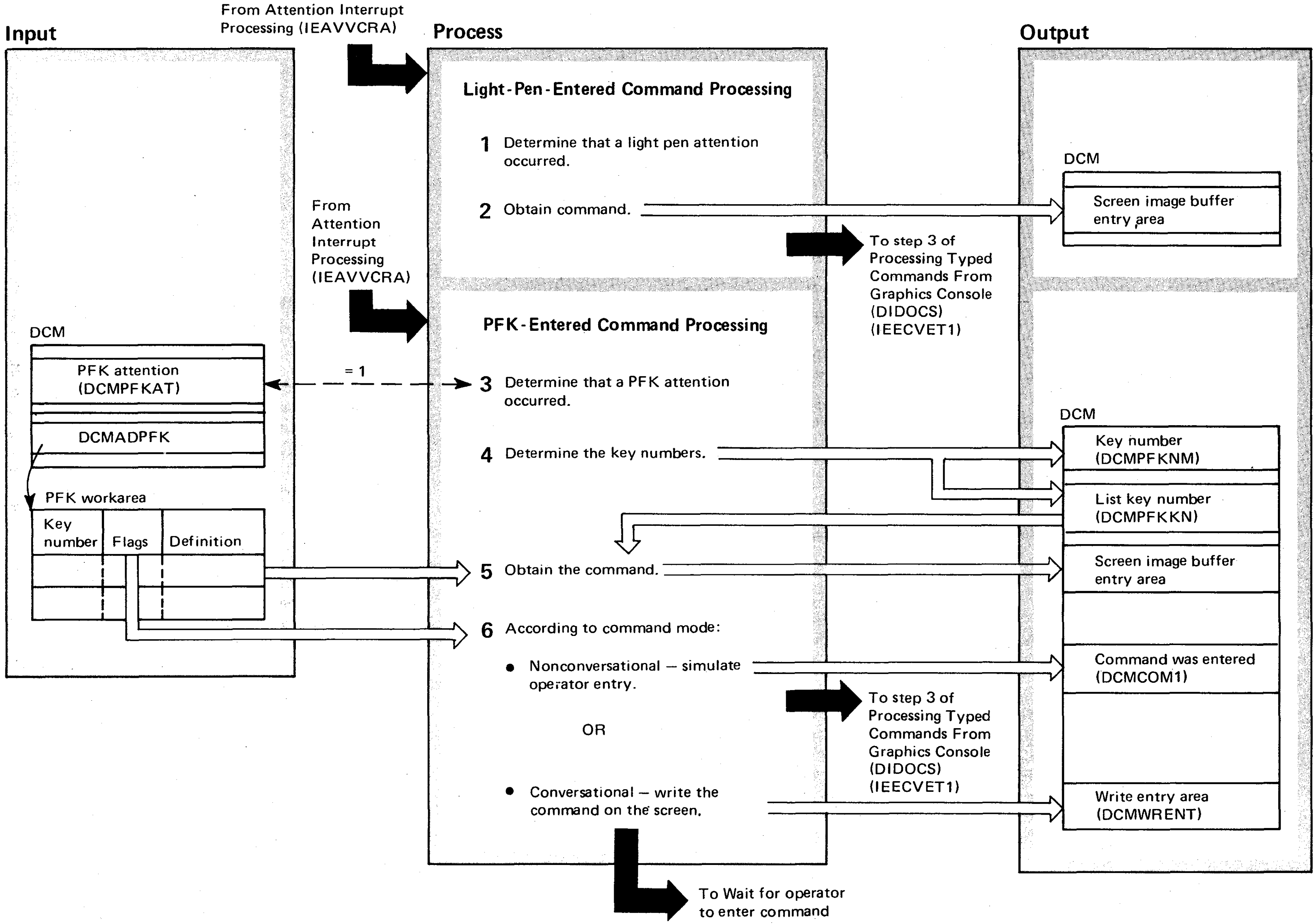


Diagram 1-27. Processing Light-Pen and PFK Commands From a Graphics Console (DIDOCS) (IEECVETF) (Part 2 of 2)

Extended Description	Module	Label
Light-Pen Entered Command Processing Most display consoles include a device that enables the operator to communicate with the system. One such device is a light pen. DIDOCS processes commands that the operator enters using a light pen, as follows:		
1 DIDOCS determines by the location of the light pen detection whether the light pen was positioned over a displayed PFK number or a screen indicator (*E, *U, *H, *D, C, K, or *F). If the location of the light pen detection is a displayed PFK number, DIDOCS processes the attention as if it were a PFK-entered command; processing continues at step 4 of "PFK-Entered Command Processing."	IEECVETF	
2 If the light pen was positioned over a screen indicator, DIDOCS places the text of the command in the entry area of the screen image buffer. Processing of the command continues beginning at step 3 of "Processing Typed Commands From a Graphics Console (DIDOCS)."	IEECVETF	

Extended Description	Module	Label
PFK-Entered Command Processing Another device that enables the operator to communicate with the system is a program function key (PFK). DIDOCS processes commands that the operator enters using a PFK, as follows:		
3 DIDOCS determines that a command was entered using a PFK in one of two ways –		
– Either a light-pen attention occurred and the light pen was positioned over a displayed PFK (see step 1 above)	IEECVETF	
– Or flag DCMPFKAT is on indicating that the operator pushed a program function key.	IEECVFTA	
4 DIDOCS places in field DCMPFKNM the key number of the PFK that caused the attention. For lists of keys, DIDOCS places the key's number in the list in field DCMPFKKN.	IEECVFTA	
5 DIDOCS compares the key being processed with allocated keys in the PFK workarea (pointed to by field DCMADPFK). If the key is valid, DIDOCS moves the first command associated with the key into the entry area of the screen image buffer. If the key is invalid, DIDOCS issues a message to the operator.	IEECVFTA	
6 DIDOCS checks flags in the key's PFK workarea entry to determine the command mode –		
• If command mode is nonconversational, DIDOCS indicates that a command must be processed by setting bit DCMCOM1. Then DIDOCS processes the command as described beginning at step 3 of "Processing Typed Commands From a Graphics Console (DIDOCS)."	IEECVFTA IEECVET1	
• If command mode is conversational, DIDOCS writes the entry area to the console device buffer; command processing continues after the operator enters the command.	IEECVETH/P/R/U IEECVET1	

Diagram 1-28. Operator Requested Message Deletion (DIDOCS) (IEECVET8) (Part 1 of 2)

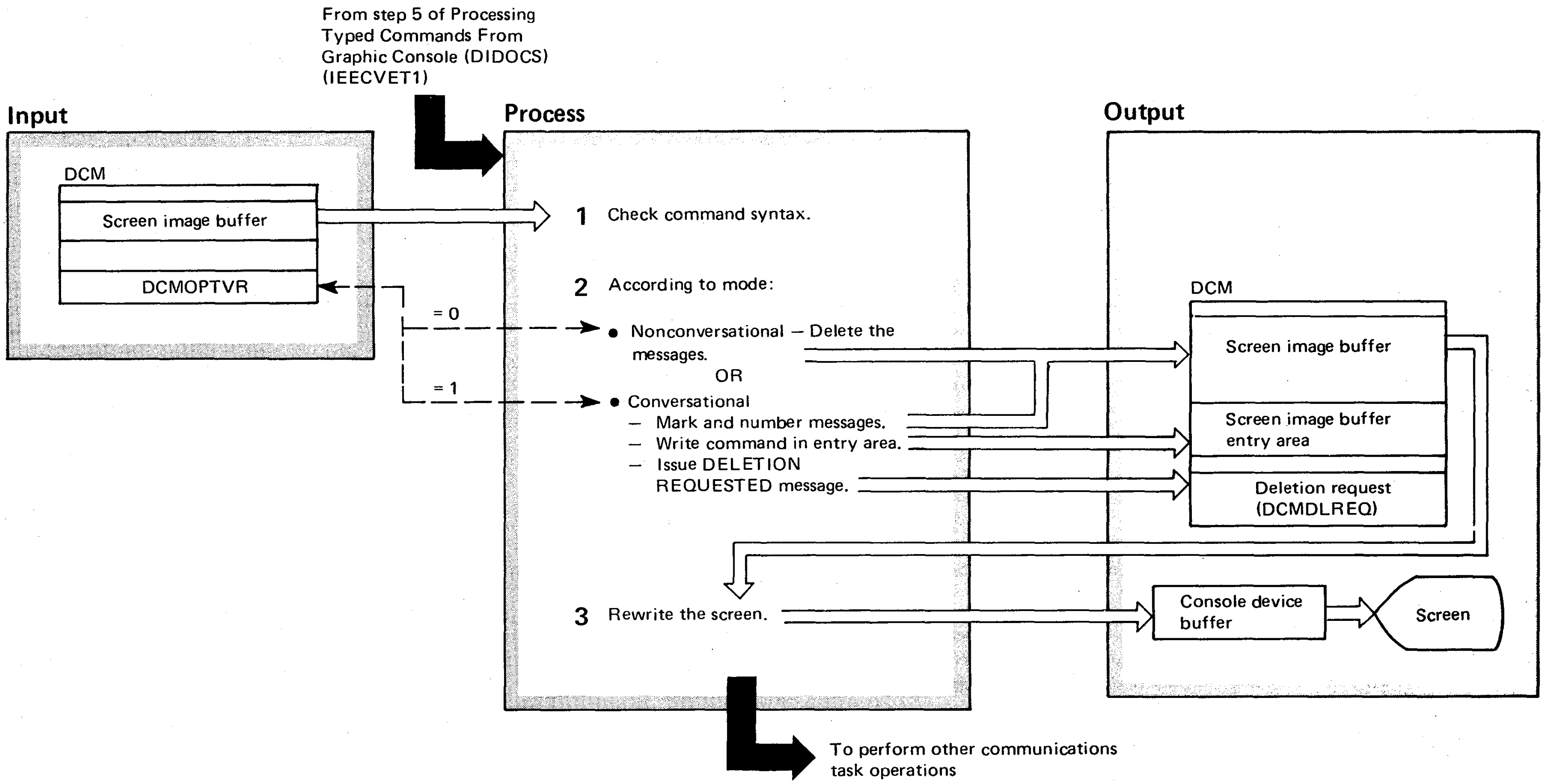


Diagram 1-28. Operator Requested Message Deletion (DIDOCS) (IEECVET8) (Part 2 of 2)

Extended Description	Module	Label
The operator uses the following CONTROL commands to delete messages from the graphics console screen:		
• K – specifies that a segment of messages should be deleted.	IEECVET8	
• K E – specifies that a segment of messages should be deleted.	IEECVET8	
• K E,SEG – specifies that a segment of messages should be deleted.	IEECVET8	
• K E,nn – specifies that a single message (nn) or a range of messages (nn,nn) should be deleted.	IEECVET6	
• K E,F – specifies that flagged messages should be deleted.	IEECVET6	
DIDOCS deletes the messages as follows:		
1 DIDOCS checks the command syntax. If a syntax error exists, DIDOCS issues an error message.	IEECVET6/8	

Extended Description	Module	Label
2, 3 DIDOCS checks bit DCMOPTVR to determine the message deletion mode –		
• If DCMOPTVR is zero, message deletion mode is non-conversational. DIDOCS removes the messages from the message area of the screen image buffer. Then, DIDOCS writes the screen image from the SIB into the console device buffer.	IEECVET9 IEECVETH/P/R/U	
• If DCMOPTVR is one, message deletion mode is conversational. DIDOCS marks deletable messages with a vertical bar, numbers the messages, and places the command in the entry area of the screen image buffer. Then DIDOCS sets bit DCMDLREQ to indicate that the DELETION REQUESTED message must be issued to the operator. When the operator enters the next command, DIDOCS processes the command in the entry area, then rewrites the screen image from the SIB into the console device buffer.	IEECVET6/8 IEECVETD IEECVET9 IEECVETH/P/R/U	

Diagram 1-29. PFK Definition or Redefinition (DIDOCS) (IEECVFTB) (Part 1 of 2)

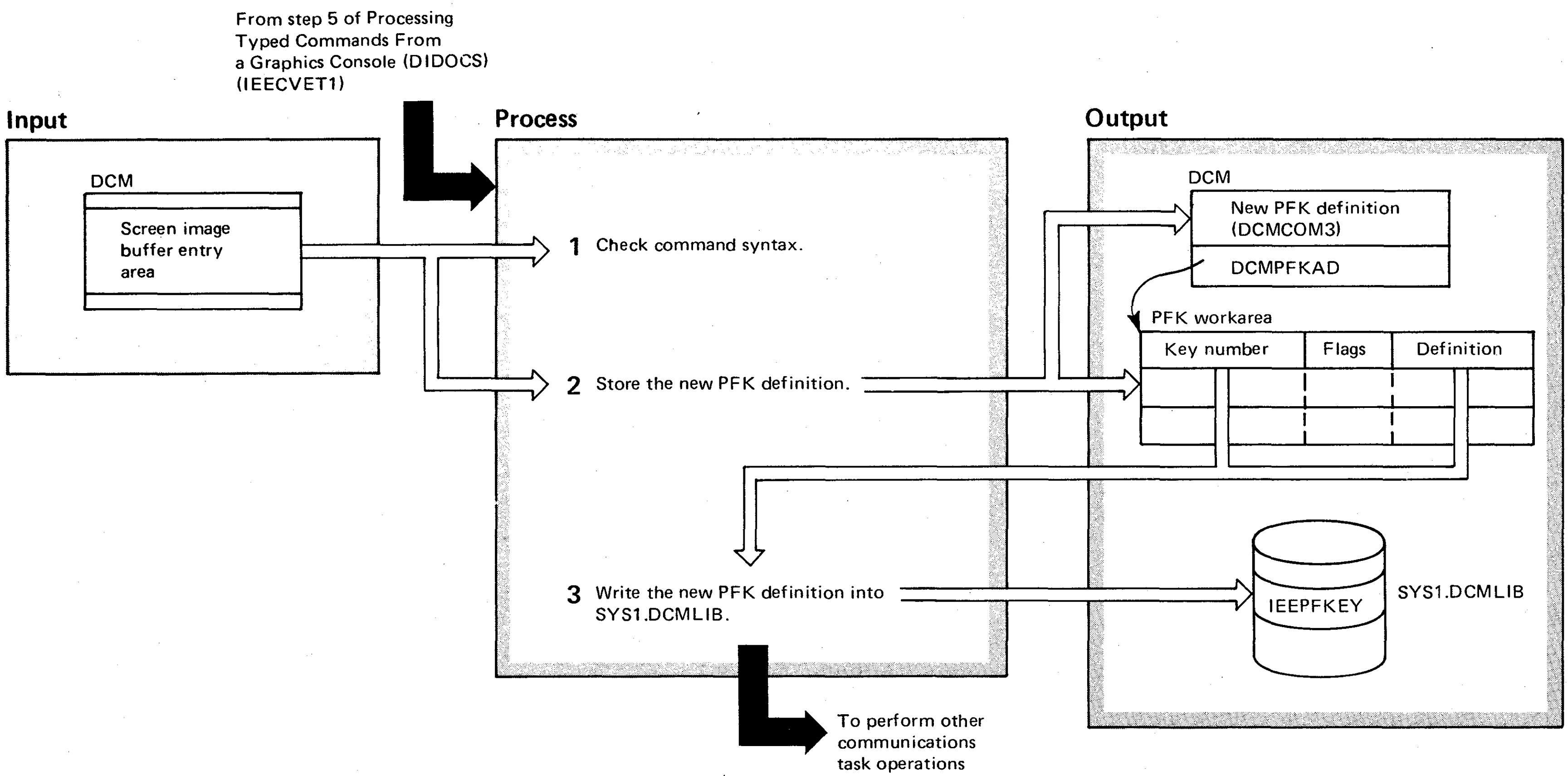


Diagram 1-29. PFK Definition or Redefinition (DIDOCS) (IEECVFTB) (Part 2 of 2)

Extended Description	Module	Label
The operator defines or redefines a program function key using the K N,PFK= command. DIDOCS processes the definition or redefinition, as follows:		
1 DIDOCS checks the syntax of the command, which is in the SIB entry area. If a syntax error exists, DIDOCS issues a message to the operator.	IEECVFTB	
2 If the syntax of the command is valid, DIDOCS moves the new definition from the SIB to the PFK workarea (pointed to by DCMPFKAD). DIDOCS sets bit DCMCOM3 to indicate that a new PFK definition has been entered by the operator.	IEECVFTB	
3 DIDOCS constructs a channel program and issues an EXCP macro instruction to write the new PFK definition in member IEEPFKEY of SYS1.DCMLIB.	IEECVFT1	

Diagram 1-30. Changing Message Deletion Specifications (DIDOCS) (IEECVETA) (Part 1 of 2)

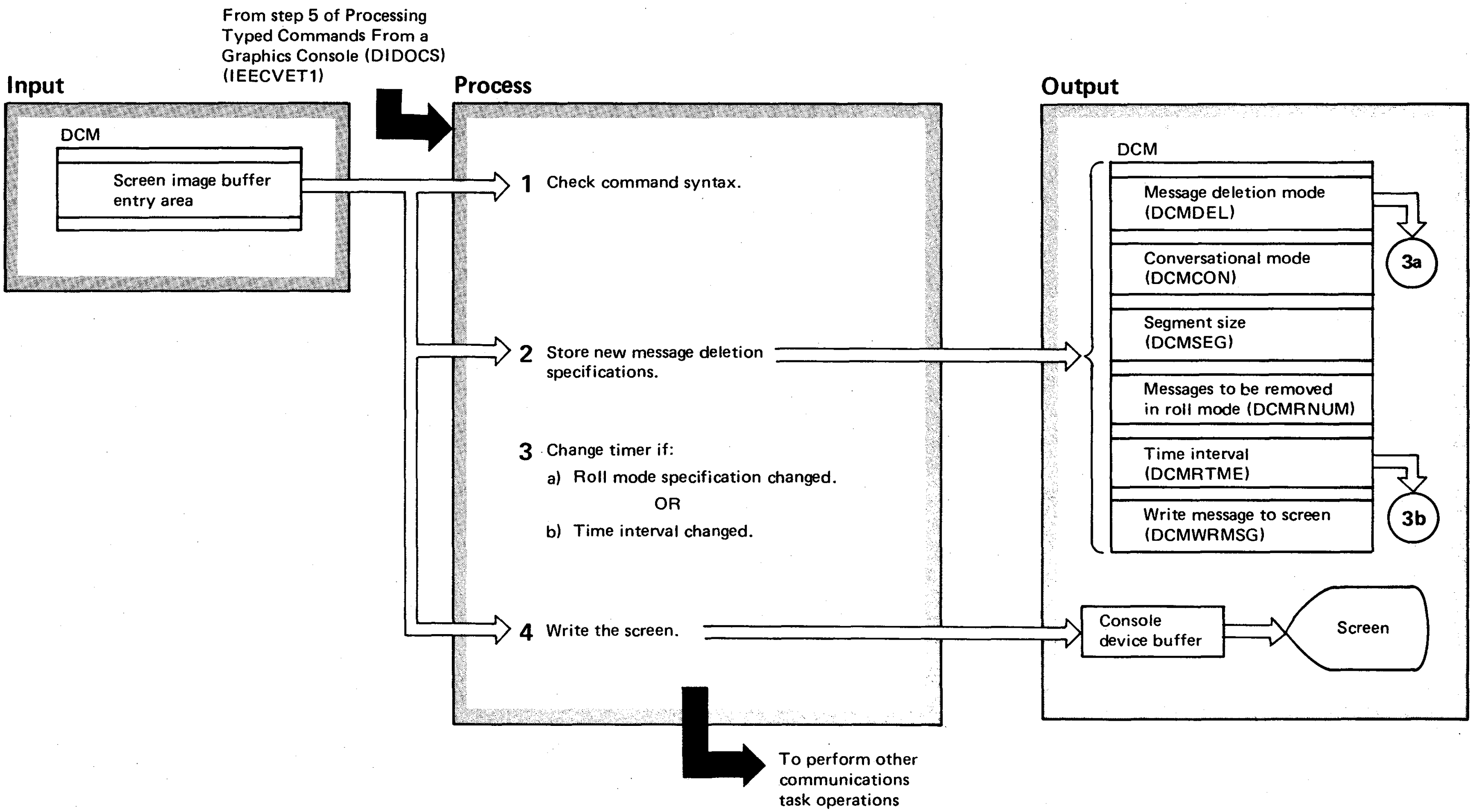


Diagram 1-30. Changing Message Deletion Specifications (DIDOCS) (IEECVETA) (Part 2 of 2)

Extended Description	Module	Label
The operator establishes message deletion specifications using the K S command. DIDOCS processes the message deletion specifications as follows:		
1 DIDOCS checks the syntax of the command, which is in the SIB entry area. If a syntax error exists, DIDOCS issues a message to the operator.	IEECVETA	
Note: If the operator entered the K S,REF command, DIDOCS moves the current definitions from DCMDEL, DCMCON, DCMSEG, DCMRNUM, and DCMRTME into the entry area, then writes the entry area. The operator then changes the specifications as required and enters the command.	IEECVETA	

Extended Description	Module	Label
2 If the syntax is valid, DIDOCS stores the message deletion specifications in the following DCM fields – • DCMDEL indicates whether deletion mode is roll mode or automatic deletion mode. • DCMCON indicates whether conversational mode is in effect. • DCMSEG indicates the number of messages in a segment. • DCMRNUM indicates the number of message lines to be removed if roll mode is in effect. • DCMRTME indicates the time interval for messages to be removed from the screen if roll mode is in effect. DIDOCS sets bit DCMWRMSG to indicate that the screen must be rewritten.	IEECVETA	
3 If roll mode specifications have changed or if the time interval for roll mode has changed, DIDOCS must reset the timer (using an STIMER macro instruction).	IEECVETK	
4 DIDOCS rewrites the screen in order to blank the warning line (remove the MODE=R message) and to blank the entry area.	IEECVETH/P/R/U	

Diagram 1-31. Erasing or Displaying the PFK Display Line (DIDOCS) (IEECVETB) (Part 1 of 2)

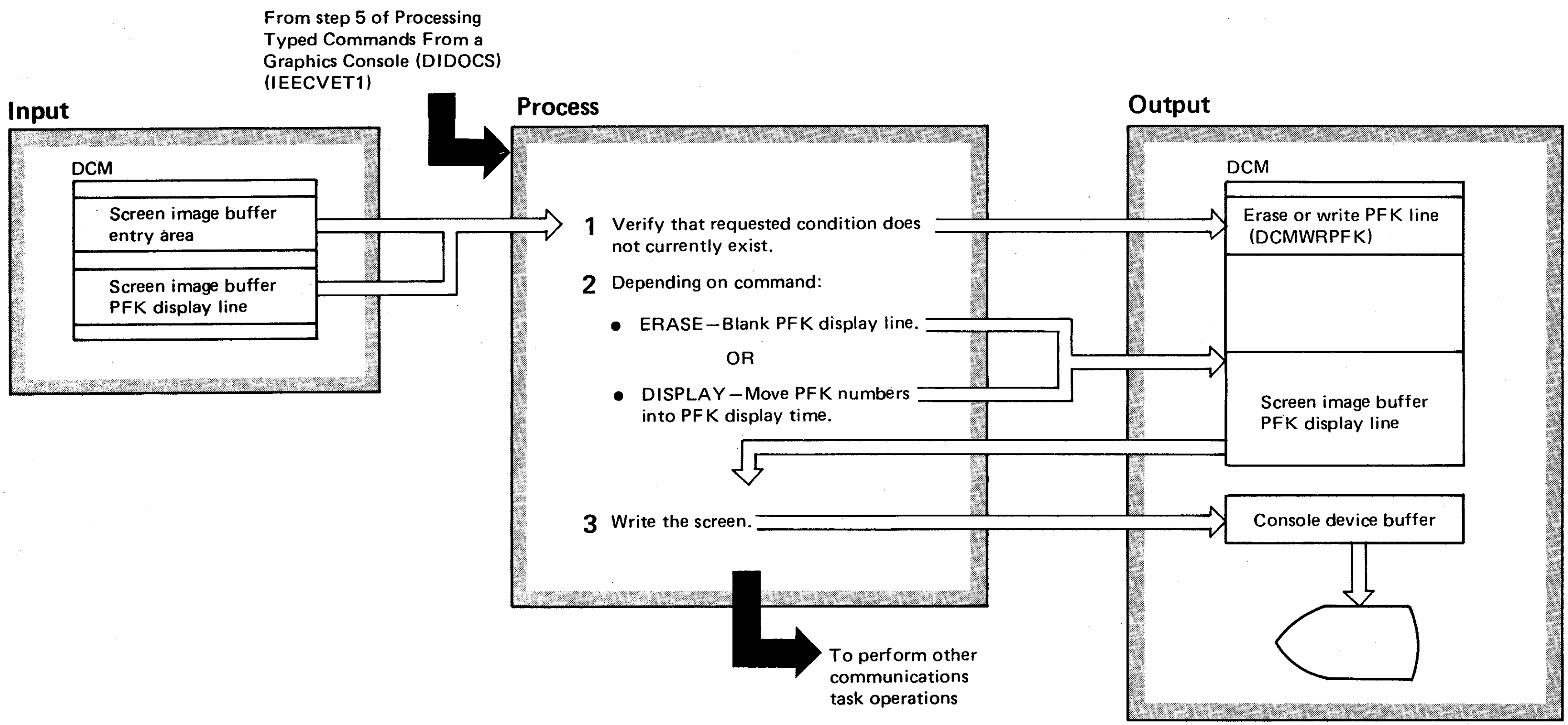


Diagram 1-31. Erasing or Displaying the PFK Display Line (DIDOCS) (IEECVETB) (Part 2 of 2)

Extended Description	Module	Label
The PFK displav line contains a display of PFK numbers that the operator uses when he/she enters commands with a light pen. The operator requests the erasing of the PFK display line by issuing the K E,PFK command. The operator requests the displaying of the PFK display line by issuing the K D,PFK command. DIDOCS processes these commands as follows:		
1 Before processing the command, DIDOCS determines whether the requested condition already exists; for example, the operator entered K D,PFK and the PFK line is already displayed. If the condition already exists, DIDOCS issues a message to the operator. If the condition does not already exist, DIDOCS sets bit DCMWRPFK to reflect the requested erase or display.	IEECVFTB	
2 • K E,PFK – If an erase was requested, DIDOCS places blanks in the SIB PFK display line.	IEECVETH/P/R/U	
• K D,PFK – If a display was requested, DIDOCS writes the PFK numbers in the SIB PFK display line.	IEECVETH/P/R/U	
3 Finally, DIDOCS writes the updated SIB to the screen.	IEECVETH/P/R/U	

Diagram 1-32. Erasing/Holding/Framing/Updating Status Displays (DIDOCS) (IEECVFTP) (Part 1 of 2)

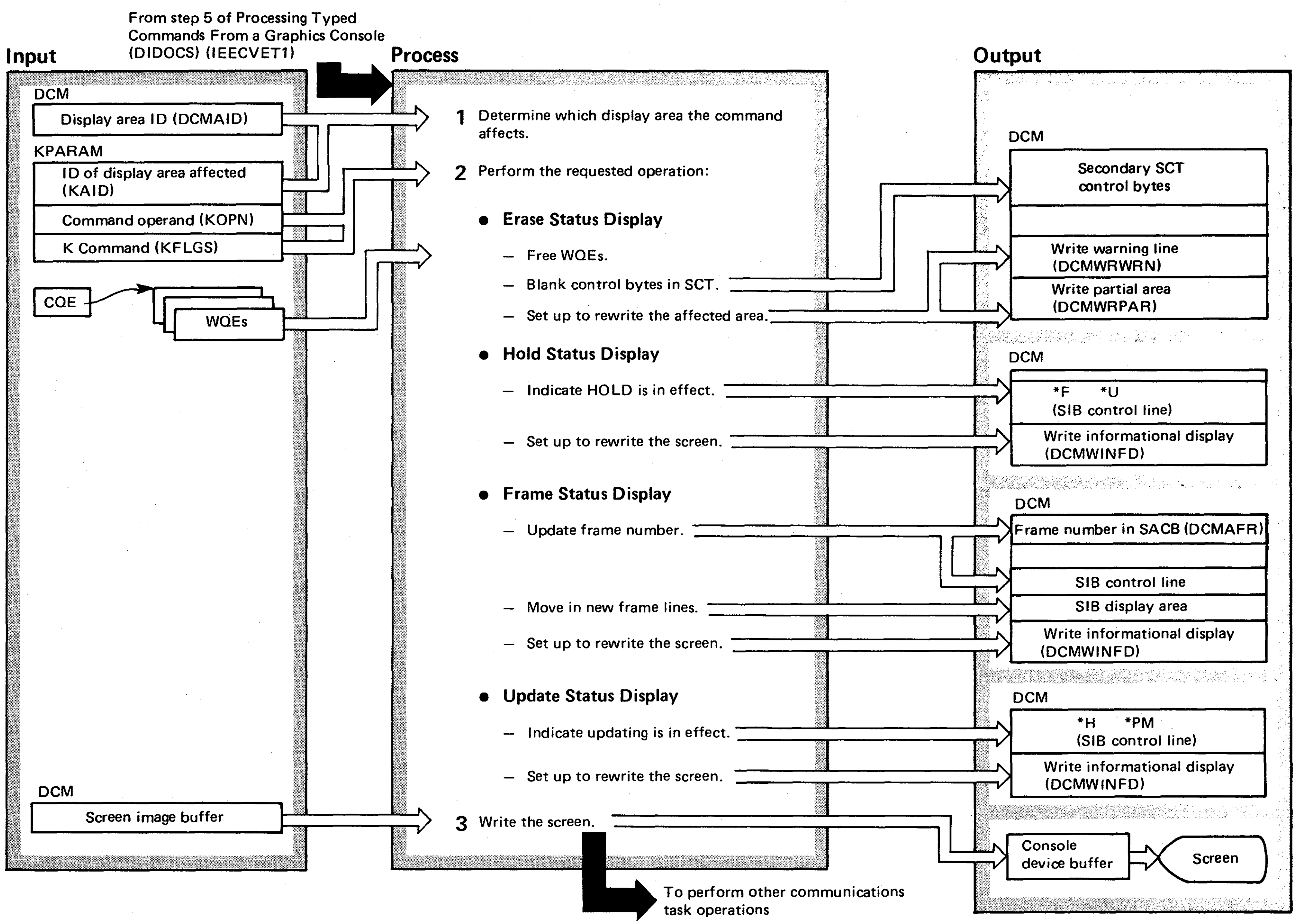


Diagram 1-32. Erasing/Holding/Framing/Updating Status Displays (DIDOCS) (IEECVFTP) (Part 2 of 2)

Extended Description	Module	Label
A status display is a formatted, multiple-line display of information about some part of the system; the status display appears at the operator's console in response to a DISPLAY or TRACK command. The operator controls status displays with the following CONTROL command options:		
• K E,D – specifies erasing of a status display.		
• K D,H – specifies holding of the dynamically updated status display.		
• K D,F – specifies moving the status display forward to the next frame; this operation is hereafter called "framing."		
• K D,U – specifies the resumption of updating of a formerly held dynamic status display.		
DIDOCS processes these commands as follows:		
1 A display area is a block of screen lines that contains a status display. Before DIDOCS performs the requested operation on a status display, it determines which display area the command affects. DIDOCS compares the area ID (KAID) in the command parameter list (KPARAM) with the area ID (DCMAID) in the DCM.	IEECVFTP	

Extended Description	Module	Label
2 From KPARAM fields KOPN and KFLGS, DIDOCS determines which status display operation was requested, then performs the operation:	IEECVFTP	
• Erasing the status display – DIDOCS frees the major and minor WQEs and blanks the control bytes in the secondary screen control table (SCT). Finally, DIDOCS sets the write-warning-line bit (DCMWRWRN) and the write-partial-area bit (DCMWRPAR to indicate that the affected portion of the entry area must be written to the screen.	IEECVFTP	
• Holding the status display – DIDOCS indicates that the status display is being held by setting *F and *U in the SIB control line. Finally, DIDOCS indicates that the screen must be written by setting bit DCMWINFD.	IEECVFTN	
• Framing the status display – DIDOCS adds one to the frame number in the SACB and moves a new control line that contains the new frame number into the SIB. Finally, DIDOCS places the new frame lines into the SIB and sets bit DCMWINFD to indicate that the screen must be rewritten.	IEECVFTN	
• Updating the status display – DIDOCS indicates that update mode is in effect by placing *H and *PM into the SIB control line. Then DIDOCS sets bit DCMWINFD to indicate that the screen must be rewritten.	IEECVFTN	
3 DIDOCS writes the screen image from the screen image buffer into the console device buffer.	IEECVETH/P/R/U	

Diagram 1-33. Roll Mode Message Deletion Processing (DIDOCS) (Part 1 of 2)

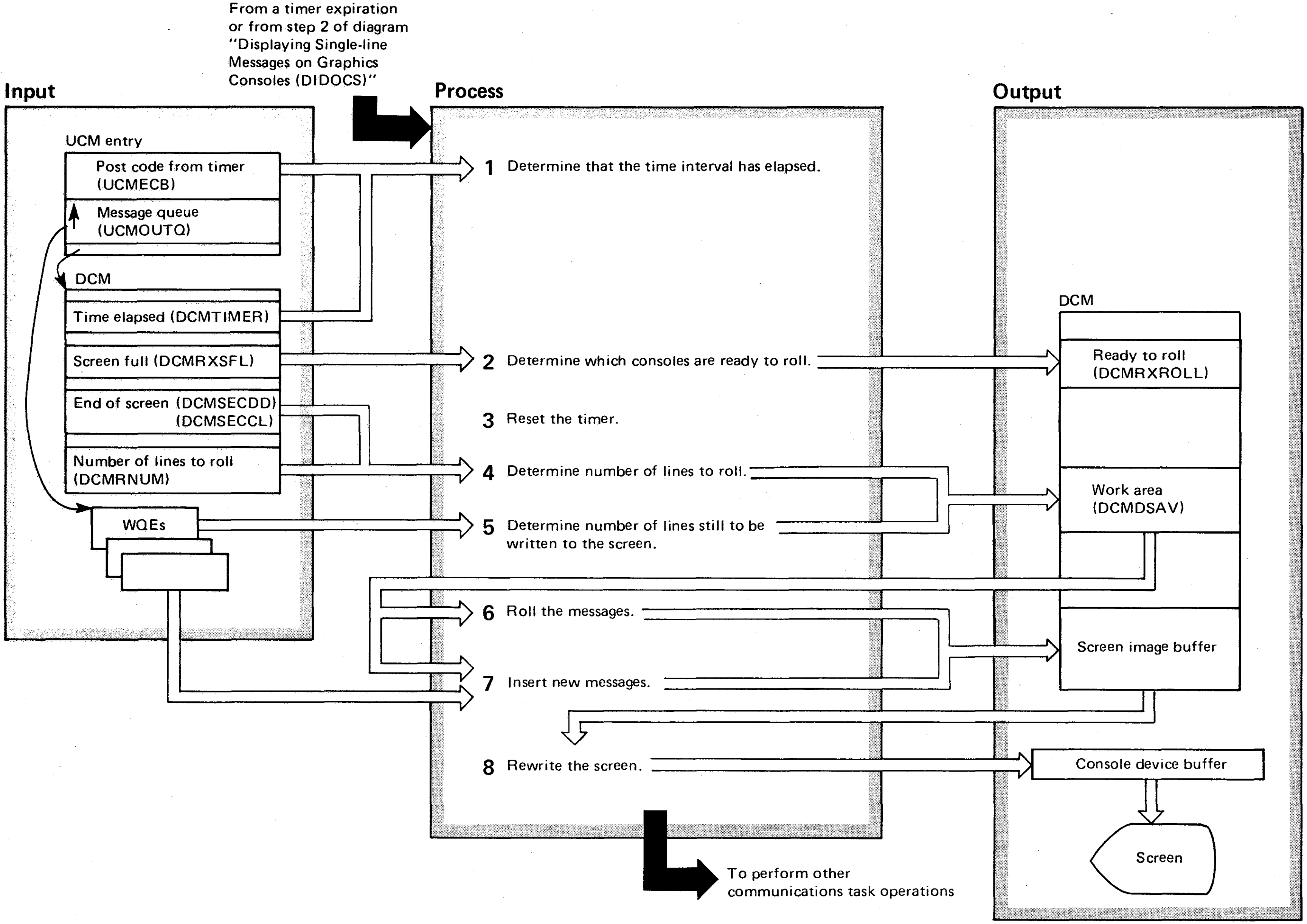


Diagram 1-33. Roll Mode Message Deletion Processing (DIDOCS) (Part 2 of 2)

Extended Description	Module	Label
When roll mode is in effect and a specified time interval elapses, DIDOCS deletes an operator-specified number of messages if the screen is full.		
1 DIDOCS determines that the time interval has elapsed by checking that the elapsed-timer bit (DCMTIMER) is on and that the post code in UCMECB is the timer post code.	IEECVET1	
2 DIDOCS examines each active console's screen-full bit (DCMRXSFL) to determine whether the console screen is full. If so, DIDOCS sets the ready-to-roll bit (DCMRXRLL) in the console's DCM.	IEECVETK	
3 DIDOCS issues the STIMER macro instruction to reset the system interval timer.	IEECVETK	
4 DIDOCS subtracts the number of lines used for status displays (DCMSECDD) from the specified roll number (DCMRNUM) to determine the number of lines to be rolled.	IEECVETJ	

Extended Description	Module	Label
5 DIDOCS searches the output queue for message lines that are waiting to be output. DIDOCS counts the number of waiting lines.	IEECVETJ	
6 DIDOCS compares the number of lines to be rolled (from step 4) with the number of waiting message lines (step 5). If the number of waiting lines is greater than the number of lines to be rolled, DIDOCS rolls the number of lines to be rolled and displays the number of message lines still waiting. If the number of waiting lines is less than the number to be rolled, DIDOCS rolls just enough lines from the screen to display the waiting message lines.	IEECVETJ	
7 To replace deleted messages, DIDOCS moves lines from the bottom line of the message area or from the message queue.	IEECVETJ	
8 DIDOCS rewrites the screen.	IEECVETH/P/R/U	

Diagram 1-34. Communication Task Functional Recovery Routine or ESTAE Controller Overview (IEAVMFRR) (Part 1 of 2)

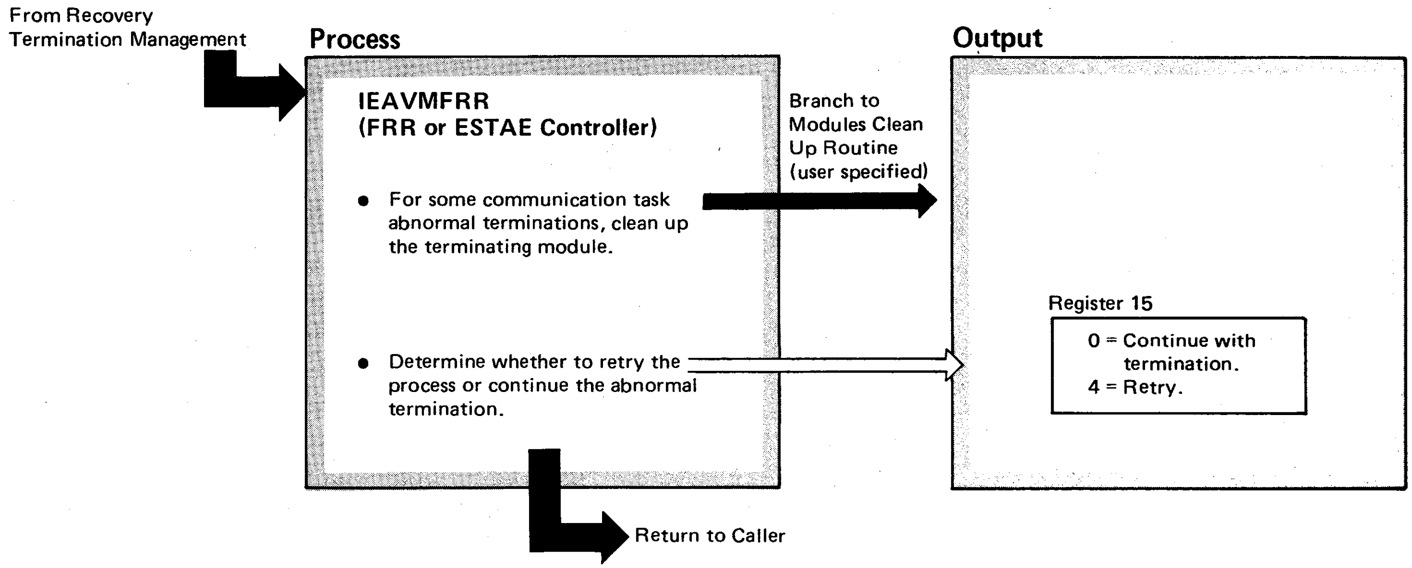


Diagram 1-34. Communication Task Functional Recovery Routine or ESTAE Controller Overview (IEAVMFRR) (Part 2 of 2)

Extended Description	Module
This routine is the common error recovery module for restoring the communication task after an abnormal termination. It receives control from the recovery-termination management (RTM) routine for both FRR and ESTAE recoveries. The FRR/ESTAE controller examines the error environment to determine whether RTM can attempt a retry of the communication task *or* continue with the termination process. When a clean up procedure has been supplied for the failing module, the FRR/ESTAE controller includes a branch to that procedure.	IEAVMFRR

Note:

FRR Entries: The CMS and local locks are still in effect.

ESTAE Entries: The CMS and local locks were freed by RTM before control was given to the FRR/ESTAE controller.

Diagram 1-35. Communication Task Functional Recovery Routine or ESTAE Controller (IEAVMFRR) (Part 1 of 10)

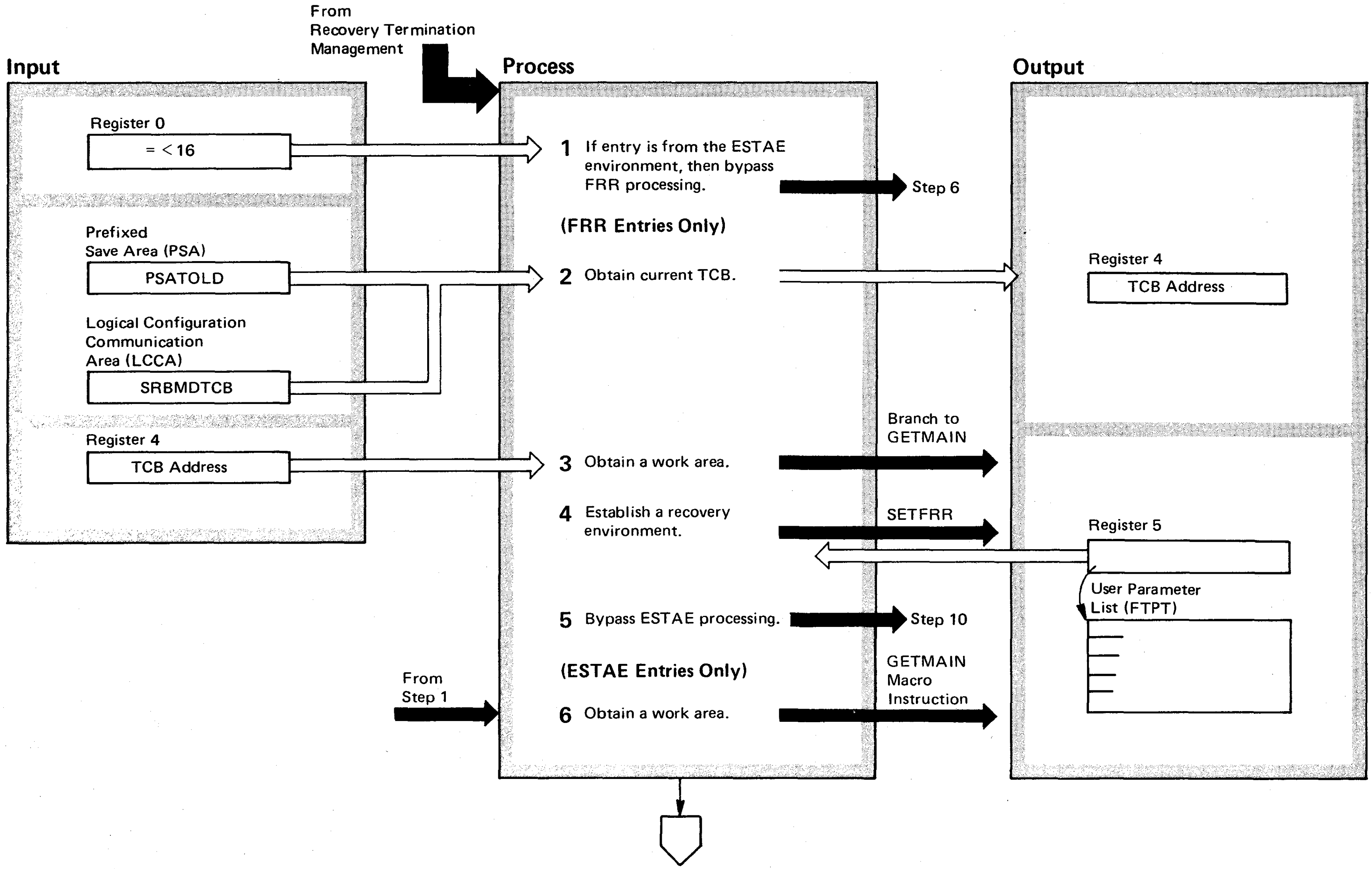


Diagram 1-35. Communication Task Functional Recovery Routine or ESTAE Controller (IEAVMFRR) (Part 2 of 10)

Extended Description	Module
1 Upon entry into the FRR/ESTAE controller, the controller tests register 0 to determine whether the entry was caused from an FRR or ESTAE. If register 0 is greater than 16, then the entry was caused by an FRR; if equal to or less than 16, then the entry was caused by an ESTAE.	IEAVMFRR
2 When the entry into the FRR/ESTAE controller was from an FRR, the address of the current task control block (TCB) is obtained and placed in register 4: the branch to the GETMAIN executed next uses this TCB address. To obtain the current TCB address, the PSATOLD field in the prefix save area is tested. When this field is zero, the current TCB address is obtained from the SRBMDTCB field of the logical configuration communication area (LCCA). When the PSATOLD is not zero, the PSATOLD field is the address of the current TCB.	
3 Using the previously obtained current TCB address this routine does a branch to GETMAIN. The GETMAIN routine obtains a work area. The branch entry into GETMAIN is necessary because the local and CMS locks were being held when the communication task encountered the error condition that caused the abnormal termination. RTM does not release the local and CMS locks for an FRR retry before giving control to the FRR/ESTAE controller.	

Extended Description	Module
4 The SETFRR macro instruction is issued to establish a recovery environment for the FRR/ESTAE controller. This step is necessary because the local and CMS locks are still in effect. If this routine should happen to abnormally terminate while processing the current error recovery, a record is made of the environment and RTM continues with the termination.	
5 FRR processing has been initialized bypass the initialization of ESTAE processing.	
6 A GETMAIN macro instruction is issued to obtain a work area. The GETMAIN macro instruction may be issued for ESTAE processing because RTM freed the local and CMS locks that may have been held before control was given to the FRR/ESTAE controller.	

Diagram 1-35. Communication Task Functional Recovery Routine or ESTAE Controller (IEAVMFRR) (Part 3 of 10)

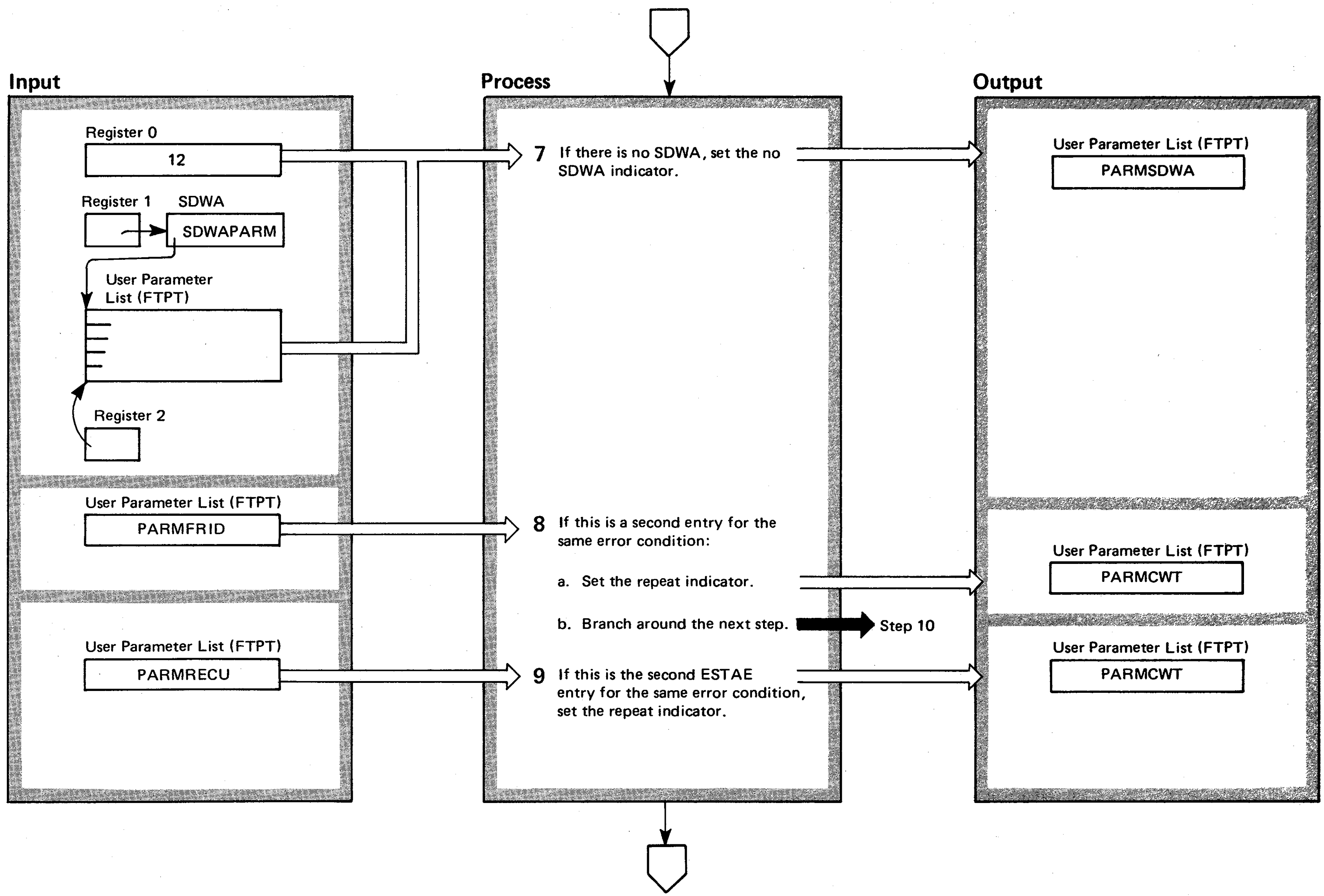


Diagram 1-35. Communication Task Functional Recovery Routine or ESTAE Controller (IEAVMFRR) (Part 4 of 10)

Extended Description | **Module**

7 If register 0 is not equal to 12, a STAE diagnostic work area (SDWA) was not passed to the FRR/ESTAE control by RTM. The PARMSDWA bit in the user parameter list (FTPT) is set accordingly.

8-9 Steps 8 and 9 determine if this entry is for repeat processing of a previous error condition. It is possible to process an ESTAE entry without having previously processed an FRR entry. The three levels of entry are:

8a An FRR entry. Although the FRR entry is not processed by these steps, previous FRR entry processing could have been done for this same error condition. The return code from this processing could have told RTM either to retry the process that failed *or* to continue termination processing.

8b First ESTAE entry. Step 8 determines if this current error condition was previously processed by an FRR entry that resulted in an attempted retry. If this is the second entry into the controller for the same error condition, the PARMCWT bit in the user parameter list (FTPT) is set to indicate that this is repeat processing. The return code could tell RTM either to retry the failing process or continue with termination processing.

9 Second ESTAE entry. Step 9 determines if this current error is the result of a retry from previously processing an ESTAE entry. Eventually, the controller tells the RTM to continue termination processing; no more retries. If this is the second ESTAE entry into the controller for the same error condition, the PARMCWT bit in the user parameter list (FTPT) is set to indicate repeat processing.

Diagram 1-35. Communication Task Functional Recovery Routine or ESTAE Controller (IEAVMFRR) (Part 5 of 10)

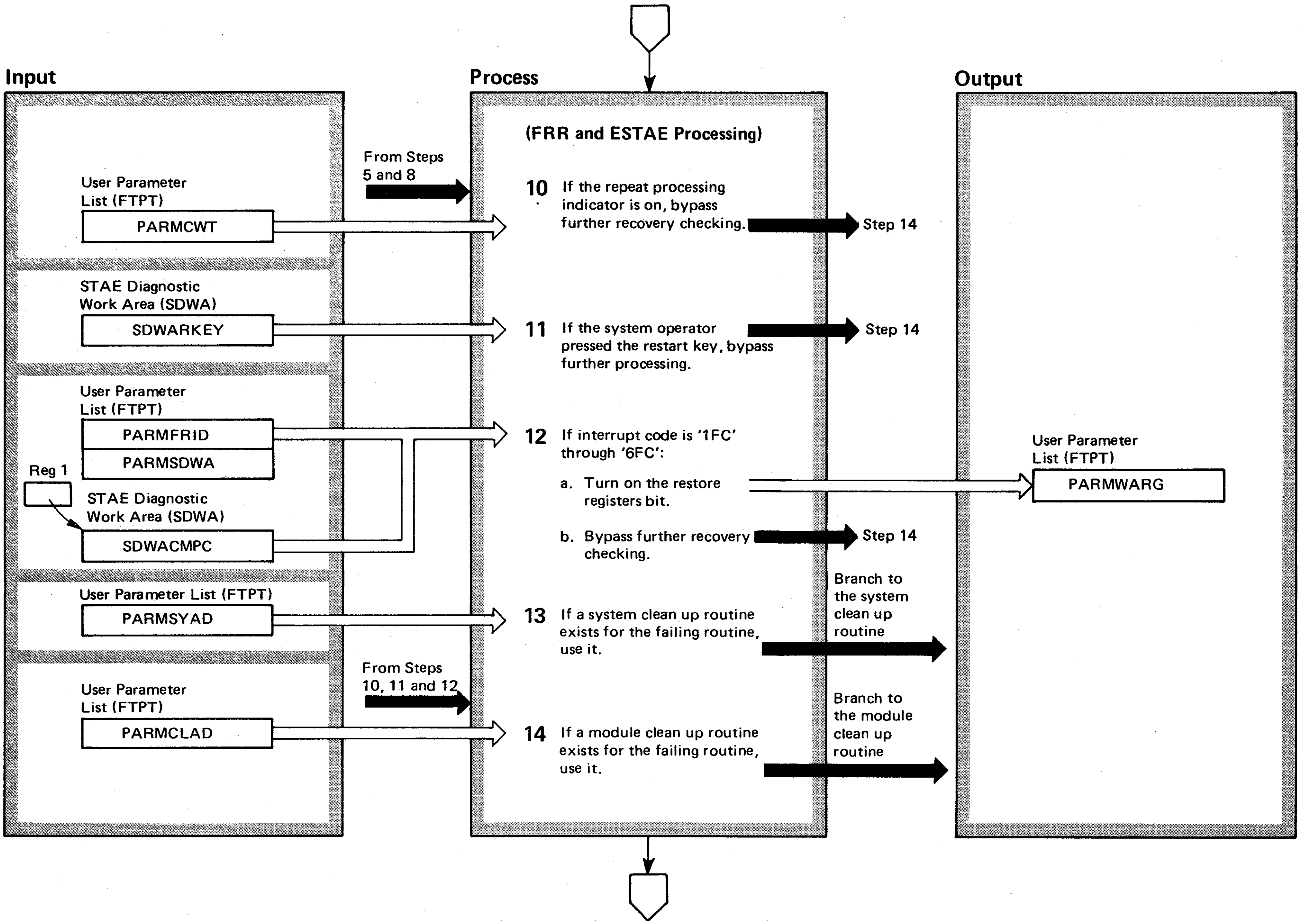


Diagram 1-35. Communication Task Functional Recovery Routine or ESTAE Controller (IEAVMFRR) (Part 6 of 10)

Extended Description **Module**

10 If the current error condition has previously been processed by the FRR/ESTAE controller, bypass further error checking.

11 The console operator may have attempted to restart a previously failing system by pressing the console restart button. The operator restart condition is indicated in the SDWARKEY field of the STAE diagnostic work area (SDWA). If the operator has attempted to restart the system and the FRR/ESTAE controller is processing an abnormal termination as a result of that operator restart, then bypass further error checking. This step is bypassed when the SDWA does not exist as determined by the PARMSDWA bit in the user parameter list (FTPT).

12 The registers will be restored from the STAE diagnostic work area (SDWA) and further recovery processing is bypassed if all of the following conditions are met:

- The SDWA exists.
- The abnormal termination code represented by SDWACMPC is any code between '1FC' and '6FC'.
- The controller was entered for ESTAE processing.

Extended Description **Module**

13-14 To allow an operation to be retried, data areas related to the failing operation may need to be restored — cleaned up. Some time prior to the failure, the failing module issued a macro instruction to set either the functional recovery routine (FRR) or the ESTAE environment. Both macro instructions permit system and module clean up routines to be designated. When these routines are designated, the FRR/ESTAE controller branches to them. When each routine finishes its process, it returns to the FRR/ESTAE controller.

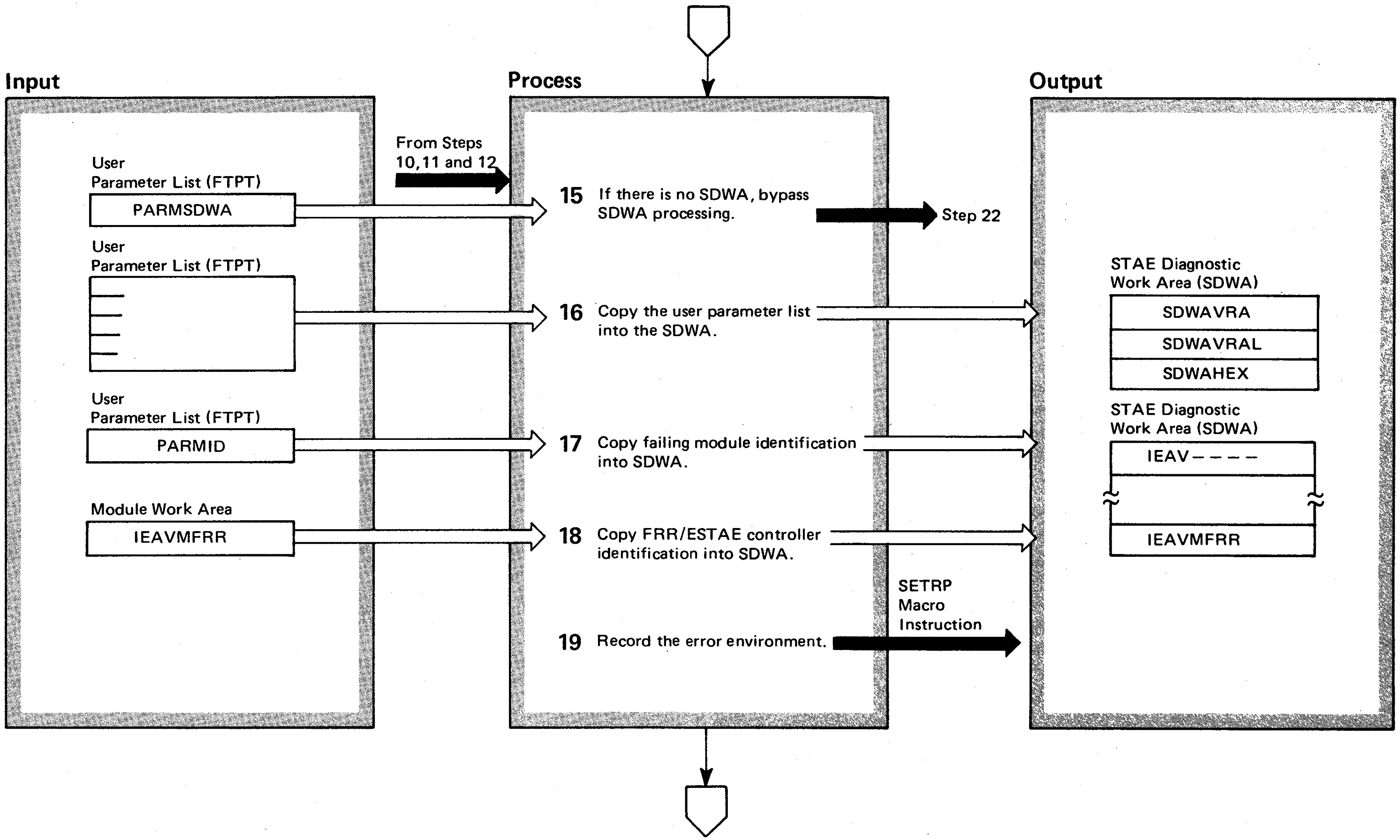
System Clean Up: Mainline routines often use supportive system functions, such as SVCs and macro instructions that execute outside the mainline routine. When using these system functions, certain data areas or registers may have been set by the mainline function or the system function. The system clean up routine restores those areas of the mainline routine related to system functions.

Module Clean Up: The module clean up routine restores data areas and registers, not related to system functions, that will permit the failing operation to be retried.

13 If the controller has the address of a system clean up module in the PARMSYAD field of the user parameter list (FTPT), the controller branches to that clean up routine. Following clean up, control is returned to the controller.

14 If the controller is given the address of the module clean up module in the PARMCLAD field of the user parameter list (FTPT), the controller branches to that clean up routine. Following the clean up, control is returned to the controller.

Diagram 1-35. Communication Task Functional Recovery Routine or ESTAE Controller (IEAVMFRR) (Part 7 of 10)



Extended Description	Module
15 If the STAE diagnostic work area (SDWA) is not available, bypass further processing that involves the SDWA.	
16 The user parameter list (FTPT) is copied to the variable area of the STAE diagnostic work area (SDWA) for future recording on SYS1.LOGREC.	
17 The identification of the failing module is copied into the STAE diagnostic work area (SDWA).	
18 The identification of the FRR/ESTAE controller is copied into the STAE diagnostic work area (SDWA).	
19 To record the error environment on SYS1.LOGREC, the SETRP macro instruction is issued. Shortly after control has been returned to recovery termination management, the environment will be asynchronously recorded.	

Diagram 1-35. Communication Task Functional Recovery Routine or ESTAE Controller (IEAVMFRR) (Part 9 of 10)

Input

User Parameter List (FTPT)

- PARMWARG
- PARMCWT
- PARMRTAD

User Parameter List (FTPT)

- PARMWARG

STAE Diagnostic Work Area

- SDWANXTL

Register 4

- TCB Address

From Steps 15 and 20

Process

20 If the supplied retry address can be used:

a. Specify retry.

b. Bypass other retry steps.

21 If an interrupt abnormal termination code was found, specify a retry.

22 Free the work area:

a. If this is an FRR entry, branch to FREEMAIN.

b. If this is an ESTAE entry, issue the FREEMAIN macro instruction.

23 Return to RTM.

SETRP Macro Instruction

Step 22

SETRP Macro Instruction

Branch to FREEMAIN (IEAVGM00)

FREEMAIN Macro Instruction

Output

Register 15

0	=	Continue with termination.
4	=	Retry.

Diagram 1-35. Communication Task Functional Recovery Routine or ESTAE Controller (IEAVMFRR) (Part 10 of 10)

Extended Description **Module**

20 The SETRP macro instruction is issued to specify a retry operation if all of the following conditions exist:

- The restore registers bit (PARMWARG) is not set.
- The user parameter list (FTPT) has a retry address (PARMRTAD) for retrying the failing routine.
- The continue-with-termination bit (PARMCWT) in the user parameter list (FTPT) is not set.

21 If the PARMWARG bit in the user parameter list (FTPT) is set, the SETRP macro instruction is issued to specify a retry operation of the failing routine. The retry address is taken from the SDWANXTL field of the STAE diagnostic work area.

22 The work area obtained for the FRR/ESTAE controller is freed.

23 The controller returns control to RTM with a return code in register 15:

0 RTM is instructed to continue termination processing.
4 RTM is instructed to attempt a retry of the failing module.

Diagram 1-36. Communication Task Recovery (STAR) Routine (IEAVSTAA) (Part 1 of 12)

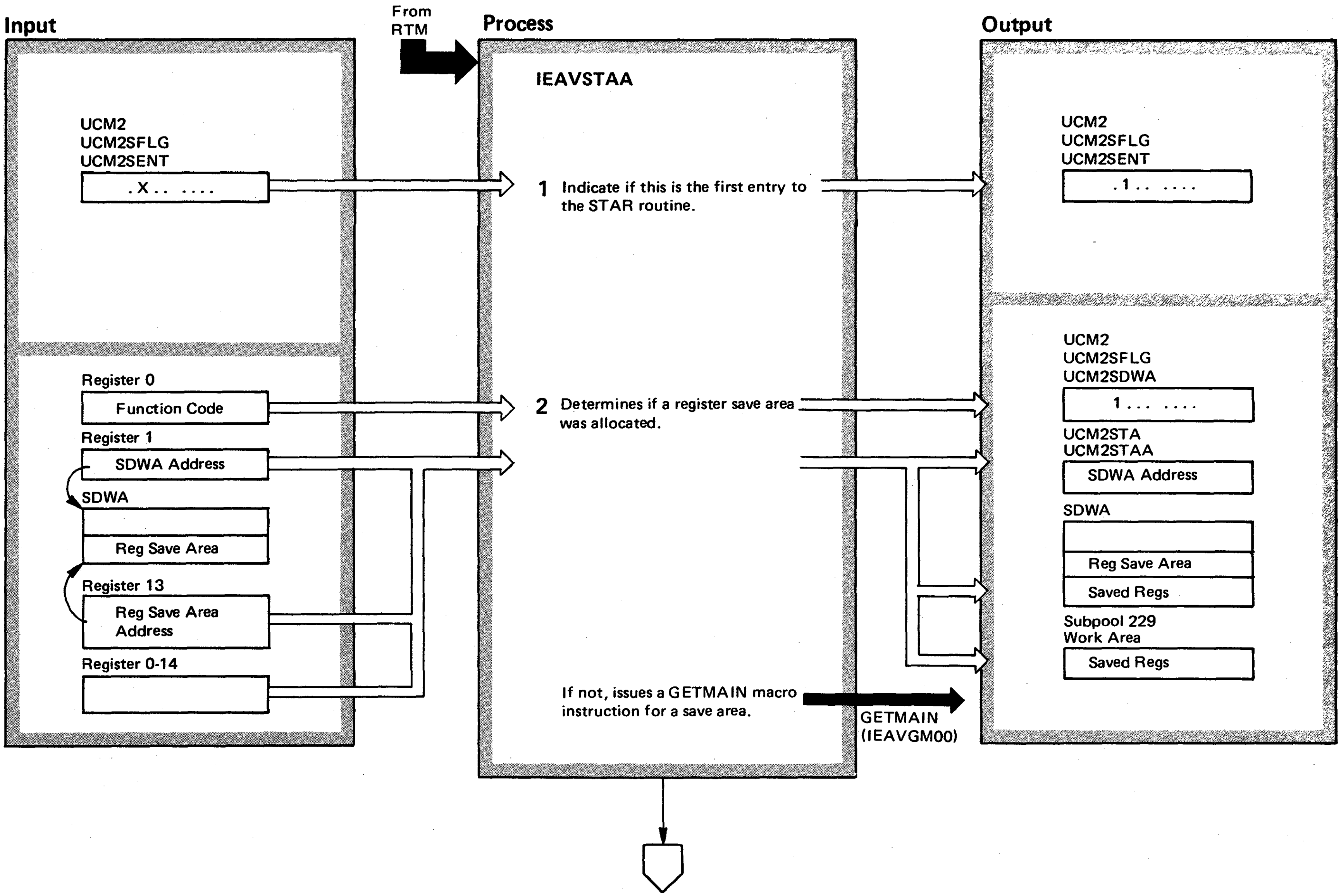
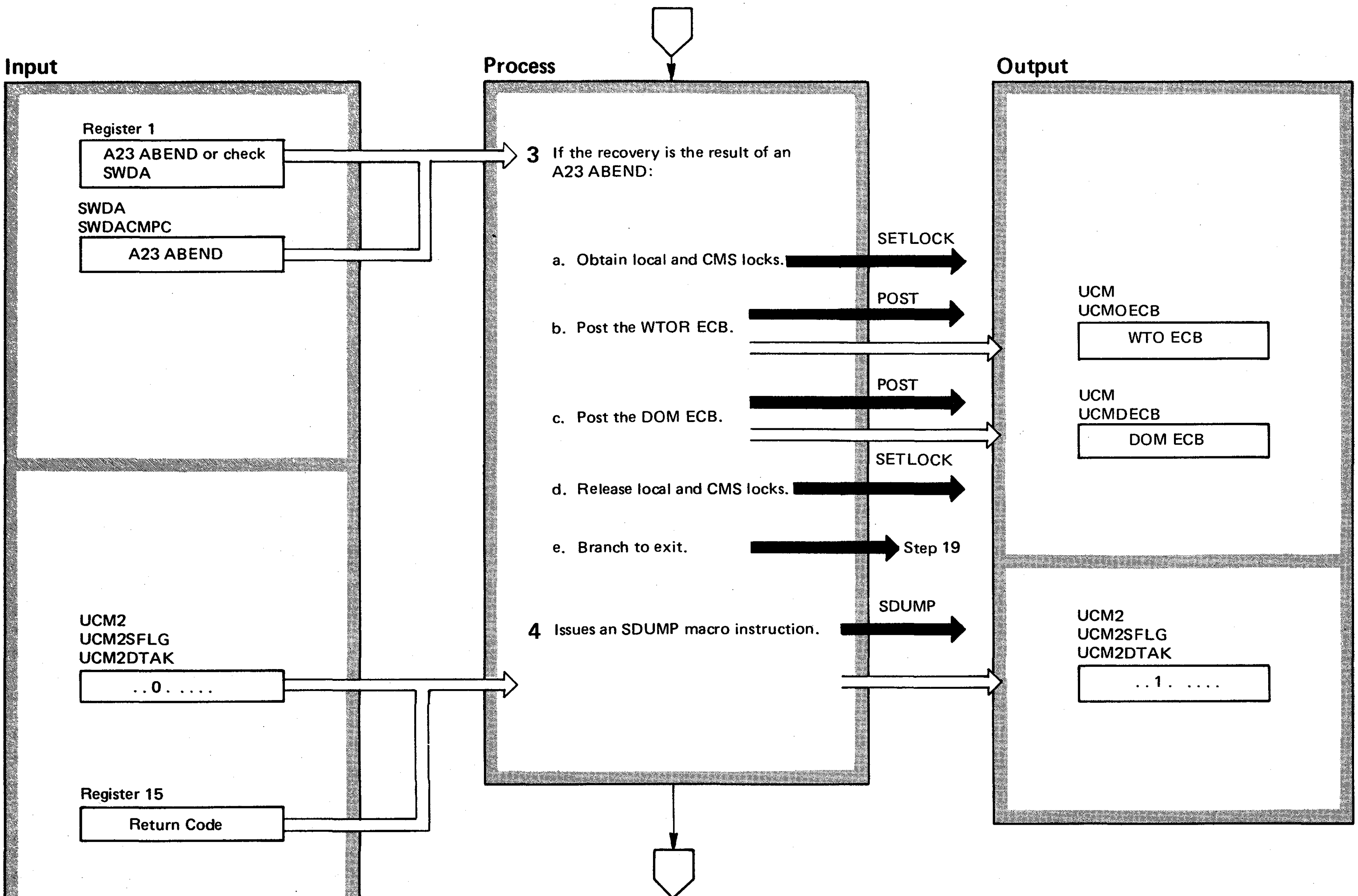


Diagram 1-36. Communication Task Recovery (STAR) Routine (IEAVSTAA) (Part 2 of 12)

Extended Description	Module
This routine is part of load module IEAVCTSK. It is entered enabled by recovery-termination management (RTM) for all abnormal terminations caused by the Communications Task.	
1 The STAR bit (UCM2SENT) is used during dump reading to determine whether the STAR routine has been entered. If this is the first entry into the STAR routine, the STAR bit is set.	IEAVSTAA
2 Determines if a work area was provided by RTM (Register 0 is not 12 and register 1 contains the address of the work area). If no work area is provided, a GETMAIN macro instruction is issued to obtain 64 bytes from Subpool 229; then the registers are saved. Otherwise, if the work area was provided, the address of the STAE diagnostic work area (SDWA) is saved (UCM2STAA in the UCM2) and bit UCM2SDWA is set. Then the registers are saved.	

Diagram 1-36. Communication Task Recovery (STAR) Routine (IEAVSTAA) (Part 3 of 12)



Extended Description | **Module**

3 There are several points in the communication task modules – or modules supplied by the communication task to other system functions – where those modules may attempt a cross-memory post of the communication task. When the attempt fails, an A23 ABEND results. This A23 ABEND may not be an error, only a legitimate attempt to post the communication task. The STAR routine, therefore, posts the communication task for the user who received the A23 ABEND. The user is not terminated. The A23 ABEND will only be issued if a general communications task recovery module (IEAVMEST) was issued as the error exit address for the XMPOST macro.

The A23 ABEND from the communication task means that either the WTO and WTOR event control block (ECB) or the DOM event control block (ECB) were being posted. To satisfy that request, the WTO and WTOR event control block (ECB) and DOM event control block (ECB) are both posted. Posting these control blocks will permit control to be given to the communication task's wait service routine to process whatever work that needs to be performed.

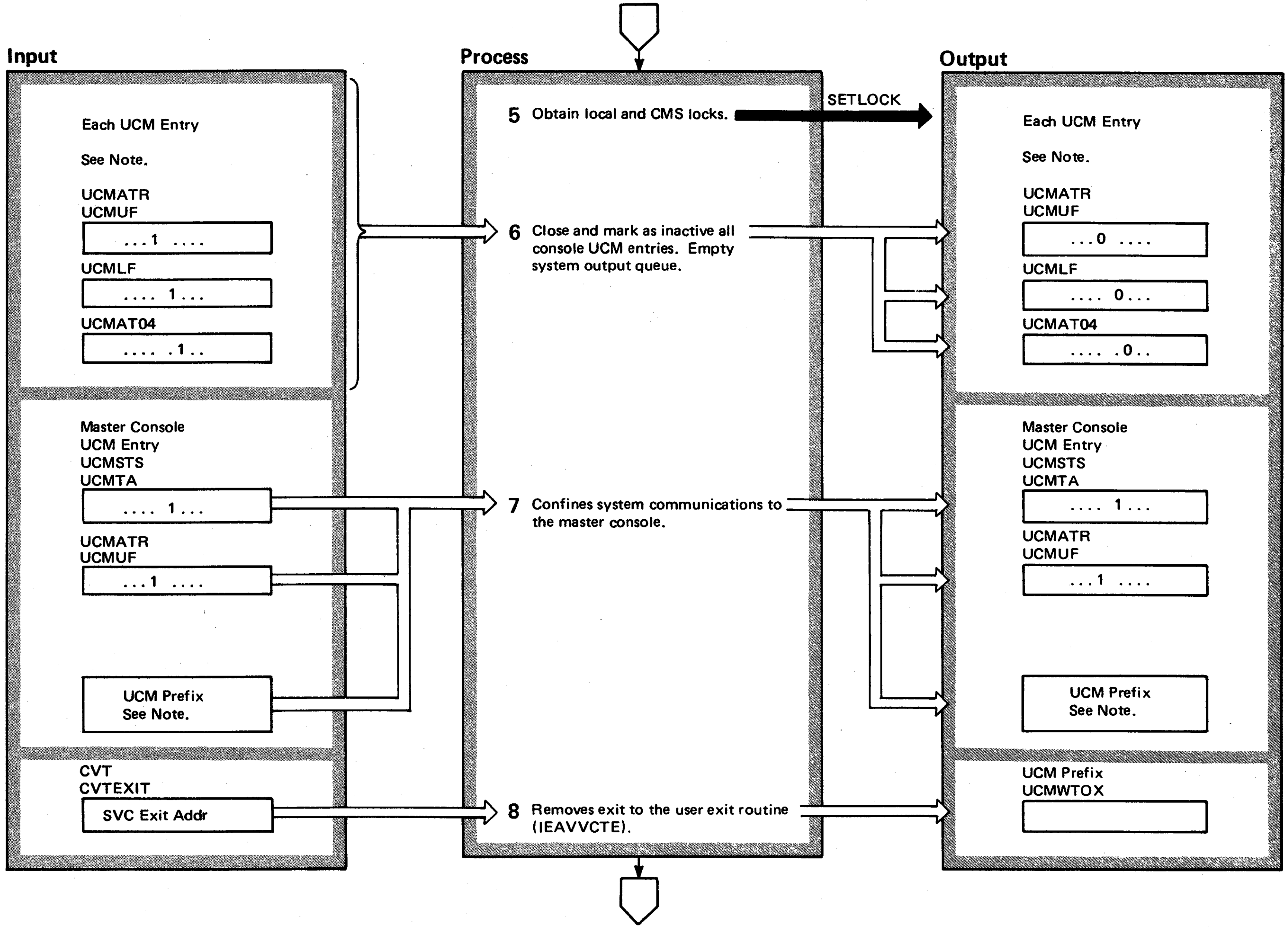
To test for the A23 ABEND: If register 0 is not equal to 12, register 1 contains the address of the STAE diagnostic work area (SDWA); the A23 ABEND code is in the SDWACMPC field of that control block. When register 0 is equal to 12, then the A23 ABEND code is in register 1.

4 The following SDUMP macro instruction is issued to start a dump:

```
SDUMP SDATA=(SQA,NUC,LSQA,LPA,SWA,CSA),
        MF=(E,CTBUF)
```

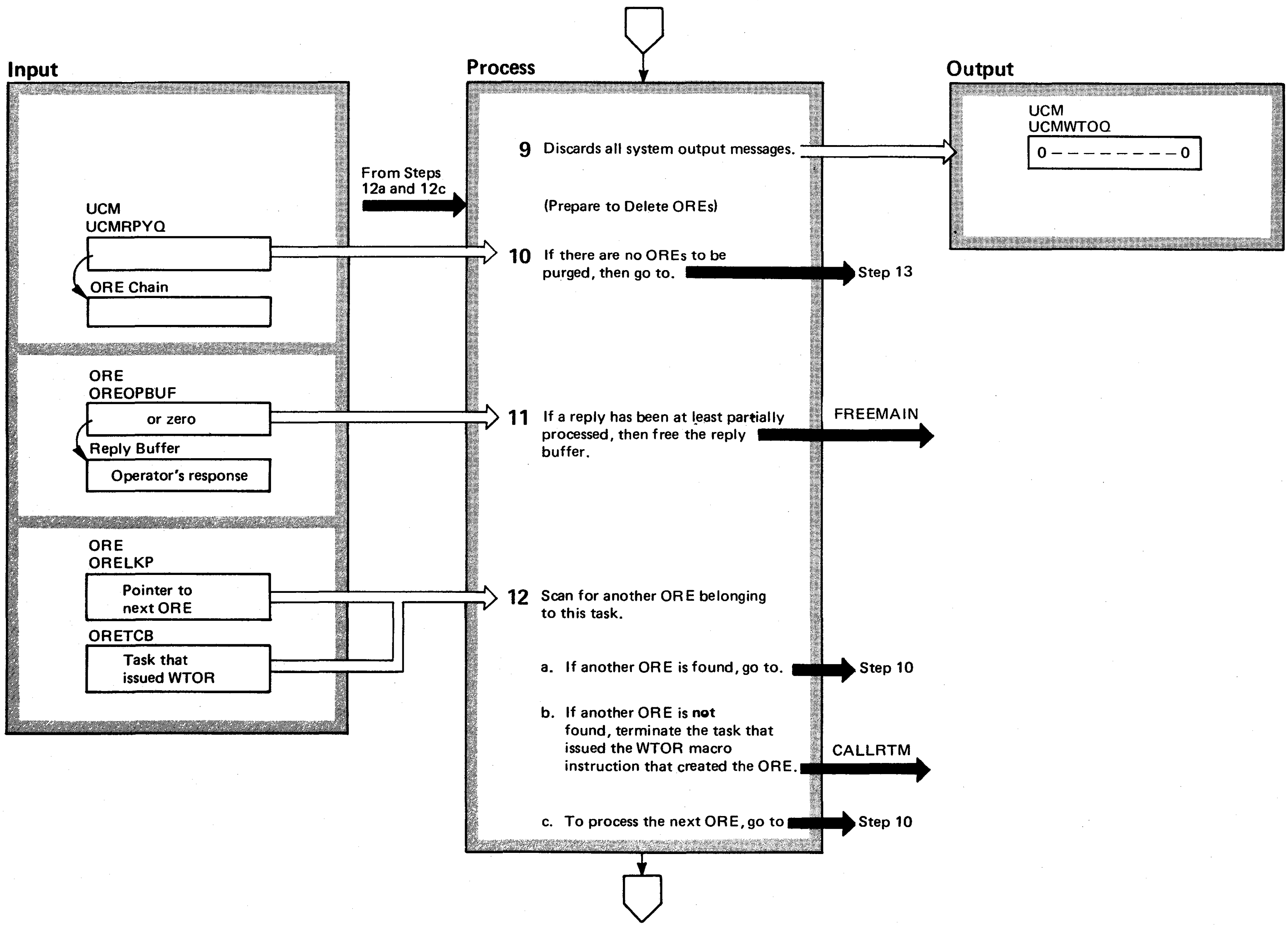
"CTBUF" contains the header label printed on the dump.

Diagram 1-36. Communication Task Recovery (STAR) Routine (IEAVSTAA) (Part 5 of 12)



Extended Description	Module
5 The local and CMS locks are obtained to serialize the use of the communication task's resources. All locks that may have previously been held were freed by recovery-termination management (RTM) before control was given to the STAR routine.	
6 The consoles are closed and made inactive, and the output queues are emptied by setting to zero the following fields in each UCM Entry: UCMDCB, UCMSTS, UCMOUTQ, UCMWLAST, UCMMLAST, UCMMSG, UCMDEVC AND UCMSDS5. Flags UCMUF, UCMLF and UCMAT04 in UCMATR.	
7 The UCM Entry for the master console is located. Flags UCMTA and UCMUF in UCMATR are set; this makes the master console the only console that the operator can use upon a retry attempt. The following fields in the UCM Prefix are set to zero: UCMSYSD, UCMDOME, UCMFLGS2, UCMXCT, UCMSDS1, UCMSYSB, UCMSYSC Field UCMCMID is set to one.	
8 The use of the user-supplied exit routine (IEAVVCTE) is prevented by storing the SVC exit address as the address of the exit routine.	

Diagram 1-36. Communication Task Recovery (STAR) Routine (IEAVSTAA) (Part 7 of 12)



Extended Description **Module**

9 The WQE chain pointer (UCMWTOQ in the UCM) is set to zero causing all system output messages to be discarded. The storage they occupied is not available for reallocation.

Prepare to delete Operator Reply Elements (OREs).

10-12 Steps 10 through 12 free space obtained for WTOR replies and terminate all tasks that are waiting for replies to a WTOR message issued from those tasks. Regardless of the number of replies any one task may be waiting for, that task is terminated with a single B23 ABEND.

10 The ORE chain is checked to determine if any outstanding OREs exist, that is, if all requested replies had been received. If none are found, control goes to step 13.

11 If an operator has made a partial response to a WTOR message, stage I of the reply command processor has obtained a buffer space for that reply and placed a pointer to that reply buffer space in the ORE (OREOPBUF). Until the space has been obtained, the pointer is zero. If the reply buffer space exists for this ORE, the space is freed.

12 For each task that is expecting a reply to a WTOR message, this step gives one, and only one, B23 ABEND regardless of the number of replies each task is expecting. By stepping down the ORE chain one ORE at a time, step 10 has selected an ORE for processing. In this step, the selected ORE's task identification (ORETCB) is compared against the task identifications for the remaining OREs on the ORE chain. When a match *is* found, the task that issued the WTOR macro instruction is *not* terminated. When a match is *not* found, the task *is* terminated.

In both cases, this routine branches back to process the next ORE on the ORE chain.

Diagram 1-36. Communication Task Recovery (STAR) Routine (IEAVSTAA) (Part 9 of 12)

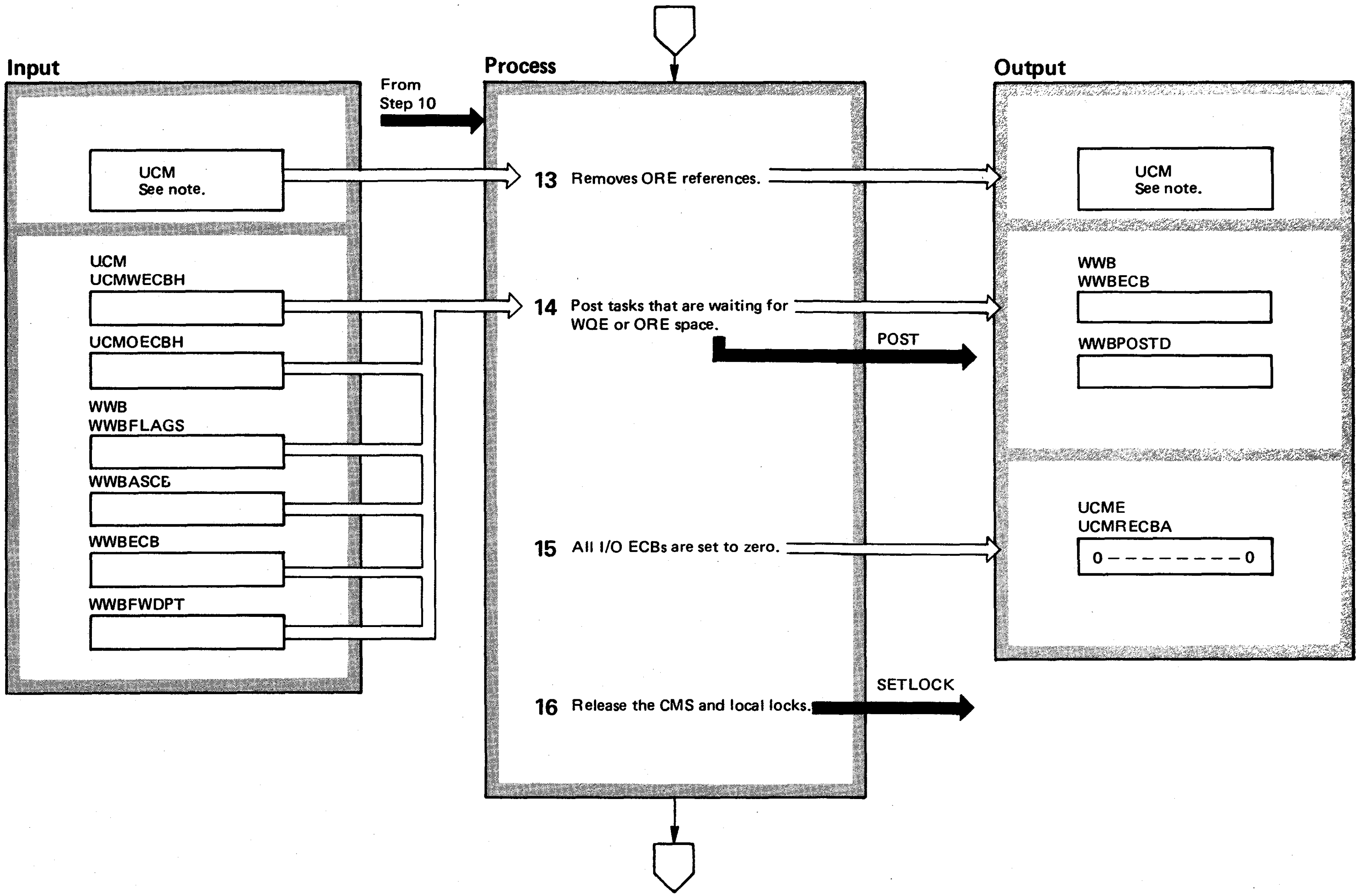


Diagram 1-36. Communication Task Recovery (STAR) Routine (IEAVSTAA) (Part 10 of 12)

Extended Description **Module**

13 The following fields are set to zero:

UCMRPYQ (ORE queue)
UCMRPYI (Reply ID assignment pattern)
UCMRQNR (ORE current counter)
UCMWQNR (WQE current counter)
UCMWQEND (Last WQE pointer)
UCMMODE (All subfields)

14 The write queue element-write wait blocks (WQE-WWBs) and operator queue element-write wait blocks (ORE-WWBs) are scanned for tasks that are waiting for WQE and ORE space. When such a task is found, the WWB post bit (WWBPOSTD) is turned on and the waiting task (WWBECB) is posted.

15 The I/O event control blocks (ECBs) are set to zero.

16 The previously obtained CMS and local locks are released.

Diagram 1-36. Communication Task Recovery (STAR) Routine (IEAVSTAA) (Part 11 of 12)

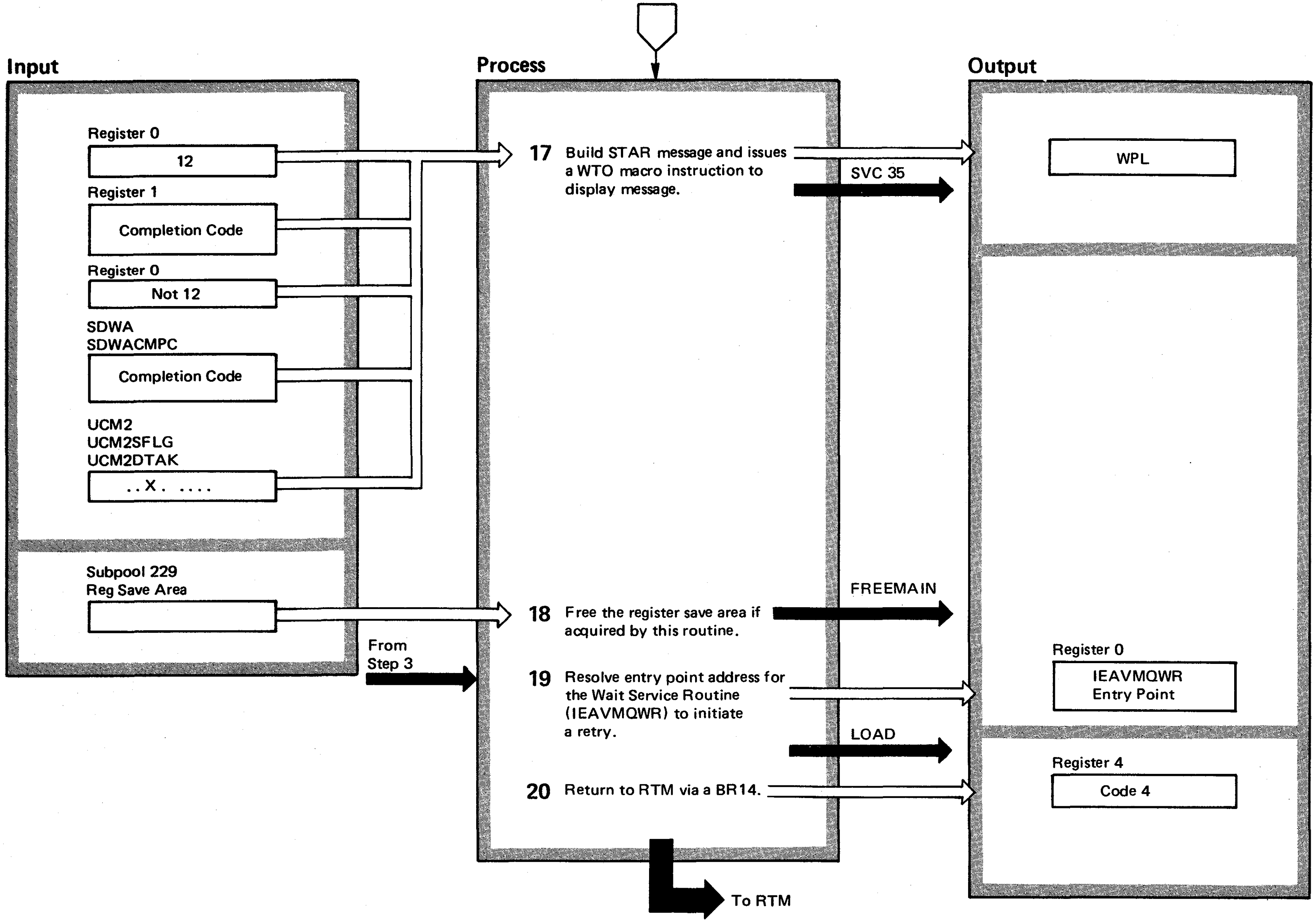


Diagram 1-36. Communication Task Recovery (STAR) Routine (IEAVSTAA) (Part 12 of 12)

Extended Description | **Module**

17 The abnormal termination completion code provided by RTM is moved into the message text. This completion code is found in one of two places: If register 0 equals 12, register 1 contains the completion code. If register 0 does not equal 12, the SDWA (SDWACMPC) contains the completion code.

A check is made (UCM2DTAK) to determine if a successful dump was taken. If not, the message text indicates 'NO DUMP TAKEN'. After the message is built, a WTO macro instruction is issued.

18 Upon return from the WTO, the register save area, if obtained by this routine, is freed via a FREEMAIN macro instruction.

19 A LOAD macro instruction is issued to resolve the entry point address of the wait service routine (IEAVMQWR).

20 A return code of 4 is placed in register 4 and a branch is issued to return to RTM.

Command processing consists of command scheduling and command execution. Command scheduling involves providing for a command task's execution and sychronizing it with other events in the system. Command execution is the performance of the function specified in the command itself. (For a list and explanation of the commands, refer to *Operators' Library* for JES2 or JES3.)

Commands (including subsystem commands) entered into the operating system are initially handled by the SVC 34 common processing routines. These routines: create an ESTAE environment to permit recovery from failures caused either by program checks, machine checks, or by abnormal end situations; and determine if SVC 34 processing is to manipulate control block chains, process a command, or schedule a command for execution by non-SVC 34 processors (such as subsystem processors or other attached processors). The use of the system macro instructions MGCR and QEDIT results in SVC 34 routines receiving control to process CSCBs (command scheduling control blocks) and/or CIBs (command input buffers).

A command may be issued from any of the following sources:

- From a graphics console via DIDOCS routines.
- From a hardcopy (paper) console via communications task routines.
- From a TSO terminal that is in operator mode and that uses SVC 100, or from a TSO terminal that uses the terminal input/output coordinator (TIOC) routines.
- From a "system key" component (that is, commands issued internally such as from checkpoint/restart routines).
- From an input stream reader via the converter subcomponent.

General Considerations

There are two main groups of command processing modules: the common processing modules that perform the same functions for all commands, and the individual command processors (which may consist of more than one module) that handle one or a group of commands.

The common processing modules:

- Establish the ESTAE environment.
- Handle message processing.
- Translate commands.
- Check command authority.
- Route commands to proper processors.
- Interface with subsystems for command identification.
- Create control blocks.
- Manipulate control block queues.
- Perform recovery functions in case of failure.

For many commands, these modules store the command in CSCBs.

CMD PROC

The individual command processors (modules) handle individual commands based on the command verbs with specific keywords. In some cases, the processors perform checking and routing for commands with multiple keywords and operands. In the case of checking and routing, an individual processor may pass control to other modules that perform the actual final processing based on specific operands, or, in some cases, may perform the final processing itself.

Command Execution

Command processing routines perform a specified function either as a new task established by the master scheduler or as a part of an existing system task. Common initialization routines establish the environment necessary for processing either type of (command) task.

The first task of the command scheduling common routines is to establish the address of an error-recovery routine. The routines then translate an input command to upper case characters and, in most cases, write it to a hardcopy device. If the command is a subsystem (for example, JES2) command, the SVC 34 routines return control to the caller (of SVC 34) and the subsystem performs the command processing. Otherwise, valid (proper verb code, syntax, authority, etc.) commands are routed to an appropriate processing module. If an error occurs during the pre-routing processing, an error message is written, SVC 34 processing stops, and control returns to the caller.

In the case of task-creating commands, the appropriate processor receives control only after further preliminary steps have been taken. For all commands in this category, control passes to the CSCB-creation module (IEE0803D). This module builds a command scheduling control block to contain the command and stores an encoded version of the command (verb and operands) in the block. If the command is either a START, MOUNT, or LOGON command, a new memory (or address space) is required. In that case, a memory-request processor communicates with the system resources management (SRM) routines and establishes the environment for the new memory. The request processor creates an address space control block (ASCB) and establishes an address space identification (ASID) for the new memory. Module IEE0803D sets on an assignment-pending indicator in the CSCB, places the newly-created CSCB on the CSCB chain, and uses the cross-memory form of the POST macro instruction to notify the master scheduler wait routine, which is the initial responder to all task-creating commands.

When the master scheduler wait routine (IEEVWAIT) receives control in response to a POST macro instruction, it searches the CSCB chain for a CSCB in pending (or availability) status. When it finds a pending CSCB, other than one for a START, MOUNT or LOGON command, it removes the CSCB from the chain and attaches the appropriate command processor in the master scheduler's region. In the case of a START or MOUNT command, the master scheduler attaches the memory-create function rather than the command processor. For a LOGON command, the master scheduler attaches the terminal input/output coordinator (TIOC) processor, which in turn gives control to the memory-create function. The memory-create routine(s) give control to the region control task (RCT) routines, which pass control to the started task control (STC) routines to initiate the processing associated with a START, LOGON, or MOUNT command. After all pending CSCBs have been processed, the master scheduler waits (via a WAIT macro instruction) until it is again posted for an event control block (ECB).

Each attached (task-creating) command processor uses the system macro instruction MGCR to free its corresponding CSCB storage area. The processor operates in supervisor state with a system key of zero and operates under a job step task control block (TCB). The processor lacks a save area because its task ceases to exist when current use of the processor is finished.

Reconfiguration Commands

There are several commands in the MVS operating system that assist in the reconfiguration of the operating system. These commands permit operations personnel to have the capability of adding components (or elements) to a running system, of removing failing components, and of taking a CPU, channels, devices, and areas of main (real) storage offline for maintenance. System reconfiguration involves a physical or logical change in the type or quantity of components available to the operating system.

Physical reconfiguration is the actual connection of components to or the disconnection of components from the system. An operator may perform physical reconfiguration on an operating system by using the QUIESCE command before the reconfiguration occurs. This command suspends system activity until the operator signals via a system restart interrupt that the system may continue.

Logical reconfiguration, which programming accomplishes, involves changing system tables to notify the control program of any physical changes. Logical reconfiguration may be performed without performing physical reconfiguration, but it should always be performed whenever a physical reconfiguration occurs. An operator (or programmer) may perform logical reconfiguration either when a system is loaded (at IPL time) or by using a form of the VARY command to change the status of CPUs, channels, devices, and main storage.

Command Processing Modifications

Changes to command processing routines from VS2 Release 1 include the following:

- Module-to-module linkage is accomplished by using branch instructions instead of the XCTL macro instruction mechanism.
- The extended save area (XSA) and the command buffer interface used throughout are obtained by the same GETMAIN macro instruction and are contiguous in storage.
- All SVC 34 processor modules reside within one load module, IGC0003D.
- In addition to the existing-task command and task-creating commands of previous releases, the task-creating commands contain the TART, MOUNT, and LOGON commands in a subset known as memory-creating commands.
- Enqueue-Dequeue logic is used to add elements to, and delete elements from, the CSCB chain.
- For time-sharing (TSO) oriented commands, the address space identification block (ASID replaces the time-sharing identification block (TJID).
- The system log has been designed to eliminate the log data sets SYS1.SYSVLOGX and SYS1.SYSVLOGY.

- The DISPLAY (A, TS or JOBS) command processor scans the CSCB chain instead of the TCB chain.
- Resources protection routines use a "lock" mechanism to replace the disabling feature of the Set System Mask mechanism.
- Jobqueue commands (such as HOLD Q) are removed.
- Reconfiguration commands (such as VARY (CPU, or STOR, or PATH, or CHAN) exist.
- Command processors use VS2 supervisor error recovery techniques—see the Recovery/Termination Management section of this publication.
- A TRACK command requires changes to the MSGRT and CONTROL commands.
- The SET and RESET commands have operands that require an interface with the system resources manager (SRM).
- A command, CHNGDUMP, permits parameter changes to the DUMP command and to ABEND dumps.
- The VARY CONSOLE (ONLINE, OFFLINE) and UNLOAD commands now are task-creating commands (that is, they are processed by processors attached by the master scheduler's routine IEEVWAIT).
- A command, TRACE, permits maintenance of the NIP trace table after system initialization.

Changes to command processing routines from VS2 Release 2 include the following support for:

- The 3850 Mass Storage System (MSS) has been added. It consists of the library with its associated read/write units, DASD staging devices, and controllers.
- Varying a range of devices online or offline is now supported by means of the VARY command.
- The MSGRT and parts of the control commands may be issued under tasks other than the communications task (i.e., by JES3). These instances are protected via the CMS and local locks.

Command to Which Diagram Relates	Diagram Title
The first seven diagrams apply either in part or in whole to many of the commands listed in the rest of this summary.	2-1 SVC 34 Common Processing Initialization (Overview) 2-2 Creating STAE Environment for SVC 34 Command Processing (IEE0003D) 2-3 SVC 34 STAE Routine (IEE5103D) 2-4 SVC 34 General Message Assembly Routine (IEE0503D) 2-5 Manipulation of Command Control Blocks (QEDIT) (IEE0303D) 2-6 Command Translation and Routing (IEE5403D) 2-7 Creating CSCB for Task-Creating Commands (IEE0803D) 2-8 Master Scheduler Wait (IEEVWAIT) 2-9 Master Scheduler Wait Recovery and Retry (IEEVWAIT)
START, LOGON, and **MOUNT** Commands	2-10 Obtaining a New Virtual Memory (IEE0803D)
CANCEL – The CANCEL command causes the cancellation of an executing task by posting the CANCEL ECB in the CSCB. This causes recovery termination management routines to terminate the task.	2-11 Cancelling Background (Batch) and Foreground (TSO) Jobs (IEE3703D) 2-12 System-Initiated Cancelling of a TSO User (IKJL4T00)
CHNGDUMP – The CHNGDUMP command causes a change in system dump parameters.	2-13 Changing Dump Parameters (IEEMB815)
CONTROL – The CONTROL command establishes and changes the functions of a graphics console, mainly in the areas of screen definition and control.	2-14 CONTROL Command Processing (IEE6703D)
DISPLAY – The DISPLAY command causes a graphic display of the current status of various system functions.	2-15 DISPLAY and TRACK Command Preprocessing (IEE3503D) 2-16 Displaying and Tracking System Status (IEECB800) 2-17 Displaying Console Status (IEEXEDNA) 2-18 Displaying CONTROL Command Operands (IEE00110) 2-19 Displaying a Matrix of System Status (IEEMPDM) 2-20 Displaying Operator-Action Requests (IEE2903D) 2-21 Display of Program-Function-Key Definitions (IEE40110) 2-22 Displaying Unit Status (IEE20110) 2-22A Displaying Parameters of Domains (IEEDISPD)
DUMP – The DUMP command interfaces with the SVC DUMP macro instruction to provide a storage dump of specified options.	2-23 Dumping Virtual Storage (IEECB866)

Figure 2-5. Command Processing Method-of-Operation Diagram Summary (Part 1 of 4)

Command to Which Diagram Relates	Diagram Title
HALT – The HALT command closes the system log, empties the SMF buffers, and stops teleprocessing operations.	2-24 HALT, SWITCH, and TRACE Command Initialization (IEE1403D) 2-25 HALT and SWITCH Command Processing (IEE70110)
HOLD – The HOLD command permits the interception of messages going to a TP station.	2-55 Holding and Releasing Teleprocessing Messages (IED1303D)
LOG – The LOG command writes text entries into the system log.	2-26 Processing LOG and WRITELOG Commands (IEE1603D)
LOGON – The LOGON command, which is an internally-issued command, causes the creation of a new memory space for a TSO-user. See the section, *Started Task Control.*	
MODE – The MODE command controls recovery management activity and displays information about the current state of recovery management facilities.	2-27 SWAP (IGF2503D) and MODE (IGF2603D) Command Processing
MODIFY – The MODIFY command sends parameters (in a command input buffer) to an executing task to modify that task.	2-28 STOP/MODIFY Command Processing (IEE0703D)
MONITOR – The MONITOR command causes a display of the status of the system to reflect changing events.	2-29 Starting (IEE7103D) and Stopping (IEE5503D) Monitoring Functions
MOUNT – The MOUNT command allocates a device to several job steps that require a given volume, and it eliminates the need for intervening mounts and demounts of the volume. See the section, *Started Task Control.*	
MSGRT – The MSGRT command routes certain status display options to a given console or screen area.	2-30 Routing Messages to Consoles (IEE6303D)
PAGEADD – The PAGEADD command adds page or swap data sets to the system.	25-31 Page Expansion (ILRPGEXP)
QUIESCE – The QUIESCE command, which is generally used in conjunction with a VARY command and in a MP environment, stops a system before the controls at a configuration's control panel are modified.	2-31 Quiescing a System (IEEMP503)
RELEASE – The RELEASE command releases previously-held messages to a TP station.	2-55 Holding and Releasing Teleprocessing Messages (IED1303D)

Figure 2-5. Command Processing Method-of-Operation Diagram Summary (Part 2 of 4)

Command to Which Diagram Relates	Diagram Title
REPLY – The REPLY command provides a facility to answer WTOR messages from the system and from problem programs.	2-32 Replying to Information Requests (IEAVVRP1)
RESET – The RESET command dynamically changes the performance group of a job currently executing.	2-33 RESET Command Processing (IEEMB810)
SEND – The SEND command provides for message communication between operators and logged-on time-sharing (terminal) users.	2-34 Sending/Saving/Listing Messages (IEEVSEND)
SET – The SET command (1) establishes the local date and time of day and (2) permits the respecification of parameters needed by the system resources manager for controlling job scheduling .	2-35 Setting Local Time (IEE0603D) 2-36 Changing IPS Values (IEEMB811)
SETDMN – The SETDMN command permits the respecification of parameters used by the system resources manager (SRM) to control the multiprogramming level in a domain.	2-60 SETDMN Command Processing (IEE8603D)
START – The START command causes the starting of a procedure that resides in SYS1.PROCLIB. See the section, *Started Task Control.*	
STOP – The STOP command halts the execution of a task by posting an ECB.	2-28 STOP/MODIFY Command Processing (IEE0703D)
STOPMN – The STOPMN command stops the processing being performed by a previously-issued MONITOR command.	2-29 Starting (IEE7103D) and Stopping (IEE5503D) Monitoring Functions
STOPTR – The STOPTR command stops the processing being performed by a previously-issued TRACK command.	2-37 Stopping Periodic Track (Status) Displays (IEE5503D)
SWAP – The SWAP command activates or deactivates dynamic device reconfiguration (DDR) for purposes of a volume exchange on device(s).	2-27 SWAP and MODE Command Processing (IEE0403D)
SWITCH – The SWITCH command permits a manual switching of SMF data sets for recording purposes.	2-24 HALT, SWITCH, and TRACE Command Initialization (IEE1403D) 2-25 HALT and SWITCH Command Processing (IEE70110)
TRACE – The TRACE command causes the master scheduler to *either terminate or continue* system tracking after initialization of the primary job entry subsystem occurs.	2-24 HALT, SWITCH, and TRACE Command Initialization (IEE1403D)
TRACK – The TRACK command permits a periodic display of job information on a display console.	2-15 DISPLAY and TRACK Command Processing (IEE3503D) 2-16 Displaying and Tracking System Status (IEECB800)

Figure 2-5. Command Processing Method-of-Operation Diagram Summary (Part 3 of 4)

Command to Which Diagram Relates	Diagram Title
UNLOAD – The UNLOAD command logically removes (demounts) a volume that was previously mounted as a result of a MOUNT command.	2-38 Unloading I/O Devices (IEEMB813)
VARY – The VARY command controls data handling resources (such as I/O units, consoles, CPUs, channels, paths, and storage) as well as the status of, and access to, these components for the system.	2-39 Routing of VARY Commands (IEE3203D) 2-40 Changing Console Status, Message Routes, and Command Authorization (IEE3603D) 2-41 VARY CN Processing (IEECB900) 2-42 VARY CN Processing (IEECB901) 2-43 Varying Devices (Consoles or I/O Units) Online and Offline (IEE4203D) 2-45 VARY HARDCPY Command Processing (IEE4703D) 2-46 Master Console Switching (IEE4303D) 2-47 Varying a CPU or Channel Offline or Online (Overview) (IEEVCPU) 2-48 Varying a CPU Online (IEEVCPU) 2-49 Varying a CPU Offline (IEEVCPU) 2-50 Varying a Channel Online (IEEVCPU) 2-51 Varying a Channel Offline (IEEVCPU) 2-52 Varying the Path to a Device (IEEVPTH) 2-53 Varying a Range of Device Addresses (IEECB904) 2-44 Varying the Status of Real Storage (IEEMPVST)
WRITELOG – The WRITELOG command activates or deactivates the system log and switches the log data sets.	2-26 Processing LOG and WRITELOG Commands (IEE1603D)
Commands entered into the system via SVC 34 routines but which are processed by components other than the master scheduler.	2-54 Teleprocessing (TP) Commands (IED1303D) 2-55 Holding and Releasing Teleprocessing Messages (IEE0803D) 2-56 Processing Commands with the "NET" Operand (ISTCFF3D)
Additional routines described in this section because of their major use by a command processor.	2-57 Stopping and Restarting (via an Interrupt) the System (IEESTPRS) 2-58 Device Information Subroutine (IEEVDEV)
Miscellaneous Routine	2-59 Deleting a Virtual Memory (IEAVEMDL)

Figure 2-5. Command Processing Method-of-Operation Diagram Summary (Part 4 of 4)

Diagram 2-1. SVC 34 Common Processing/Initialization – Overview (IGC0003D) (Part 1 of 2)

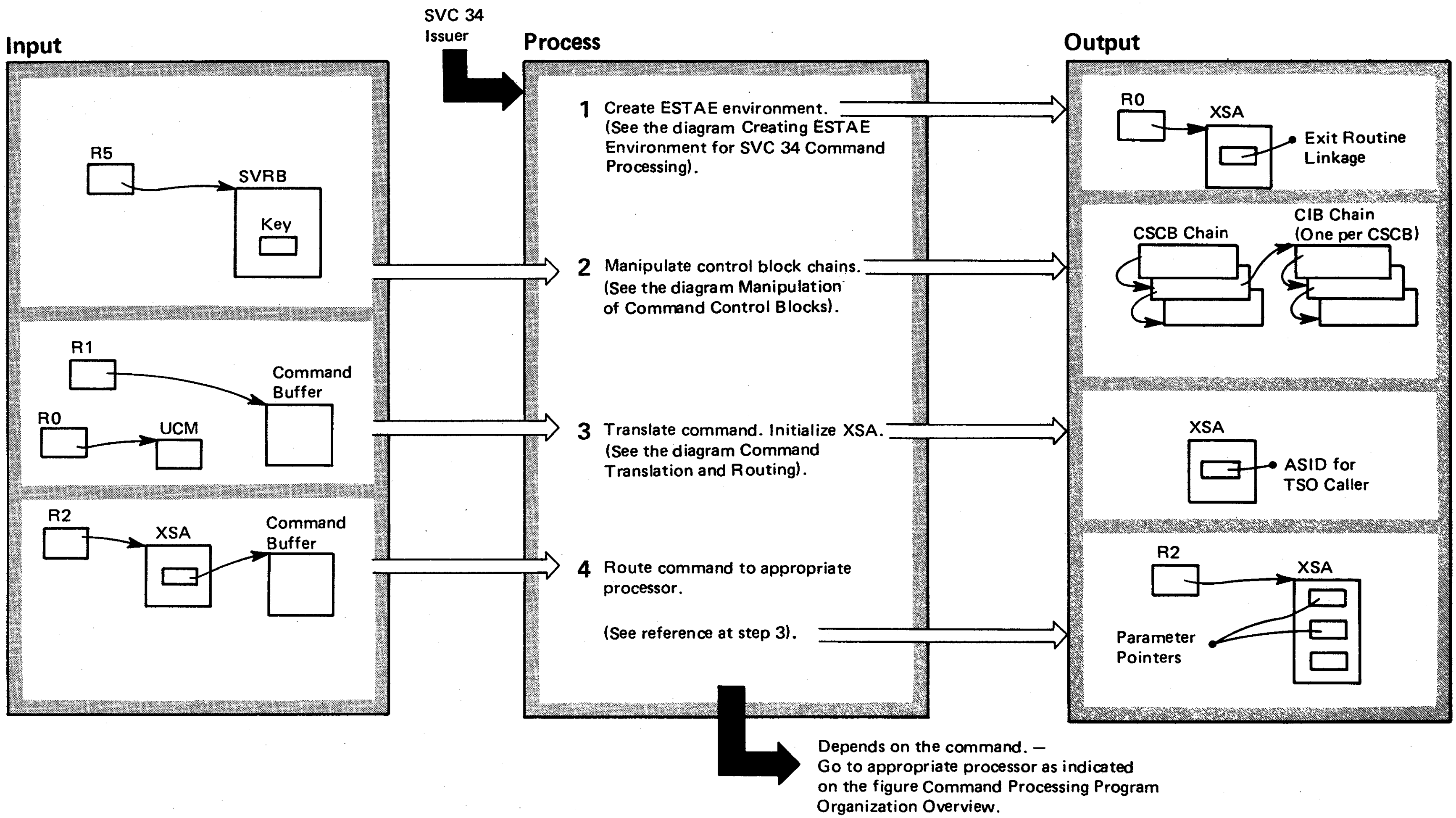


Diagram 2-1. SVC 34 Common Processing/Initialization – Overview (IGC0003D) (Part 2 of 2)

Extended Description	Module	Label
This processing prepares the system for handling of a command by the appropriate processor.		
1 This environment protects the command scheduler (SVC 34) from an abnormal end (ABEND).	IEE0003D IEE5103D	
2 Check system authority. Set up CSCB and CIB chains. Handle ABTERM requests.	IEE0303D	TABLE XCHAIN XEOT
3 Translate syntax. Initialize XSA. For the multiple-console support option, check hardcopy log requests.	IEE5403D	
4 First, check the validity of the command authority. Then, route the command to the appropriate processor.	IEE0403D	
If the authority is invalid, control goes to the error routine, IEE0503D.		

Diagram 2-2. Creating STAE Environment for SVC 34 Command Processing (IEE0003D) (Part 1 of 2)

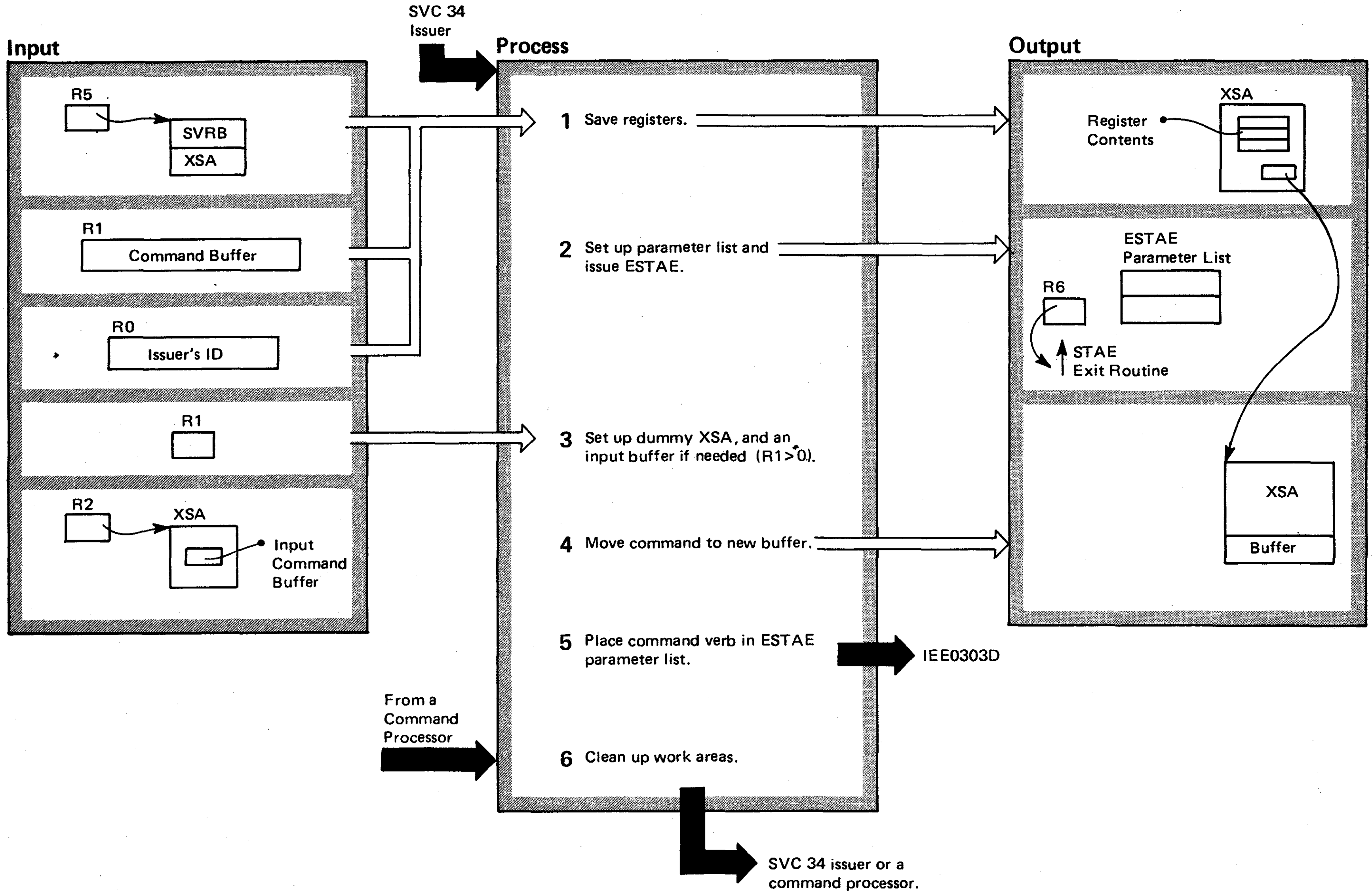


Diagram 2-2. Creating STAE Environment for SVC 34 Command Processing (IEE0003D) (Part 2 of 2)

Extended Description	Module	Label
This STAE environment handles ABEND situations occurring in command processing routing.		
1 Save registers 0, 1, 5, 11, 14, 15. The XSA is contiguous to the SVRB. Register 0 indicates if the issuer of SVC 34 is one of the following: • An input stream command. • A console (the ID is given). • A TSO terminal (the ID is given). • The operating system.	IEE0003D	
2 The parameter list contains a one-word address of the retry routine and a one-word field containing both the number of the subpool from which the parameter list storage was obtained and the size of the parameter list. This information is used when the work areas are freed.		
3 If R1 <0, a buffer is not needed. Routine IEE0303D receives control to handle control block manipulations. A GETMAIN macro instruction is issued for the XSA.		
4 The buffer is at the end of the XSA.		
5 This action makes verb available for later insertion into message if an ABEND occurs. Control now passes to the block chain handler to set up for the action defined by the command.		XDISCAN
6 On return from the proper command processor (other than one attached via IEEVWAIT), storage is freed for the dummy XSA and the parameter list.	IEE0003D	COMMEXIT

Diagram 2-3. SVC 34 STAE Routine (IEE5103D) (Part 1 of 2)

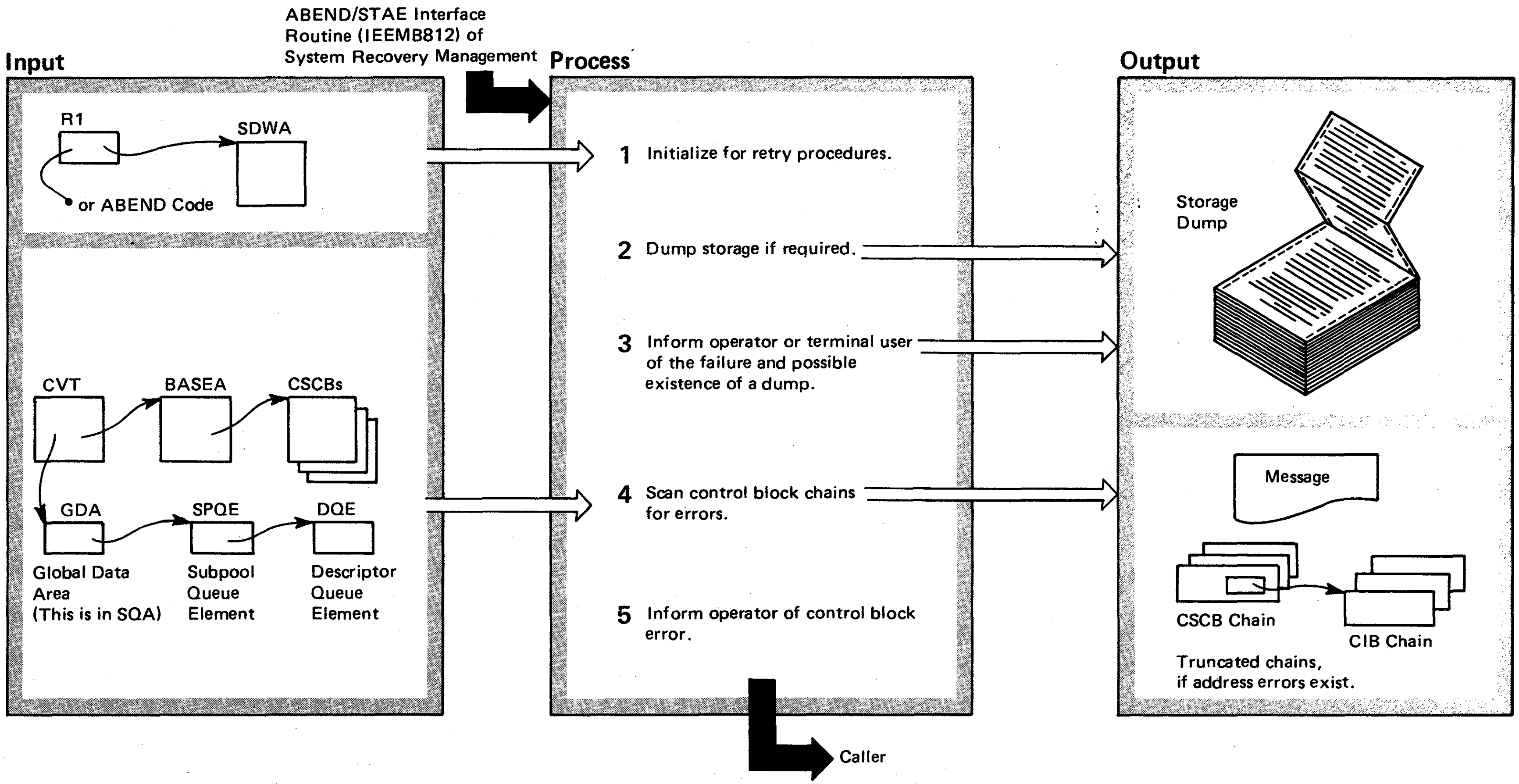


Diagram 2-3. SVC 34 STAE Routine (IEE5103D) (Part 2 of 2)

Extended Description	Module	Label
This process provides a storage dump and message for ABEND situations. For each command it receives, the ESTAE recovery routine (IEE0003D) makes the name of this STAE routine available for system recovery management routines.		
1 This step permits successive entries without getting in a loop. If R0 is other than 12, then R1 points to the system diagnostic work area (SDWA).*	IEE5103D	STAE0020

SDWA

↑ Parameter list
ABEND code
PSW at ABEND time
Last problem program PSW before ABEND
Registers contents at ABEND
Name of ABENDed local module
↑ Module that is ABENDed

*If R0=12, then R1 contains the ABEND code and R2 points to the parameter list.

Extended Description	Module	Label
2 A dump (using the SDUMP macro instruction (SVC 51)) is taken for a system failure, for a program check, or if "RESTART" key is depressed.		STAE0050
3 The routine uses a WTO or a TPUT macro instruction. The message includes the ABEND code and an indication of the success of the dump.		STAE0218 STAE0120
4 Each CSCB and its associated CIBs are scanned for boundary and region requirements within the SQA. If an error is found, the rest of the chain is truncated.		STAE01S0
5 The operator receives a message indicating that the control block chain(s) are truncated.		

Diagram 2-4. SVC 34 General Message Assembly Routine (IEE0503D) (Part 1 of 2)

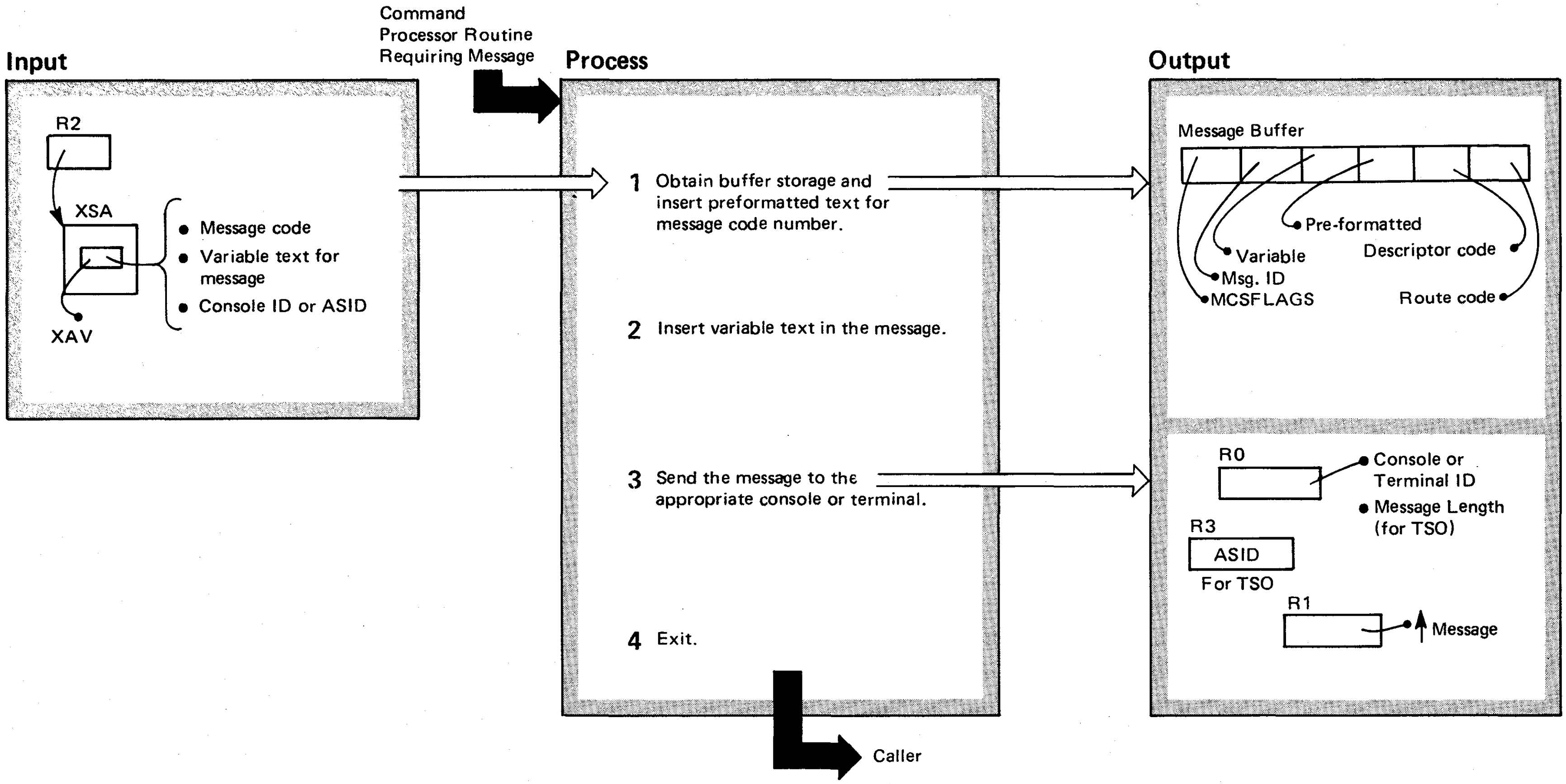


Diagram 2-4. SVC 34 General Message Assembly Routine (IEE0503D) (Part 2 of 2)

Extended Description	Module	Label
The command processor'message routine informs the operator or terminal user of processing success and of errors occurring during command processing.		
1 The message code number in the XSA (of the SVRB) must be valid. Buffer storage comes from either the LSQA (first choice) or the SQA. The inserted text is part of the WTO parameter list.	IEE0503D	IGG2103D
2 The type of user determines what this entry will be: for example — it may be a command verb or a job-name, etc.		GOTCORE
3 For console messages, the routine uses the WTO macro instruction. For terminal messages, the routine uses the TPUT macro instruction. (If an outstanding TPUT require-ment prevents this message output, one retry is attempted.)	IEE0503D	CONSOLE TPUTMSG
4 The storage buffer work area (used for the WTO parameter list) is freed prior to returning to the calling routine.		FREECORE

Diagram 2-5. Manipulation of Command Control Blocks (QEDIT) (IEE0303D) (Part 1 of 2)

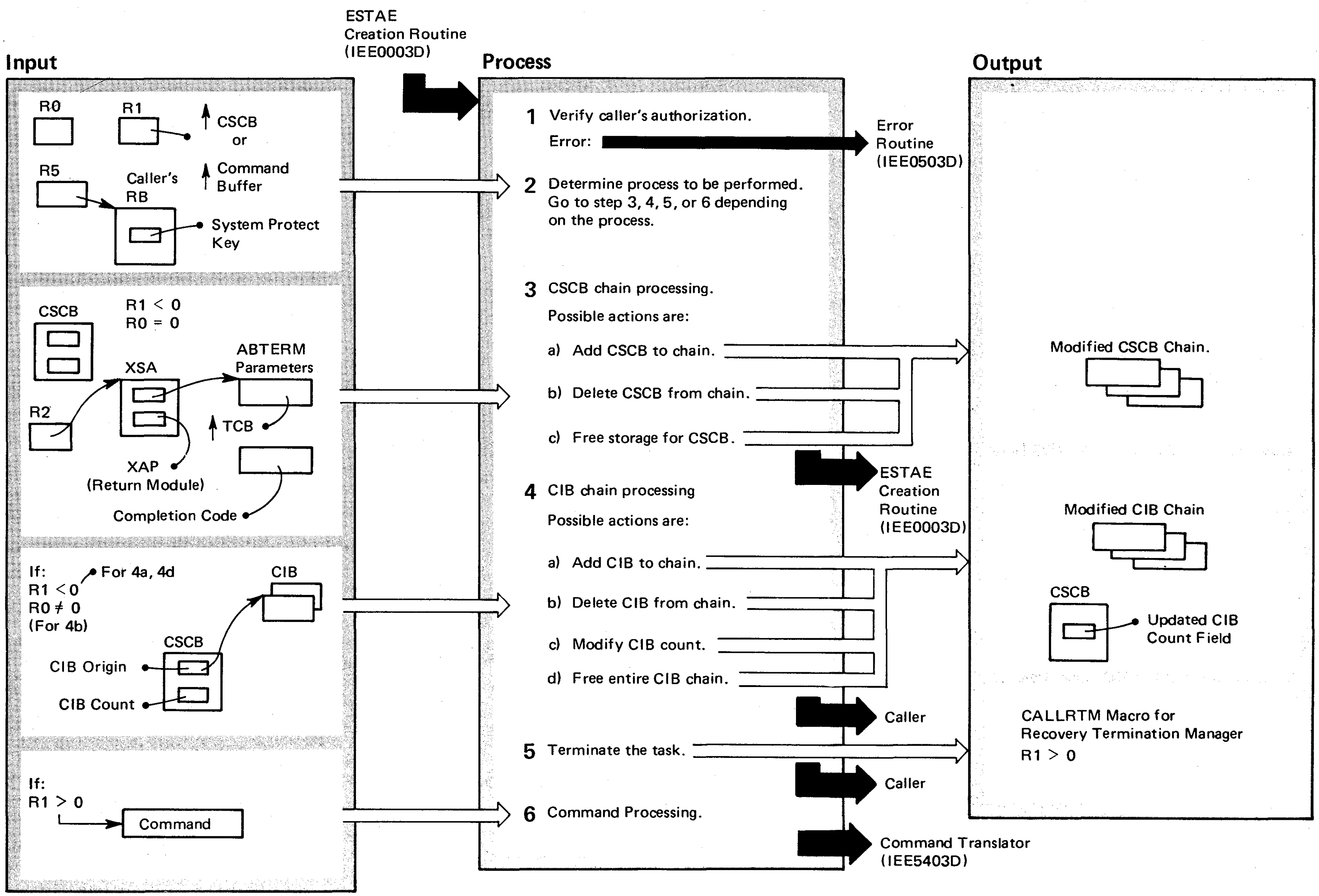


Diagram 2-5. Manipulation of Command Control Blocks (QEDIT) (IEE0303D) (Part 2 of 2)

Extended Description	Module	Label
By modifying counts and chaining control blocks, this routine manipulates (1) CIBs for the QEDIT macro instruction and (2) CSCBs for system processing.		
1 Only system-key programs can issue commands and manipulate CSCBs. Commands may also be issued by using the QEDIT or MGCR macro instruction.	IEE0303D	TABLE
2 The following relationships exist between the input register contents and the subsequent processing action that occurs.		IEE0303D

R0	R1	Key	Processing Action*	Notes
0	Neg.	System	CSCB Processing.**	R1 = 2's complement of CSCB block address.
Pos.	Neg.	System	CIB Processing. (To add a CIB)	R1 = 2's complement of CIB block address.
Pos.	Neg.	Any	CIB Processing. (To delete a CIB.)	R0 = Address of origin of CIB block.
Pos.	0	System	Free the CIB chain.**	R0 = Address of origin of CIB block.
Neg.	Neg.	Any	Set the CIB count in the CSCB to zero.	
Neg.	Neg.	Any	Place positive R1 value in CIB count field of CSCB.	R1 = CIB count (2's) complement. R0 = 2's complement of block's address
0	Pos.	System	ABTERM processing. The routine uses the CALLRTM macro instruction, and it branches to the recovery termination manager.	The CHABT bit must be on.
	Pos.	System	A command is to be processed.	For a system task issuing an SVC 34 instruction.
	Pos.		Routine returns control to the caller.	A problem program is not allowed to issue an internal command.
0	0		Routine returns control to the caller. This is an error condition.	

*A system protect key >8 indicates a problem program action. Otherwise, a system program action is indicated.

**This processing is allowed only if the system key <8. Problem programs cannot set flags in protected storage.

Extended Description	Module	Label
3 This processing occurs when R0 is zero and R1 is negative. Flags in the CSCB status byte determine which of the actions occurs. CSCBs are added at the end of a chain. Enqueue (ENQ) logic is used by the routine to serialize the resource. For task termination (that is, if the ABTERM bit CHABT in the CSCB is equal to 1), the ABTERM parameters are passed to the recovery termination manager via the CALLRTM macro instruction.	IEE0303D	XCHAIN XEOT
4 This processing includes modification of the CIB count for the particular CSCB. In this case, R1 contains the two's complement of the CIB count.		XSETCNT
If a CIB is on the chain, it is removed and its storage space is freed. CIBs are added at the end of a chain.		XCIBCHN
5 SVC 34 requires translation and routine processing services for the individual commands.		

Diagram 2-6. Command Translation (IEE5403D) and Routing (IEE0403D) Routines (Part 1 of 2)

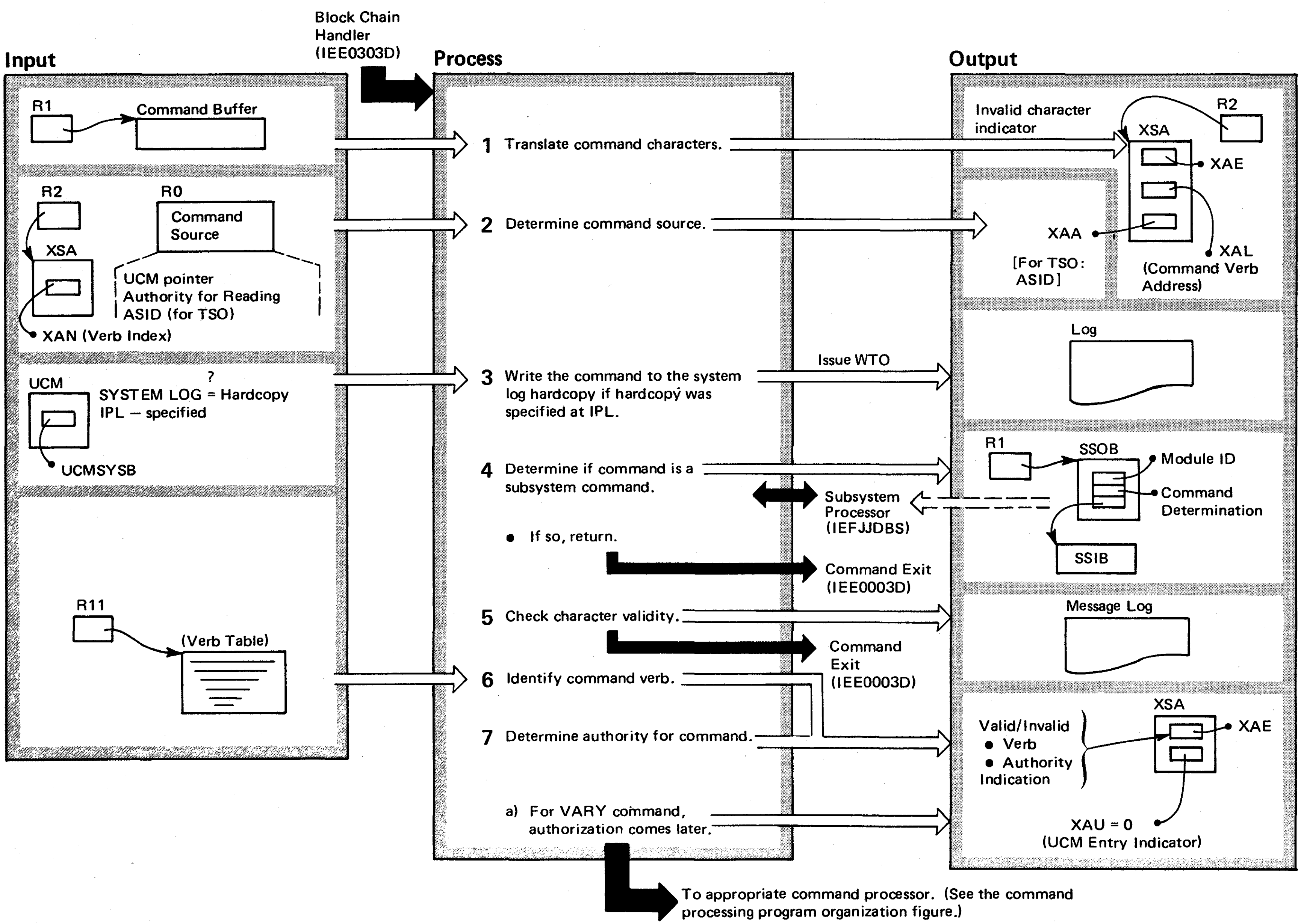
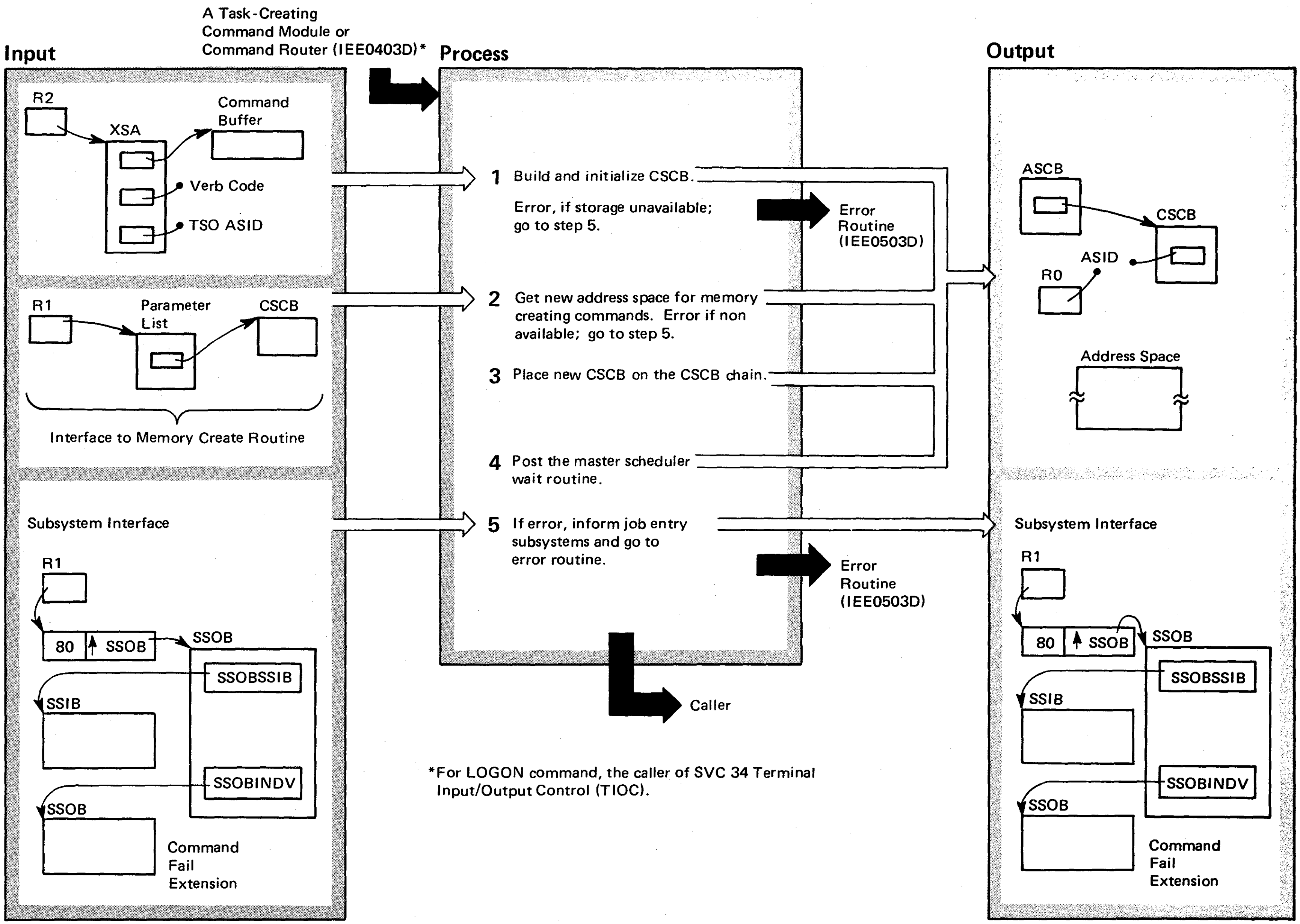


Diagram 2-6. Command Translation (IEE5403D) and Routing (IEE0403D) Routines (Part 2 of 2)

Extended Description	Module	Label
These routines scan and translate commands, set up control blocks, determine authority for a command, and pass the command to the appropriate processing routine.		
1 This routine changes lower case letters to upper case. It uses an internal translate table to do this.	IEE5403D	TRANSO
There is an exception to this: Characters within single quotes remain unchanged.		
2 [For invalid characters, IEE0503D issues a message.] Either an input stream, TSO terminal, or console command.	IEE5403D (IEE0503D)	CHKMCS
3 For all commands except REPLY and CONTROL. This step is bypassed in the case of REPLY and CONTROL commands.	IEE5403D	HCFLG
4 Master subsystem processor returns the indication of the command form (that is, subsystem or otherwise).	IEFSSREQ	
5 See step one.	IEE0403D	GETMSG–
6 The routine issues this message when it encounters an invalid verb. The verb table contains a list of all acceptable command verbs.	IEE0403D	STPTR
7 Authority required only for externally-issued command.	IEE0403D	XAUTH
• Invalid authority, issue message.	IEE0503D	

Diagram 2-7. Creating CSCB for Task Creating Commands (IEE0803D) (Part 1 of 2)



*For LOGON command, the caller of SVC 34 Terminal Input/Output Control (TIOC).

Diagram 2-7. Creating CSCB for Task Creating Commands (IEE0803D) (Part 2 of 2)

Extended Description	Module	Label
For task-creating commands, this routine constructs a CSCB. If a memory creating situation exists, memory request routines perform initialization.		
1 Storage for the CSCB comes from subpool 245 (that is, the SQA). The CSCB will contain the command verb code, the size of the CSCB, the ID of the issuing console, the screen area ID for the receiving console, and the ASID for the address space.	IEE0803D	CSCB
2 START, MOUNT, and LOGON commands are the memory-creating commands and they require the ASCB.	IEAVEMRQ	
3 ENQ-DEQ protection of the CSCB chain is used while chaining takes place. A SYSEVENT macro instruction causes the SRM to make the current memory non-swappable until the dequeue is complete.	IEE0803D	ENQCSCB
4 After the routine sets the assignment-pending bit (for the particular CSCB) in the CSCB itself, it uses a cross-memory POST macro instruction to post the master scheduler wait routine for further action.	IEE0803D	POST0019
5 If either a CSCB or an address space was unavailable for a START command that was not issued under the comm task, the routine passes an SSOB to all active subsystems. The SSOB contains the address of the command buffer and a return code that indicates the type of failure. If the return code from the subsystem is nonzero, this routine's error message is suppressed and control returns to the caller. (In this case, the subsystem issues its own error message.) Otherwise, control goes to IEE0503D to issue message IEE328I ("xxxx COMMAND ABORTED"). For failing LOGON and MOUNT commands, this message is always issued. For all three commands (START, LOGON, and MOUNT) the return code is set to 08 to indicate command failure.	IEE0803D	EXIT3A

Diagram 2-8. Master Scheduler Wait (IEEVWAIT) (Part 1 of 2)

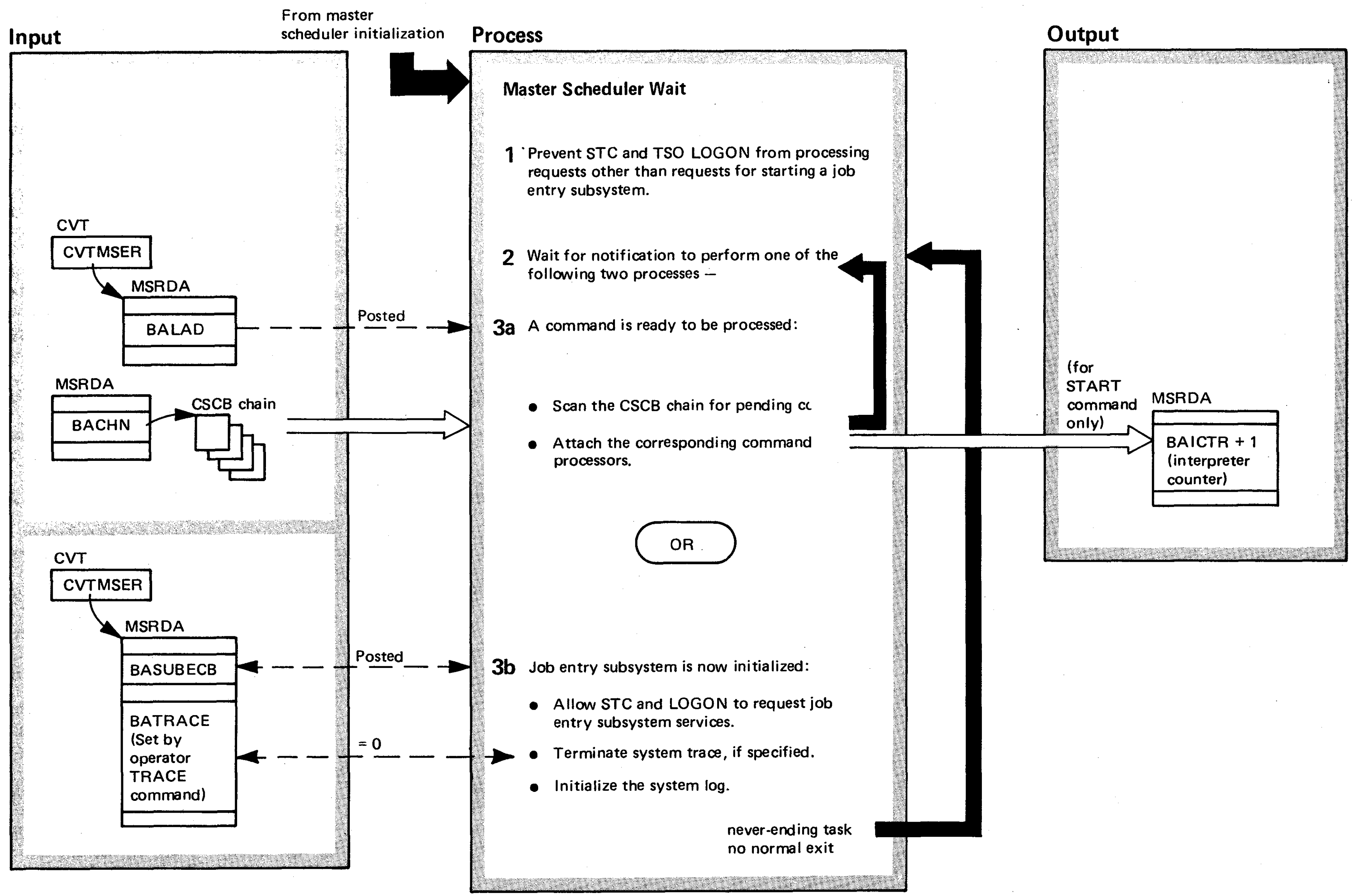


Diagram 2-8. Master Scheduler Wait (IEEVWAIT) (Part 2 of 2)

Extended Description	Module	Label
When the master scheduler (IEEVWAIT) receives control at system initialization time, it first processes the START command for the job entry subsystem and any other pending commands. (The automatic commands contained in SYS1.PARMLIB are pending at this time.) After the job entry subsystem has initialized itself, IEEVWAIT terminates system trace, if specified, and attaches the system log task. After system initialization time, IEEVWAIT's only function is to scan the CSCB (command scheduling control block) chain when posted to do so and to process any pending commands by attaching the proper command processor. The initialization function of IEEVWAIT is more fully described in *OS/VS2 System Initialization Logic,* SY28-0623. The recovery function of IEEVWAIT is described in the next diagram, "Master Scheduler Wait Recovery."	IEEVWAIT	
1 At initialization time, IEEVWAIT enqueues on the STC and TSO internal readers: • STC internal reader – major name SYSIEFSD, minor name STCQUE. • LOGON internal reader – major name SYSIEFSD, minor name TSOQUE. It holds these resources until the job entry subsystem (JES2, for example) has initialized itself. While the job entry subsystem is initializing itself, any START/LOGON/MOUNT commands can be processed up to the point where STC needs the job entry subsystem to write JCL to the spool data set. At this point STC enqueues on one of the internal readers. Thus, STC cannot request subsystem services until the job entry subsystem is initialized and IEEVWAIT has dequeued from the internal readers.	IEEVWAIT	WAIT0000
2 Master scheduler wait issues a wait on two ECBs. Depending on which ECB is posted, either Step 3a or Step 3b of this diagram is performed.	IEEVWAIT	WAITING
3a If the wait ECB is posted, master scheduler wait scans the CSCB chain until it finds one with the pending bit on. It then attaches the processor corresponding to the command verb in the pending CSCB. Master wait repeats the scan until no pending CSCBs are left on the chain. Then master wait returns to Step 2 processing to wait for further notification.	IEEVWAIT	WAIT0030
3b If the subsystem ECB is posted, the job entry subsystem has completed its initialization. Master scheduler wait releases the serialization resources it obtained in Step 1, so that STC and TSO LOGON can request subsystem services. IEEVWAIT terminates system tracing by replacing the trace-active instruction in the CVT with a dummy instruction, setting to zero all the PSA pointers to the trace table, and deleting the trace table itself. Also, the system log task can now be attached for initialization processing. Refer to the topic "System Log" in this publication.	IEEVWAIT	WAIT0060

Diagram 2-9. Master Scheduler Wait Recovery and Retry (IEEVWAIT) (Part 1 of 2)

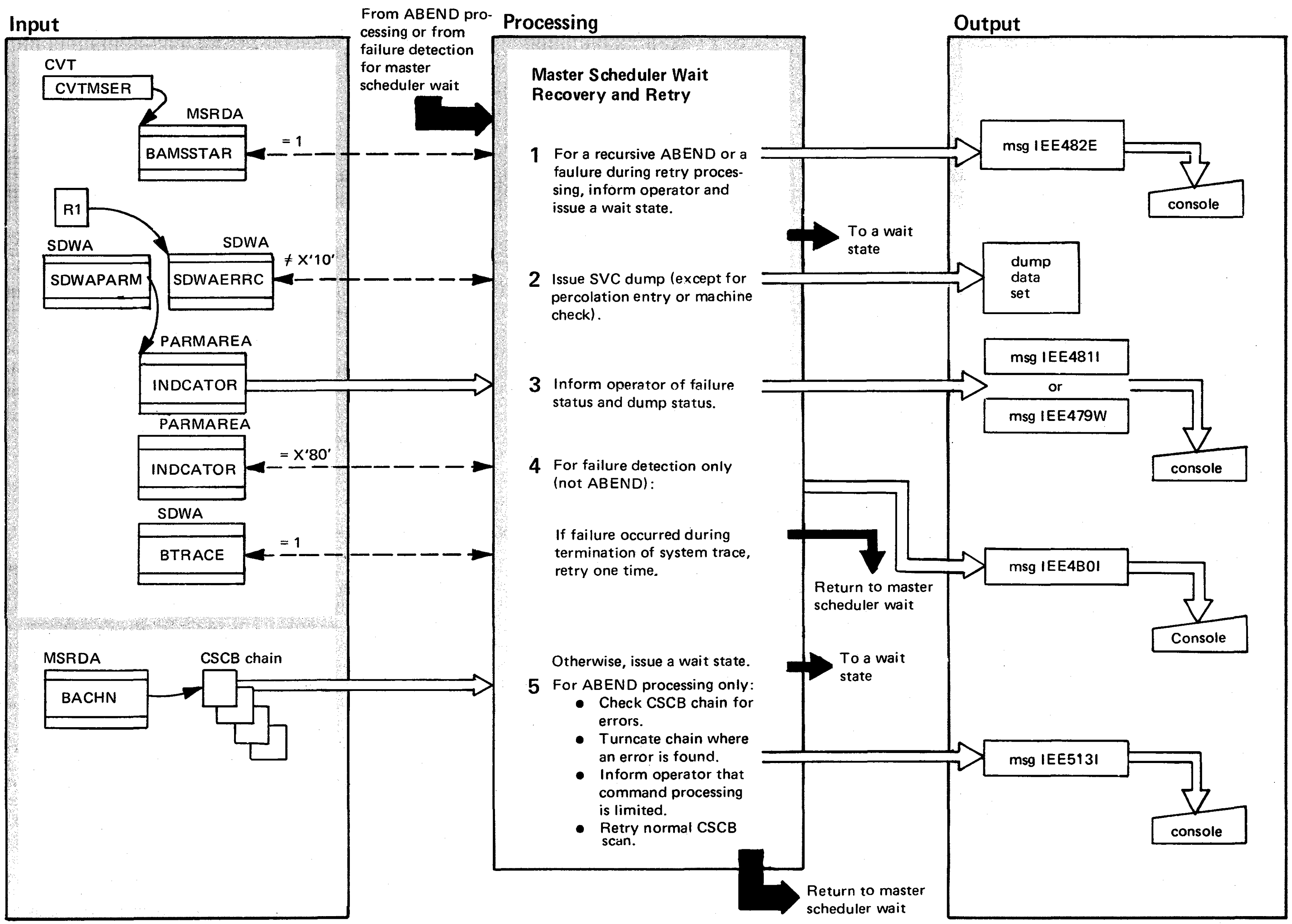


Diagram 2-9. Master Scheduler Wait Recovery and Retry (IEEVWAIT) (Part 2 of 2)

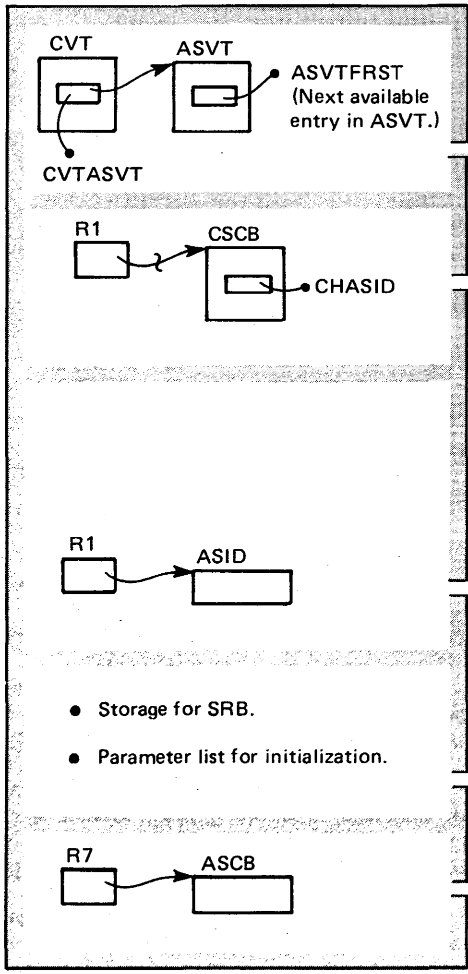
Extended Description	Module	Label
The recovery portion of master scheduler wait either puts the system into a wait state after issuing a dump and operator messages, or attempts a retry of master scheduler wait after correcting the CSCB chain.	IEEVWAIT	STAE0000
1 If repeated attempts at executing this recovery code result in ABENDs or if the retry of the CSCB scan fails, message IEE482E is issued to inform the operator and the system is put into a wait state.	IEEVWAIT	STAE0000
2 If another recovery routine has passed control to the master wait's recovery code (an event called percolation), a dump is not necessary.	IEEVWAIT	STAE0000
3 The operator messages indicate whether an ABEND or a failure occurred and whether or not an SVC dump was successfully taken.	IEEVWAIT	STAE0070

Extended Description	Module	Label
4 A failure during the termination of system trace results in a retry of the initialization code in IEEVWAIT that executes following the initialization of the primary job entry subsystem. If the retry attempt abnormally terminates, message IEE480I is sent to the operator requesting him to re-IPL. The retry is attempted only once.	IEEVWAIT	
5 Master wait recovery checks the CSCB chain for errors and truncates the chain where one of the following errors is found: • A CSCB is not located in the SQA (subpool 245). • A CSCB is not on a doubleword boundary. Next, the normal CSCB scan is retried. If this scan does not cause another ABEND, the master scheduler is considered to be restarted. Message IEE513I informs the operator that command processing is limited (that is, some commands may have been deleted from the CSCB chain during retry).	IEEVWAIT	STAE0140

Diagram 2-10. Obtaining a New Virtual Memory (Part 1 of 4)

Block Chain Handler (IEE0803D) for START, LOGON, or MOUNT Commands (IEE0403D)

Input



Process

1 Create CSCB.

2 Build control blocks and assign an address space.

3 Notify SRM of desired memory.

4 Post the master scheduler.

From IEEVWAIT (START or MOUNT)

From IEDAY3 (LOGON)

5 Build segment and page tables.

6 Enqueue the ASCB.

7 Initialize and schedule SRB for memory initialization.

8 Free the SRB storage.

Output

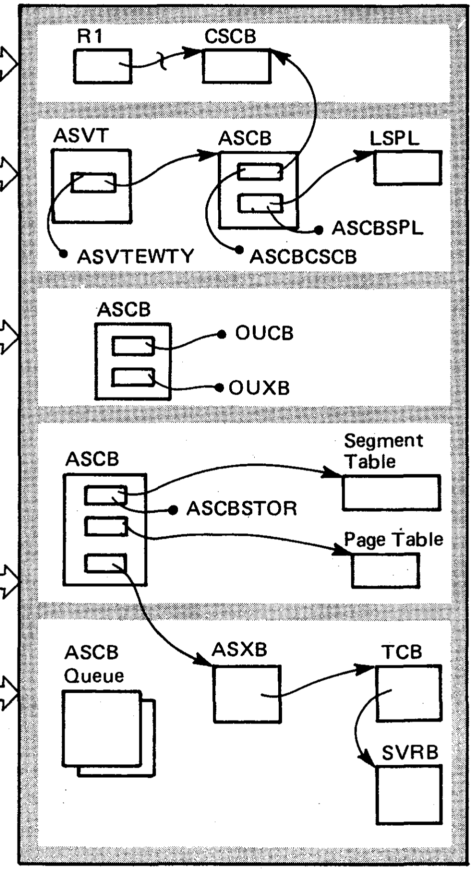


Diagram 2-10. Obtaining a New Virtual Memory (Part 2 of 4)

Extended Description	Module	Label
This routine obtains and initializes a new memory in response to a START, LOGON, or MOUNT command.		
1 The CSCB is placed on the CSCB chain.	IEE0803D	
2 The global dispatcher lock is used to serialize the ASVT alterations. Page fixing services prevents page faults while holding the lock. The format of the LSPL (local service priority list) appears below:	IEAVEMRQ*	MRQFIXP

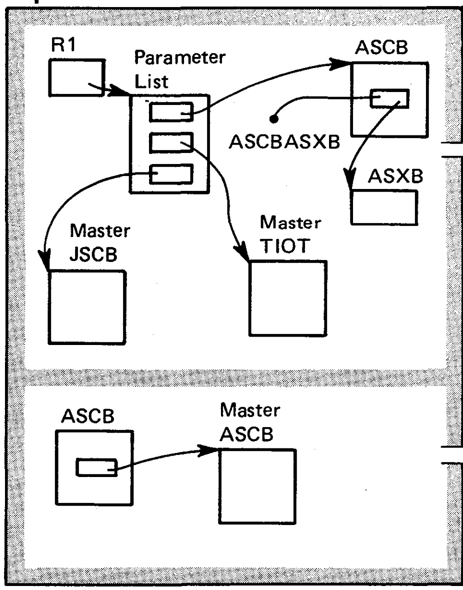
↑First non-quiescable SRB	4
↑Last non-quiescable SRB	4
↑First system SRB	4
↑Last system SRB	4

Extended Description	Module	Label
3 The routine uses the SYSEVENT MEMCREAT macro instruction (SVC 95) to inform SRM that a new memory is being created. Control goes to module IRARMINT.	IEAVEMRQ	
4 Transfer from caller's memory to master scheduler's memory.	IEE0803D	
5 Give control to Virtual Storage Memory (VSM) routine (IEAVGCAS) to create an address space. Assign LSQA storage for ASXB.	IEAVEMCR*	
6 Use ASCBCHAP subroutine to enqueue the ASCB on the ready queue.	IEAVEMER	
7 The SRB for memory initialization is scheduled for the Service Priority List (SPL) of the ASCB. It will operate in the new memory without locks. The SRB's storage area comes from SQA via a GETMAIN macro instruction.	IEAVEMCR	
8 The routine uses a FREEMAIN macro instruction for deleting the SRB.	IEAVEMCR	

*Error routine for IEAVEMRQ is at MRQFRR and MRQESTAE. Error routine for IEAVEMCR is at MCRESTAE.

Diagram 2-10. Obtaining a New Virtual Memory (Part 3 of 4)

Input



Process

9 Initialize the ASXB, TCB (for the region control task), and work areas.

10 Inform SRM that new memory may be swapped if needed.

11 Post the memory creation function.

Dispatcher for Region Control Task (IEAV0003D)

Output

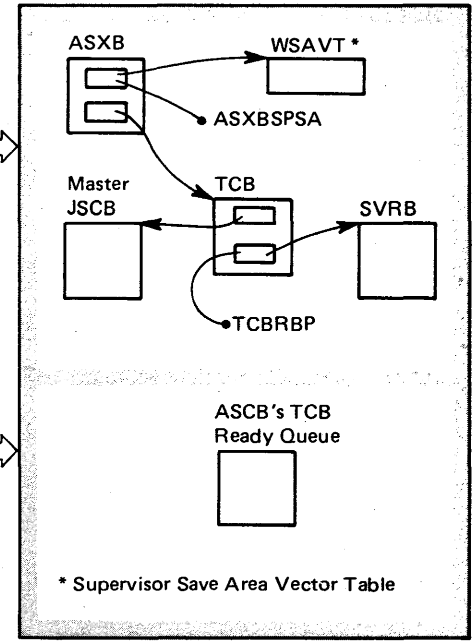


Diagram 2-10. Obtaining a New Virtual Memory (Part 4 of 4)

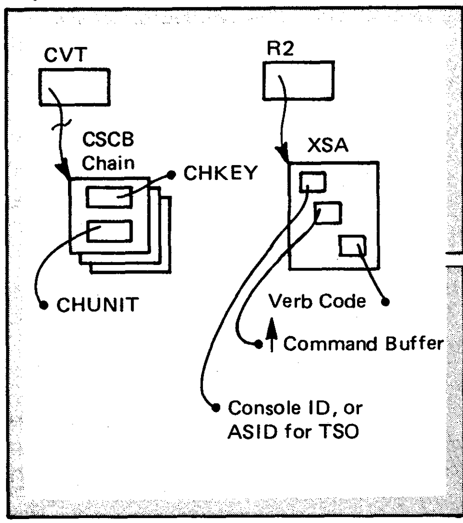
Extended Description	Module	Label
9 Place a dispatchable TCB/SVRB on the ASCB's ready queue.* The work area is obtained from LSQA.	IEAVEMIN	
10 Issue SYSEVENT OKSWAP macro instruction to allow the initialized address space to be swapped. At the time the SYSEVENT MEMCREAT is issued (step 3), the new memory is uninitialized and is marked unswappable to prevent the SRM from scheduling SRBs to the new memory.	IEAVEMIN	
11 A cross-memory post to the memory create routine indicates that a memory is ready. (Control goes to module IEA0PT01 to do the posting.	IEAVEMIN	

Note: If the memory creation processing fails, the routine IEAVEMCR uses either a WTO macro instruction to inform the operator that the START, MOUNT, or LOGON command failed, or a TPUT macro instruction to inform a terminal user that a LOGON request failed. Then the memory create routine posts the memory termination controller to clean up the partially-created memory and exits to the caller.

*Until this new memory is initialized, a lock to serialize the use of resources is unnecessary since another task is unable to execute in the memory.

Diagram 2-11. Cancelling (C) Background (Batch) and Foreground (TSO) Jobs (IEE3703D) (Part 1 of 2)

Input



Command Router (IEE0403D)

Process

1 Check operands and syntax of command.

Error: → Message Routine IEE0503D

2 Match input command job name with those on the CSCB chain.

Error: → Message Routine IEE0503D

3 Obtain the ASCB for the current memory.

4 Cancel the specified job.

5 Release control of resources.

6 Put out message.

→ Message Routine (IEE0503D)

Output

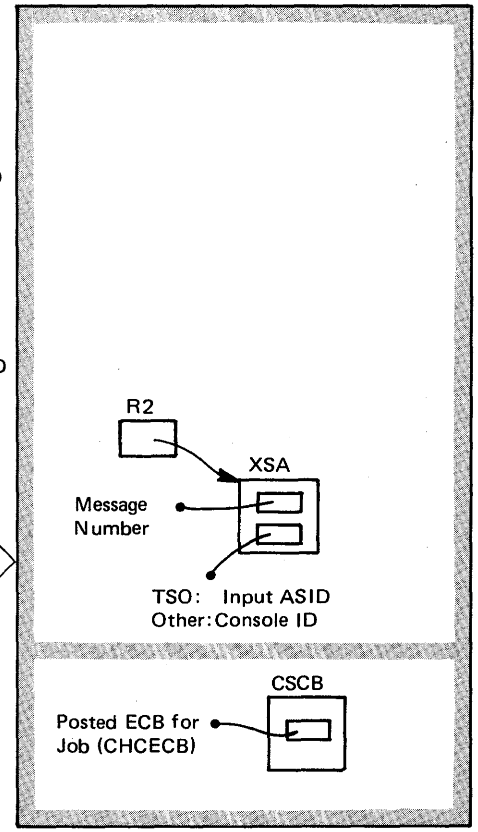


Diagram 2-11. Cancelling (C) Background (Batch) and Foreground (TSO) Jobs (IEE3703D) (Part 2 of 2)

Extended Description	Module	Label
This process is used to terminate background and foreground jobs that are currently executing.		
1 The job name or a device identifier must be supplied by the calling routine.	IEE3703D	IEE3703D
2 A SYSEVENT macro instruction directs the system resources manager (SRM) to make the current memory unswappable. The CSCB chain search requires ENQ-DEQ protection during its process of determining the existence of a job.	IEE3703D	CMSCAN
3 If there is no match, the task is not active and a message is issued.		POST
4 Set cross-memory services locks and local locks. For TSO jobs, the system-initiated cancel routine receives control. Module IKJEFLF schedules the SRB routine IKJL4T00 to handle operator or line-disconnect cancellations.	IKJEFLF	
For background jobs, this routine cross-memory posts the initiator ECB for the job specified in the command. For all jobs, a SYSEVENT macro instruction directs the System Resources Manager (SRM) to swap-in the address space being canceled.	IEE3703D	DOPOST
5 The CSCB resource is released and memory swapping is again permitted.	IEE3703D	CMBRCH
6 Issue message indicating that the CANCEL command function was accepted by the system.	IEE0503D	

Diagram 2-12. System-Initiated Cancelling (C) of TSO Users (IKJEFLF and IKJL4T00) (Part 1 of 4)

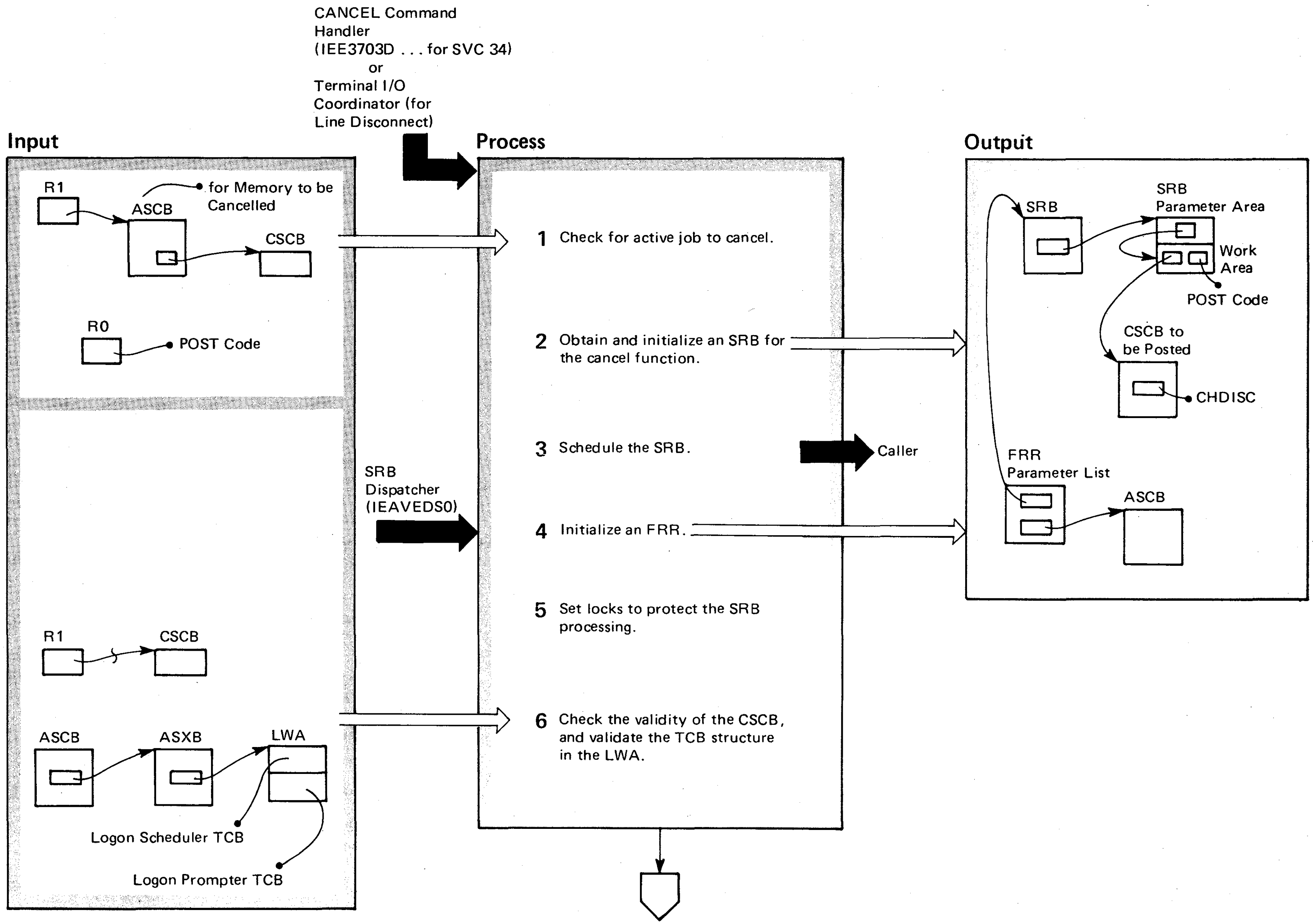


Diagram 2-12. System-Initiated Cancelling (C) of TSO Users (IKJEFLF and IKJL4T00) (Part 2 of 4)

Extended Description	Module	Label
This process provides for an orderly cancellation of a TSO user. It includes the scheduling of an SRB routine to synchronize events between the TIOC (terminal I/O coordinator) routines and the logon scheduler.		
1 The routine verifies that the job's CSCB is active and is for a time-sharing user.	IKJEFLF	
2 Storage for the SRB is obtained from subpool 239. At the same time, the routine also gets storage for a work area. This work storage contains information to be used by the SRB (supervisor request block) processing routine, IKJL4T00.		
3 This step places the SRB on a queue for use by the SRB dispatching routine. It is done by using a SCHEDULE macro instruction.		
Note: The routine that calls the SRB scheduling routine must be in PSW key zero in the supervisor state.		
4 A functional recovery routine (FRR) and an RMTR (recovery management termination routine) protects the SRB function and gets control if the SRB routine fails.	IKJL4T00	ENTRYSRB
5 Local and CMS locks are used to protect the processing.		GETLOCK
6 Ensure that the ASCBCSCB pointer field is not zero. A non-zero TCB address (in the LWA) for the logon scheduler must exist.		

Diagram 2-12. System-Initiated Cancelling (C) of TSO Users (IKJEFLF and IKJL4T00) (Part 3 of 4)

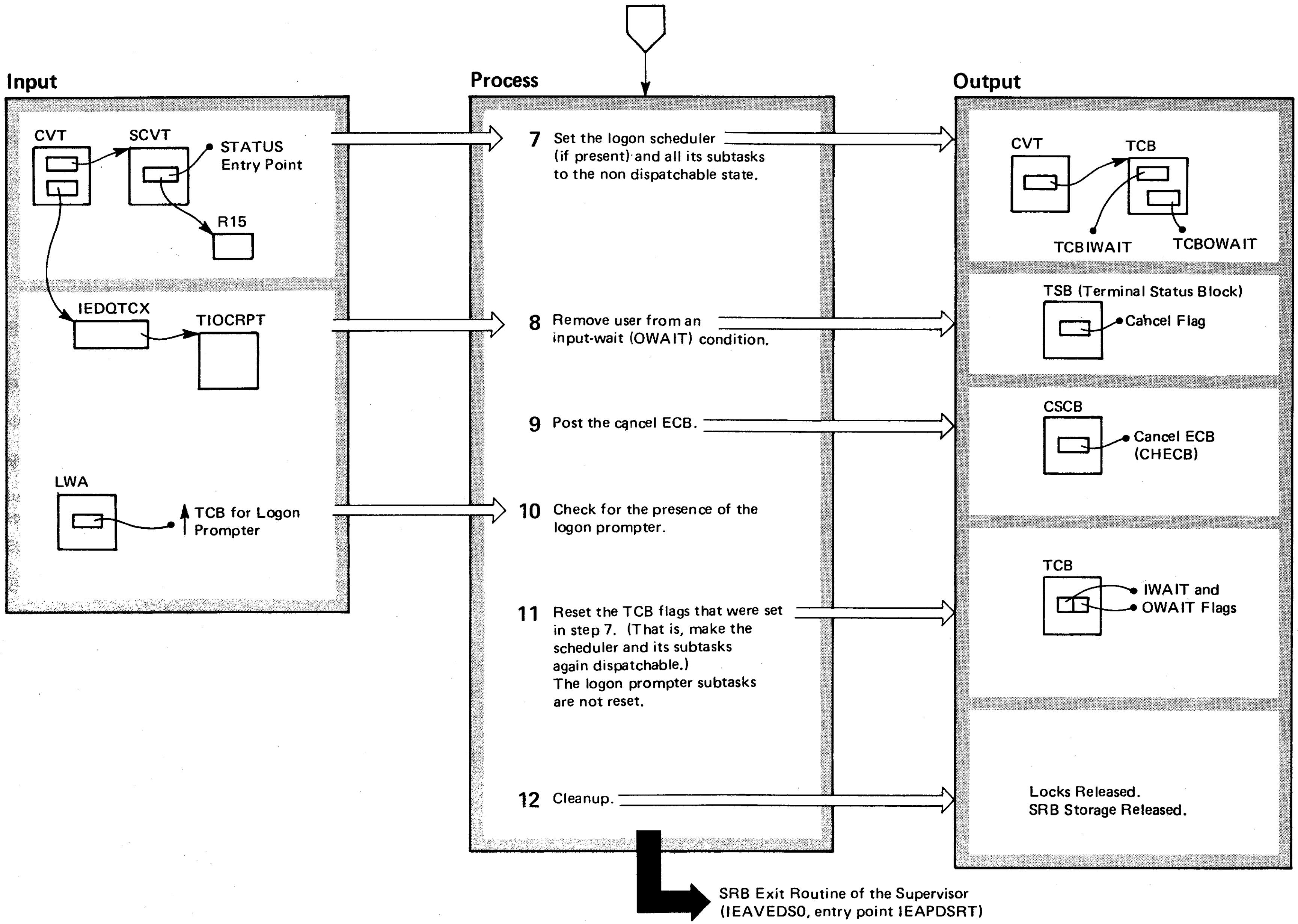


Diagram 2-12. System-Initiated Cancelling (C) of TSO Users (IKJEFLF and IKJL4T00) (Part 4 of 4)

Extended Description	Module	Label
7 Turn on non-dispatchability flags to preclude terminal I/O activity. This will permit the scheduler to respond properly to the posting of the cancel ECB. A branch entry to the STATUS function (see SVC 79) is used since the SRB routine is uninterruptable via an SVC.	IKJL4T00	
8 If the ASCBCSCB field is zero and if either there is no LWA or the TCB pointer in the LWA is zero, the QTIP subroutine (see SVC 101) receives control via a branch entry. This subroutine prevents the endless 'wait' of a task that has done a TPUT to the user terminal being cancelled; (that is, it removes the task from the "O-wait" condition).		QTIPRTN
9 By having the post function occur under local lock conditions, the posting will not cause the logon scheduler to abend before the QTIP and/or STATUS functions receive control.		
10 The nondispatchability flags are set to zero (off). (See step 7.) The logon prompter must be a subtask of the logon scheduler. The STATUS function receives control via a branch entry to perform this.		
11 This action removes the logon prompter from input/output wait status.		
12 The SRB is released. Locks are removed, and the FRR environment is deleted.		RESTREGS

Diagram 2-13. Changing Dump (CD) Parameters (IEEMB815) (Part 1 of 2)

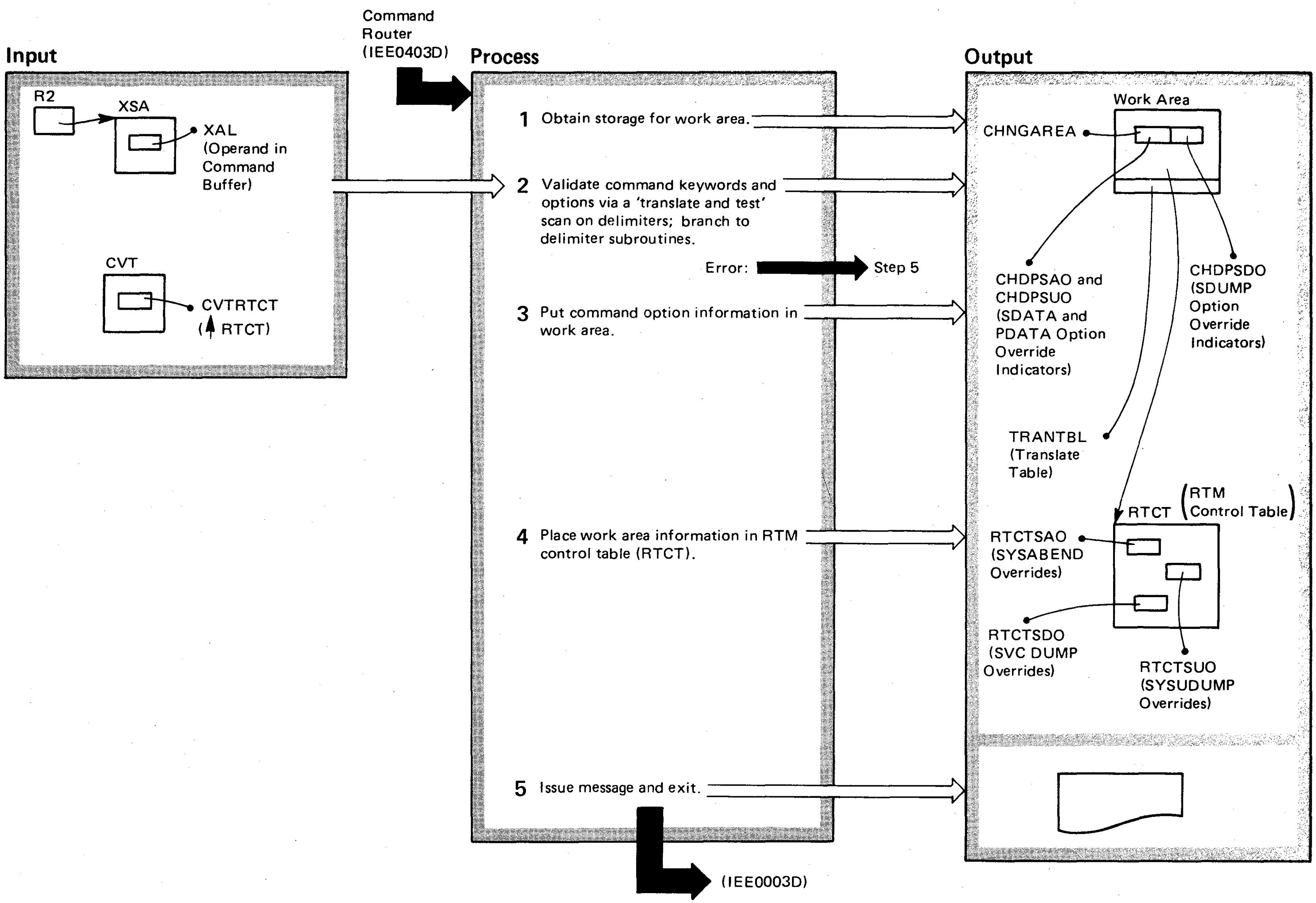
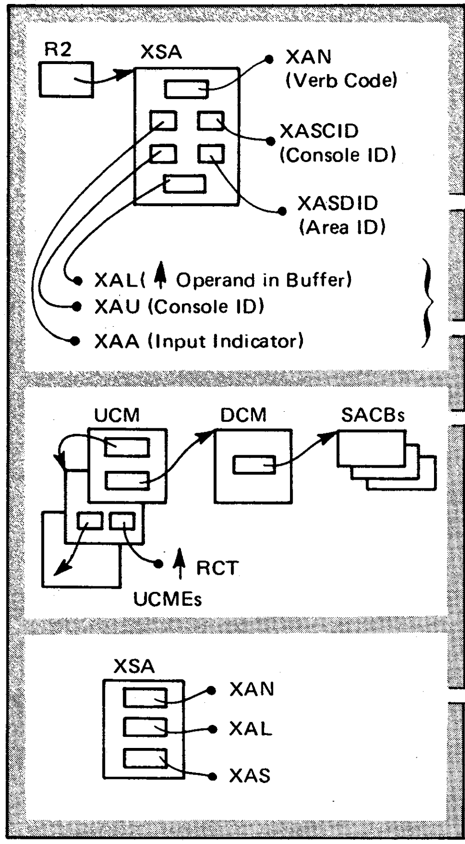


Diagram 2-13. Changing Dump (CD) Parameters (IEEMB815) (Part 2 of 2)

Extended Description	Module	Label
This routine provides either a temporary override of dump options that may exist in a system (via SYS1.PARMLIB or requested on either the ABTERM, CALLRTM or SETRP macro instructions) or a deletion of specified override parameters.		
1 Storage comes from subpool 253 (LSQA).	IEEMB815	CHDINIT
2 If syntax error is encountered, the operand scan terminates and the module puts out an appropriate message.	IEE2103D	CHDCNTRL
3 This information is requested by the command issuer.		
4 The fields of the RTCT are used by ABDUMP processor (IGC0001C) and the SVC DUMP processor (IEECB866).	IEEMB815	CHDCDSS
5 Message indicates that the command has been accepted.	IEE2103D	

Diagram 2-14. CONTROL (K) Command Processing (IEE6703D) K Command (Part 1 of 10)

Input



Command Router IEE0403D

Process

1. Save registers for JES2.
2. Set local and CMS locks.
3. If requester is JES2, go to Step 9
4. Determine if command is not executing under COMTASK. If not, indicate in XSA.
5. Check the "L" operand, and remove from buffer.
6. If routing parameter is missing, obtain default values.
7. Check for and, if found, validate the console ID and target display area.
8. Determine console authority, activity, and status conditions.
9. Check validity of JES2's screen area request.
10. Check validity of first operand in buffer.

Output

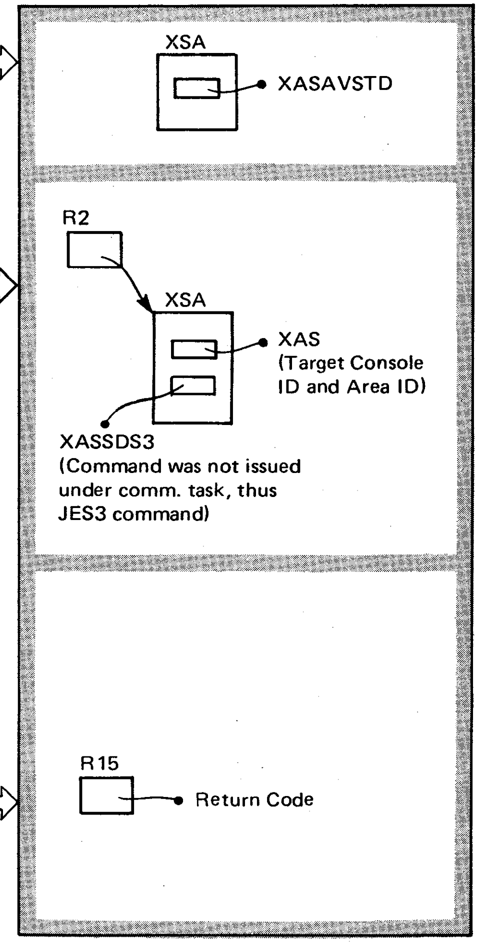


Diagram 2-14. CONTROL (K) Command Processing (IEE6703D) K Command (Part 2 of 10)

Extended Description	Module	Label
This routine processes the various operands of the CONTROL command. It also checks the source input validity and the validity of the target console selected.		
1 This entry point (IEE7603D) and input is for JES2 processing only. The entry point for non-JES2 commands is IEE7503D.	IEE7503D IEE7503D	IEE7603D IEE7503D
2 Serializes the UCMEs and SACB.		LOCKSET
3 For JES2 commands, the validity of the request requires checking.		
4 If a command is not executing under COMTASK, set XASSDS3 bit on. (e.g., K commands issued on a JES3 associated console.)		
5 If the operand is invalid, default values are placed in the XSA (field XAS). (The "L" operand is invalid on CONTROL commands other than those listed in steps 11 and 13.)		
6 Default values may come from the routing control table (RCT) or the issuing console. For further details concerning steps 5-7, see steps 4-6 of the diagram, Stopping Periodic Track (Status) Displays.		
7 The XAA field must indicate (by a value of zero) other than a time-sharing terminal input.		LFOUND NOL
8 A target CRT (cathode ray tube) console must be active and have a defined screen area control block (SACB).		VALIDITY
9 Determine if console is valid and active and has an area available to receive status displays.		JES2CODE
10 Except for the command K C,D, all the K command targets must be graphics consoles.	IEE6703D	GRAPHICS

Diagram 2-14. CONTROL (K) Command Processing (IEE6703D) K Command (Part 3 of 10)

Input

XSA
XAL

XSA
XAL

Process

11 Determine operand and branch accordingly;

For K C,D Go to step 15.
For K V Go to step 18.
For K T Go to step 23.
For K A Go to step 28.
For K A,ℓℓ Go to step 31.
For K A,NONE Go to step 35.

12 Check validity of the operands D,H; D,U; D,F.

13 For K D, U, post the TRACK command processor. (See note in extended description.)

14 Release serialization locks.

15 Post the DIDOCS routines. (See note in extended description.)

Command Router (IEE0003D)

Output

SACB
DCMATECB

UCM
UCMECB

RDCM
KPARM
Routed Command Parameter List

Extended Description	Module	Label
11 Depending on the command, the branch instructions occur at the labels shown.	IEE6703D	
K C,D		D7803D
K V		D7703D
K T; K A; K A, NONE		D6903D
K A, ℓℓ, ℓℓ		D6803D
(These commands are explained in the Display Consoles Manual.)		
All K commands that are routable with the L=CCA operand are valid on a JES3-associated console, except K V,USE.		
12 Each valid operand must have the correct syntax.		SETFLG SETFLGHD MODSET
13 See step 10 of the diagram, Stopping Periodic Track (Status) Displays.	IEE6703D	POSTRACK
14 Local and CMS locks are released.		LOCKSOFF
15 See step 12 of the diagram, Stopping Periodic Track (Status) Displays. The parameter list KPARM is a communication list between DIDOCS routines and the CONTROL command processor.		POST

Note: (for steps 12 and 14): If the command is not executing under COMTASK, the post will be a cross-memory post.

Diagram 2-14. CONTROL (K) Command Processing (IEE7803D) K Command (Part 5 of 10)

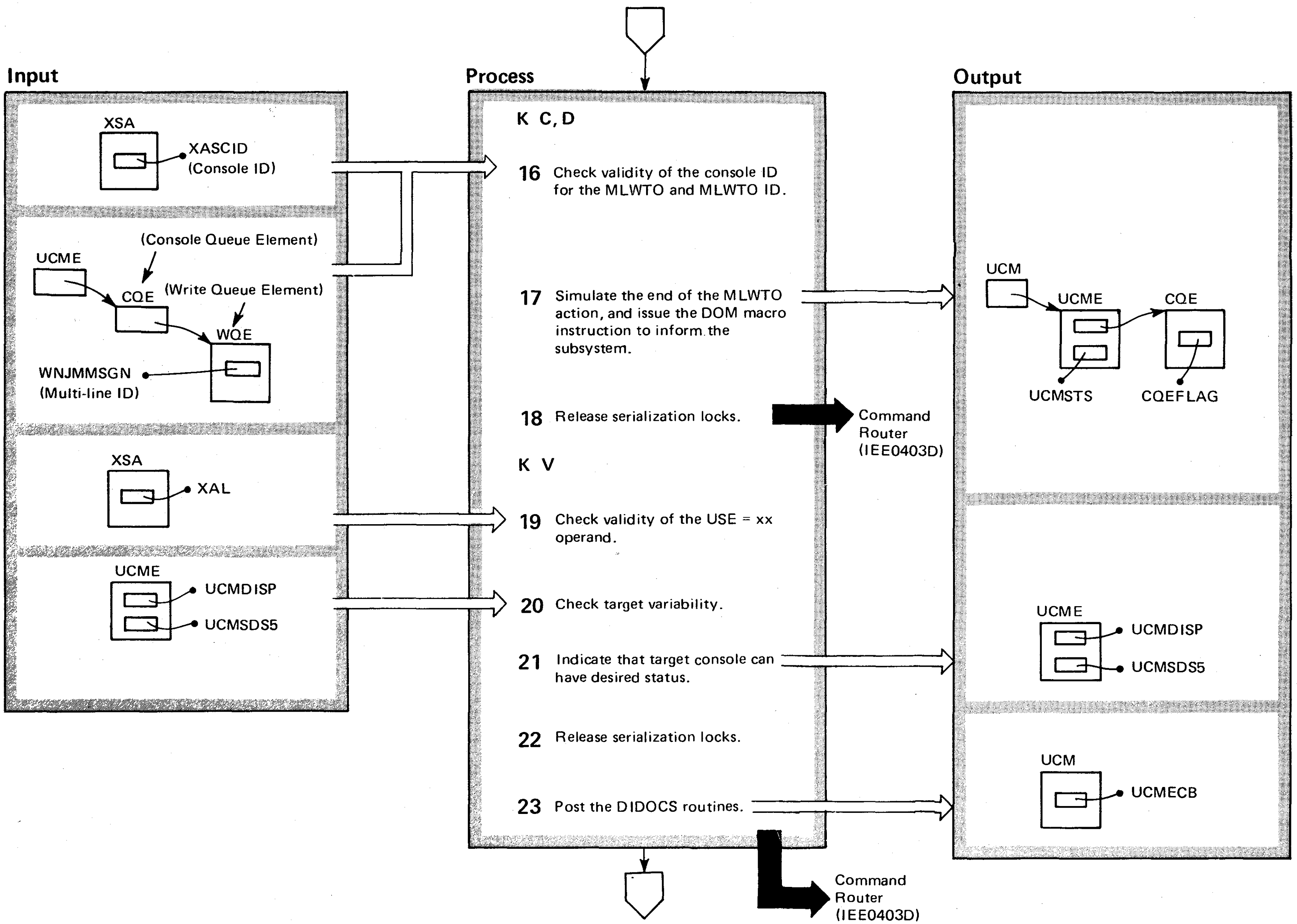


Diagram 2-14. CONTROL (K) Command Processing (IEE7803D) K Command (Part 6 of 10)

Extended Description	Module	Label
K C,D		
16 This is the ID operand on the K C,D,ID command. This K command is the only one valid for a target paper-output console.	IEE7803D	KCORTN
17 Routine reduces the use count of the WQE chain.		SETSUBQ
18 Local and CMS locks are released.	IEE7803D	ENDSRCH
K V		
19 Operands must be K V,USE=xxb̸	IEE7703D	VALID
20 The ability to vary the target console as specified must exist.		
21 Possibilities are for console to have full capability (FC), message status (MS) capability, or status display (SD) capability.		
22 Local and CMS locks are released.		LOCKSOFF
23 DIDOCS routines handle the remaining processing for the operand "V".		SETPOST

Diagram 2-14. CONTROL (K) Command Processing (IEE6903D) K Command (Part 7 of 10)

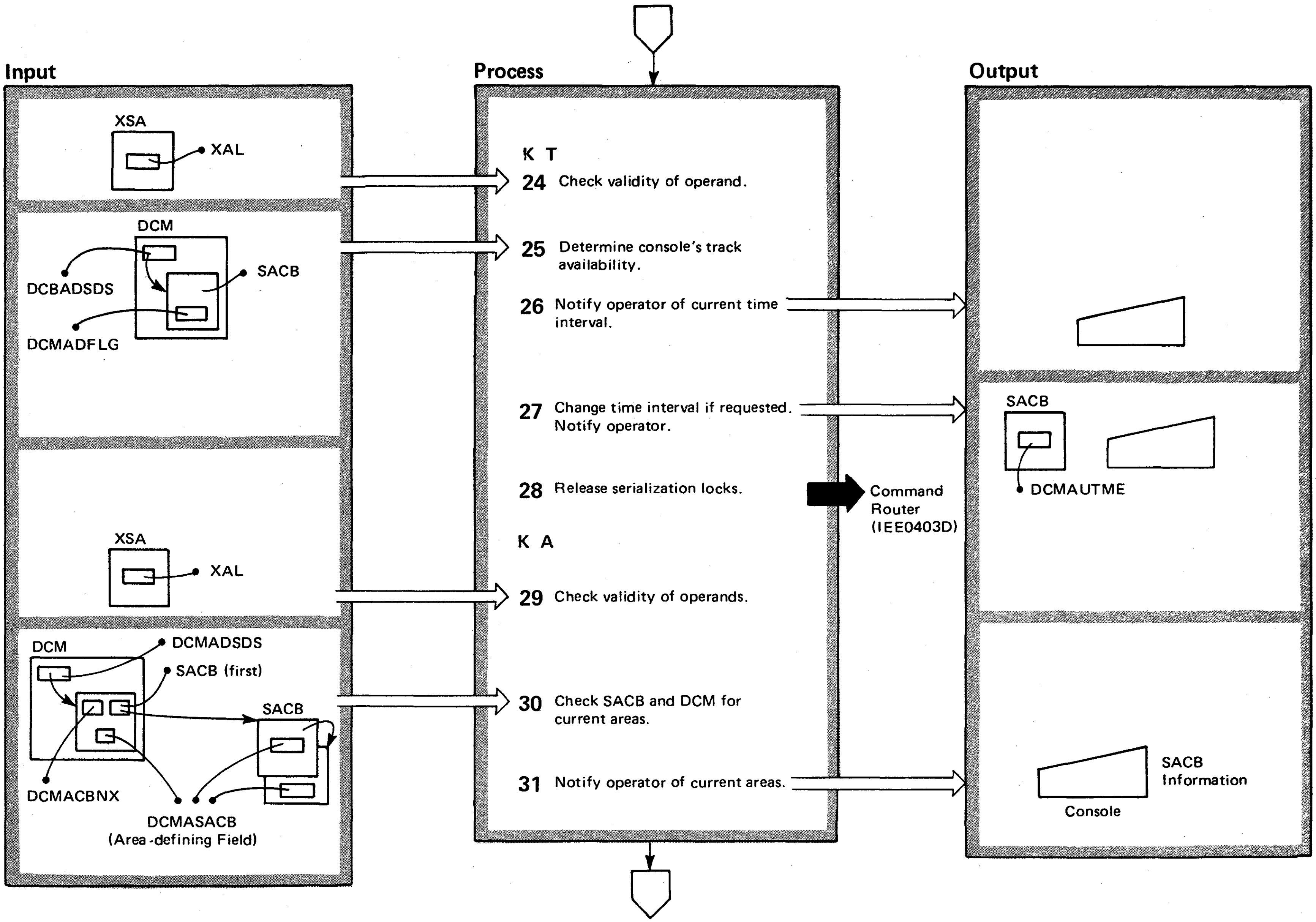


Diagram 2-14. CONTROL (K) Command Processing (IEE6903D) K Command (Part 8 of 10)

Extended Description	Module	Label
K T		
24 The routine checks for the correct operand syntax for the K A command.	IEE6903D	
25 A TR command must be active on the console for the K T command to be valid.		
26 This applies to the K T or K T,REF command. The routine uses the WTO macro instruction.	IEE6903D	KTREFRTN
27 Use the specification in the command K T,UTME. Routine uses the WTO macro instruction.		KUTMERTN
28 Local and CMS locks are released.		FREELOCK
K A		
29 The commands K A and K A,REF are equivalent. These commands are valid only if the target console is a graphics device.	IEE6903D	
30 These are areas defined for the target CRT either via the command K A, $\ell\ell$, $\ell\ell$ or at SYSGEN time for the SYSGEN SACB. The first SACB is contained in the DCM.		MOVEINKA
31 Routine uses a WTO to notify the operator.		KREFRTN

Diagram 2-14. CONTROL (K) Command Processing (IEE6903D) K Command (Part 9 of 10)

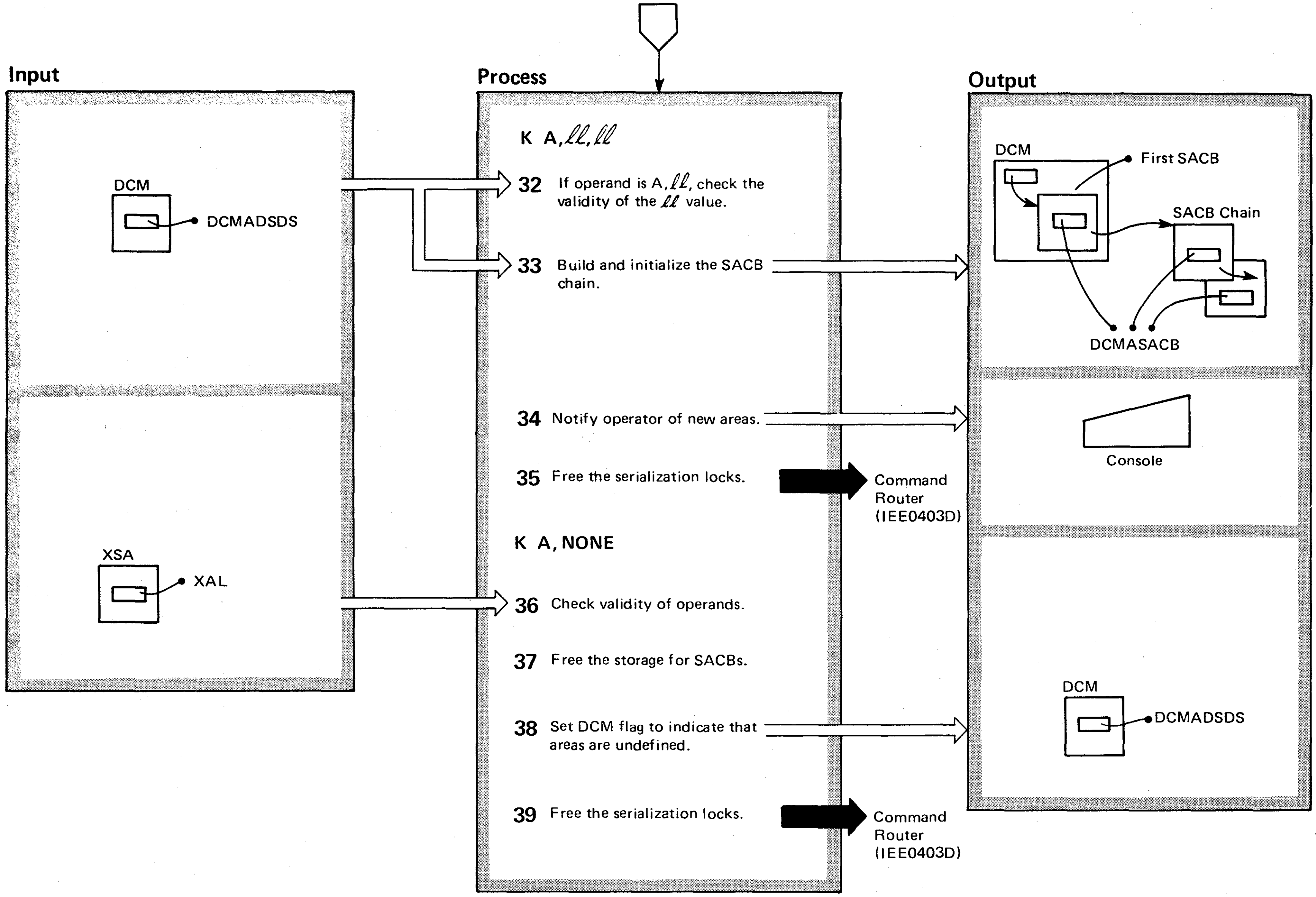


Diagram 2-14. CONTROL (K) Command Processing (IEE6903D) K Command (Part 10 of 10)

Extended Description	Module	Label
K A, $\ell\ell$, $\ell\ell$		
32 The operand must be greater than or equal to 4 and less than the screen size (total number of lines available).	IEE6803D	LLLOOP
33 There is 1 SACB per screen area as defined by the $\ell\ell$ operand. If DCMADSDS is 0, there currently exist no SACBs.		ENDLL
34 Routine uses the WTO macro instruction.		
35 Local and CMS locks are released.	IEE6803D	
K A,NONE		
36 This is the first procedure of the K A, NONE Command processing.	IEE6903D	
37 Screen areas must be inactive in order to free the SACBs. The routine uses a FREEMAIN macro instruction.		KNONERTN
38 Screen areas are defined in the SACBs.		FREESACB
39 Local and CMS locks are released.		FREELOCK

Diagram 2-15. DISPLAY (D) and TRACK (TR) Command Preprocessing (IEE3503D and IEE7503D) (Part 1 of 8)

Command Router (IEE0403D)

Input

R2

XSA

XAN (Command Verb)

XAA (ASID Indicator)

XAU (UCM Indicator)

XAL (↑ Command Operand)

XSA

XAS

Process

1 Check first operand for a valid value.

Error → Step 19

Process according to operand:

- T (TIME)

2 Issue TIME macro instruction.

Error → Step 19

a) Issue message to console or time sharing terminal. → SVC 34 (IEE0003D)

- TP (TCAM) or NET (VTAM)

3 Give control to appropriate processing routine. → TCAM Display (IED1303D or ISTCFF3D)

- PFK (Program Function Keys)

4 Set routing information field to zero.

Output

Message

TIME=
DATE=

R2

XSA

XAS

Diagram 2-15. DISPLAY (D) and TRACK (TR) Command Preprocessing (IEE3503D and IEE7503D) (Part 2 of 8)

Extended Description	Module	Label
This routine performs the preliminary checking and initialization needed by the routines that actually put out the requested display information.		
1 If processing a TRACK command, go to step 6 after this step.	IEE3503D	DISPLAY TRACK
2 Either a TPUT (for a terminal) or a WTO (for a console) macro instruction puts out a time message.		DDTIME
3 Either module IED1303D (for a TCAM display) or module ISTCFF3D (VTAM command processor for D NET) receives control to continue the processing.		DDTP DDNET
4 The routine changes the verb code (XAN) and gives control to the CSCB creation routine IEE0803D.		DDPFK

Diagram 2-15. DISPLAY (D) and TRACK (TR) Command Preprocessing (IEE3503D and IEE7503D) (Part 3 of 8)

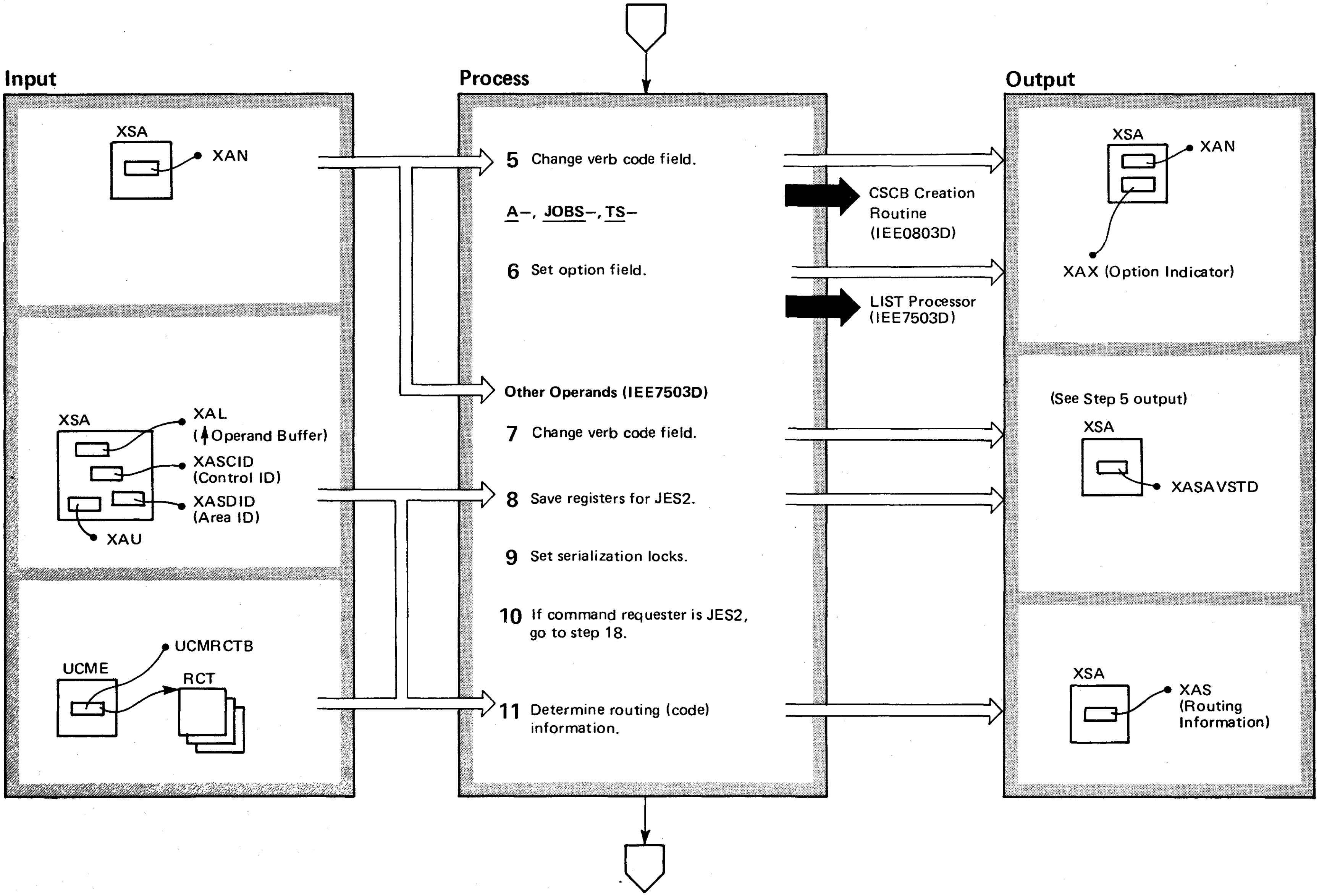


Diagram 2-15. DISPLAY (D) and TRACK (TR) Command Preprocessing (IEE3503D and IEE7503D) (Part 4 of 8)

Extended Description	Module	Label
5 For each operand, the routine assigns a unique value to the field.	IEE3503D	
6 The option field bits indicate the requested operands for the appropriate command.		DACTIVE
7 The verb code, XAN, is set to indicate the appropriate command operand.		
8 This step and input applies only to JES2 command requests.	IEE7503D	IEE7603D
9 The CMS and local locks serialize the use of the UCMEs.		LOCKSET
10 Non-JES2 command processing is bypassed.		
11 The routine checks for a L=cca operand. If this operand is non-existent (that is, not coded), the routine checks for a routing control table (RCT) set up by		NOTINTEN
a previous MSGRT command. If an RCT does not exist, the display information is sent to the console issuing the command. Internally issued commands are routed to the master console.		NOCOMP

Diagram 2-15. DISPLAY (D) and TRACK (TR) Command Preprocessing (IEE3503D and IEE7503D) (Part 5 of 8)

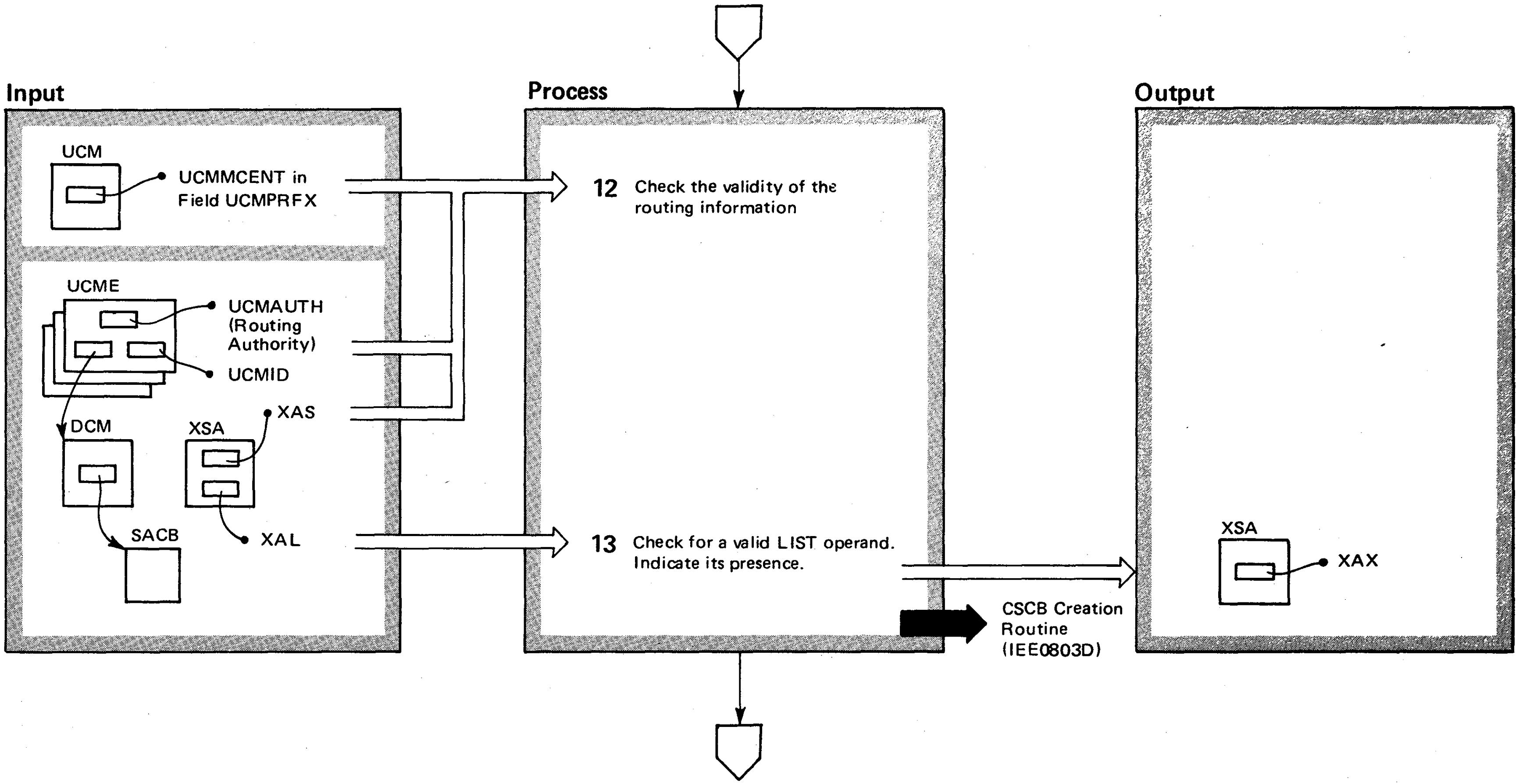


Diagram 2-15. DISPLAY (D) and TRACK (TR) Command Preprocessing (IEE3503D and IEE7503D) (Part 6 of 8)

Extended Description	Module	Label
12 For example, the "cc" value in the L=cca operand of the command must be equal to or less than 99 in order to be valid.	IEE7503D	VALIDITY
System commands cannot be routed to a JES3 console with the L=CCA operand (UCMDISPK bit is on).		DACTIVE
13 This information will be used by the CSCB creation routines.		
The check is made only for the operands TS, A, and JOBS. The "L" operand is removed from the buffer (pointed to by field XAL) after it is checked.		
Note: The TRACK command is invalid on a JES3 console. (UCMDISPK bit is on.)		

Diagram 2-15. DISPLAY (D) and TRACK (TR) Command Preprocessing (IEE3503D and IEE7503D) (Part 7 of 8)

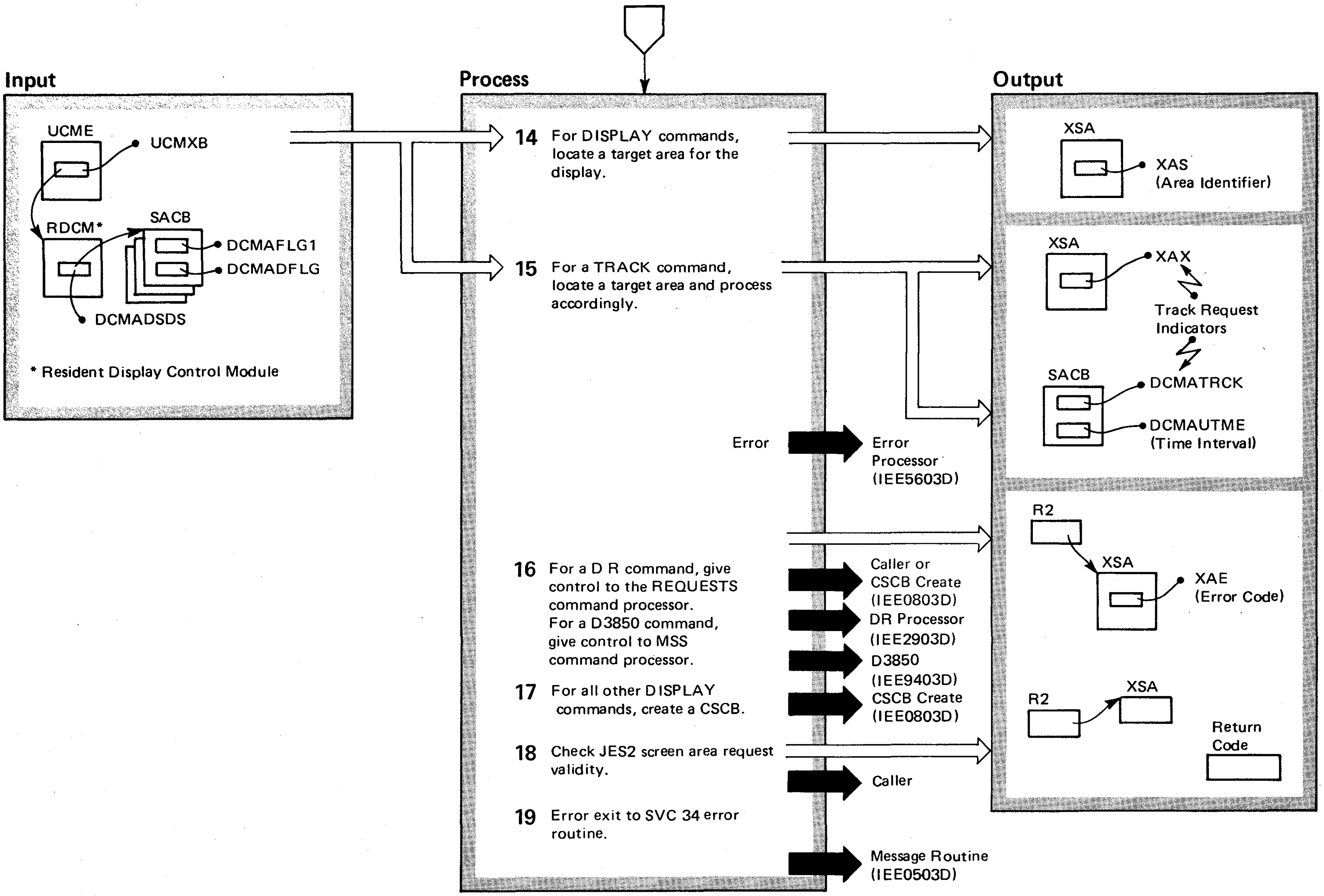
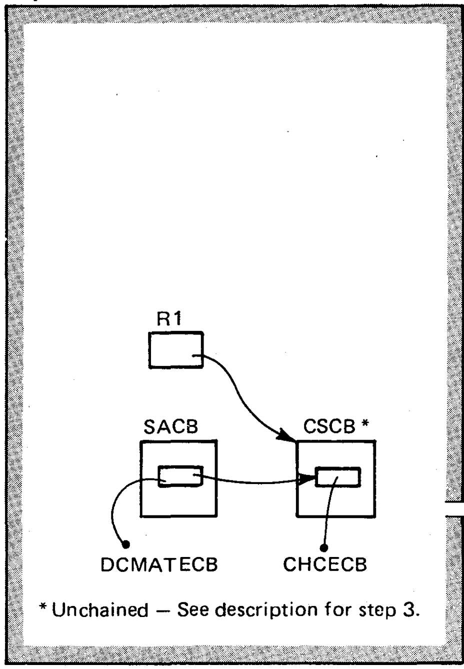


Diagram 2-15. DISPLAY (D) and TRACK (TR) Command Preprocessing (IEE3503D and IEE7503D) (Part 8 of 8)

Extended Description	Module	Label
14 An existing dynamic (TRACK command) display will not be overlaid. If the routine is unable to find an area for a non-status display console, it defaults the message to an inline display.	IEE7503D	DISPLAY
15 If there is a target area (as found by searching the SACB chain), the routine updates the track entry and returns control to the caller.		TRACK
If a target area is missing or undefined, the command being processed is the first one for the given console. The CSCB creation routine (IEE0803D) then receives control to build a CSCB so the master scheduler wait routine (IEEVWAIT) can attach the processor for this console.		
16 The routine changes the verb code, XAN, and relinquishes control.		D0803D
17 For all other commands (that is, D M; D U; D C,K; D CONSOLES; and D A), the routine makes a determination and, if necessary, an assignment of, a display area before giving control to the CSCB creation module.		
18 The routine determines the availability of a valid, active console with a free area that can receive status displays. For return codes from this step, see the Diagram, Stopping Periodic Track (Status) Displays.		JES2CODE
19 For invalid commands or parameters, the error routine receives control.		

Diagram 2-16. Displaying (D A) and Tracking (TR A) System Status (IEECB800) (Part 1 of 6)

Input



Master Scheduler Wait Routine (IEEVWAIT)

Process

1 Create an ESTAE environment.

For DISPLAY commands, go to step 10. → Step 10

For TRACK commands, continue.

2 Issue another ESTAE for the TRACK command processing.

3 Get local and CMS locks.

4 Determine if the TRACK command processor's ECB has been posted (signifying track termination) by a console-closing function.

a) Yes: Release local and CMS locks. → Step 16

No

5 Determine if TRACK processor has been posted by a STOPTR command.

a) Yes: Turn off track flags in the SACB. Release all locks. → Step 16

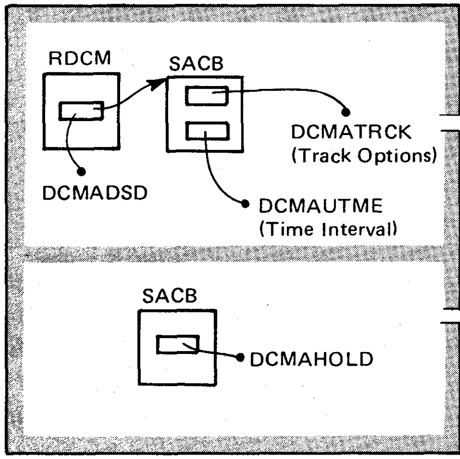
No

Diagram 2-16. Displaying (D A) and Tracking (TR A) System Status (IEECB800) (Part 2 of 6)

Extended Description	Module	Label
This routine provides for a display of system status information about active tasks and jobs and time sharing terminals. TRACK command requests appear on a graphics (screen) device. DISPLAY command requests may appear on either a graphics or a paper-output console device.		
1 This environment protects the routine against ABEND situations.	IEECB860	
2 This second ESTAE handles the cleanup of the interface between the TRACK command and DIDOCS routines.		
Note: The TRACK command is invalid from an input stream and from any time-sharing terminal. It is valid only for CRT devices in status-display mode or in full-capability mode.		
3 This permits serialization of the SACB and the ECB in the TRACK command's CSCB, which module IEEVWAIT has removed from the CSCB chain.	IEECB800	AWAKE
4 Console errors and a VARY OFFLINE command applied to a target console may cause this.		ECBPOST
5 If the track processor is posted, the track functions for the target (specified) console will stop.		
a. This informs DIDOCS routines that the console screen should be cleared.		

Diagram 2-16. Displaying (D A) and Tracking (TR A) System Status (IEECB800) (Part 3 of 6)

Input



Process

6 Put current track options and time interval in CSCB.

7 Determine if display is currently being held.

a) No → Step 9

b) Yes

8 Release serialization locks. Set time interval. Wait on the STIMER ECB and the TRACK ECB.

9 Release serialization locks and set the time interval as in step eight.

10 Get storage for message buffers.

Output

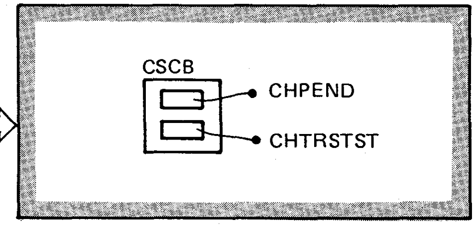


Diagram 2-16. Displaying (D A) and Tracking (TR A) System Status (IEECB800) (Part 4 of 6)

Extended Description	Module	Label
6 Track options are changed by TRACK or STOPTR commands. Time intervals are changed by a "CONTROL T,UTME=" Command.	IEECB800	
7 A "CONTROL D,H" command would cause the display to be held. A "CONTROL D,U" command causes the display to be updated.		HOLDMODE
8 The routine uses the STIMER macro instruction for the interval listed in step six.		
9 The routine again uses the STIMER macro instruction.		
10 This storage will contain the blocks of 10 lines to be used for the MLWTO.		DISPLAY

Diagram 2-16. Displaying (D A) and Tracking (TR A) System Status (IEECB800) (Part 5 of 6)

Input

CSCB
- CHTRSTAT

CSCB Chain
- CHTRKID

CSCB Fields.
- CHKEY (TSID or Step Name)
- CHCLS (Job Name)
- CHRGNAD (Region Address)
- CHRGNSZ (Region Size)

Process

11 Put time value in message.

12 Scan CSCB chain and put "active tasks" count in message(s).

13 Determine the function to be displayed and list the number of active jobs, initiators, and/or TS users in the system.

14 For LIST operand, obtain job names, step names, and region addresses for jobs and tasks, and user ID for TS users.

15 Issue messages and dequeue off the CSCB chain.
For DISPLAY command. → Step 17

16 For TRACK command, issue a WAIT on the STIMER ECB and the TRACK processor ECB.

17 Release the CSCB storage.

→ SVC EXIT (SVC 3)

Output

Message
- Task Counts

Message Header

Message

Message

R1 → Parameter List
- ↑ CSCB
- ↑ Message Buffer

Diagram 2-16. Displaying (D A) and Tracking (TR A) System Status (IEECB800) (Part 6 of 6)

Extended Description	Module	Label
11 A TIME macro instruction provides the time stamp.	IEECB800	
12 Routine enqueues on the CSCB chain before scanning.		
13 The numbers of active tasks for each type are placed in the appropriate subheader sections of the display areas.		
14 The region address (and size) fields apply to V=R jobs.		
15 Uses a MLWTO or a TPUT macro instruction to put out the message, 10 lines at a time.	IEECB801	ECBWAIT
16 The routine uses the MGCR macro instruction to free the CSCB and then returns control to the caller.		
17 Routine uses the system macro instruction, MGCR.	IEECB800	NOTIMER

Diagram 2-17. Displaying Console (D C) Status (IEEXEDNA) (Part 1 of 2)

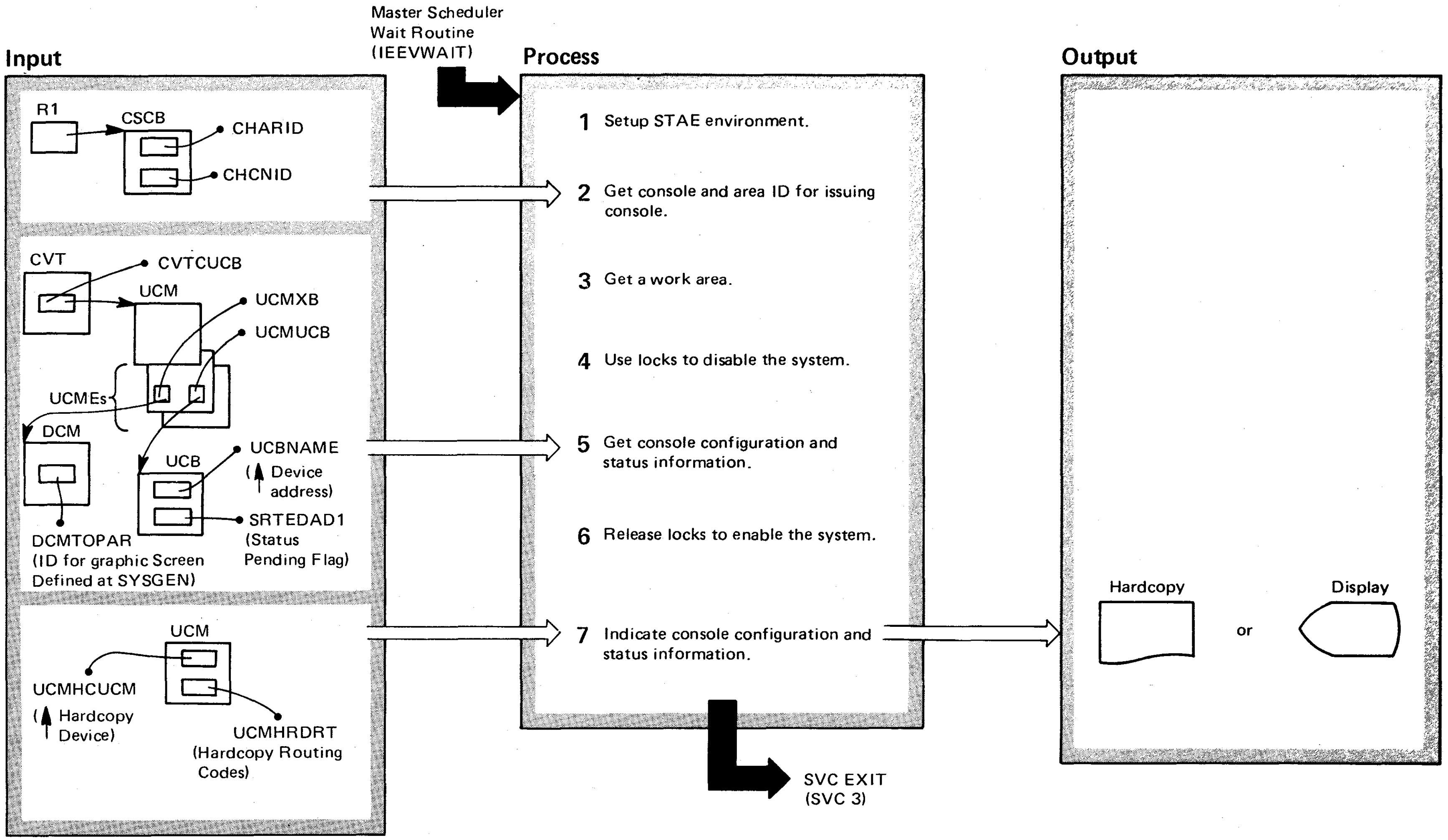
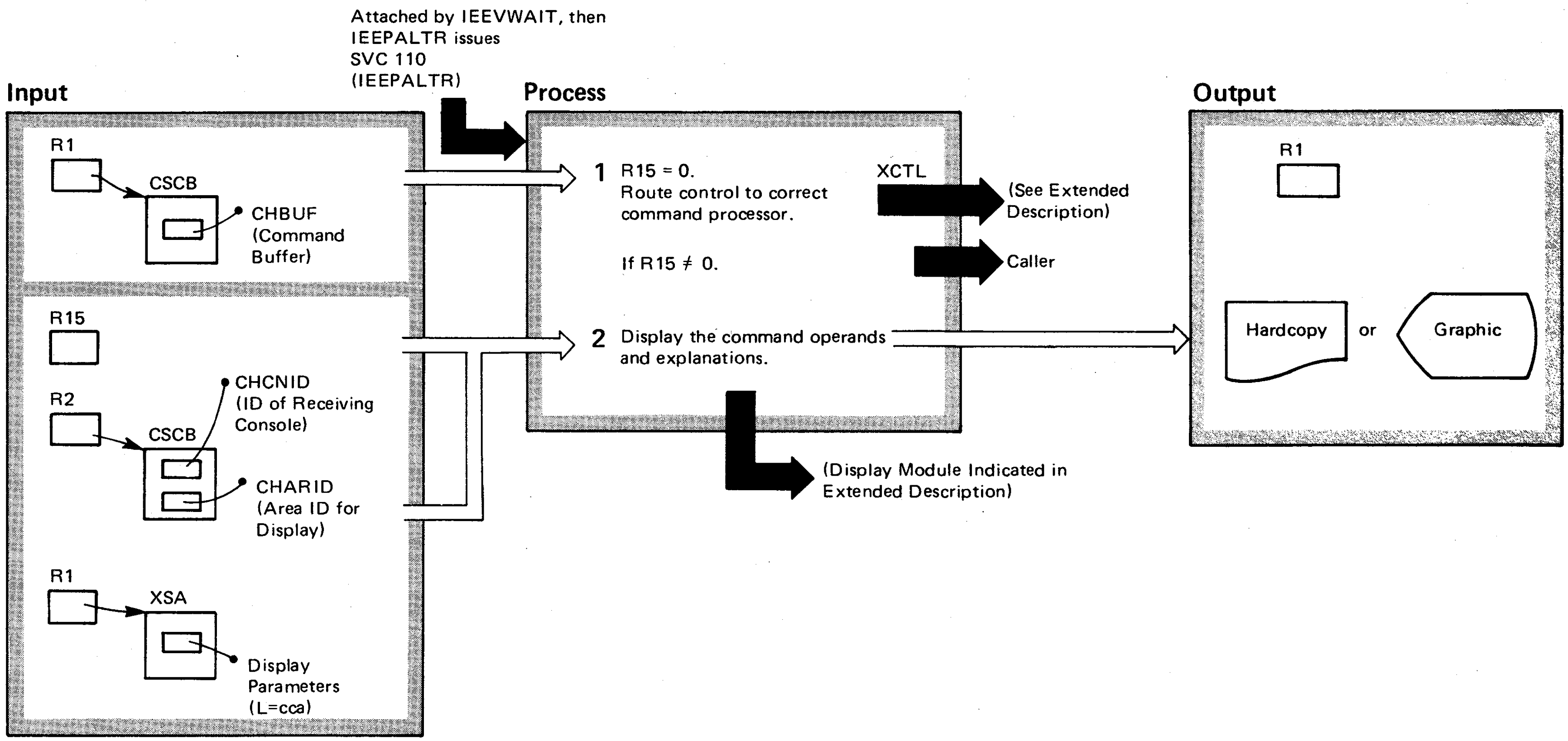


Diagram 2-17. Displaying Console (D C) Status (IEEXEDNA) (Part 2 of 2)

Extended Description	Module	Label
This procedure displays information about console configurations. It builds a display for graphic screen or hardcopy output.		
1 Load and branch to module IEECB860 to set up this routine for handling ABEND situations.	IEEXEDNA	
2 After this step, the routine releases the CSCB via the use of the MGCR macro instruction.		
3 This area will contain the message to the operator and module work space.		
4 The use of the SETLOCK macro instruction to obtain local and CMS locks prevents changes from being made to the UCMEs and UCBs by another user.		SETLOCK
5 This routine uses information in UCME, DCM, and UCB. These blocks contain information about device displays and console characteristics.		GETDATA
6 The routine uses a SETLOCK macro instruction to release the locks.		FREELOCK
7 The routine builds and issues a MLWTO macro instruction(s) to write information to console.		MSGSET

Diagram 2-18. Displaying CONTROL Command Operands (D C, K) (IEE10110) (Part 1 of 2)



* This routine receives control from a command processor or when an SVC 110 is issued.

Diagram 2-18. Displaying CONTROL Command Operands (D C, K) (IEE10110) (Part 2 of 2)

Extended Description	Module	Label
This routine builds and displays selected CONTROL command operands.		
1 Determine information to be displayed. Depending on the command being processed, IEE00110 gives control according to the following:	IEE00110	DISPCNTL

Command	*Module Receiving Control*
D C,K	IGC10110
D U	IGC20110
D PFK	IGC40110

If R15 ≠ 0, IEE0110 first uses the MGCR macro instruction to free the CSCB before returning control to the caller.

2 Issue WTO macro instruction to write out desired information. First module indicated writes lines 1-12. Second module indicated writes lines 13-26. Third module indicated writes lines 27-end of display. (The L=cca operands were previously stored in the XSA by module IEE7503D.)	IEE10110 IEE11110 IEE12110	

Diagram 2-19. Displaying a Matrix (D M) of System Status (IEEMPDM) (Part 1 of 2)

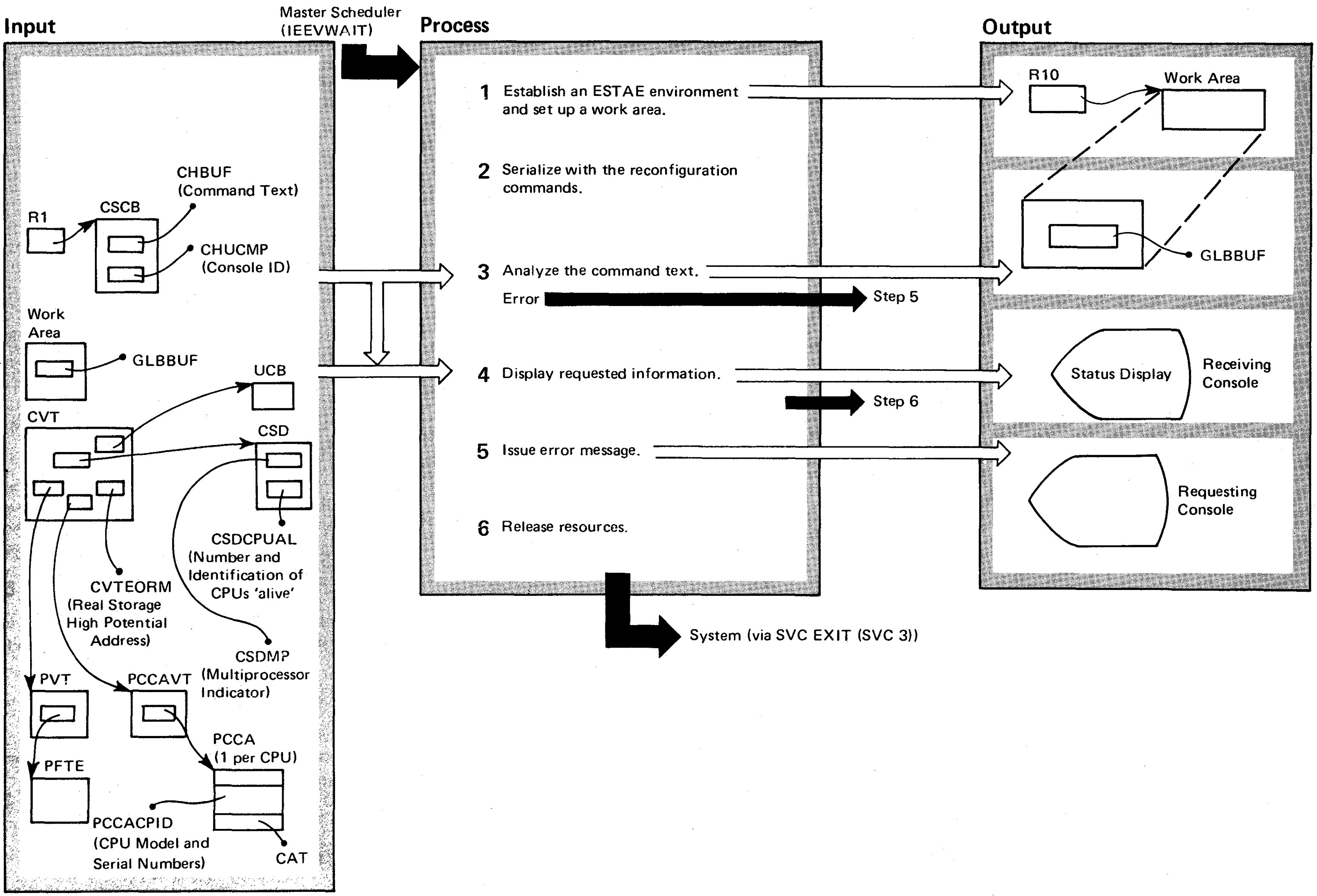


Diagram 2-19. Displaying a Matrix (D M) of System Status (IEEMPDM) (Part 2 of 2)

Extended Description	Module	Label
This routine produces a display of system status to the operator's console. It displays information such as the status of CPUs, channel(s), paths, and real storage.		
1 The ESTAE routine (ESTAERTN) handles abnormal end situations.	IEEMPDM	
The work area contains a function mask that has flags for each requested component to be displayed.		
2 By serializing the reconfiguration commands (through use of the ENQ macro instruction for the SYSZVARY, CPU resource), other reconfiguration commands are prevented from executing while the current command is in control.		
3 The individual operands (for the requested functions) in the command are analyzed, and the appropriate flags are set in the global flag area (GLBBUF) of the work area.		PARSE
4 For each requested option, the routine uses a multi-line WTO macro instruction to display to the operator on the receiving console the status for the option.		WTORTN
The display is presented serially in the order: CPU, Channel, Devices, High Storage Address, and status of Real Storage offline or scheduled to go offline. Only the information requested is displayed.		
The routine writes a multi-line display to the receiving console.		

Extended Description	Module	Label
The first line of the display is a control line. Then appear the data lines for the requested items. An end line completes the display. The following items require the inputs indicated:		
• **CPU and Channel**: From the common system data (CSD), an indication of the multiprocessing state and which CPUs are 'alive' (active). From the physical configuration communication area (PCCA) for a given CPU, the channel information in the channel availability table (CAT). Also from the PCCA, the CPU model and serial numbers.		CPUCHAN
• **Devices**: Channel and device information from the CAT and the UCB, respectively.		DEVICE
In displaying device data, the routine uses the IOSGEN macro instruction twice: once with the UCBLOOK operand to obtain the UCB address, and once with the MAP operand to obtain path (to a device) information.		
• **High Storage Address**: The high potential address from the CVT.		
• **Storage**: The page frame table entries (PFTEs) contain storage status information. The real storage reconfiguration (RSR) routine of the real storage management (RSM) component processes the entries in the PFTE and returns the information to module IEEMPDM. A search of the PFTEs is also made to determine any reconfigurable storage units defined to the system.		STORAGE
5 A WTO macro instruction is used to produce a single line message output to the requesting console.	IEEMPDM	WTORTN
6 The routine frees the work area, uses a DEQ macro instruction to release the resource SYSZVARY, CPU, and releases the console from the multi-line environment.		

Diagram 2-20. Displaying Operator-Action Requests (D R) (IEE2903D) (Part 1 of 2)

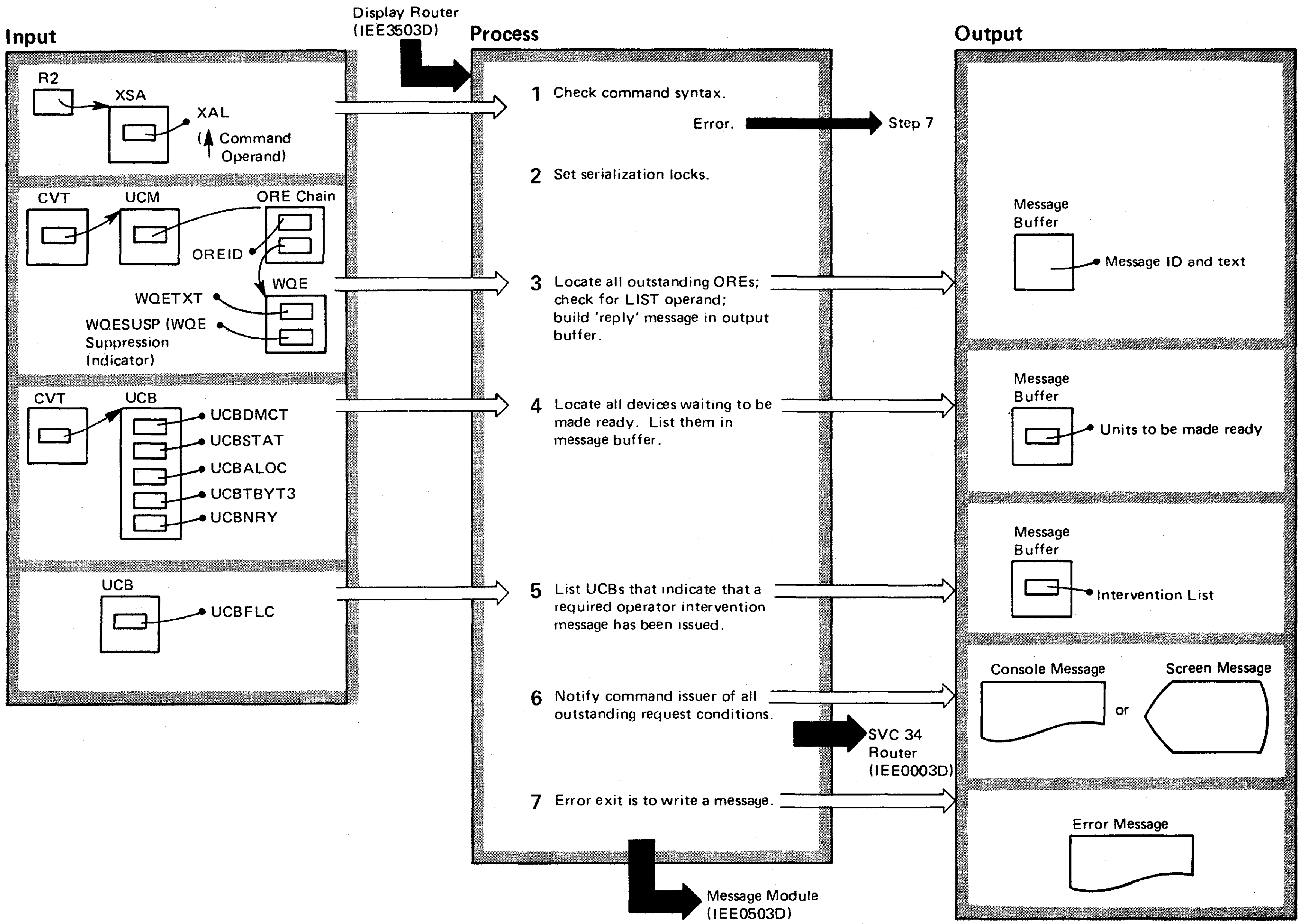


Diagram 2-20. Displaying Operator-Action Requests (D R) (IEE2903D) (Part 2 of 2)

Extended Description	Module	Label
This routine builds a console display of information related to unanswered WTOR messages, outstanding mount requests, and pending operator-intervention requests.		
1 Routine checks the format of the LIST operand.	IEE2903D	SETLOCK
2 The routine uses local and CMS locks to serialize the use of operator reply elements (OREs) and write queue elements (WQEs).		SETLOCAL
3 If any of the following conditions are met, an ORE is considered as not outstanding: • The ORE has been marked as deleted (a delete operator message (DOM) has been issued). • The ORE has been partially processed (a temporary buffer exists). • The ORE has been marked as suspended. If LIST is specified, up to 65 text characters are inserted in the message buffer for each outstanding ORE. If the text is greater than 65 characters and the sixty-sixth character is non-blank, the text will be truncated after the last complete word before the sixty-sixth character.		RPIDSRCH
4 Each tape or direct access UCB that has a mount message pending and is currently allocated is considered to be *not* ready. Tape devices must also be *marked* as *not* ready. The routine moves the unit numbers for these UCBs to the message area.		RDUTSRCH
5 The routine lists the UCBs by means of their unit numbers.	IEE2903D	IRTEST
6 This is done by using multi-line WTOs or TPUTs (for TSO user). The routine also releases the serialization locks.		MLWTOSEG

Diagram 2-21. Display of Program-Function-Key Definitions (D PFK) (IEE40110) (Part 1 of 2)

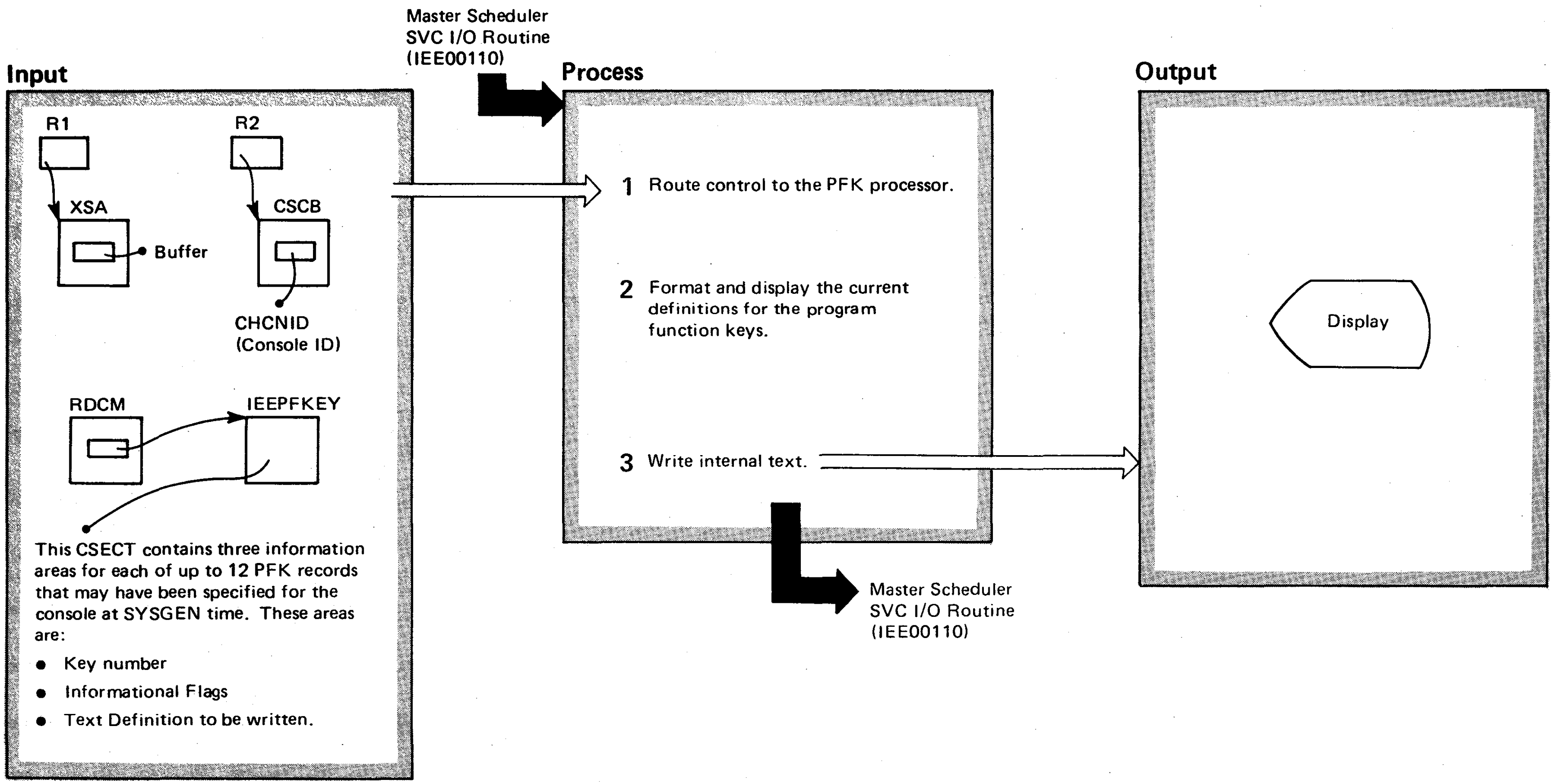


Diagram 2-21. Display of Program-Function-Key Definitions (D PFK) (IEE40110) (Part 2 of 2)

Extended Description	Module	Label
This routine satisfies a request to display pre-defined program-function-key (PFK) information.		
1 This processing occurs after the SVC34 load module has posted the master scheduler. The scheduler attaches a SVC 110 routine, which gives control to the PFK processor.	IGC0003D IEEVWAIT IEEPALTR IEE00110	
2 Move the definitions into the WTO parameter list.	IEE40110	START
3 Issue the WTO macro instruction to write the internal text to the console indicated in the CSCB.	IEE40110	SUBROUT

Diagram 2-22. Displaying Unit Status (D U) (IEE20110) (Part 1 of 2)

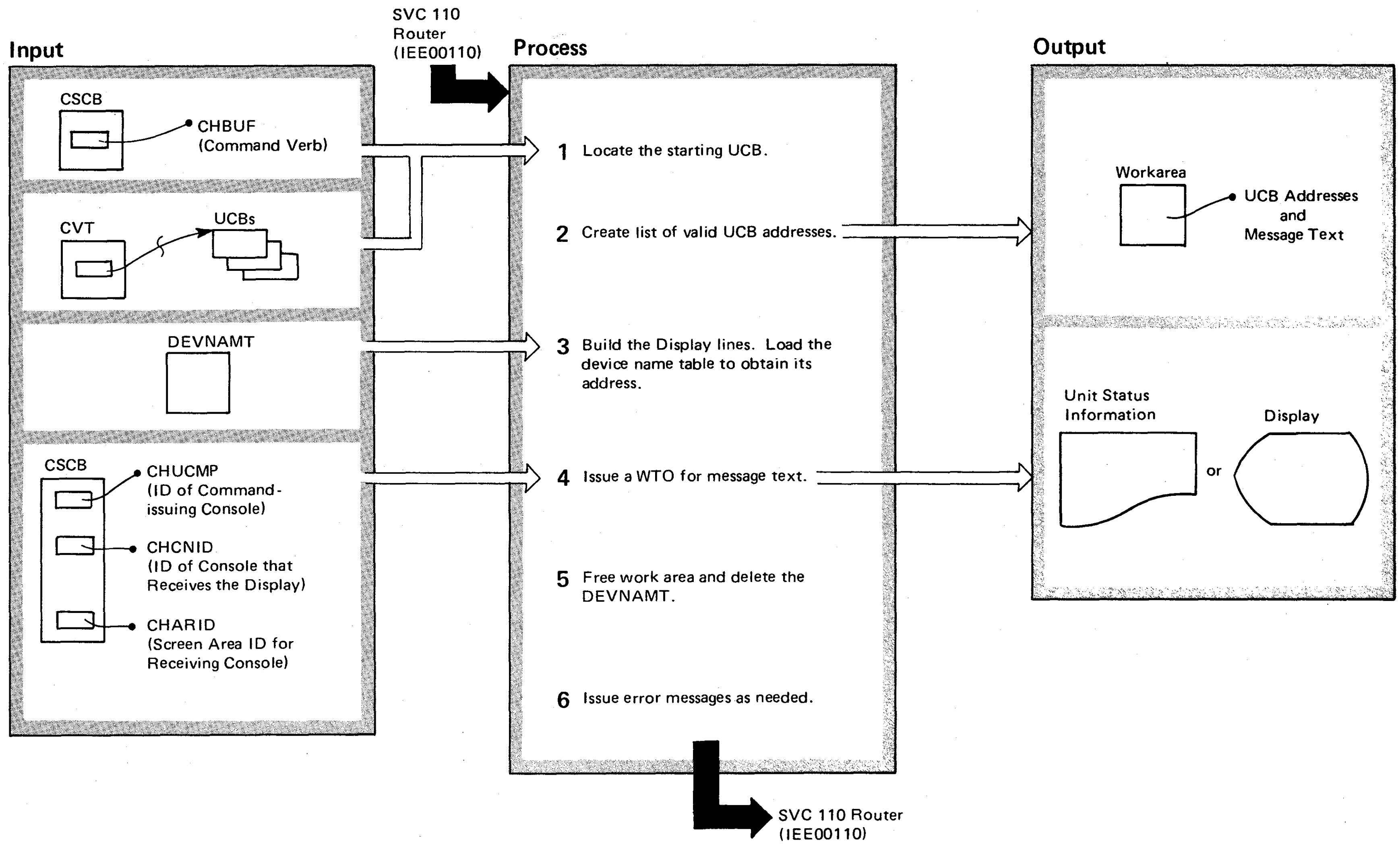


Diagram 2-22. Displaying Unit Status (D U) (IEE20110) (Part 2 of 2)

Extended Description	Module	Label
This routine satisfies a request for a tabular display of unit status information.		
1 The routine uses a GETMAIN for a work area. It saves the "to" and "from" console IDs. It verifies syntax and determines initial UCB.	IEE20110	IEE20110
2 Find UCBs that satisfy the command. Order the UCBs by device address. Indicate the end of the display.	IEE23110	VALIDCAK COMPSET OSU
3 The device name table (DEVNAMT) is established at SYSGEN time. It resides in the link pack area library (LPALIB) and is loaded into the work area. One half of the text line uses data from the UCBs.	IEE21110	
4 Issue SVC 35 for the title, the label lines, and the text.	IEE23110	WTORTN
5 The routine uses a DELETE macro instruction for the device name table and a FREEMAIN macro instruction for both the DEVNAMT and the work area storage.		
6 The routine writes any necessary error messages.	IEE22110	

Diagram 2-22A. Displaying Parameters of Domains (IEEDISPD) (Part 1 of 4)

Master Scheduler Wait Routine (IEEVWAIT)

Input

R1

CSCB

Area ID (CHARID)

Console ID (CHCNID)

CSCB

DMN operand (CHBUF)

R15

Return code from scan function IEEBUFSC

Process

1 Establish ESTAE environment.

— if unsuccessful proceed to step 13.

2 Get console and area ID for the target console.

3 GETMAIN storage for a workarea.

4 Enter SRM to obtain a copy of the domain table.

5 Scan for 'DMN' operand. If not found, proceed to step 7.

6 Display domain table and proceed to step 13.

7 Scan for 'DMN=value' operand.

8 Evaluate return code and issue message.

IEE0503D

Output

Display of Domain Table

Error Message

Diagram 2-22A. Displaying Parameters of Domains (IEEDISPD) (Part 2 of 4)

Extended Description	Module	Label
This process displays the Domain Descriptor Table (DMDT).		
1 This ESTAE environment handles ABEND situations. If the ESTAE is not established, storage for the CSCB is released before returning control.	IEECB860 IEEDISPD	 FREECSCB
2 Save console and area information to use in MLWTO.		
3 This area (obtained by GETMAIN) will contain the MLWTO parameter list and data obtained via Sysevent number 40 processing. Storage is from subpool 253.		GETSTOR
4 Enter SRM via SYSEVENT 40. SRM module IRARMEVT will return a copy of the domain table and the count of the number of entries.		
5 The character string 'DMN' is searched for in the buffer.		DMNSCAN
6 This routine issues a multiple-line WTO macro instruction to write the domain information to the console.		MSGSET

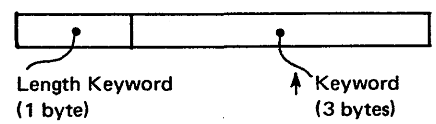
Extended Description	Module	Label
7 This routine uses the IEEBUFSC macro instruction to find the DMN keyword and its value.		

Input to IEEBUFSC:

R1 (points to the beginning of the buffer)

R0 (points to the last byte of the buffer +1)

R15



Output from IEEBUFSC:

R1 (length of keyword value)

R14 (points to the first byte of keyword value)

R15 Return code

- 0 = success
- 4 = DMN value invalid
- 8 = DMN keyword not found

8 If register 15 contents are non-zero, IEE0503D is loaded and given control to issue the error message: 'IEE708I DMN KEYWORD VALUE INVALID'. Otherwise, Step 8 executes next.	IEE0503D	

Diagram 2-22A. Displaying Parameters of Domains (IEEDISPD) (Part 3 of 4)

Input

CSCB
DMN operand (CHBUF)

CSCB
DMN operand (CHBUF)

Process

9 Validate domain value.
- If invalid. → IEE0503D
- Otherwise proceed to step 10.

Step 12

10 Check for extra operands in the buffer.
- If found. → IEE0503D
- Otherwise proceed to step 11.

Step 12

11 Display specified domain.

12 Free work area and CSCB.

13 Return using register 14.

Return to Caller (IEEVWAIT)

Output

Error Message

Error Message

Display of Domain

Diagram 2-22A. Displaying Parameters of Domains (IEEDISPD) (Part 4 of 4)

Extended Description	Module	Label
9 If the domain value is not in the range of 1-128, its length exceeds three, or the domain is not defined in the domain table, error message 'IEE708I DMN KEYWORD VALUE INVALID' is issued.	IEEDISPD IEE0503D	VALDMN
10 Search the buffer for extraneous operands. If any are found, issue the error message, 'IEE535I DISPLAY INVALID PARAMETER'.	IEEDISPD IEE0503D	
11 A MLWTO (multiple-line WTO) is issued to write the specified entry.	IEESISPD	MSGSET
12 The work area (subpool 253) and CSCB are freed.		FREESTOR FREECSCB
13 IEEDISPD returns using the contents of register 14 initially passed at entry by IEEVWAIT.		

Diagram 2-23. Dumping (DUMP) Virtual Storage (IEECB866) (Part 1 of 2)

Master Scheduler Wait Routine (IEEVWAIT) (via ATTACH)

Input

R1
DUMP CSCB

UCM
UCMID
CHUCMP
CSCB

Operator Reply

Virtual Storage
Amount is determined by the parameter list

Process

1 Create ESTAE environment.

2 Check Console's Command authority.
Invalid authorization. → Error Routine (IEE0503D)

3 Get header text and data to be dumped.

4 Dump the requested information.

5 Check return codes and return.

a) If code = 0. → System via BR14

b) For non-zero code, put out message.

Error Routine (IEE0503D)

Output

SDUMP Parameter List

SYS1.DUMPnn (nn = 00-09)

Master Console

Diagram 2-23. Dumping (DUMP) Virtual Storage (IEECB866) (Part 2 of 2)

Extended Description	Module	Label
The DUMP command causes a dump of virtual storage to a preallocated data set. The dump routine runs in the master scheduler region.		
1 This environment protects the dump processing in case of abnormal end. An ESTAE exit will dump the master memory, using SDATA options, to the dump data set.	IEECB860	GETESTAE
2 Only the master console is authorized to issue the console DUMP command.	IEECB866	CMDCHECK
3 Header text data is specified in the operand of the DUMP command. The header text contains a maximum of 100 characters. Dump operand data is specified in the REPLY command, which the operator inserts in response to a WTOR command issued by the dump routine. The parameter list has the format shown below:	IEECB866	SETUP DMPREPLY

<table>
<tr><td>Flag 0</td><td>Flag 1</td><td>Dump Data 1</td><td>Dump Data 2</td></tr>
<tr><td colspan="4">Reserved</td></tr>
<tr><td colspan="4">↑Storage list</td></tr>
<tr><td colspan="4">↑Header record</td></tr>
<tr><td colspan="4">Reserved</td></tr>
<tr><td colspan="2">0</td><td colspan="2">User's ASID to be dumped</td></tr>
</table>

The flag and dump data field contents are as follows (blank indicates 'reserved'):

Bit	Flag 0	Dump Data 1 (SDUSDAT1)	Dump Data 2 (SDUSDAT2)
0		Dump the PSA	Dump the CSA
1	Storage list is specified		Dump the SWA
2	Header record is specified	Dump the nucleus	
3		Dump the SQA	
4	ASID is specified	Dump the LSQA	
5		Dump the private region	
6		Dump the active LPA	
7		Dump the trace table/GTF buffers	

- Flag 1 = X'80'
- The storage list contains the beginning and ending addresses of the areas to be dumped.

Extended Description	Module	Label
4 Routine IEECB866 issues SVC 51 (via the SDUMP macro instruction) to have information put on a preallocated data set, SYS1.DUMPnn (nn = 00-09).	IEECB866	ISUSDUMP
5 The CSCB for the command is freed before returning to the caller.	IEECB866	CMDCHECK
a.		
b. The message module issues a message to the master console for error conditions due to operand syntax or lack of command keywords.	IEE0503D	NODUMP

Diagram 2-24. HALT (Z), SWITCH (I), and TRACE (TRACE) Command Initialization (IEE1403D) (Part 1 of 2)

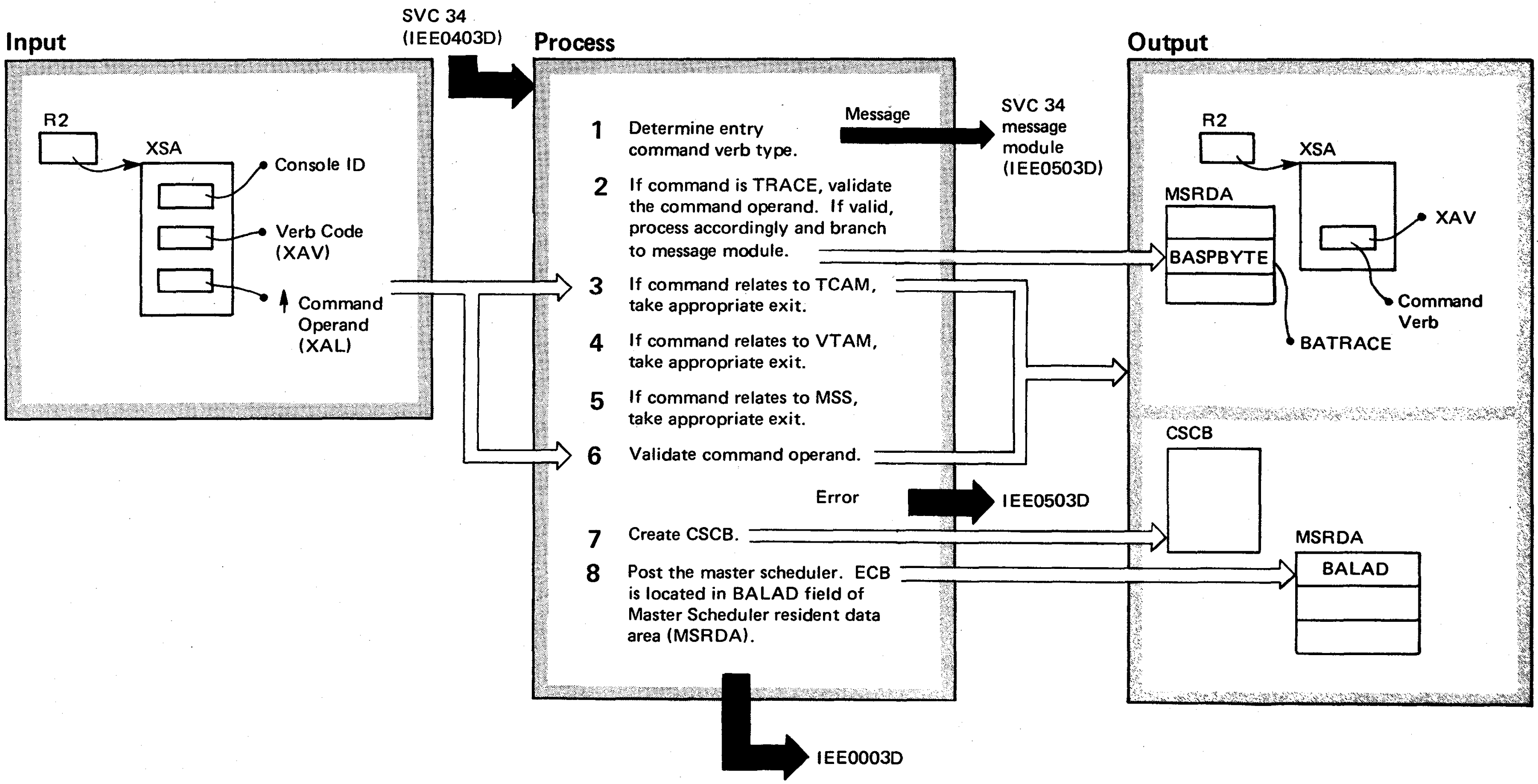


Diagram 2-24. HALT (Z), SWITCH (I), and TRACE (TRACE) Command Initialization (IEE1403D) (Part 2 of 2)

Extended Description	Module	Label
The HALT command initiates a shutting down of a VTAM or TCAM system, or the 3850 MSS, or it prepares for closing down the entire operating system. The command also causes the closing of the system log and, as part of shutting down the TP access method, it halts the transmission of terminal-oriented messages.		
1 A HALT, SWITCH, or TRACE command causes entry to this routine.	IEE1403D	IEE1403D
2 If command is TRACE ON, so indicate in the MSRDA. (Flag BATRACE is set on in the BASPBYTE field.) If command is TRACE OFF, so indicate in MSRDA. (Set BATRACE flag off.) If command is TRACE STATUS, or no operand is provided, set code for appropriate message and branch to IEE0503D. TRACE processing will be completed by IEEVWAIT, according to how BATRACE was set.	IEE1403D	TRACE
3 For TCAM, terminate the SVC 34 processing; exit to TCAM processor via module IED1303D.	IED1303D	
4 For VTAM, terminate the SVC 34 processing. Exit to VTAM processor via module ISTCFF3D.	ISTCFF3D	
5 For MSS, validate the presence of "S," with MSS operands, and exit to IEE9403D to post MSS.	IEE9403D	
6 For SWITCH, SMF is the only valid operand. If syntax is invalid, issue message IEE305I.	IEE1403D	
7 Creating a CSCB will avoid the processing of the command by the communications task. (See the diagram, Creating CSCB for Task-Creating Commands.)	IEE0803D	
8 The master scheduler will attach the HALT/SWITCH/TRACE command processor to perform the appropriate closing function.	IEEVWAIT	

Diagram 2-25. HALT (Z EOD) and SWITCH (I SMF) Command Processing (IEE70110) (Part 1 of 4)

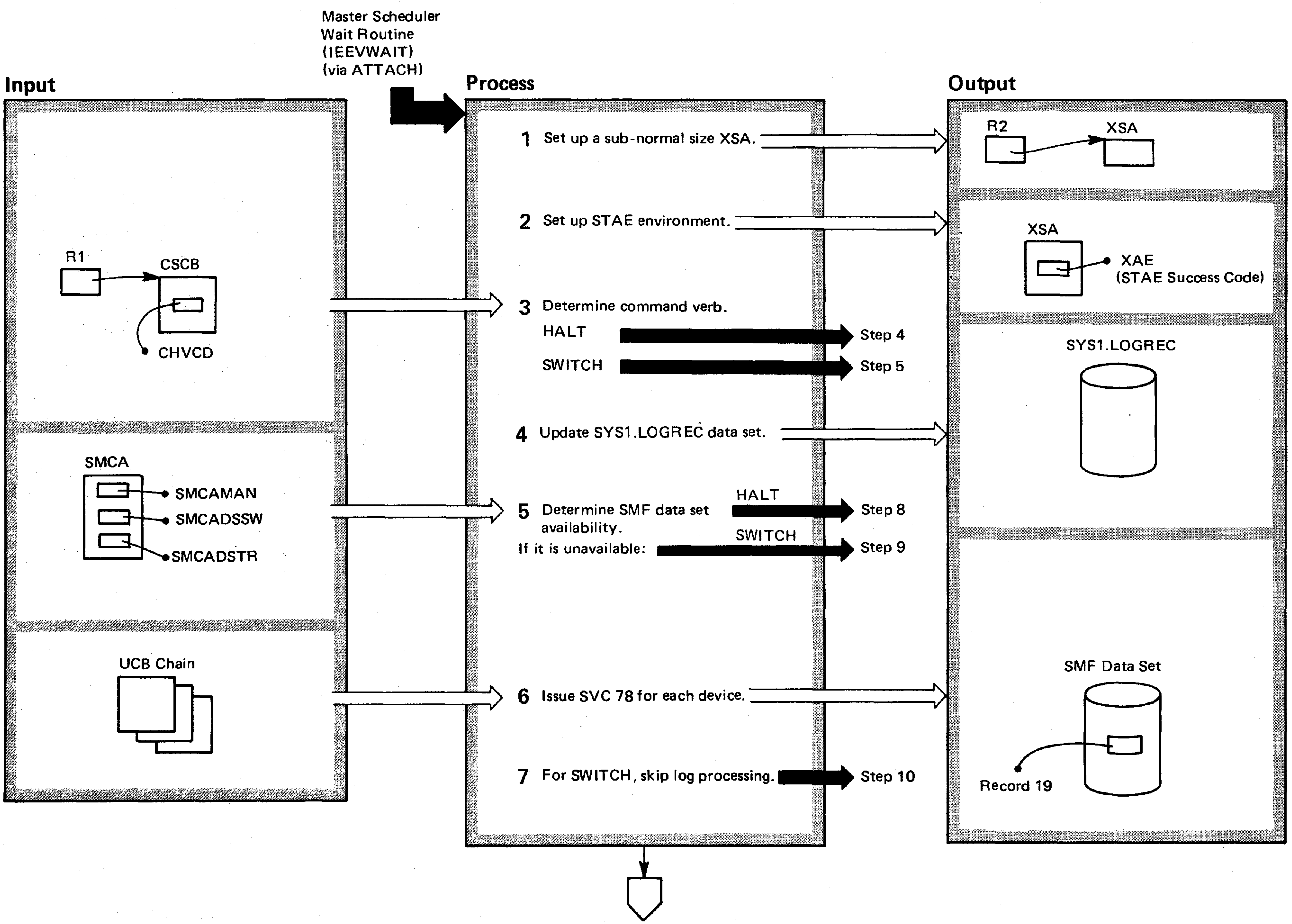


Diagram 2-25. HALT (Z EOD) and SWITCH (I SMF) Command Processing (IEE70110) (Part 2 of 4)

Extended Description	Module	Label
This routine performs two functions: • It preserves the status of system log data sets and moves data from internal storage to the SYS1.LOGREC data set. • It switches the recording of SMF data from one data set to another.		
1 This XSA is only 88 bytes long.	IEE70110	
2 STAE handles abnormal end situations.	IEECB860	
3 A HALT command must have the EOD operand. A SWITCH command must have the SMF operand.	IEE70110	
4 The routine uses SVC 76 to do the update.		
5 For the SMF active state, the routine uses SVC 83 to inform SMF routines of the current processing.		
6 The routine uses the LSPACE macro instruction (SVC 78) to generate SMF record 19 and place it on the SMF data set. This is done for each online direct-access device (or UCB).		
7 If the command is 'SWITCH,' the log is not posted.		

Diagram 2-25. HALT (Z EOD) and SWITCH (I SMF) Command Processing (IEE70110) (Part 3 of 4)

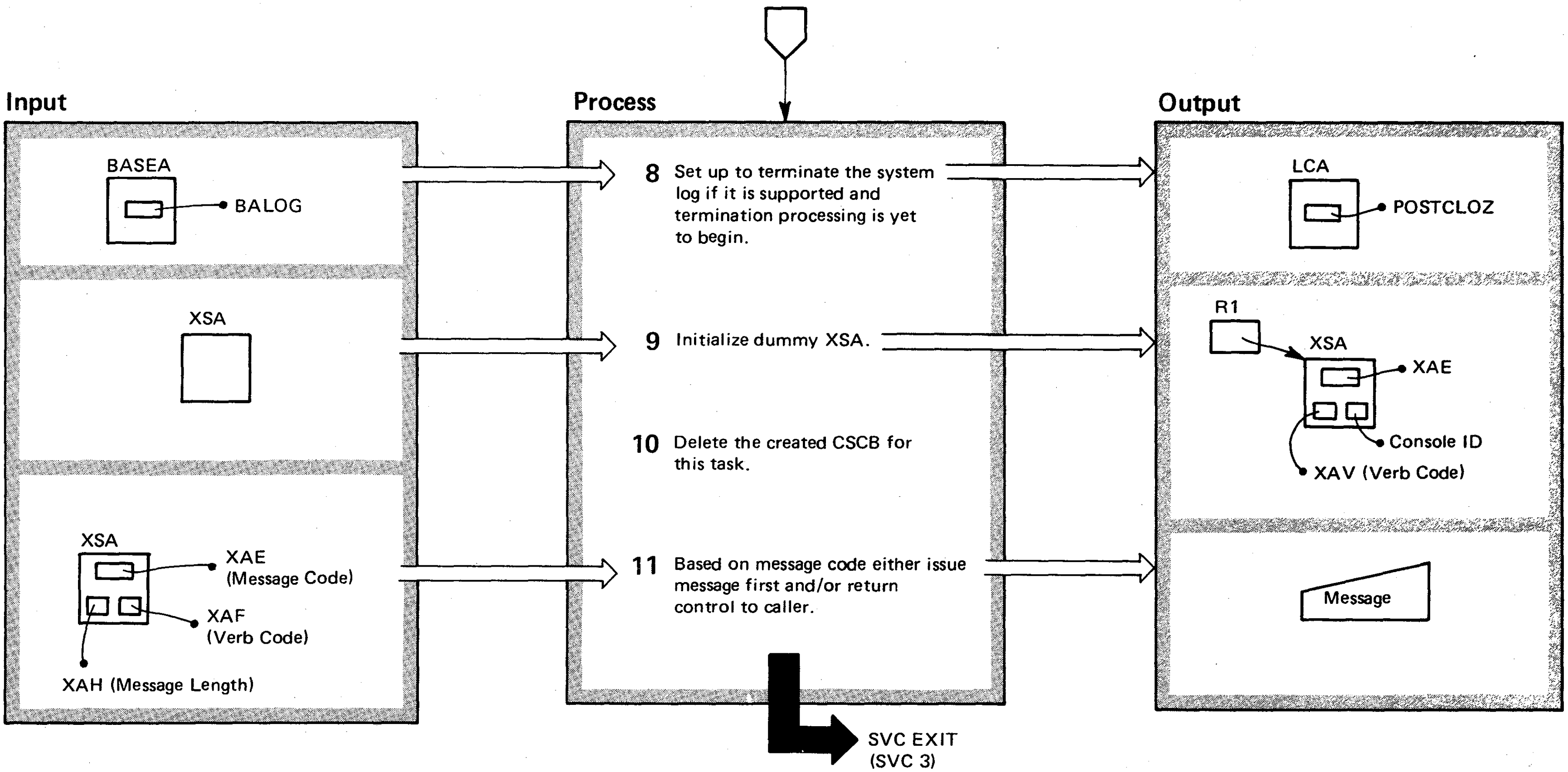


Diagram 2-25. HALT (Z EOD) and SWITCH (I SMF) Command Processing (IEE70110) (Part 4 of 4)

Extended Description	Module	Label
8 The routine posts the 'log close' ECB to permit the occurrence of log termination. The post is for module IEEMB803.	IEE70110	
9 The routine sets the XSA fields that relate to the command to be processed.		
10 The CSCB was created by module IEE0803D. The routine uses the MGCR macro instruction (SVC 34) to delete the CSCB.		
11 If message code indicates a successful SMF switch, the message module is bypassed, and return is directly to the caller. A test also indicates whether a STAE-failure occurred. The message module issues the WTO macro instruction to put out the message that indicates the success or failure of the processing.	IEE90110	

Diagram 2-26. Processing LOG (L) and WRITELOG (W) Commands (IEE1603D) (Part 1 of 4)

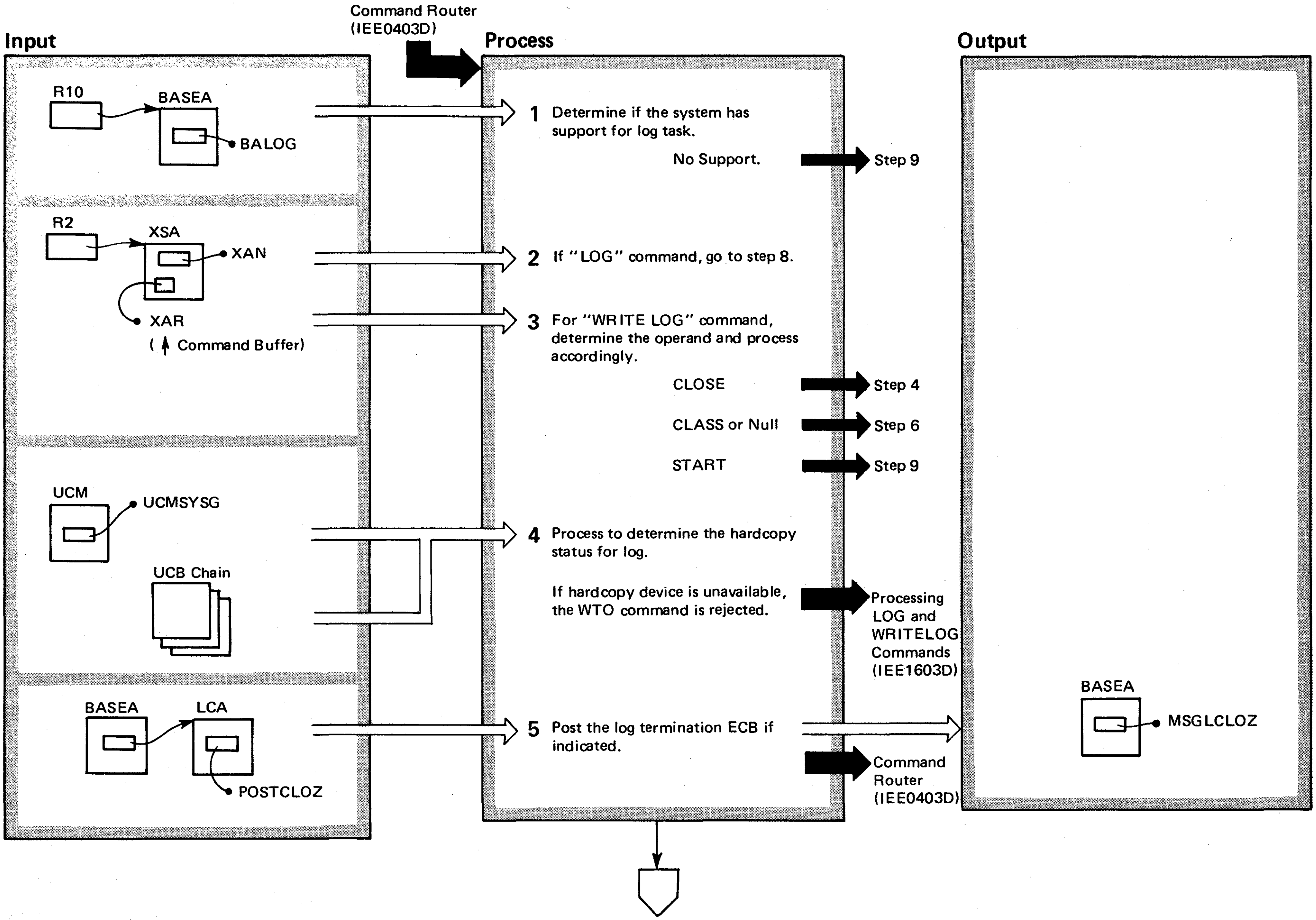


Diagram 2-26. Processing LOG (L) and WRITELOG (W) Commands (IEE1603D) (Part 2 of 4)

Extended Description	Module	Label
This routine processes the LOG and WRITELOG Commands and either puts out a message or posts an ECB.		
1 If the BALOG value is not 0, the system supports the log task.	IEE1603D	
2 If the XAN value = X'1C', the command is "LOG."		
3 A request has been made either to close the system log, to make the system log a part of an output class, or to schedule the writing of the system log.		
4 If the log is a hardcopy device, the system rejects all WRITELOG CLOSE commands until the hardcopy function is assigned (varied) to another device.		
5 This step is indicated if the log is already in the process of terminating.		

Diagram 2-26. Processing LOG (L) and WRITELOG (W) Commands (IEE1603D) (Part 3 of 4)

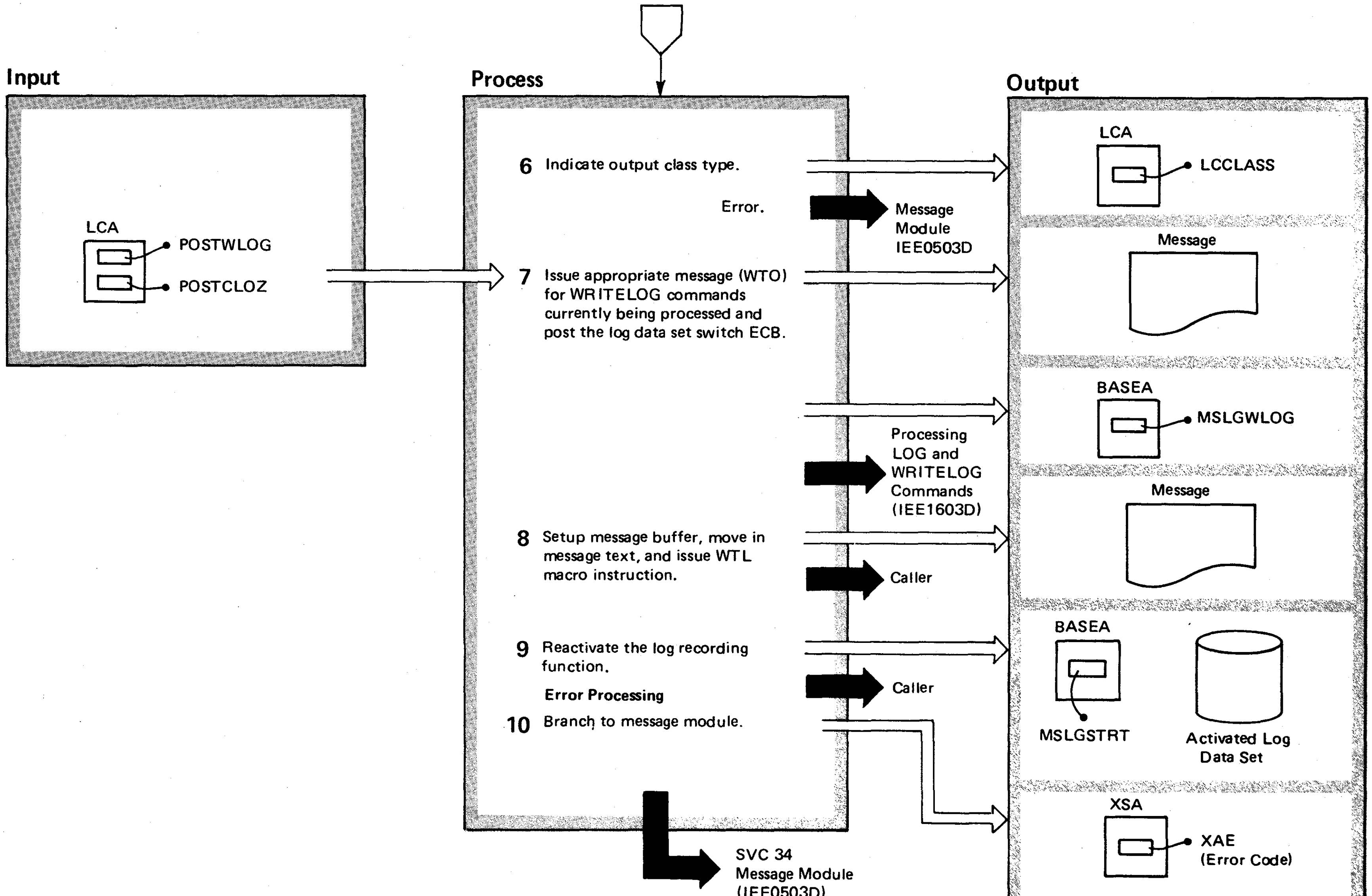


Diagram 2-26. Processing LOG (L) and WRITELOG (W) Commands (IEE1603D) (Part 4 of 4)

Extended Description	Module	Label
6 Valid output classes are those from A-Z or 0-9. This is the output class to be used when printing the contents of the system log. If the output class designation is omitted, a default class of 'A' is assumed.	IEE1603D	
7 The routine posts the ECB for the switching of the log data set.		
8 Use the WTL macro instruction to write a message to the system log.		
9 The WRITELOG START command initiates support of the system log. The routine posts the appropriate ECB.		
10 Message codes are set during various stages of the processing.	IEE0503D	

Diagram 2-27. SWAP (G) (IGF2503D) and MODE (MODE) (IGF2603D) Command Processing

Command Translator (IEE5403D)

IEE0403D

Process

1 Command Verb.

a) SWAP. → IGF2503D (A)

b) MODE. → IGF2603D (A)

(A) For further logic details, see the publication, **OS/VS2 Release 2 MVS/RMS Program Logic Manual,** SY27-7250.

Diagram 2-28. STOP (P) and MODIFY (F) Command Processing (IEE0703D) (Part 1 of 2)

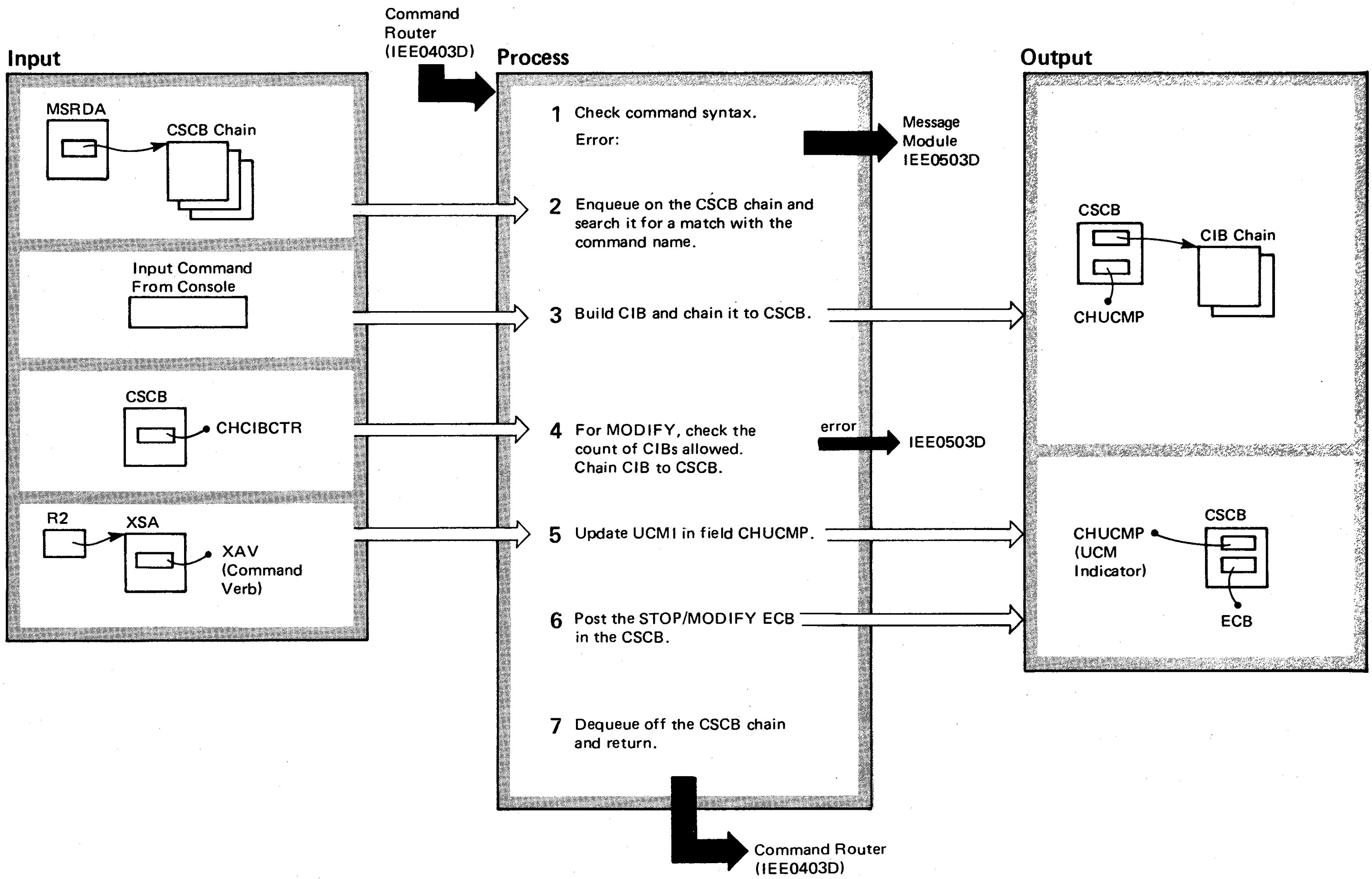
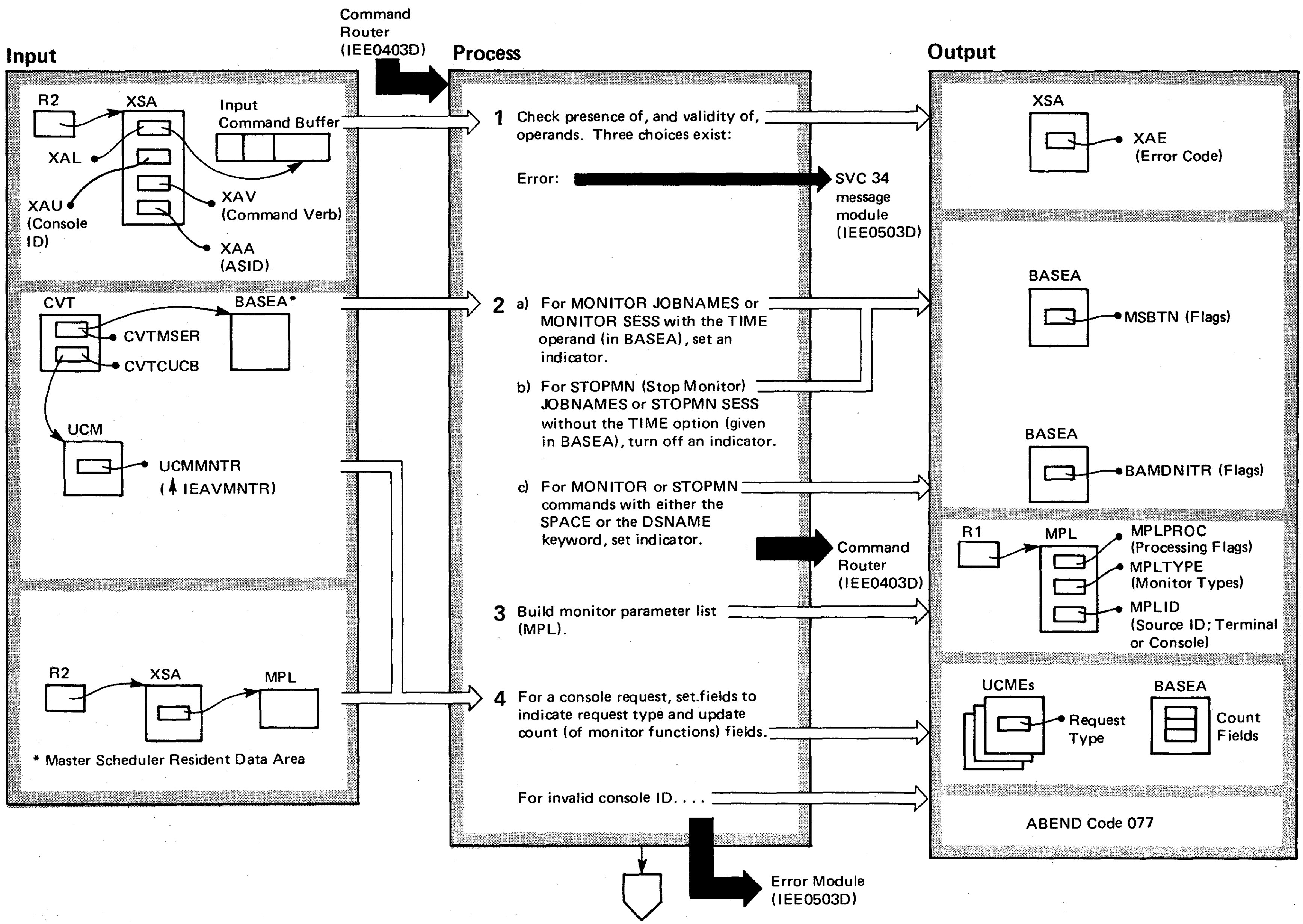


Diagram 2-28. STOP (P) and MODIFY (F) Command Processing (IEE0703D) (Part 2 of 2)

Extended Description	Module	Label
This processing either stops the operation of, or changes the characteristics of, system components such as readers and writers, and performs the same services for appropriately-loaded problem programs.		
1 Operands beyond the job name or job identifier are prohibited on the STOP command, but are required on the MODIFY command.	IEE0703D	IEE0703D
2 If no match with the job name or job identifier is found, error message results.		CMST1
3 For MODIFY, move parameters to CIB.		CMCIBLD CM002
4 If more CIBs are allowed, add them to the chain. If the CIB chain contains the maximum allowable number of CIBs, the processor rejects the command and leaves the UCMI unchanged. For STOP, the CIB is chained unconditionally.		XTEST3
5 The UCM indicator of the console issuing the command overlays the UCM indicator, in the CSCB, of the console that issued the START command.		CM003
6 A cross-memory posting occurs here. Each command has its own posting code.		SETECB
7 The dequeueing occurs after all CSCBs on the CSCB chain have been examined and processed, if necessary.		CMDEQ

Diagram 2-29. Starting (IEE7103D) and Stopping (IEE5503D) Monitoring Functions (Part 1 of 4)



Extended Description	Module	Label
This processing handles requests to start or stop event-driven displays of direct access space, data set names and job names, and so on. The processing uses a communications task routine, IEAVMNTR, to adjust fields in response to MONITOR and STOP MONITOR commands with the operands SESS, STATUS, or JOBNAMES.		
1 If the input stream contains the command, the console ID = 0.	IEE7103D (For Monitor)	
2 The time field in BASEA is cleared with the following operands: JOBNAMES, SESS, STATUS, SPACE, or DSNAME, as follows:	IEE5503D (For Stop Monitor)	

	Input Stream or Console	Time Sharing (Operator Mode)	Subsystem Console (JES3)
JOBNAMES(,T)	valid	valid	invalid
STATUS	valid	valid	invalid
SESS(,T)	valid	valid	invalid
SPACE	valid	invalid	valid
DSNAME	valid	invalid	valid

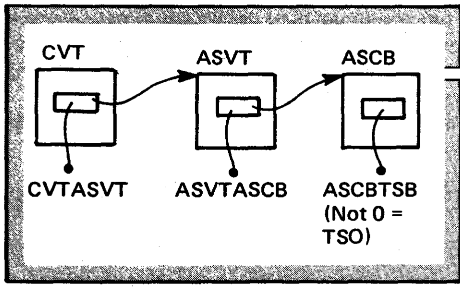
The MONITOR and STOP MONITOR operands and their validity by source.

3 This parameter list address is at field XAR in the XSA.		
4 The request type bits indicate the stopping or starting of a monitor request. Count fields indicate function (STATUS, JOBNAMES, or SESS). The count indicates the total number of consoles (and terminals) that are monitoring a given function (one count field per function).	IEAVMNTR	

Note: CMS and local lock enable a serial use of the UCME.

Diagram 2-29. Starting (IEE7103D) and Stopping (IEE5503D) Monitoring Functions (Part 3 of 4)

Input



Process

5 For a request from a terminal, create and/or update the monitor queue element (MQE). Update BASEA count fields.

a) For an error during creation of MQE, set return code. → Step 6

b) If stopping monitor action and all MQEs are off the chain, post the termination ECB for the entry point IEAVTPUT in module IEAVMASV.

6 Free storage and return.

→ IEE0003D

Output

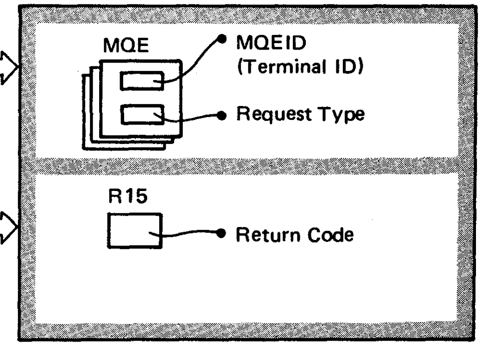


Diagram 2-29. Starting (IEE7103D) and Stopping (IEE5503D) Monitoring Functions (Part 4 of 4)

Extended Description	Module	Label
5 For terminal requests, monitor queue elements (MQEs) are chained together. An MQE is removed from the chain if all request type bits are off. The MQEs reside in the Common Service Area (CSA). TSO use of SPACE and DSNAME operands is invalid. (That is, module IEAVMNTR does not handle these two operands.) **Note:** CMS and local locks enable a serial use of the MQE.	IEEVMNTR	
6 Storage was used for initial saving of register contents.	IEAVMNTR	

Diagram 2-30. Routing Messages (MR) to Consoles (IEE6303D) (Part 1 of 2)

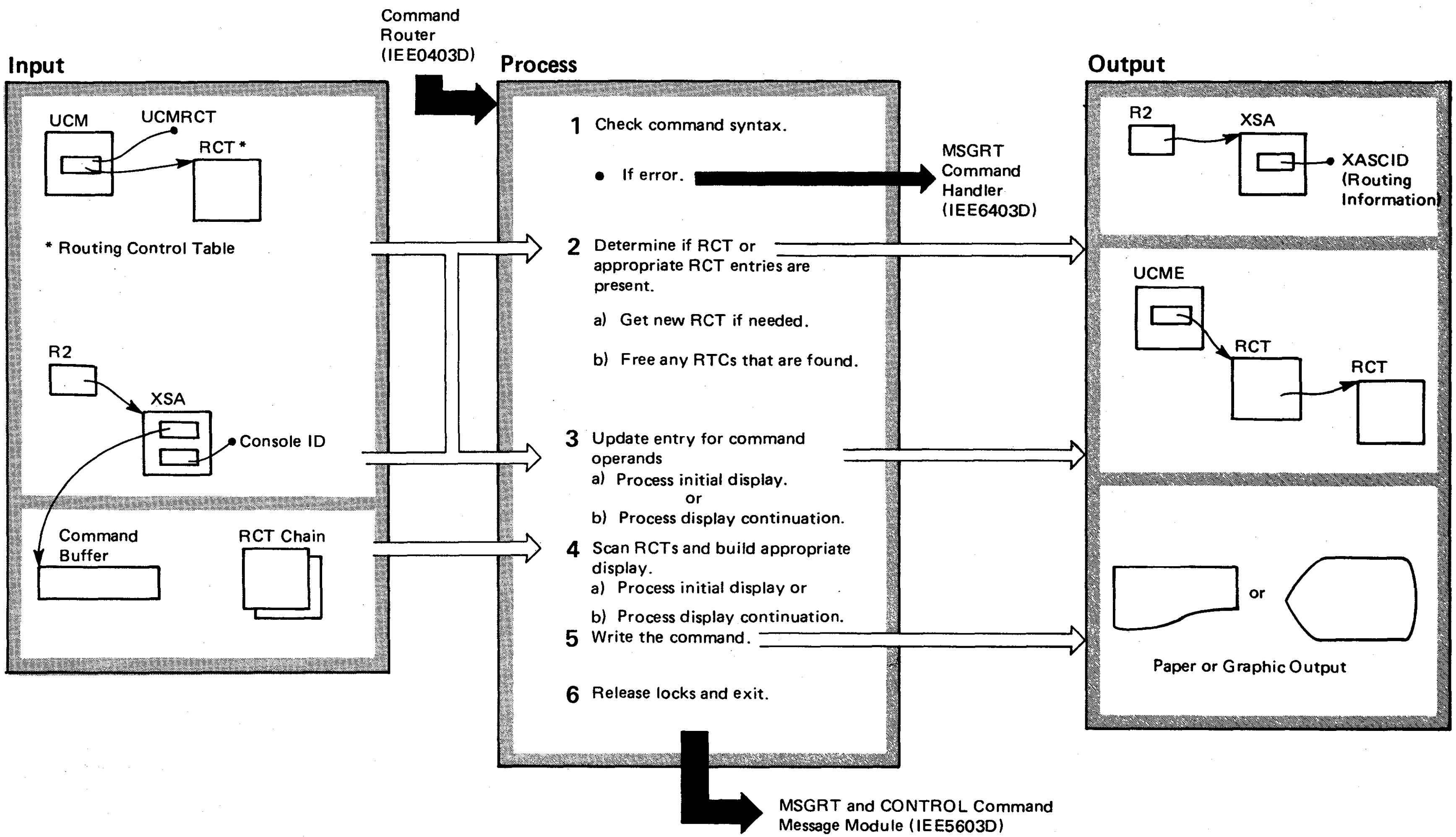


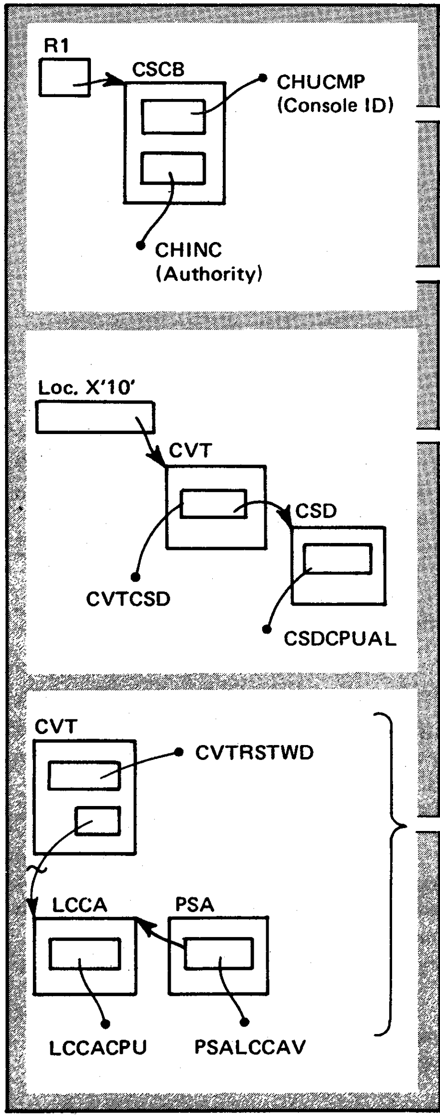
Diagram 2-30. Routing Messages (MR) to Consoles (IEE6303D) (Part 2 of 2)

Extended Description	Module	Label
Message routing commands direct the output of status-display commands to the indicated display area on a console.		
1 If the command has the REF operand or if there is a default due to the lack of an operand, go to step 4. If it does not have a REF operand, the routine establishes default routing for DISPLAY, TRACK, STOPTR, and CONTROL commands.	IEE6303D	IEE6303D
2a The routing control table (RCT) has a prescribed number of entries. If the table is full, the routine uses the GETMAIN macro instruction to get storage for a new RCT and chains the new RCT to last RCT.		TABLECK
Note: Local and cross-memory-services (CMS) locks are held during this process.		
2b For the operand NONE, the storage for any existing RCTs is released.	IEE6303D	NONERTN
3 There is one 8-byte RCT-entry per command operand. An RCT entry contains the command parameters and the console and display area IDs.		VERBCOD
Note: The UCME pointing to the RCT chain is the UCME for the console that issues the command.		

Extended Description	Module	Label
4 The display is the MSGRT command that would generate the routing requests defined by the RCT entries. If the command is not executing under the Communications Task (i.e., the command is issued by JES3), then the RCT scan to build the message is done while holding the CMS and local locks.	IEE6403D	IEE6403D
a. Test next operand to see if it will fit in the output buffer. If it will not, set up continuation processing and place the CONT operand in the message.	IEE6403D	BUFCHK
b. If the CONT operand is being processed from command input and the output buffer is full, purge the buffer and re-initialize with RCT entries not yet displayed.		TSTCONT
5 For a CRT (display) console, the message is inserted in the instruction line in the display control module (DCM). For hardcopy output, use the WTO macro instruction. If the command is not executing under the Communications Task (i.e., the command is issued by JES3), the WTO macro instruction is always used.	IEE6403D	

Diagram 2-31. Quiescing (QUIESCE) a System (IEEMPS03) (Part 1 of 4)

Input



Process

Master Scheduler wait routine (via ATTACH) (IEEVWAIT)

1 Determine ID of the receiving console.

2 Determine if command-issuing authority is valid.

If error, → Step 8

3 Perform initialization.

If error, → Step 8

4 Fix the work area and problem program pages in storage.

If error, → Step 9

5 Establish protection for restart processing resources.

6 Obtain resource for stop/restart processing.

If error, → Step 8

Output

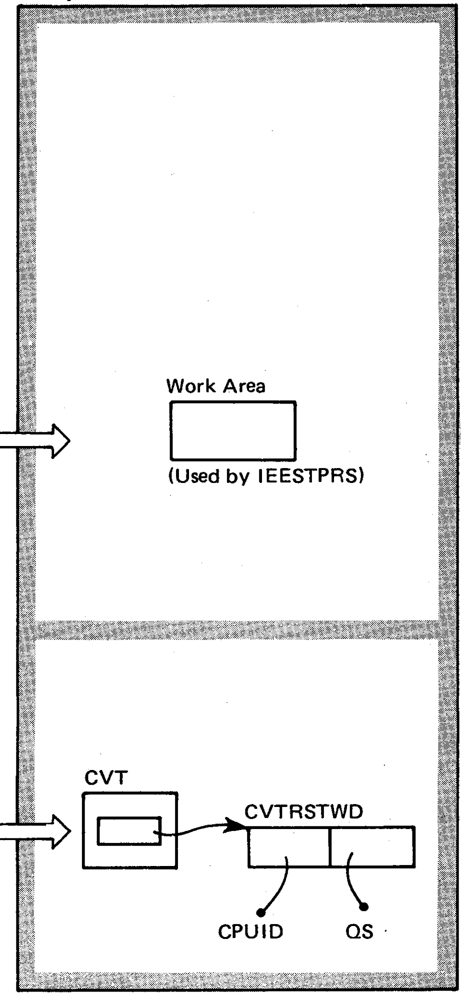


Diagram 2-31. Quiescing (QUIESCE) a System (IEEMPS03) (Part 2 of 4)

Extended Description	Module	Label
This routine suspends system activity by placing the active CPUs in a manual or wait state. If accurate job step timing is to be maintained across a stopped state, the quiescing function must be used.		
1 This is the console that will receive operator messages. For the master console and reader sources, the master console receives the messages. Messages from invalid console sources are sent to the requesting (or issuing) console.	IEEMPS03	
2 This command may be issued only from the master console or from a reader with the same authorization as the master console.		
3 The routine establishes a recovery routine (an ESTAE exit for abnormal end situations), it serializes the reconfiguration commands by using the ENQ macro instruction on the resource, SYSZVARY, CPU to serialize command processing in a multiprocessing environment, and it gets storage for a work area in the local system queue area.		SETUP
4 The routine uses the PGFIX macro instruction to make both its own code and the dynamic data area unpageable. This prevents page faults while the CPU is disabled.		PAGEFIX
5 The routine uses the SETFRR macro instruction to build FRR protection for the resources that are obtained by the compare and swap routine.		
6 The routine first builds a resource word in its own storage area. This word contains both the CPU ID (of the CPU on which the command is being processed) and the EBCDIC value 'QS' (to indicate that QUIESCE command processing is in progress). Then the routine enters a loop in which it checks the restart word CVTRSTWD for a null value. A null value indicates the availability of the restart resource. Throughout the compare and swap loop, the routine uses the WINDOW macro instruction to enable the system for malfunction alerts (MFAs) and emergency signals (EMSs) that will be pending if the other CPU (in a tightly-coupled MP system) fails. If MFAs or EMSs are pending, then the routine will invoke the alternate CPU recovery (ACR) routines. When the stop/restart resource is available, (CVTRSTWD has the value zero), module IEEMPS03 inserts its resource word value into the field CVTRSTWD to serialize the use of the restart resource. This permits a safe changing of the restart new PSW.		SWAPWORD

Diagram 2-31. Quiescing (QUIESCE) a System (IEEMPS03) (Part 3 of 4)

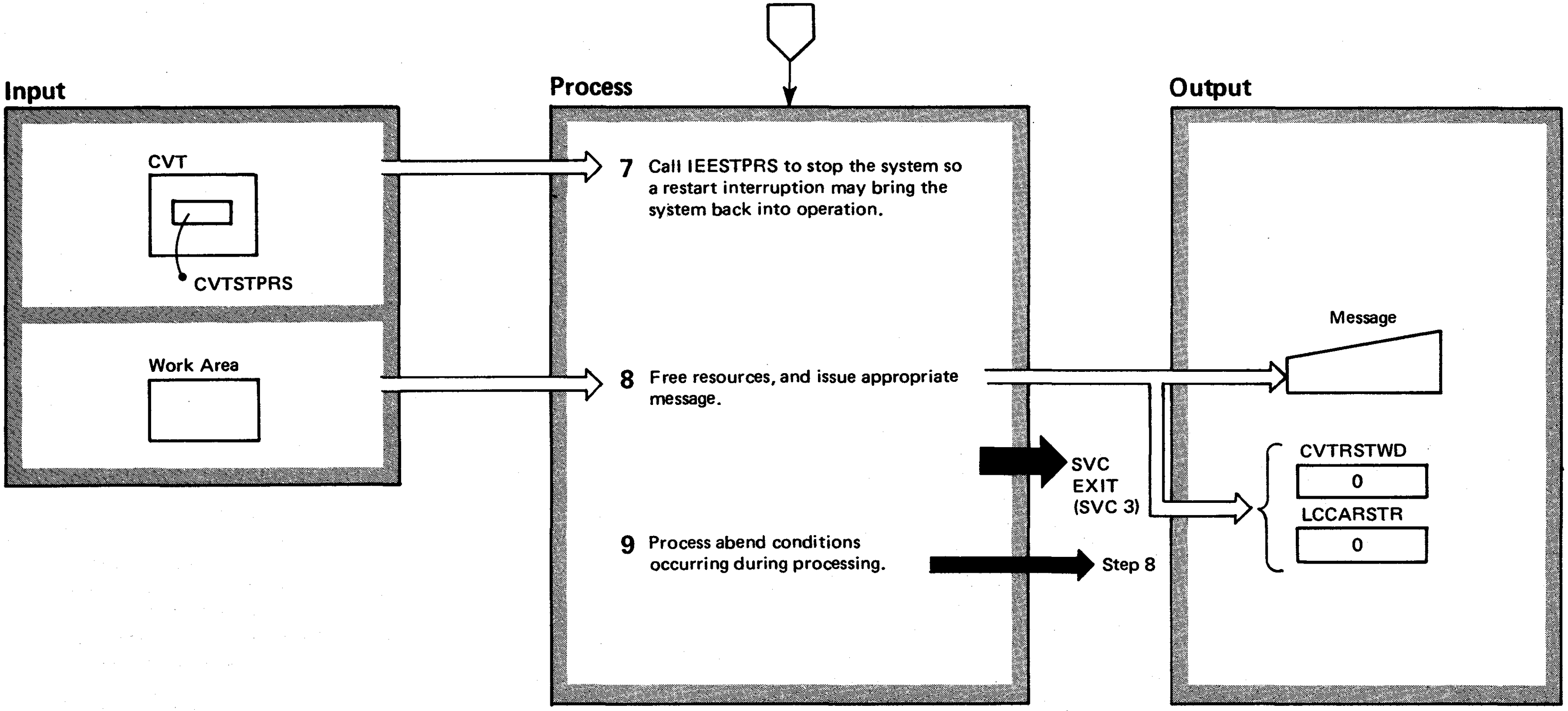


Diagram 2-31. Quiescing (QUIESCE) a System (IEEMPS03) (Part 4 of 4)

Extended Description	Module	Label
7 The stop and restart routine (IEESTPRS) is used by the QUIESCE command processor to stop all operating CPUs. See the diagram, Stopping and Restarting the System.	IEEMPS03	STPRSTRT
Depressing the RESTART button causes the CPUs to restart.		
8 This processing includes resetting the resource word in the CVT, resetting the dispatcher lock, freeing page-fixed storage and work areas, dequeueing the SYSZVARY, CPU resource, and issuing messages via the WTO macro instruction.		CLEANUP
9 For ABEND conditions occurring while the module holds the CVTRSTWD or LCCARSTR resource, a functional recovery routine frees the resource.		FRRRTN
For all ABEND conditions, an ESTAE routine records the diagnostic work area (if one is available), dumps dynamic storage, and gives control to the CLEANUP subroutine.		ESTAERTN

Diagram 2-32. Replying (R) to Information Requests (IEAVVRP1) (Part 1 of 6)

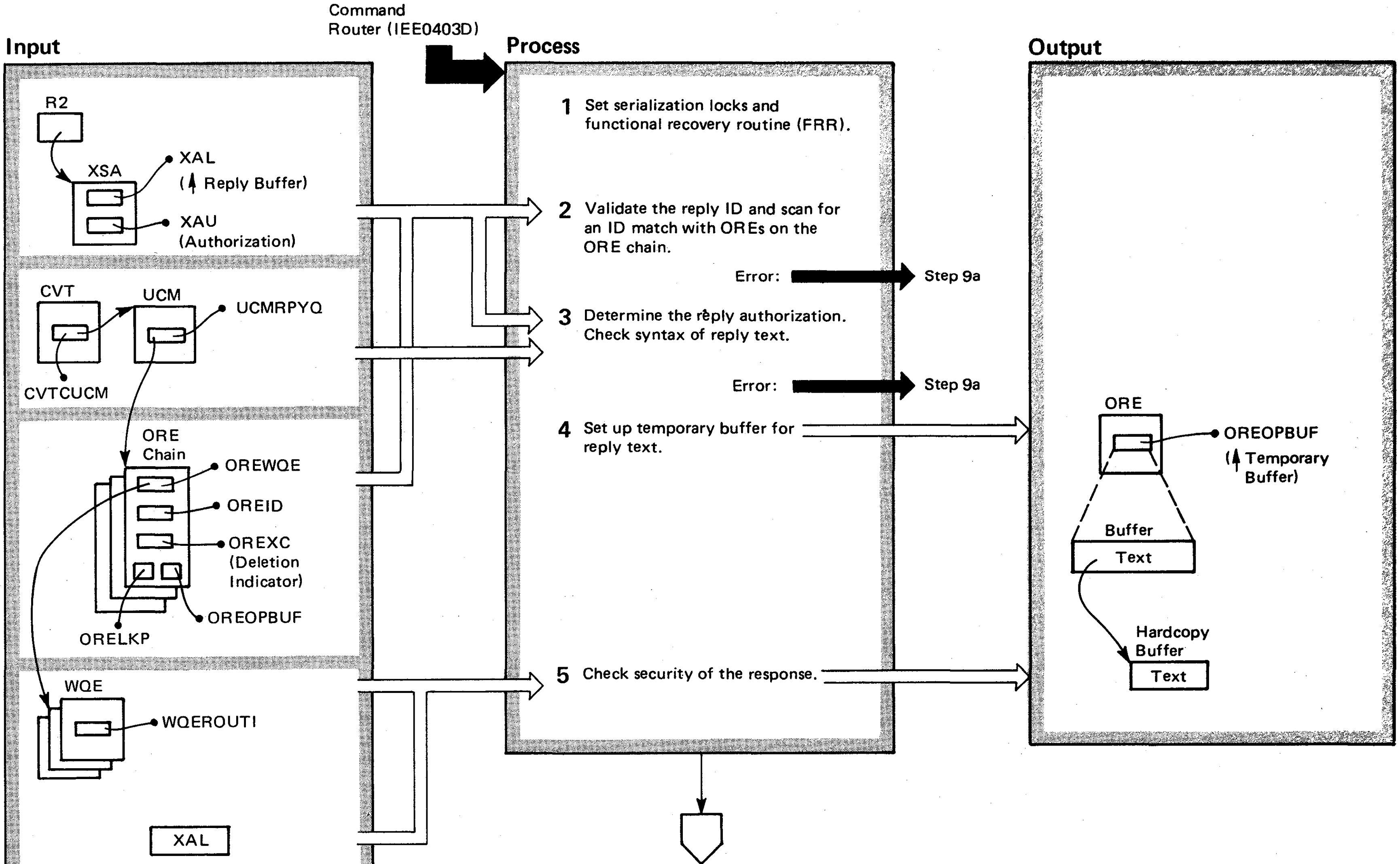
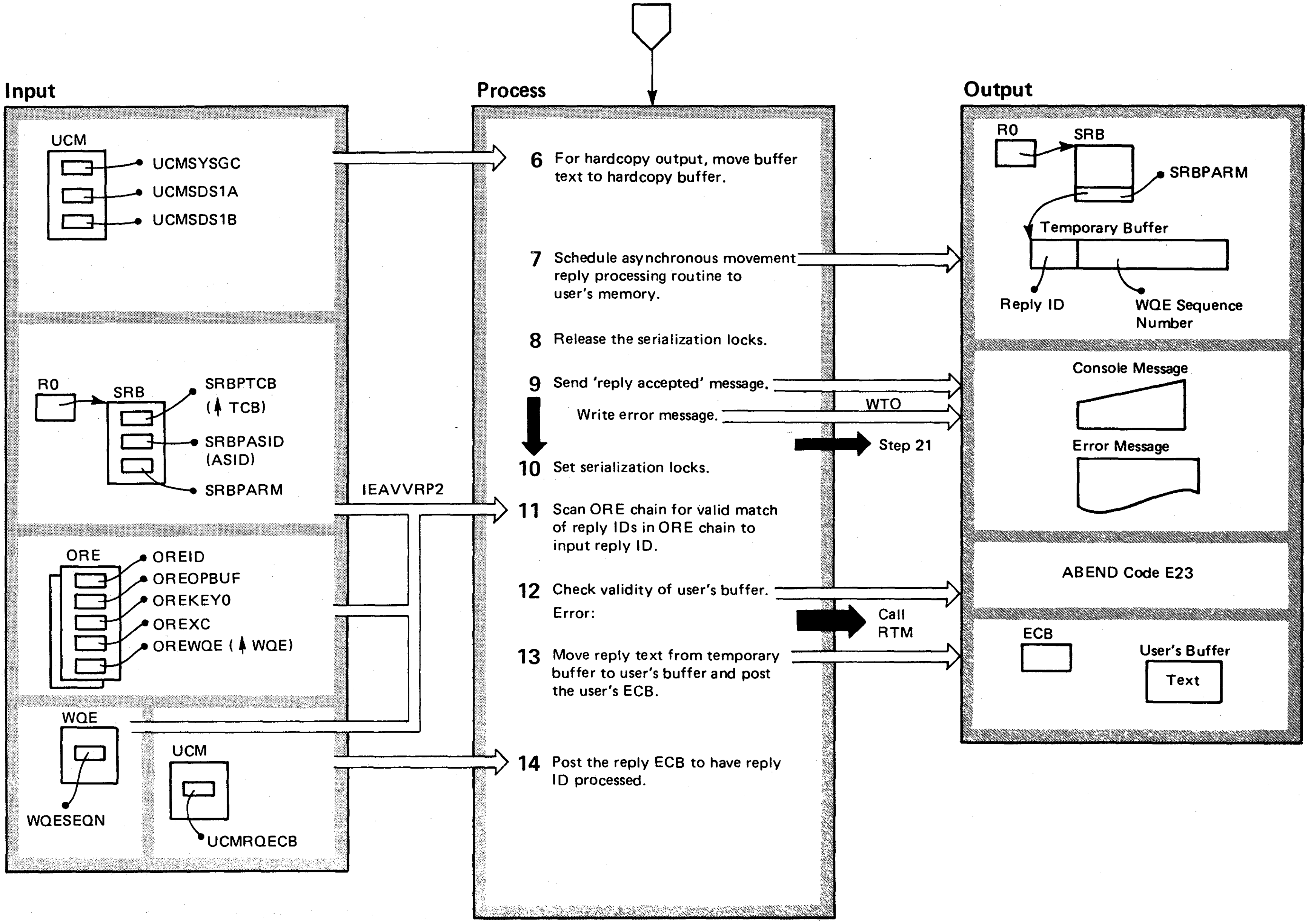


Diagram 2-32. Replying (R) to Information Requests (IEAVVRP1) (Part 2 of 6)

Extended Description	Module	Label
This routine places the text of a message response into a user's buffer and provides a console message to indicate reply acceptance.		
1 Local and CMS locks serialize the use of the ORE and WQE.	IEAVVRP1	SETLOCK
2 If the reply ID is valid (that is, a one or two digit decimal number) the ORE chain is searched for a matching ID. An ORE is not scanned if it is either marked for deletion, marked 'suspended', partially processed, or lacks an associated WQE.		VALIDATE SCANID
3 Input stream authority is indicated by XAU=0. Console authority requires input from either the master console or a console with a matching ID, or a console (whose ID is given in register 0) that is queued conditionally, or a uniquely specified console, or a console that has either route code matching or message type matching.		AUTHOR
For a non-null reply, the reply text must have proper quotation mark syntax and a length compatible with the WTOR user's buffer.		SYNTAXCK
4 A buffer is needed for moving the reply to the user's buffer, which is in the user's memory. The buffer is obtained from the common storage area.		GETBUF
5 When the associated WQE indicates that an operator's reply is a security response (routing code=9), the routine overlays the text portion of the operator's reply with the word "suppressed."		SECURITY

Diagram 2-32. Replying (R) to Information Requests (IEAVVRP1) (Part 3 of 6)



Extended Description	Module	Label
6 A WTO macro instruction indicating "Hardcopy only" writes the message to the hardcopy device.	IEAVVRP1	HARDCOPY
7 The SCHEDULE macro instruction provides for further processing to occur in the user's memory. The service request block (SRB) acts as a cross-memory interface.		SCHED
8 These are the local and CMS locks.	IEAVVRP1	RELEASE
9 This message goes to all consoles that received the original WTOR message.		ACCEPTED
The operator receives the error message.		
10 The local and CMS locks serialize the use of the ORE and WQE.	IEAVVRP2	SCANID
11 The reply ID, received from IEAVVRP1, is compared with ORE IDs. If a match is found with a valid ORE (see step 2), the WQE sequence numbers are compared.		
12 If the user is not in key 0, the routine checks the beginning and ending addresses of the user's buffer, and checks the ECB for a valid value.		VALCHK
13 The reply text is placed in the user's buffer for output and the module posts the user's ECB.		MOVEPST
14 The reply ID (field UCMRPYI) is set to indicate the reply is available. The associated ORE is removed from the ORE chain. The reply ECB is posted with a cross-memory POST macro instruction.		AVALID OREREMN POSTOECB

Diagram 2-32. Replying (R) to Information Requests (IEAVVRP2) (Part 5 of 6)

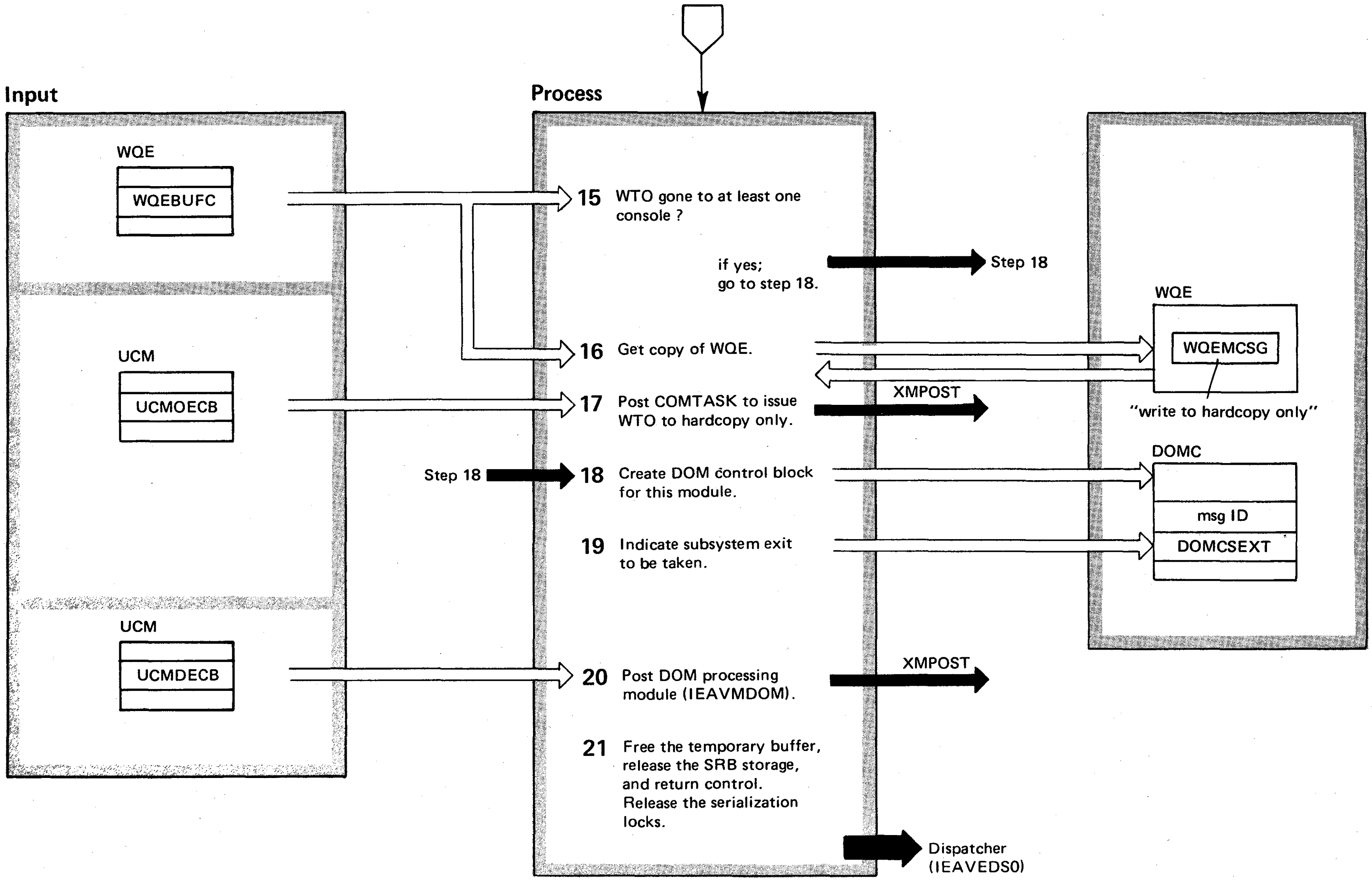


Diagram 2-32. Replying (R) to Information Requests (IEAVVRP2) (Part 6 of 6)

Extended Description	Module	Label
15 Because a reply can be accepted before the message has gone to any console, this condition (the WTO having gone to at least one console) must be tested before the message is deleted from the system.	IEAVVRP2	POSTOECB GETCELL GETEXT
16 If the message has not gone to at least one console, a copy of the associated WQE is made and a flag is set to route the message to hardcopy only.		
17 The COMTASK will be posted to write the message to hardcopy only. This is to insure that there is a copy of the message issued.		
18 To interface with DIDOCS to delete answered WTORs from the graphic screen, a DOM control block is built containing the message ID.		
19 A flag in the DOM control block (DOMCSEXT) is set to indicate the subsystem exit is not taken for this DOMC.		
20 The DOM ECB is posted to process this DOM control block.		
21 The routine releases the resources used in the processing.	FREEBUF CLEANUP	

Diagram 2-33. RESET (E) Command Processing (IEEMB810) (Part 1 of 2)

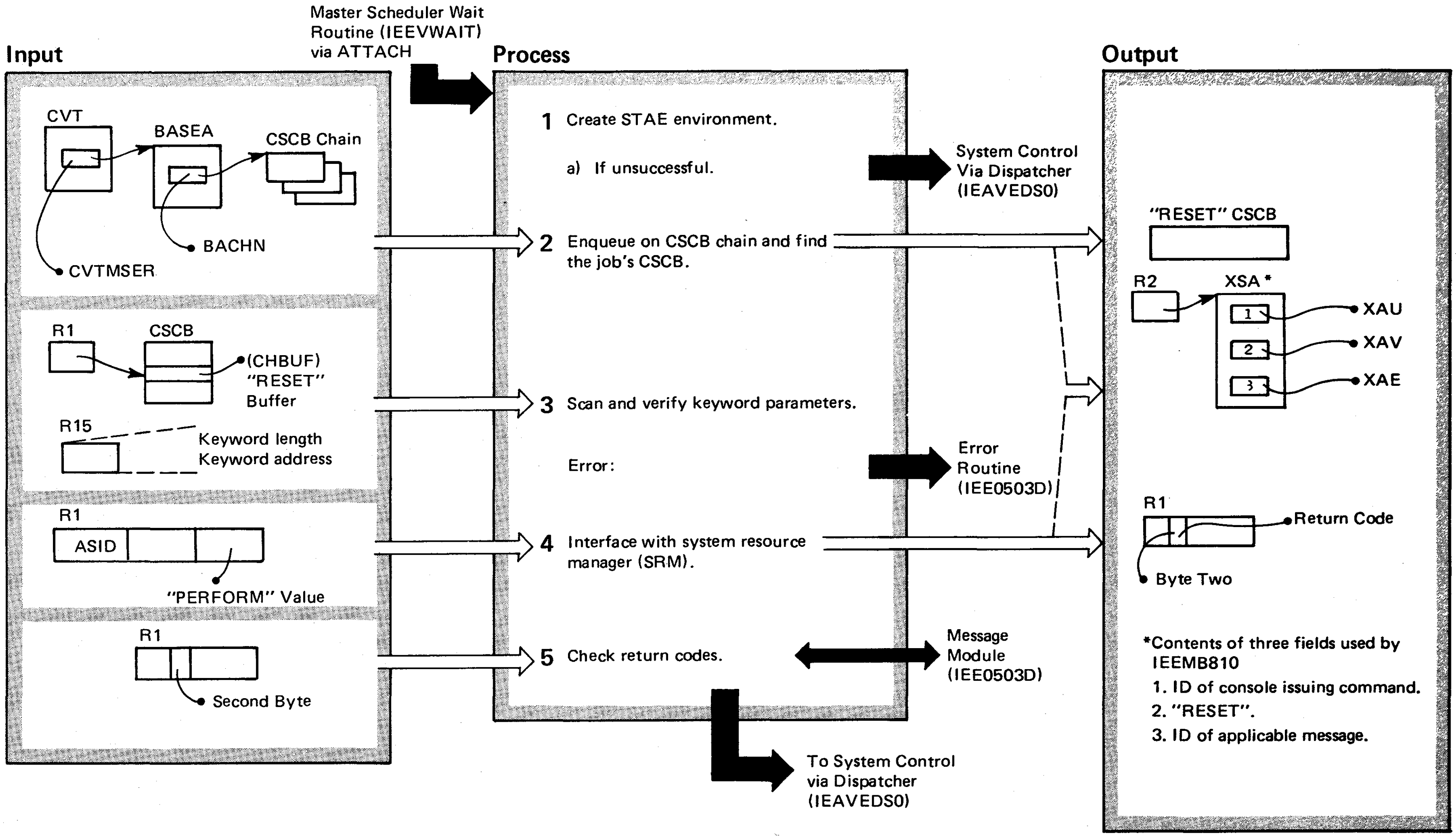


Diagram 2-33. RESET (E) Command Processing (IEEMB810) (Part 2 of 2)

Extended Description	Module	Label
This routine changes the performance group specification for an executing job and passes the information to the system resource manager (SRM). The change is in effect only for the duration of the current job step.		
1 This environment handles ABEND situations.	IEECB860	
a. The CSCB for the command is released before return to the SVC 34 issuer.	IEEMB810	ISSUSTAE
2 If the CSCB is absent from chain, the routine issues an error message.	IEE0503D	
3 The RESET command buffer contains job ID and keyword parameters.	IEEMB810	SCANNAME
The routine uses the IEEBUFSC macro instruction to do the scan.		GETBUFSC
Invalid parameters result in an error.	IEE0503D	
4 The SYSEVENT macro instruction with the RESETPG and ASID options provides the interface to pass information to the SRM to change the performance group values.	IEEMB810	SYSEVENT
5 Return codes other than 4 and 8 indicate performance value change is accepted by the SRM.	IEEMB810	CHECKRC2

Note: Module IEEMB810 constructs a simulated XSA for use by the message module, IEE0503D.

Diagram 2-34. Sending/Saving/Listing (SE) Messages (IEEVSEND) (Part 1 of 3)

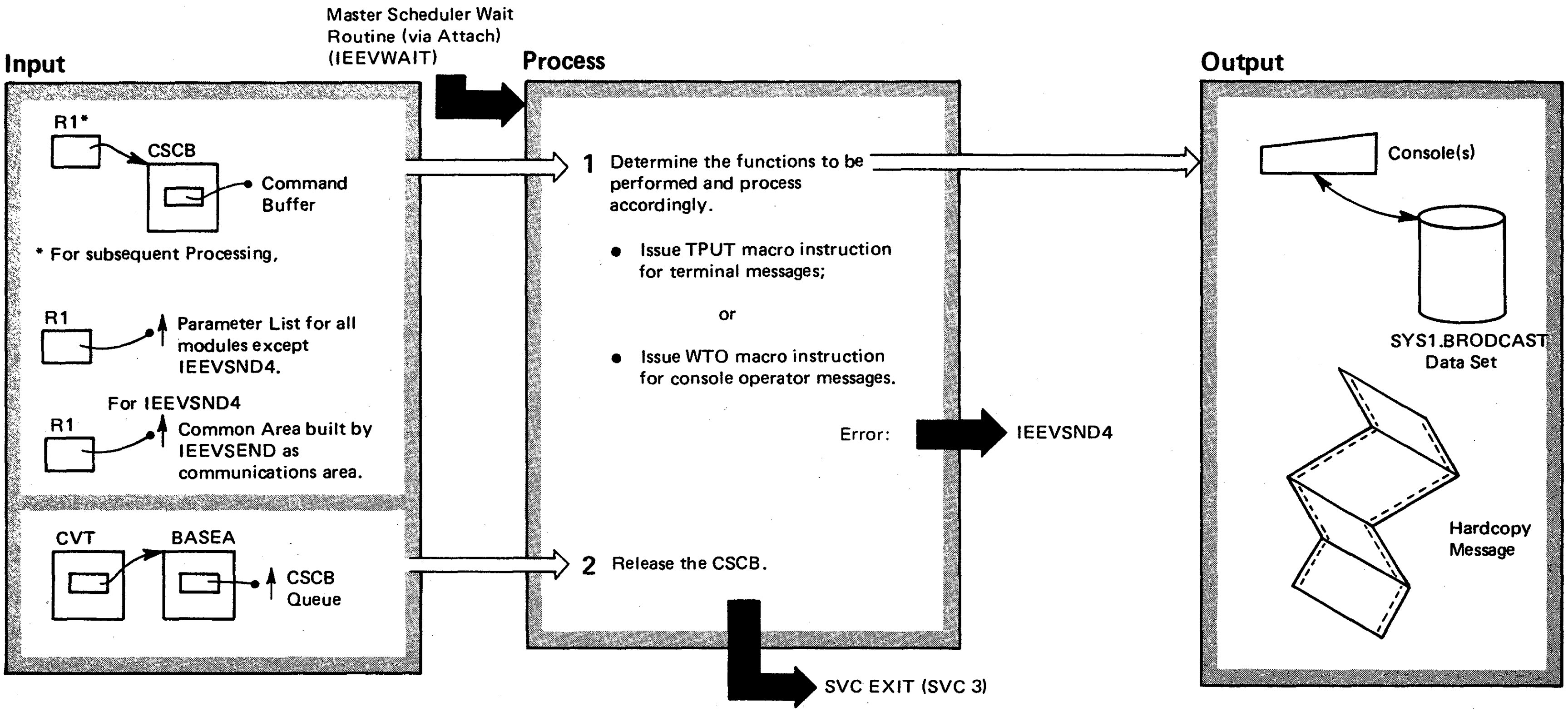


Diagram 2-34. Sending/Saving/Listing (SE) Messages (IEEVSEND) (Part 2 of 3)

Extended Description	Module	Label
This routine provides communication (messages) between an operator and the console by using the SYS1.BRODCAST data set. Prior to the SEND command processor receiving control, preliminary checking and processing has been performed by modules IKJ803D and IEE0803D. (The latter module has built the CSCB for this task processor.)		
1 The command operand field contains necessary function requests. The operand syntax is checked. The processor operates in one of the following modes as indicated by the keyword shown on the command: LIST TEXT MESSAGE NUMBER (MSGNO)	IEEVSEND	IEEVSEND
Based on the processor mode operand and subsequent descriptive operands, processing occurs in the manner, and by the module(s), indicated after the description for step 2. (See Part 3.)		
For error conditions, retrieve the appropriate error message text. Send message either to console operator via WTO macro instruction or to terminal user (in operator mode) via TPUT macro instruction; then return to caller.	IKJEES20 IEEVSND4	
2 Control returns to the calling routine.		

Diagram 2-34. Sending/Saving/Listing (SE) Messages (IEEVSEND) (Part 3 of 3)

Processing description referred to in step 1:

Mode	Operand(s)	Action	Module
TEXT	ALL		
	NOW	Uses high priority TPUT macro instruction to send message to all logged-on users.	IEEVSND6
	LOGON	Uses low priority TPUT to send messages to all logged-on users. Then save the message in the 'notice' section of the broadcast data set.	IEEVSND6 IEEVSND8 IEEVSND5
	SAVE	Saves the message in the 'notice' section of the broadcast data set.	IEEVSND8 IEEVSND5
	USER		
	NOW	Uses high priority TPUT macro instruction to send message to all specified users who are logged on.	IEEVSND6
	LOGON	Uses low priority TPUT macro instruction to send message to all specified users who are logged on and can receive messages. Otherwise, saves the message in the 'mail' section of the broadcast data set for each specified user who did not receive the message.	IEEVSND6 IEEVSND2 IEEVSND5
	SAVE	Saves the message in the mail section of the broadcast data set for each specified user.	IEEVSND2 IEEVSND5
	BRDCST	Uses the WTO SVC (35) to send the message to all active consoles.	IEEVSND6
	CN	Uses the WTO SVC to send the message to the specified console.	IEEVSND6
TEXT	OPERATOR	Uses the WTO SVC to send the message to the indicated functional area.	IEEVSND6
LIST	(none)	Retrieve all messages from the notices section of the broadcast data set and send them either to the console operator via a WTO macro instruction, or to the terminal user (in operator mode) via a TPUT macro instruction.	IEEVSND3 IEEVSND5

Mode	Operand(s)	Action	Module
MSGNO	ALL		
	NOW	Retrieves the specified message from the broadcast data set. Then proceed as for TEXT ALL/NOW.	IEEVSND3 IEEVSND5 IEEVSND6
MSGNO	ALL		
	LOGON	Retrieves the specified message from the broadcast data set. Then uses a low priority TPUT macro instruction to send the message to all logged on users.	IEEVSND3 IEEVSND5 IEEVSND6
	SAVE	This is a meaningless command since the message denoted by 'MSGNO' already exists in the 'notices' section of the broadcast data set. Therefore, no action is taken.	
	USER		
	NOW	Retrieves the message from the broadcast data set and continues processing as for TEXT USER/NOW.	IEEVSND3 IEEVSND5 IEEVSND6
	LOGON	Retrieves the specified message from the broadcast data set and then continues processing as for TEXT USER/LOGON.	IEEVSND3 IEEVSND5 IEEVSND6 IEEVSND2 IEEVSND5
	SAVE	Retrieves the specified message from the broadcast data set and proceeds as for TEXT USER/SAVE.	IEEVSND3 IEEVSND5 IEEVSND2 IEEVSND5
MSGNO	BRDCST	Retrieve the specified message. Use WTO SVC to send the message to all active consoles.	IEEVSND3 IEEVSND5 IEEVSND6
	CN	Retrieve the specified message. Use the WTO SVC to send the message to the specified console.	IEEVSND3 IEEVSND5 IEEVSND6
	OPERATOR	Retrieve the specified message. Use the WTO SVC to send the message to the indicated functional area.	IEEVSND3 IEEVSND5 IEEVSND6
	LIST	Retrieve the specified message. Send it to either the console operator via a WTO macro instruction or to the terminal user in operator mode via a TPUT macro instruction.	IEEVSND3 IEEVSND5
	DELETE	Delete the specified message from the 'notices' section of the broadcast data set.	IEEVSND8 IEEVSND5

Diagram 2-35. Setting (T) Local Time (IEE0603D) (Part 1 of 4)

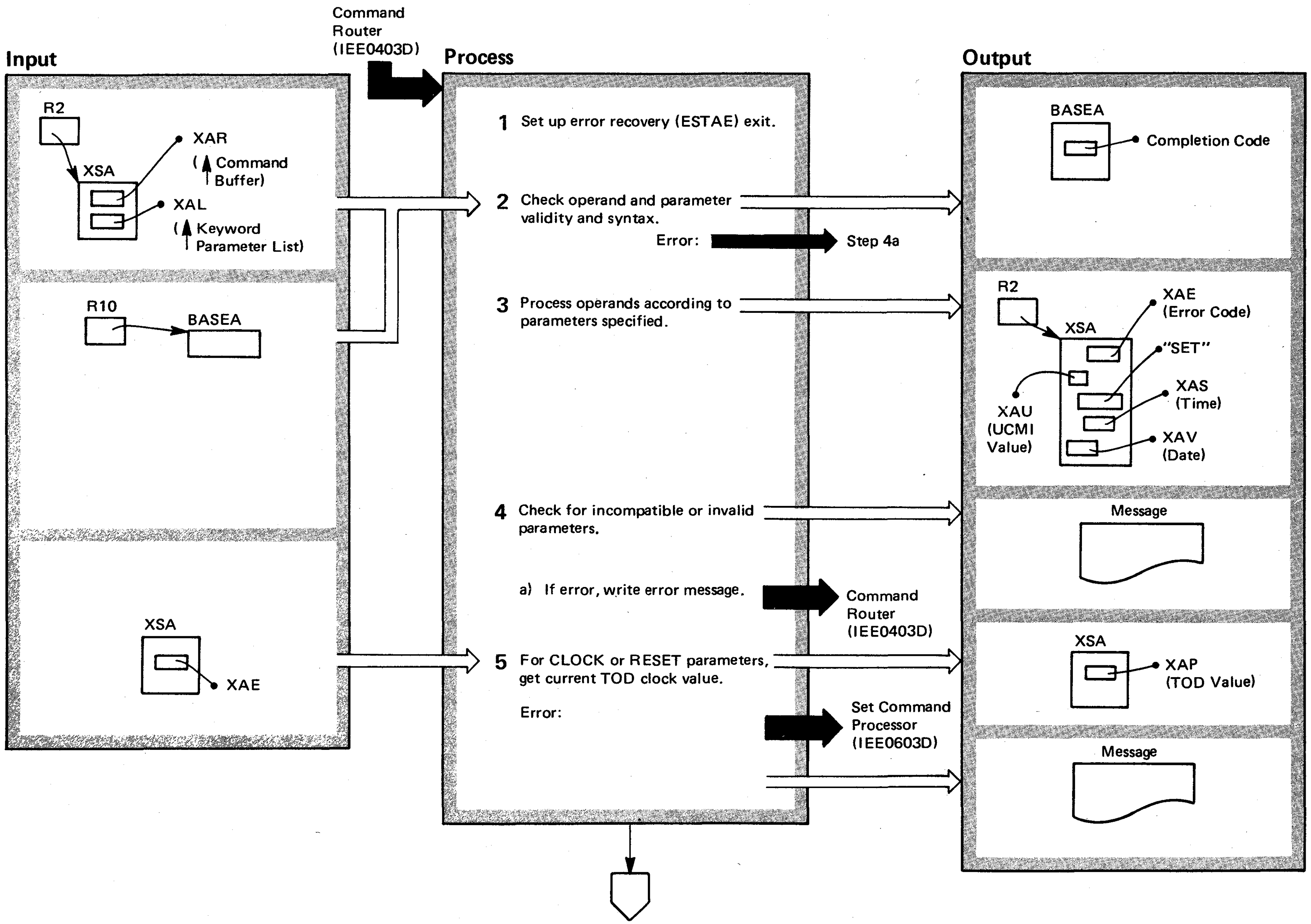


Diagram 2-35. Setting (T) Local Time (IEE0603D) (Part 2 of 4)

Extended Description	Module	Label
This routine satisfies requests to change the local date and time of day after the initial IPL setting.		
1 For unexpected failures (for example, program checks). This subroutine gets control, sets the completion code, and return to the supervisor's recovery termination management routines.	IEE0603D	
2 The parameters are checked against the values in a keyword table.		SNEWKY
3 The appropriate operand values are "packed" into a save area.	IEE0603D	SCLOCK SDATE
4 For example, specifications of the RESET *and* DATE parameters, or specification of the GMT parameters.		
5 The routine uses a STCK (store clock) instruction on the CPU that is currently executing. If this procedure is unsuccessful, the routine issues a TIME macro instruction to obtain the current value of the TOD clock setting from another CPU. If this processing fails, the operator receives notice that his request cannot be satisfied — that is, the system lacks a *good* TOD clock.	IEE6503D IEE6603D	

Diagram 2-35. Setting (T) Local Time (IEE0603D) (Part 3 of 4)

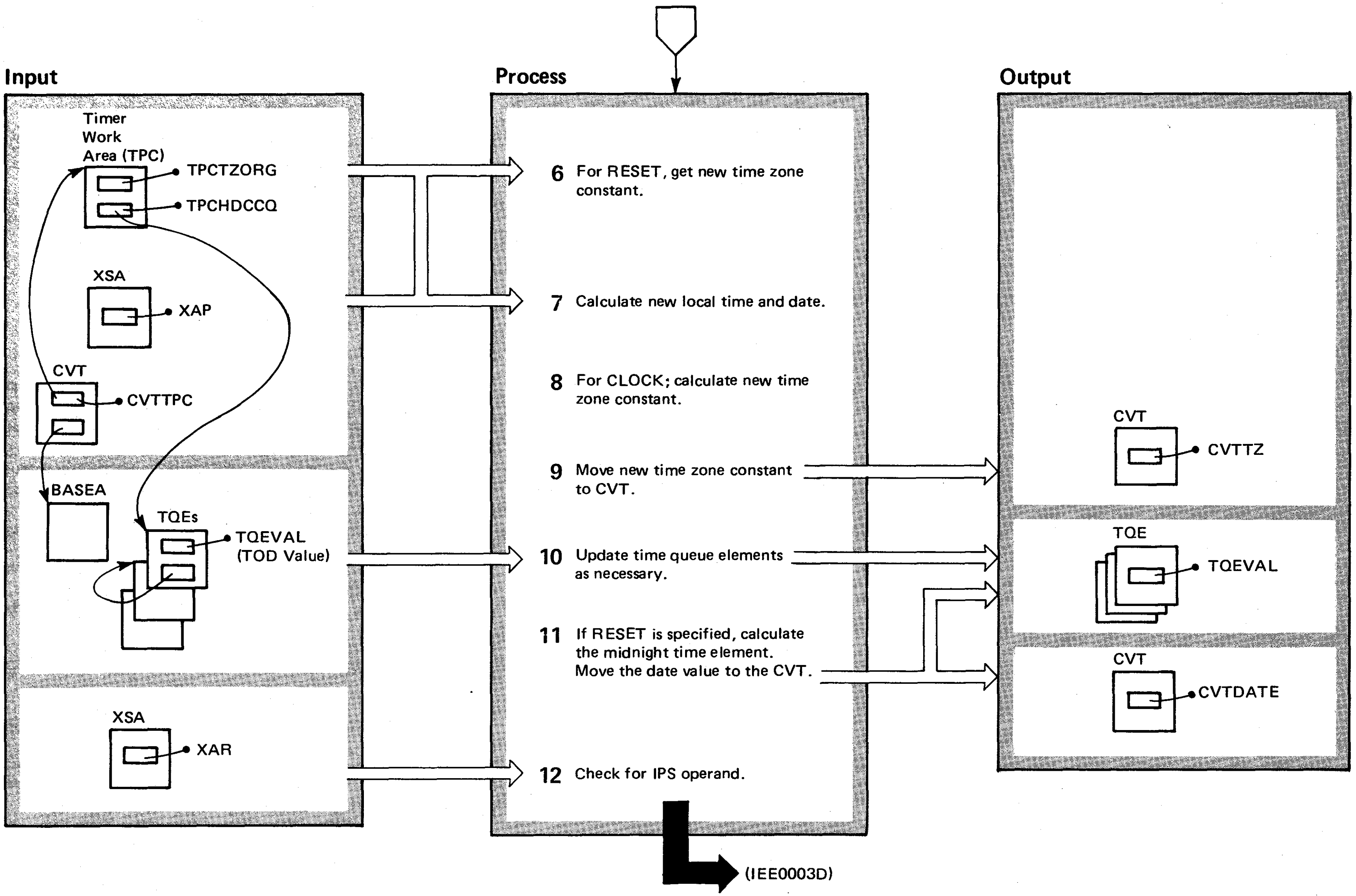


Diagram 2-35. Setting (T) Local Time (IEE0603D) (Part 4 of 4)

Extended Description	Module	Label
6 This IPL-generated constant is needed to reset the local time. The value was saved in field TPCTZORG at IPL time.	IEE6503D	
7 The routine uses the TOD value previously obtained (see step 5) and the time zone constant (from field TPCTZORG – see step 6) for this process.		
8 The routine uses the stored TOD clock value and the operator-entered clock value.		
9 The time zone constant is stored in the CVT.		
10 The TQE queue search ends when a dummy TQE (that is, the last one on the chain) is found. The elements are updated with a previously calculated correction factor. The correction factor equals the difference between the old and new TOD values. If the only operand is DATE, the factor=0.		
11 If other than RESET, the routine uses the correction factor (see step 10) to update the midnight time element. It also moves the date (if specified) to the CVT.		
12 If the IPS operand is found, control goes to module IEE0803D. If it is not found, return control to IEE0003D, via a branch on register 14.	IEE0603D	

Diagram 2-36. Changing IPS Values (T IPS) (IEEMB811) (Part 1 of 2)

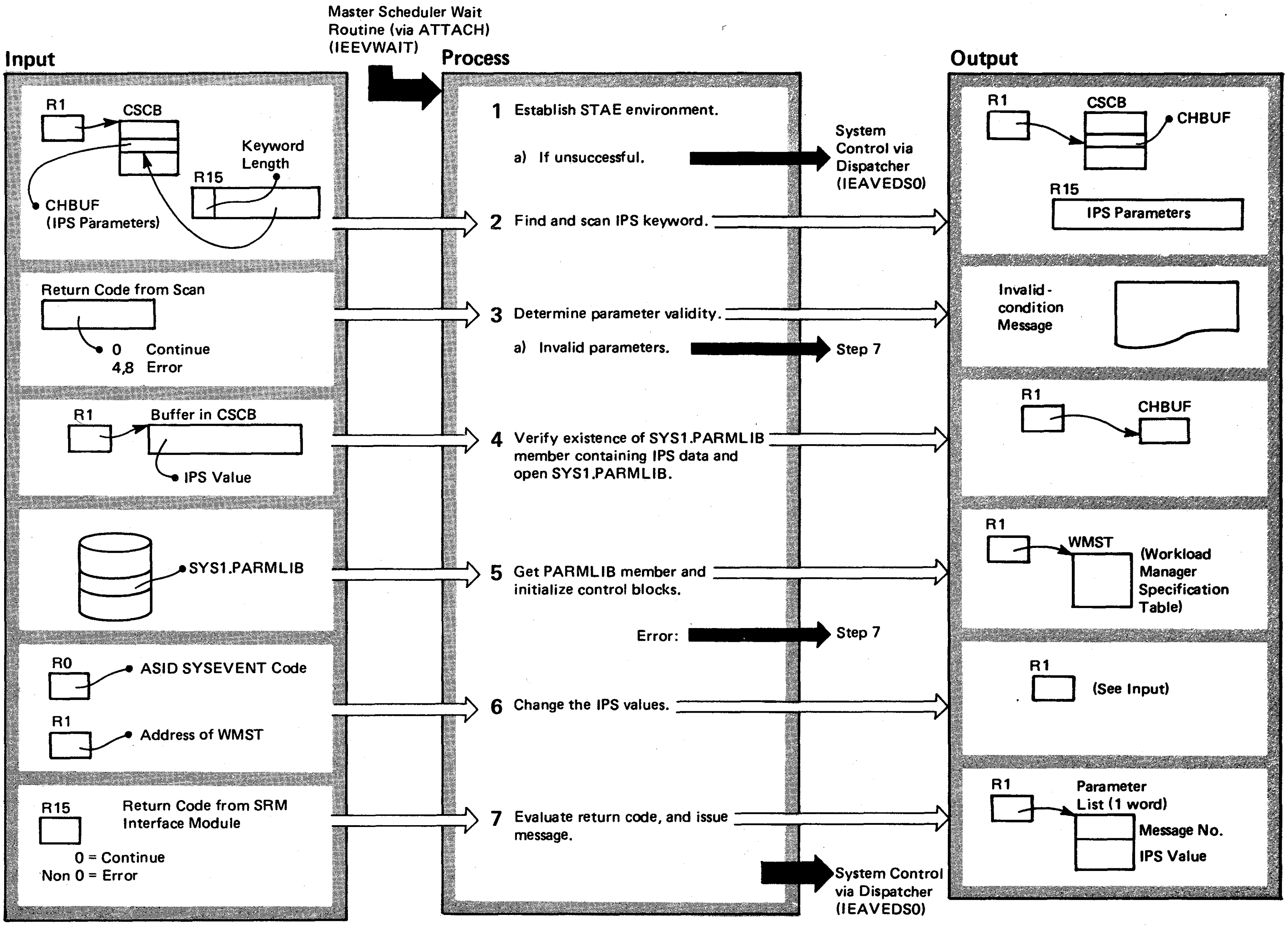


Diagram 2-36. Changing IPS Values (T IPS) (IEEMB811) (Part 2 of 2)

Extended Description	Module	Label
This process changes the system installation performance specifications (IPS) under which a system is operating.		
1 This environment handles ABEND situations.	IEECB860	
a. Storage for the appropriate CSCB is released before returning control.	IEEMB811	ISSUSTAE
2 The IEEBUFSC macro instruction searches the buffer (CHBUF) for the IPS keyword and associated parameters.	IEEMB811	SCANNAME
3a The error message is issued by either of two modules, based on message information in the one-word parameter list. For module IEE0503D, module IEEMB814 constructs a simulated XSA to contain the message information. Module IEEMB814 gives control to module IEE0503D for messages based on a scan of the IPS keyword of the SET command.	IEEMB814 IEE0503D	
4 The PARMLIB member defines the System Resource Manager (SRM) interface to set up for the IPS modifications.	IEEMB812	

Extended Description	Module	Label
5 The SRM performs a syntax scan on the IPS list in SYS1.PARMLIB and builds the new WMST.	IRARMIPS*** IRBMFANL*	
6 The system resources manager receives control via the SYSEVENT NEWIPS macro instruction.	IRARMEVT	IGC095 (via SVC 95)
SRM invokes the set-to new-IPS subroutine via IRARMEVT. This routine places each user into a valid performance group in the new IPS. It then indicates these changes to the workload manager. Finally it posts the SET IPS keyword processor and updates the pointers to the new IPS in the SRM control table (RMCTWMST). The address of the old WMST is returned to permit freeing its area.	IRARMINT IRARMEVT** IRARMSET IRARMWLM	 IRARME32 IRARMIPS IRARMWMN
7 Control actually returns from SRM via IEEMB812. Module IEEMB814 either writes a message based on the return code from module IEEMB812 or gives control to module IEE0503D for the message based on a scan of the IPS keyword of the SET command.	IEEMB811 IEEMB814	CKRETURN

*This module is described in the MF/1 section of this book.

**This module is described in the System Resource Manager section of this book.

***This module is described in *OS/VS2 System Initialization Logic.*

Diagram 2-37. Stopping Periodic Track (Status) (PT) Displays (IEE7503D) (Part 1 of 4)

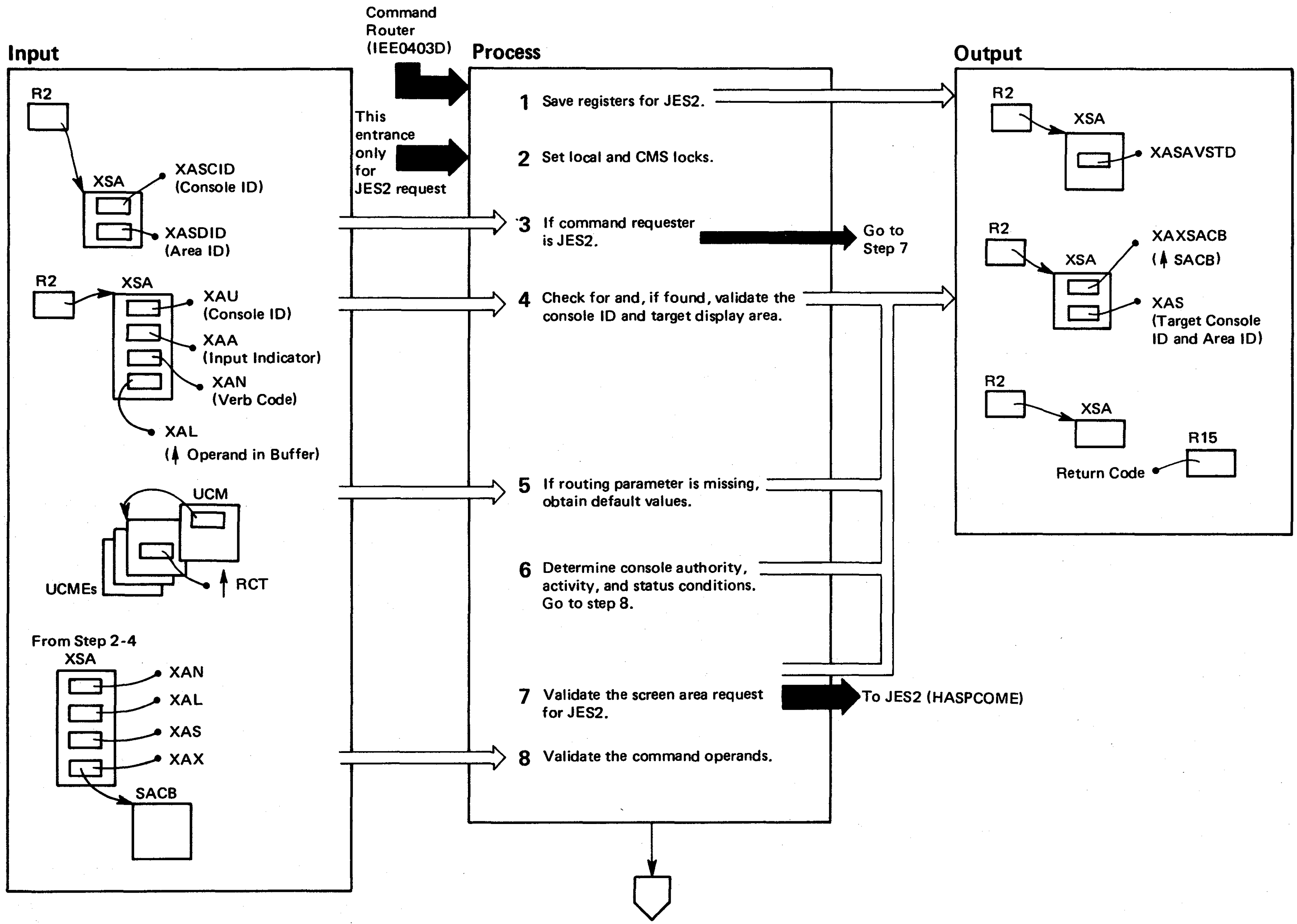


Diagram 2-37. Stopping Periodic Track (Status) (PT) Displays (IEE7503D) (Part 2 of 4)

Extended Description	Module	Label
This routine halts or reduces the display of information being produced as a result of a previous TRACK command.		
1 This step and input (to step 3) applies to JES2-issued command requests only.	IEE7503D	IEE7603D
2 The locks serialize the use of the SACB and the UCME.	IEE7503D	
3 For JES2 command requests, console validation processing is bypassed.		
4 The XAA field in the XSA must contain X'00' indicating a non-TSO terminal input. (A TSO terminal request for this processing is invalid.) The target display area is the screen area that has been receiving the information.		
5 Use as default values either those values in the routing control table (RCT) built to satisfy a previous MSGRT command for TRACK command processing or the routing parameter ("L") information applying to the requesting console.		
6 An example of console parameters follows: • A target CRT console must be active. • A source/receiving console must have routing authority. • A target CRT console must have a requested screen area that is defined by a screen area control block (SACB) and that contains an active track display.		
7 Determine the availability of a valid, active console with a free area that can receive status displays.	IEE7503D	JES2CODE

Return codes passed to JES2:

code	meaning
0	Valid routing request (that is, the display may be written).
4	Area contains a status display.
8	Area contains a track (dynamic) display.
12	Console is invalid.
16	Area is invalid.

Extended Description	Module	Label
8 Check that the TS, A, or JOBS operand is correctly specified.	IEE5503D	IEE0503D

Diagram 2-37. Stopping Periodic Track (Status) (PT) Displays (IEE7503D) (Part 3 of 4)

Process

9 Turn off all existing option indicators in SACB.

If options remain, release locks. → SVC 34 (IEE0003D)

Otherwise, go to step 10.

10 Post the TRACK command processor.

11 Release locks.

12 Post DIDOCS routines.

→ (IEE0003D)

Output

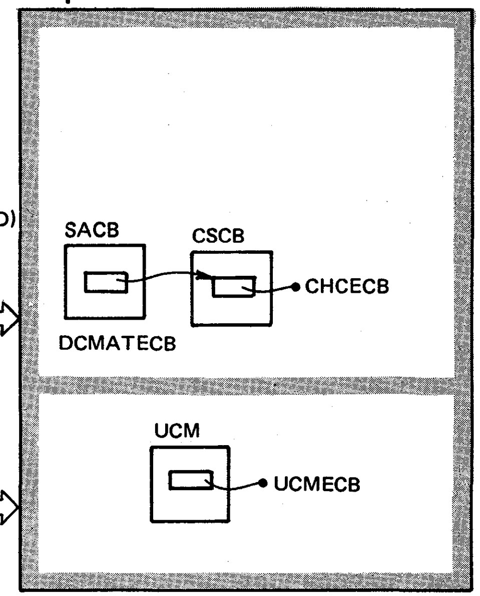


Diagram 2-37. Stopping Periodic Track (Status) (PT) Displays (IEE7503D) (Part 4 of 4)

Extended Description	Module	Label
9 The routine releases the locks that module IEE7503D had previously set.	IEE5503D	TRACKOFF
10 The TRACK processor provides clean-up functions prior to returning to the caller when the task terminates.	IEE6703D IEECB800	
11 The routine releases the local and CMS locks.		
12 The DIDOCS routines clear the CRT screen area.		

Diagram 2-38. Unloading (U) I/O Devices (IEEMB813) (Part 1 of 2)

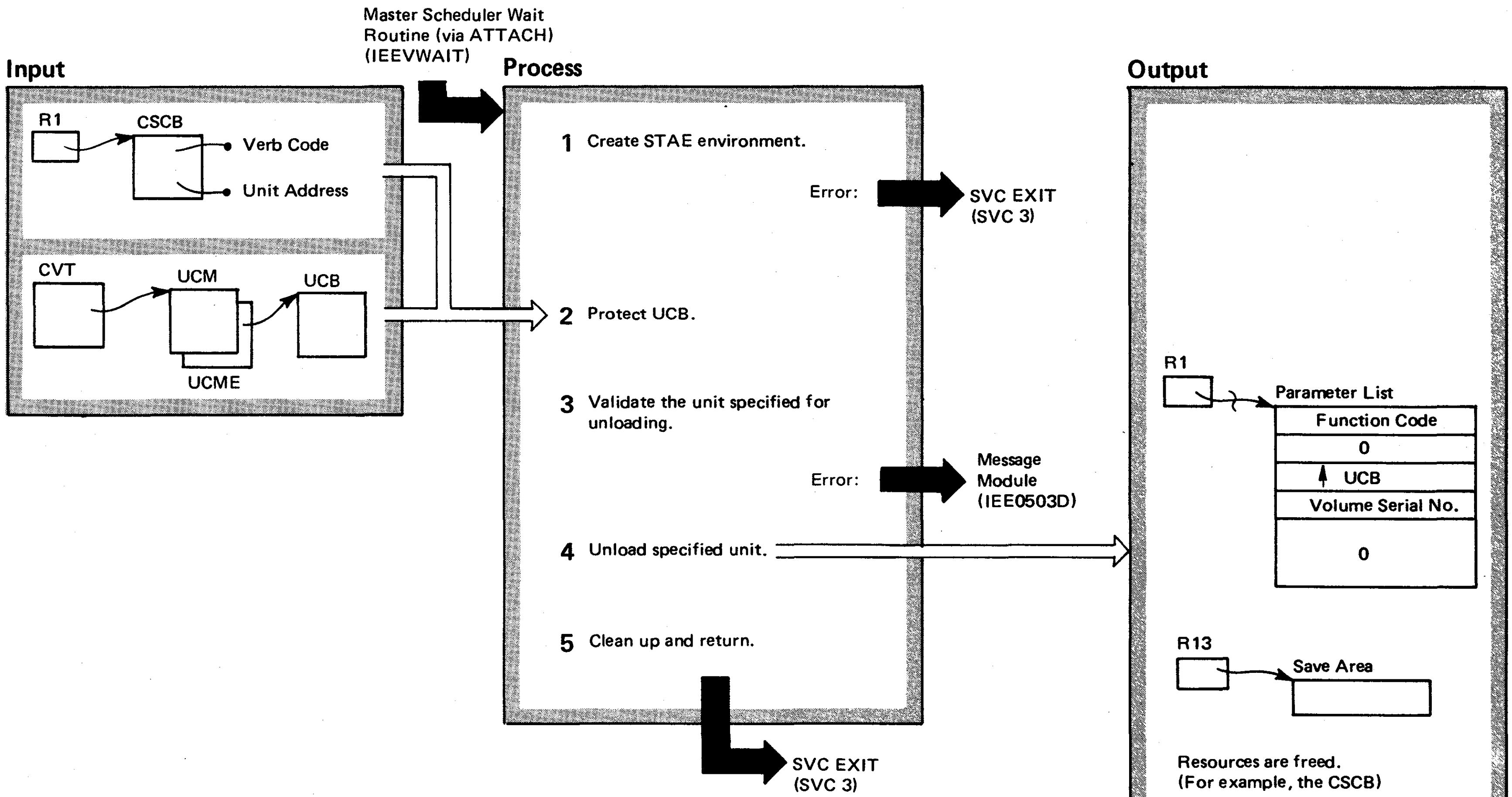


Diagram 2-38. Unloading (U) I/O Devices (IEEMB813) (Part 2 of 2)

Extended Description	Module	Label
This routine prepares a unit (device) for physical demounting (if desired) by logically unloading the unit. The routine operates in the master scheduler's region.		
1 To process for ABEND situations. If the STAE environment cannot be created, further "unload" processing ends.	IEEMB813 (IEECB860)	ISSUESTE
2 The ENQ/DEQ feature provides protection against use of the UCB by either another UNLOAD command, allocation routines, or by VARY command routines.		
3 For a unit to be unloaded immediately, the following must be in effect:		
• The unit address must have 3 characters.		
• The Unit must have a UCB.		
• The Unit must be either a tape or a direct access unit.		
• The unit must be other than a system-resident or permanently-resident device.		
• The unit must currently be on line.		
• The unit must be ready or available for unloading.		
• The unit must currently be unallocated.		
If all (of #3) except the last item is true, a UCB indicator is set to defer the unloading until allocation/termination routines get control.	IEEMB813	IEEMB813
4 The unit is logically removed, and if it is a tape, it is unloaded.	IEFAB49C	
5 The MGCR macro instruction is used to free the CSCB.	IEEMB813	

Diagram 2-39. Routing of VARY (V) Commands (IEE3203D) (Part 1 of 2)

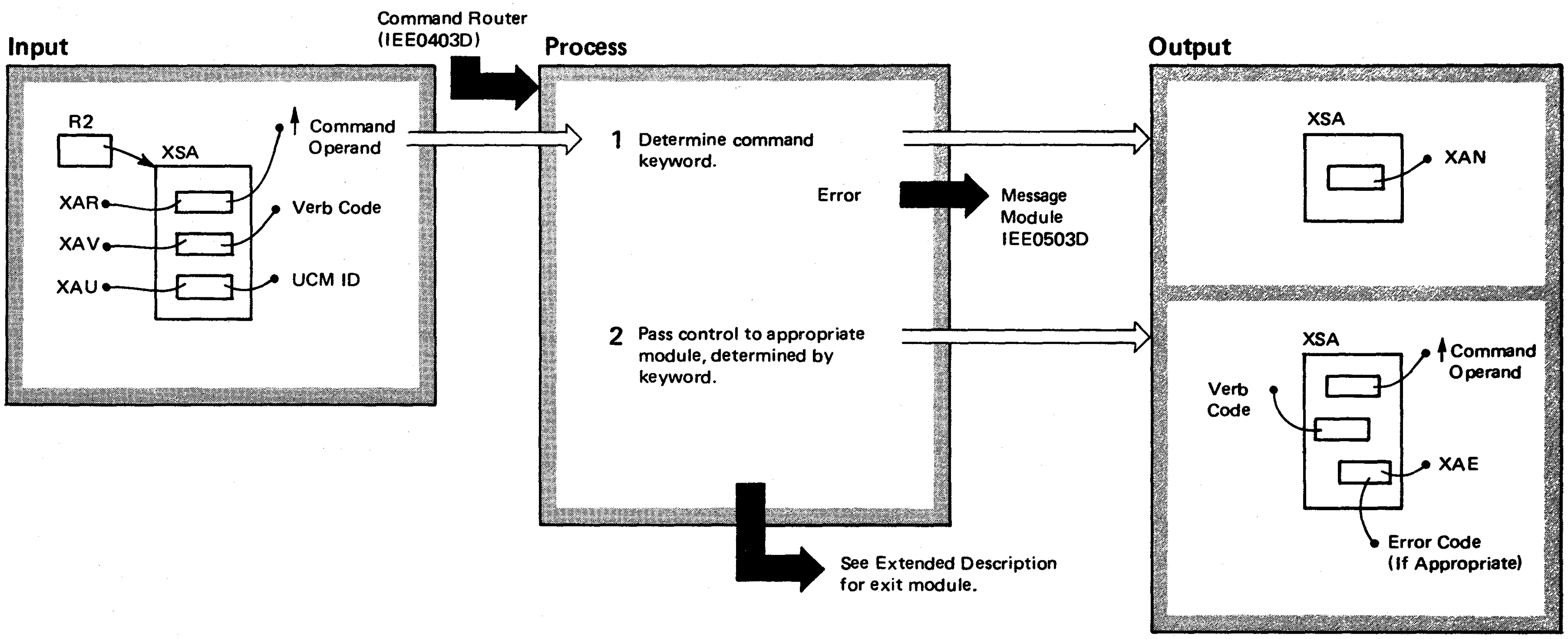


Diagram 2-39. Routing of VARY (V) Commands (IEE3203D) (Part 2 of 2)

Extended Description	Module	Label
This routine determines the correct module to handle an input VARY command.		
1 Compare the command keyword with table of acceptable values. First determine if the command contains a primary keyword. If none, check for secondary keyword.	IEE3203D	PRIMKEY

Primary Keyword: NET, PATH, CPU, STOR, CH, CN

Secondary Keyword: ONLINE, OFFLINE, CONSOLE, HARDCPY, MSTCONS, ONTP/OFFTP, ONLINE,S, OFFLINE,S

2 Keyword options are tested in IEE3203D. Error conditions are tested and error messages issued, if necessary, as follows:

Error	*Message ID*
delimiter error	IEE307I
term length error (embedded blank)	IEE308I
undefinable keyword	IEE309I
parameter missing	IEE311I
parameter conflict (incompatible keywords)	IEE312I
command length exceeds maximum (excessive length of total operands)	IEE908I

	Module	Label
If NET, processing control goes to:	ISTCFF3D	
If ONTP or OFFTP, processing control goes to TCAM.	IED1303D	
If HARDCOPY, processing control goes to:	IEE4703D	IEE4703D
If MSTCONS, processing control goes to:	IEE4303D	IEE4303D
If ONLINE,S or OFFLINE,S, processing control goes to:	IEE9403D	
Otherwise, processing control goes to: (A CSCB must be created, and the appropriate routine will be attached by the master scheduler wait routine.)	IEE0803D	

Diagram 2-40. Changing (V) Console Status, Routing Codes, and Command Authorization (IEE3603D) (Part 1 of 6)

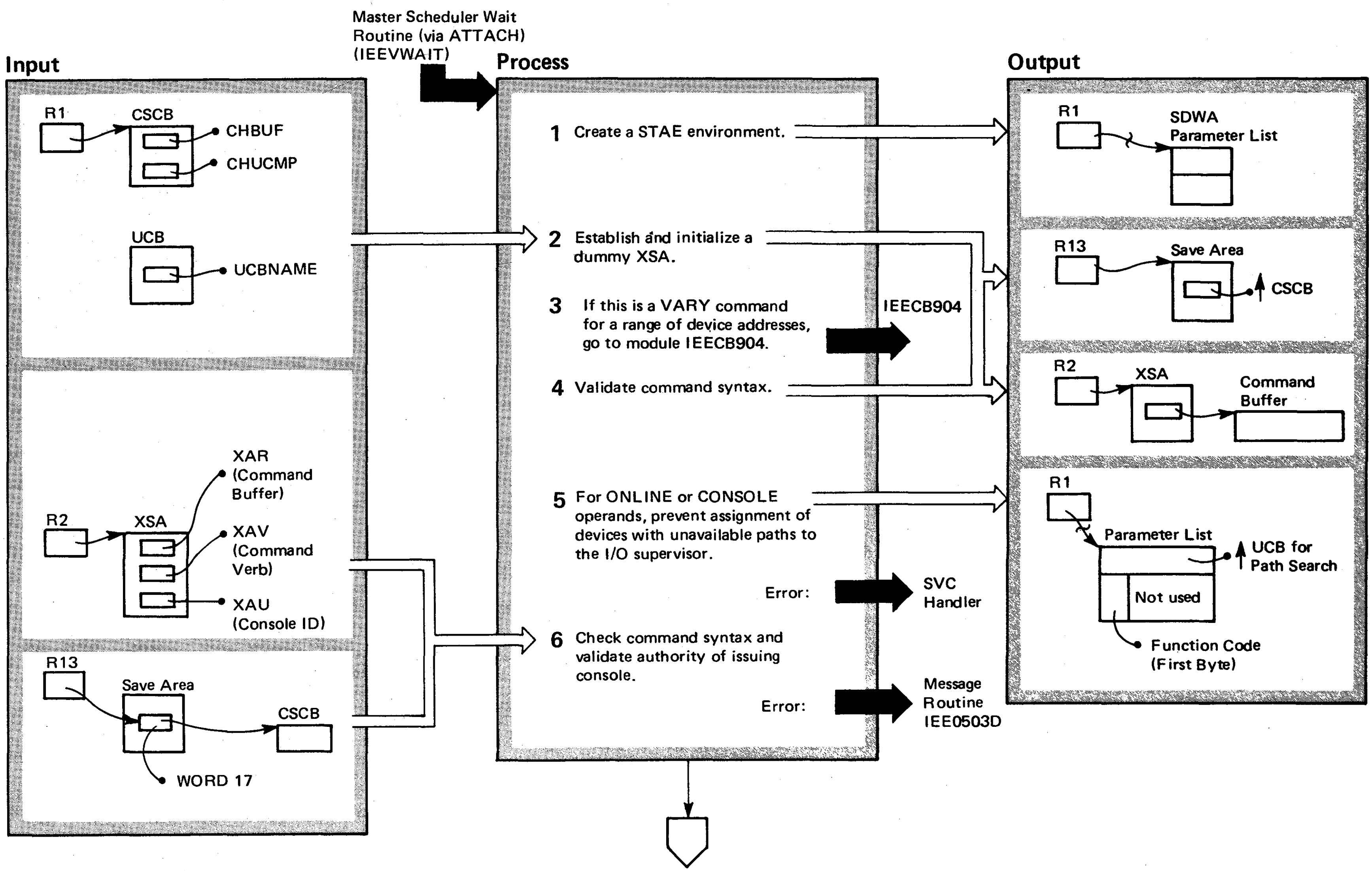
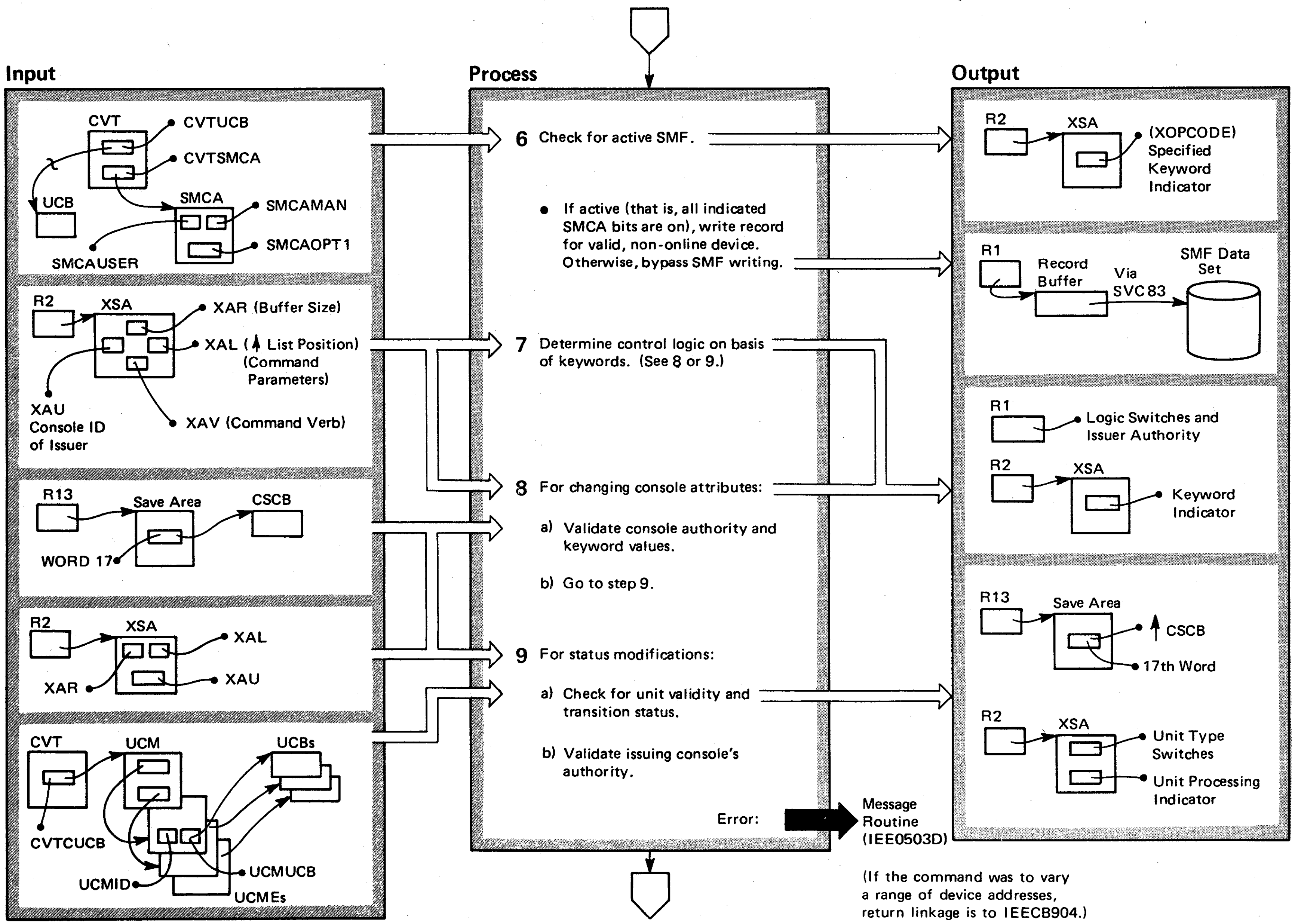


Diagram 2-40. Changing (V) Console Status, Routing Codes, and Command Authorization (IEE3603D) (Part 2 of 6)

Extended Description	Module	Label
This procedure makes a device unavailable for use as a console and either available or unavailable for system use.		
1 For ABEND protection purposes.	IEECB860	
2 The online/offline/console processors will use this XSA for reference.	IEE3603D	XSAINIT
3 If this is a VARY command for a range of device addresses, branch to module IEECB904 to establish the necessary work areas and return here for normal VARY processing.		
4 Enqueue/dequeue environment provides protection against contention for UCBs. When checking the device operands, the enqueuing function gives protection against allocation, OLTEP, and another VARY command.	IEE3603D	IEE36020
5 Use path checking subroutine IEEVDEV to look for access paths to the device.	IEE3603D	IEE361A0
6 A unit specified as 'input only' is invalid. Authority (which must be either 2 or 3) depends on the keyword specified. The master console or the hard copy log function must be assigned to another console before a request to vary the console having it offline or online is issued.	IEE3303D	CONTINUE

Diagram 2-40. Changing (V) Console Status, Routing Codes, and Command Authorization (IEE3603D) (Part 3 of 6)



Extended Description	Module	Label
6 If job accounting is requested, an SMF Type 9 record indicating changing devices can be written to the SYS1.MANx data set. (The routine uses SVC 83 for this step.) Unit specifications must refer to devices with proper availability and capabilities. (This availability includes not being tested by OLTEP.)	IEE3303D IEE2303D	SMFRTN IEE2303D
7 For secondary status change (ONLINE or OFFLINE) or assigning a console (CONSOLE), control goes to the UCME scan/router routine, IEE4203D. For changing console attributes (AUTH=, ROUT=, etc), control goes to the keyword scanner for the VARY command routine (IEE4403D) and then to IEE4203D.	IEE3303D	NOSMF
8 Only devices specified as consoles at system generation time may be varied by the CONSOLE command.	IEE4403D	KEYSCAN
9 Specified units must be available for use as requested: a) A device that is specified as an alternate console must be different from the primary console. b) The command authorization for a V CONSOLE command must be 3.	IEE4203D	UCOMP2

If a VARY command is being processed for a range of device addresses when an error message is issued, go to IEECB904 to process more ranges. If no more ranges are to be processed, clean up the work area and exit from IEECB904.

Diagram 2-40. Changing (V) Console Status, Routing Codes, and Command Authorization (IEE3603D) (Part 5 of 6)

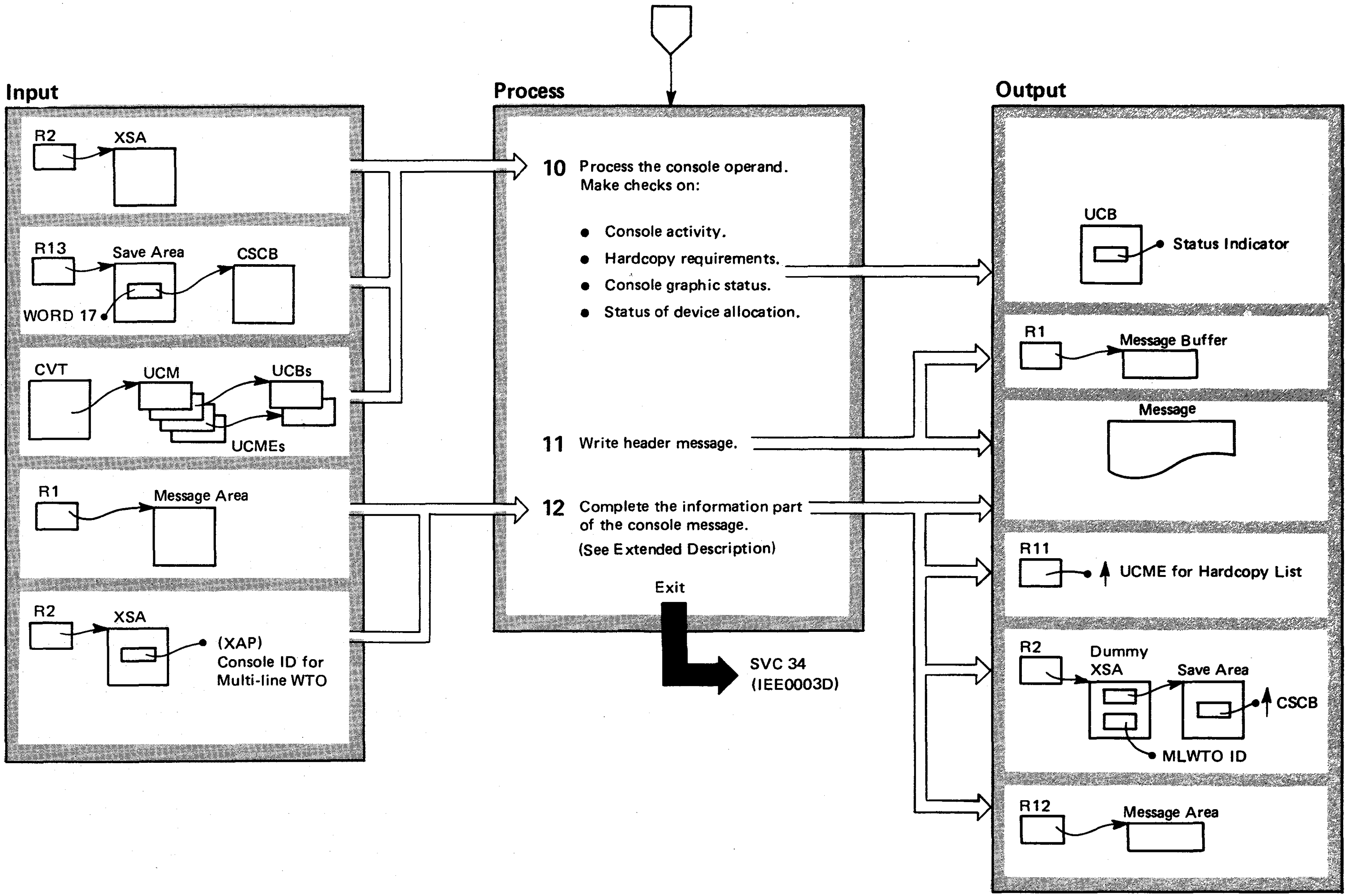


Diagram 2-40. Changing (V) Console Status, Routing Codes, and Command Authorization (IEE3603D) (Part 6 of 6)

Extended Description	Module	Label
10 System local locks provide protection against the communications task.	IEE4903D	CSETLOCK
The processing at this step involves the following considerations:		
• Determining if the device to be varied to the console state is already a console.		
• A hard copy device is required if the system has two or more consoles.		
• With two graphic devices available, one unit will act as a console.		
• If a device is unallocated, a bit is set in the UCB and processing continues.		
11 Issue MLWTO macro instruction for header portion of message.	IEE4903D	CHEADER
12 The first module named actually fills in the message buffer and the second module issues the MLWTO to the operator for the balance of the message.	IEE4803D* IEE7303D	CPROC GRPXH+

*The output register contents shown for this step refer to the output from IEE4803D when it gives control to IEE7303D.

Diagram 2-41. VARY Console (V CN) Processing (IEECB900) (Part 1 of 2)

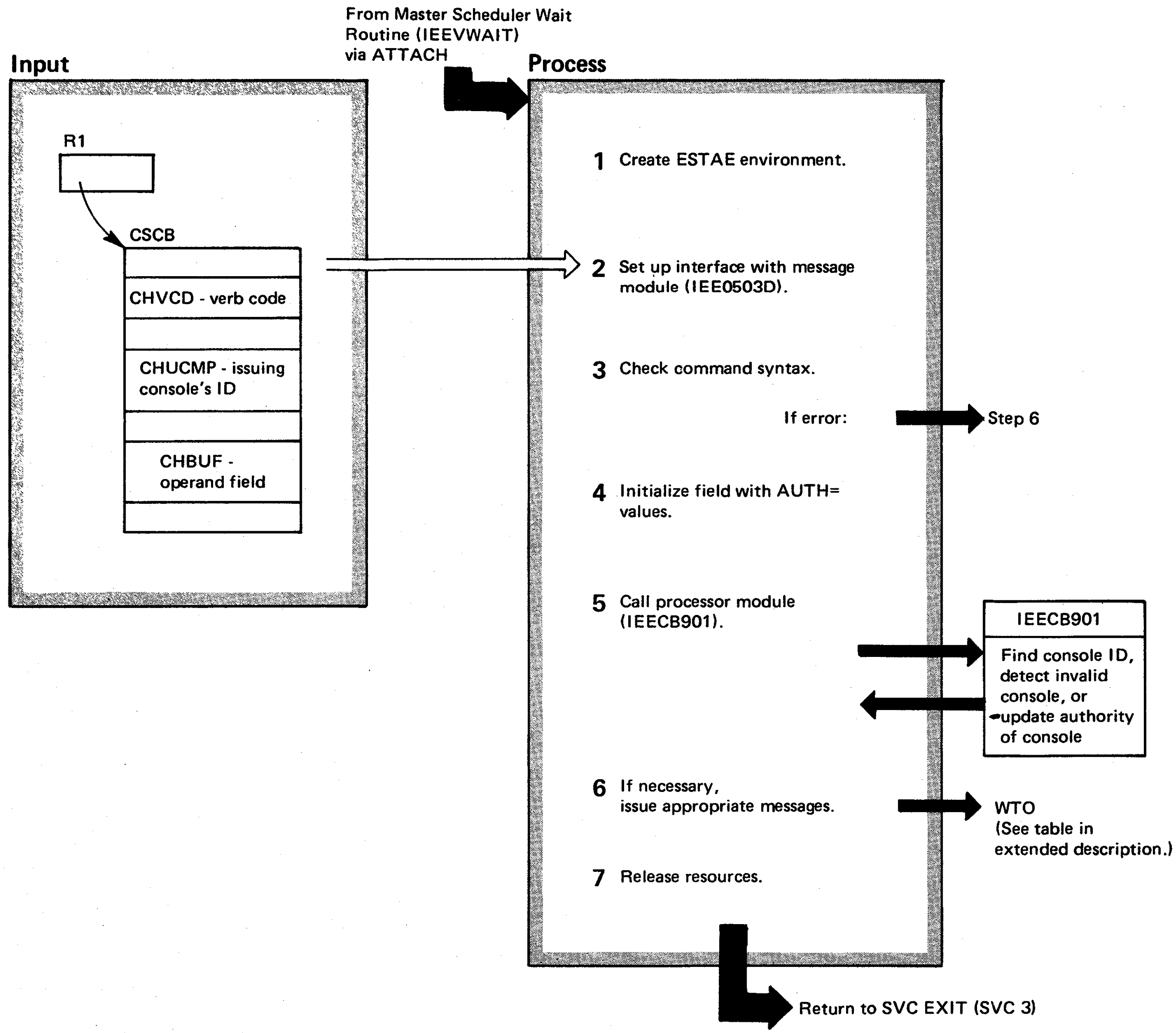


Diagram 2-41. VARY Console (V CN) Processing (IEECB900) (Part 2 of 2)

Extended Description	Module	Label
This processing changes the console authority for consoles other than the master console. The command is processed by the VARY Router, and the module is attached in the Master Scheduler region. The module checks that the command is syntactically correct, and verifies that the console is eligible to change console authority values.	IEECB900	
1 Creates an ESTAE environment via module IEECB860. If the return code in register 15 is not zero, goes to cleanup at step 7, terminating the command.		STAERTN
2 Loads message module IEE0503D to obtain its entry point address which is saved for later use in issuing messages (step 6).		
3 Verifies that the command has balanced parentheses on CN parameter, correct length for console ID(s), and decimal numeric value for console ID(s). Ensures that the AUTH= keyword is specified and that the value of the keyword is any of these: ALL, CONS, INFO, IO, and SYS. (These values, with the exception of ALL and INFO, may be specified in a parenthesized list.) Checks that the issuing console is the master console.		CNCHECK DELIMRTN
4 Initializes a bit mask corresponding to the AUTH= keyword values. This mask will be used by IEECB901 in updating unit control module entries (UCMEs).		AUTHRTN
5 Passes control to processor module IEECB901, via BALR, to process the consoles specified on the command.		

Extended Description	Module	Label
6 The appropriate message is issued, via IEE0503D, if a message code is specified in the MSGCODE field. The table summarizes the possible error conditions and their corresponding message codes and IDs.		MSGRTN

Error Condition	Message Code	Message ID
Not 'CN('	X'0A'	IEE310I
Unpaired parentheses	X'07'	IEE307I
Not master issuer	X'29'	IEE345I
Invalid operands	X'3E'	IEE535I
Invalid 'AUTH' values	X'3D'	IEE708I
Invalid 'CN' values	X'06'	IEE306I
Null/missing parameters	X'08'	IEE311I
Invalid unit	X'0D'	IEE313I
Command completion message	X'03'	IEE712I

Extended Description	Module	Label
7 Deletes the message module (IEE0503D) and frees the command scheduling control block (CSCB), via the MGCR macro.		

Diagram 2-42. VARY Console (V CN) Processing (IEECB901) (Part 1 of 2)

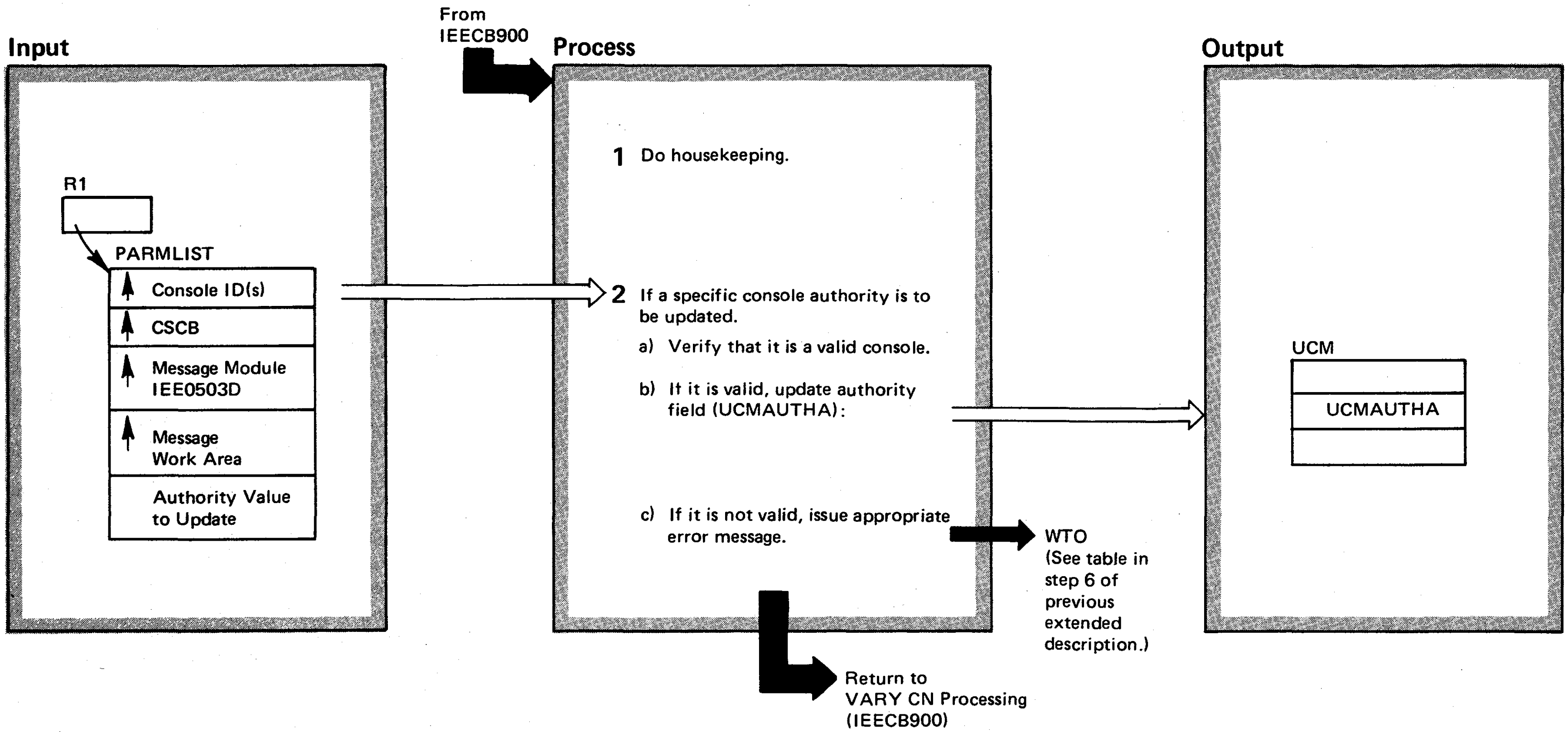


Diagram 2-42. VARY Console (V CN) Processing (IEECB901) (Part 2 of 2)

Extended Description	Module	Label
1 Sets loop control to process all console IDs which were passed from IEECB900.	IEECB901	
2 The address of the console ID list in the command buffer (CSCB) is passed from IEECB900. Processes all console IDs sequentially in the following manner:		DELIMSCN
a) Indexes to the unit control module entry (UCME) for a particular console ID, and verifies that the target console has a UCME, and that the target console is not the master console.		
b) Updates the UCMAUTHA field of the UCME with the authority mask which was passed from IEECB900.		CONSCAN
c) Issues error message via IEE0503D if failure occurs in any validity check in either module (IEECB900 or IEECB901). See message code table in step 6 of previous hipo.		MSGSETUP

Diagram 2-43. Varying Devices (Console or I/O Units) Online and Offline (IEE4203D) (Part 1 of 4)

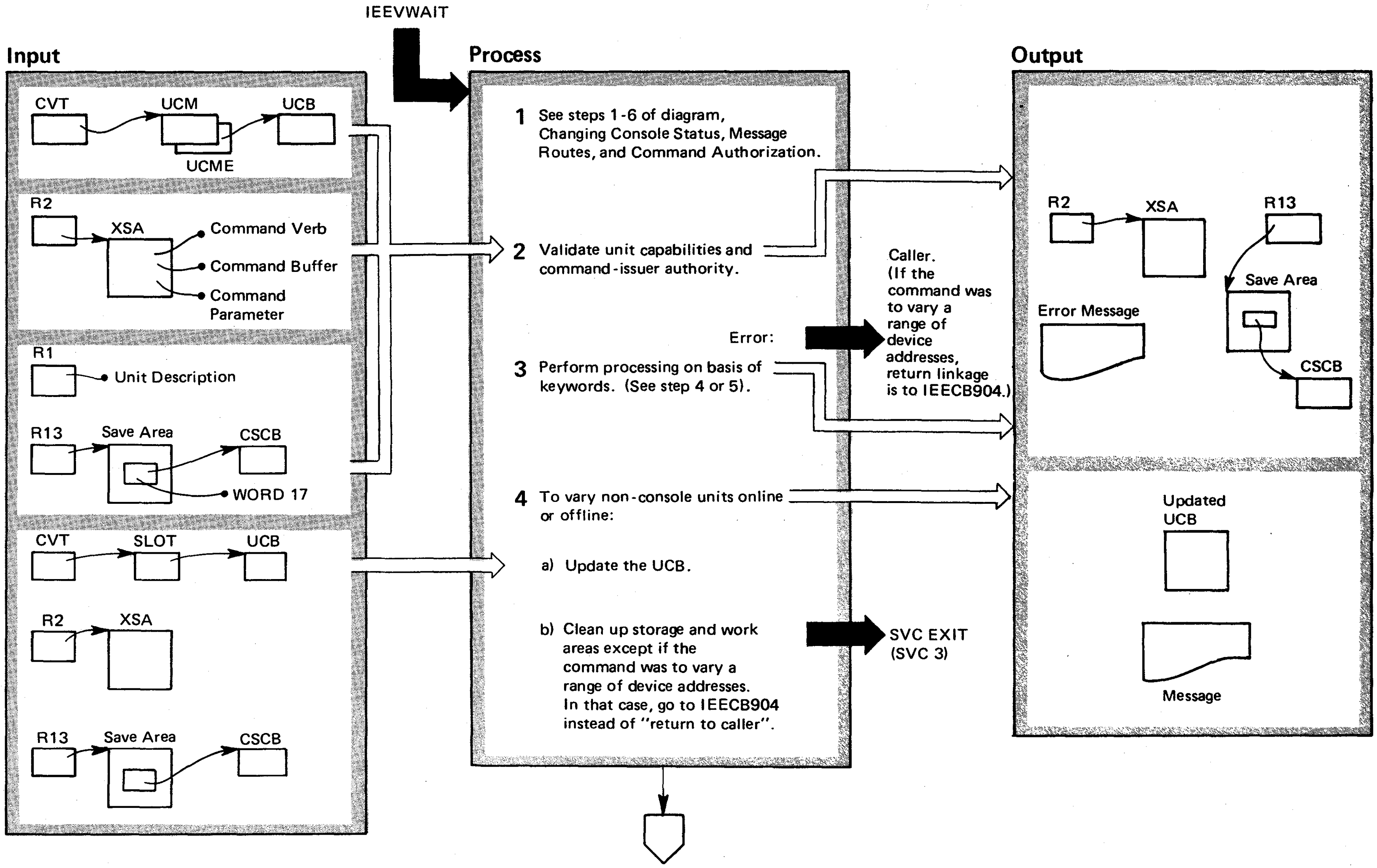


Diagram 2-43. Varying Devices (Console or I/O Units) Online and Offline (IEE4203D) (Part 2 of 4)

Extended Description	Module	Label
This processing changes the status of secondary consoles or I/O devices.		
1 These steps describe the processing that occurs after the VARY command pre-processor module has been attached by the master scheduler wait routine and before module IEE4203D gets control.		
2 General Considerations regarding unit validity, unit capability, and the authority of the command issuer.	IEE4203D	UCOMP2
• Units (including consoles) must have I/O capabilities to perform as requested.		
• Issuer's authority, if varying a console device, must be 3 or, if varying I/O units (units without a UCME) must be 2.		
• Devices specified for varying must be in a steady (unchanging) state.		
3 • If a command specifies multiple units that are all designated as "console" units at system generation time, complete the processing shown in step 5. (Note: These units will contain both a UCME and a UCB.)	IEE4603D	
• If a command specifies only multiple I/O units (those having only UCBs), complete the processing shown in step 4.	IEE3103D	VMLTUNT
• If a command specifies multiple units, some of which are designated as "console" units and some of which are I/O units, complete the processing by handling first the console units and then the I/O units.	IEE4603D IEE3103D	OSTART VMLTUNT

Extended Description	Module	Label
a) **To vary a device online:**	IEE3103D	VONLIN
• Leave on-line units as they are.		
• Change off-line units to the online state.		
• Issue appropriate message, except if the command was to vary a range of device addresses.		
To vary a device offline:	IEE3103D	VOFFLN
• Leave offline units as they are, and issue appropriate message.		
• Place online, unallocated units in the offline state, and issue appropriate message, except if the command was to vary a range of device addresses.		
• Designate online, allocated units as ready to be placed offline. (When these units become unallocated, termination routines complete the process of varying the units offline.)		
b) For both situations (4a and 4b), the final processing involves releasing the XSA and save area, and dequeuing the UCB chain. If a VARY command is being processed for a range of device addresses, return linkage is to module IEECB904 for further processing and/or releasing the XSA and dequeuing the UCB chain.	IEE3103D	NORMEXIT

Diagram 2-43. Varying Devices (Console or I/O Units) Online and Offline (IEE4203D) (Part 3 of 4)

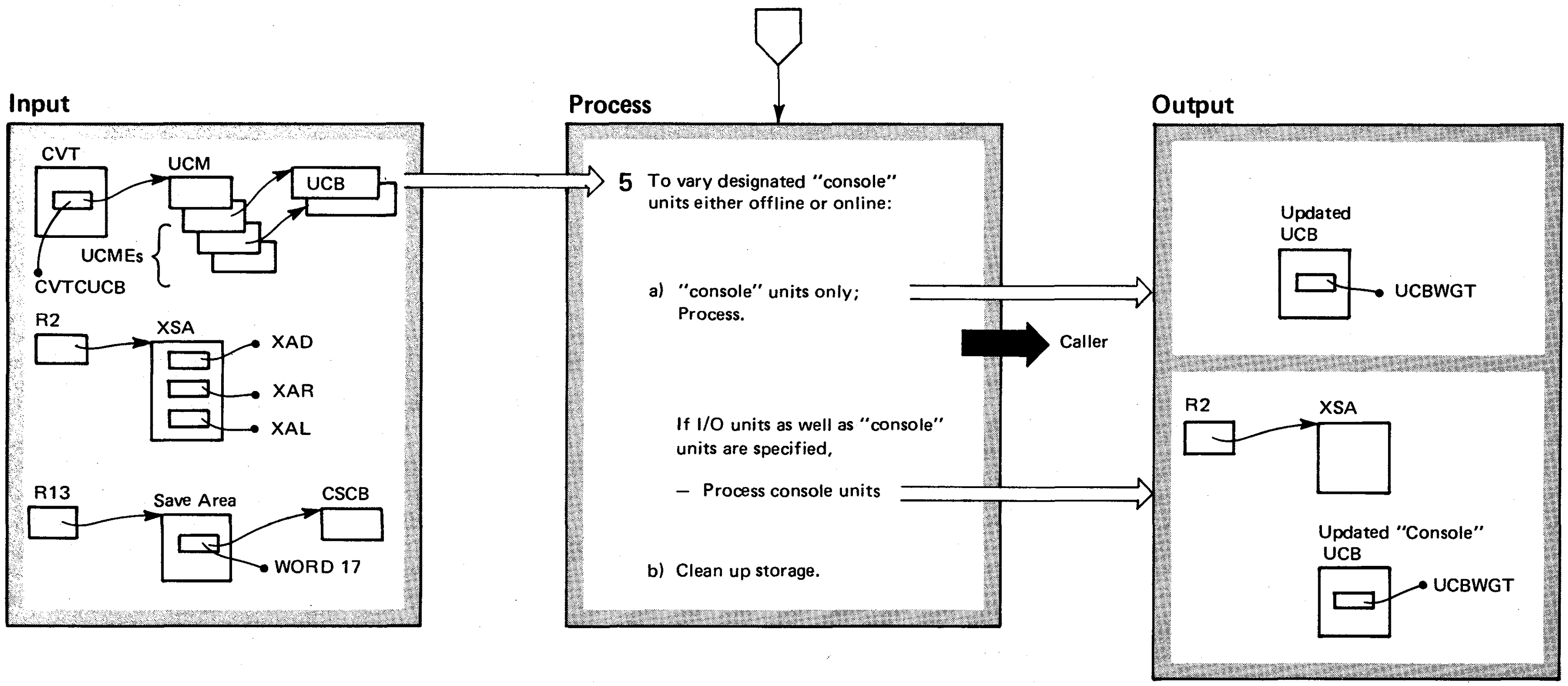
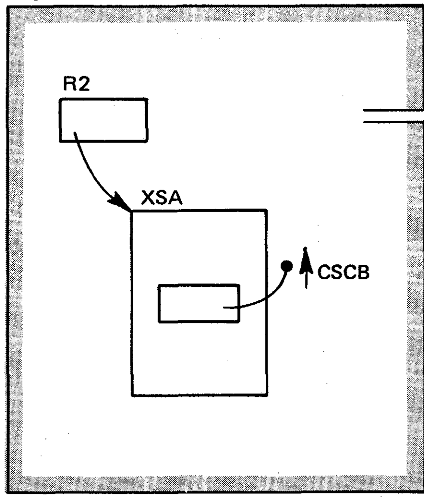


Diagram 2-43. Varying Devices (Console or I/O Units) Online and Offline (IEE4203D) (Part 4 of 4)

Extended Description	Module	Label
5a To vary units online	IEE4603D	OONLINE
• For units in an inactive console state, see function step 4a for online processing.		
• For units currently working as active consoles, the routine designates them as pending to be changed from console to online status. In this situation, the communications task will complete the processing.		
To vary units offline	IEE4603D	OFFLN
• For units in an inactive console state, see function step 4b for offline processing.		
• For units currently working as active consoles, the routine designates them as pending to be changed from console to offline status. In this situation, the communications task will complete the processing.		
5b The subroutine IEAVMNTR clears UCB bits representing the commands: MONITOR JOBNAMES, MONITOR STATUS, and MONITOR SESSION. This will prevent any monitoring messages from going to a device being varied from a console to a non-console status. The messages would be lost to the system in this case.		OCONT+

Diagram 2-44. Varying a Range of Device Addresses (IEECB904) (Part 1 of 2)

Input



VARY Processing Module (IEE3603D)

Process

1 Determine if this is the first entry to IEECB904.

If not, go to step 4.

2 Check command syntax.

If in error, issue message and go to step 5.

3 Expand address range as single unit addresses in ascending order in the buffer.

VARY Processing Module (IEE3603D)

4 Determine if more address expansion is needed in current range set.

If so, go back to step 3 to expand the addresses.

If not, issue DISPLAY UNITS command to indicate acceptance of the addresses in the specified range.

5 Determine if address expansion is needed in other address ranges.

If so, go back to step 2.

6 Release resources.

VARY Processing Module (IEE3603D)

Output

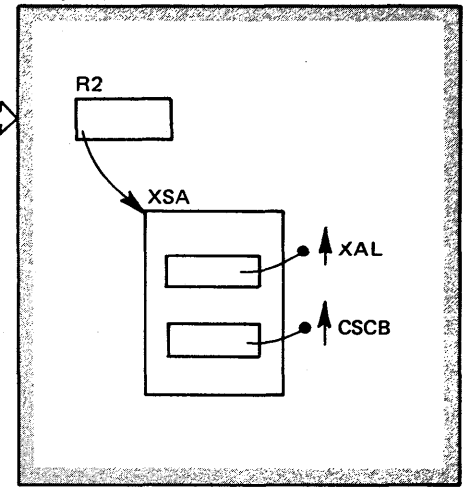


Diagram 2-44. Varying a Range of Device Addresses (IEECB904) (Part 2 of 2)

Extended Description	Module	Label
This processing is implemented through the use of IEECB904 as a driver to generate the VARY command. (e.g. V 100-102, ONLINE is changed to V (100, 101, 102), ONLINE.) If all specified units are valid, IEECB904 issues a unit status message (IEE450I).		
1 When a command to vary a range of device addresses is entered for the first time, IEECB904 will establish the necessary work areas by issuing a GETMAIN for subpool 230 and updating the XAL pointer. The XAL contains the address of the storage which mainline VARY uses as the command buffer.	IEECB904	GETSAVE
2 An invalid command results in an error message. If there are no more ranges to be processed, a message is issued and control is returned to the caller.	IEECB904	WRITEOUT
3 Control is passed to IEE3603D for VARY processing as single units.	IEE3603D	IEE36ENT
4 When the highest address of the current range set has been processed, control is returned to IEECB904, and the units whose addresses are in the specified range are displayed.	IEECB904	DISPLAY
5 The end of the command has been reached when there are no more ranges to be expanded into specific addresses.	IEECB904	
6 When exit is to be made, dequeue resources, free work areas, and issue the MGCR macro to free the CSCB.	IEECB904	TERMEXIT

Diagram 2-45. VARY HARDCOPY (Vx, HARDCOPY) Command Processing (IEE4703D) (Part 1 of 2)

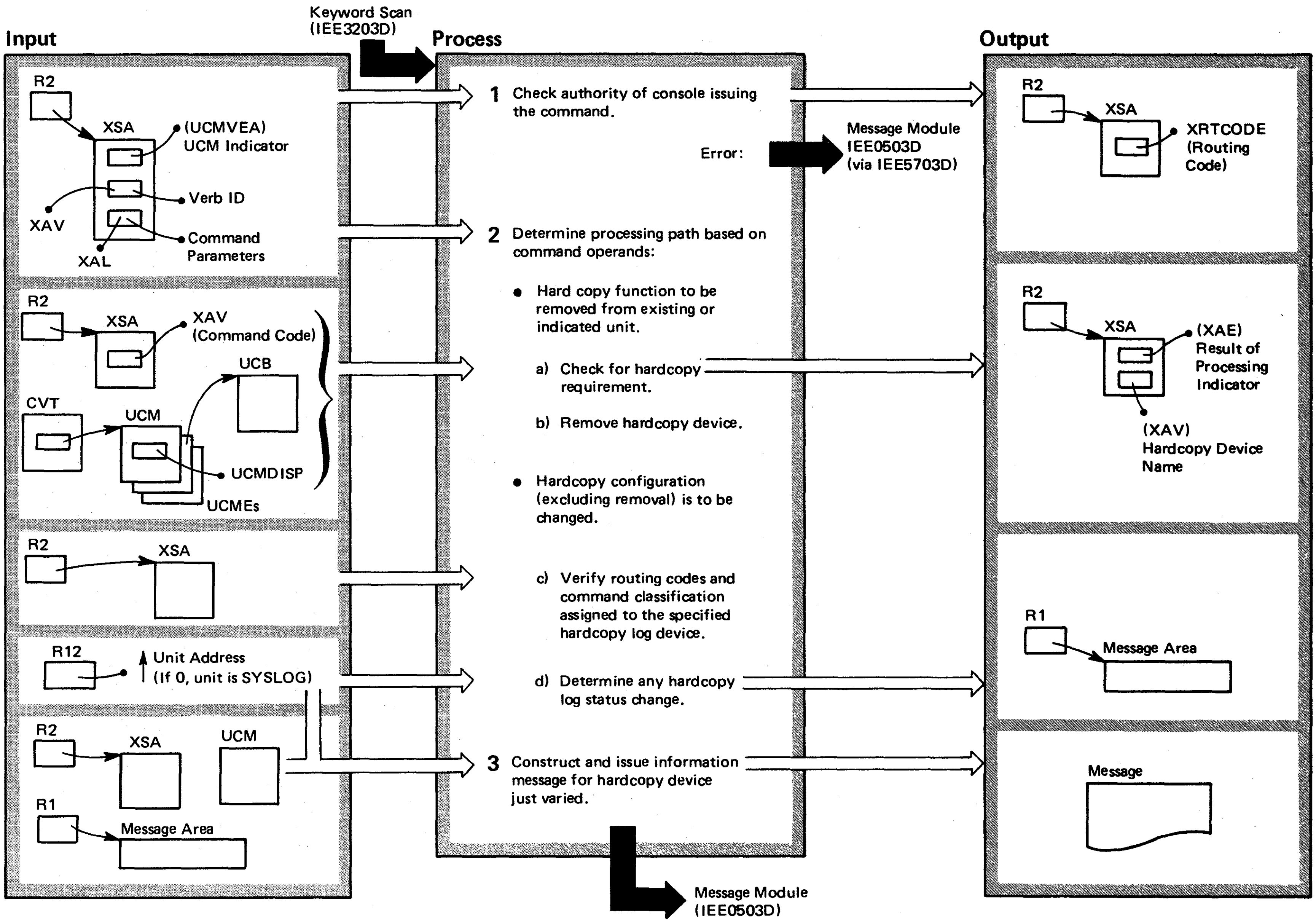


Diagram 2-45. VARY HARDCOPY (Vx, HARDCOPY) Command Processing (IEE4703D) (Part 2 of 2)

Extended Description	Module	Label
This processing either assigns a unit as a hardcopy log device or changes message routing to the hardcopy log. The process will also discontinue the hardcopy log if requested (that is, remove hardcopy log from the system).		
1 Only the system or the master console may issue this command.	IEE4703D	HERR1
2 Either the hardcopy function will be removed from the system or the hardcopy configuration will be changed. If SYSLOG is specified, it must be supported. If no unit is specified at SYSGEN time, a hardcopy unit must exist. A device specified for modification must not be in a state of change from or to console status. The system requires a hardcopy device if it has either a) more than one active console, or b) one or more active graphic consoles.	IEE4703D	HKEYFND

Extended Description	Module	Label
Remove Hardcopy		
2a The following considerations are examined: • Does the system require a hardcopy device? • Does the specified unit represent the current hardcopy device? • In the absence of a specified unit, does the system currently have a hardcopy device? Local and CMS locks are used to protect the UCM and UCB.	IEE5703D	BRANCH
2b Set an indicator in XSA to show result of the processing.	IEE5703D	INVNUM
Change Hardcopy		
2c Requested modifications apply to the current hardcopy log device in lieu of other specified devices.	IEE7203D	HSCAN
2d Consider for a new console or for the existing hardcopy log.	IEE7203D	HSCAN+
3 Local and CMS locks protect the UCM against another VARY command request. Use the WTO macro instruction to write the message.	IEE4103D	HSETLK CISSUE1

Diagram 2-46. Master Console (Vx, MSTCON) Switching (IEE4303D) (Part 1 of 2)

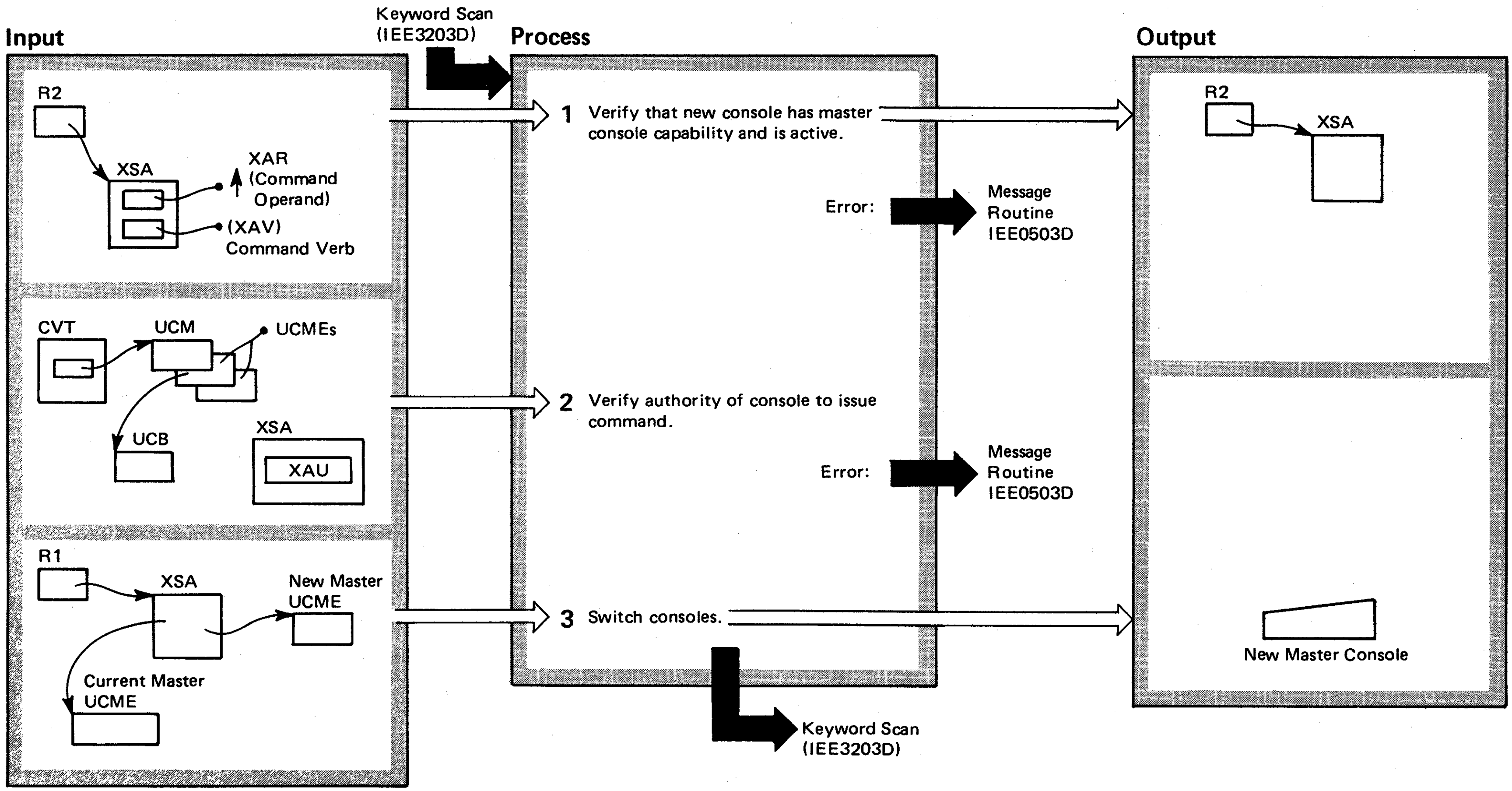


Diagram 2-46. Master Console (Vx, MSTCON) Switching (IEE4303D) (Part 2 of 2)

Extended Description	Module	Label
This process prepares another device to be the master console.		
1 The selected console must have I/O capability. Composite (console) units (established at SYSGEN time) must be active console devices. If console activity status is changing or pending a change, the command is rejected.	IEE4303D	MPROCESS
2 The current (existing) master console can issue this command. If the master console is inoperative because of hardware problems, its alternate or any console or the converter/interpreter can issue the command.	IEE4303D	
The routine reserves the UCM and UCB resources by means of the locking interface for local and CMS locks. This protects the fields being tested against changes by the communication task and/or another VARY command.		
3 Issue SVC 72.		MACTIVE

Diagram 2-47. Varying a CPU (V CPU) or Channel (V CH) Offline or Online (Overview) (IEEVCPU) (Part 1 of 2)

Master Scheduler Wait Routine (via ATTACH) (IEEVWAIT)

Input

R1

CSCB

CHBUF (Command Text)

CHINC (Command Authority)

Work Area

(Command Type)

Varying a CPU Online (IEEVCPU)
Varying a CPU Offline (IEEVCPU)
Varying a Channel Online (IEEVCPU)
Varying a Channel Offline (IEEVCPU)

Varying a CPU Online (IEEVCPU)
Varying a CPU Offline (IEEVCPU)
Varying a Channel Online (IEEVCPU)
Varying a Channel Offline (IEEVCPU)

Process

1 Check command authority.

Error → Step 6

2 Build work area.

3 Check command syntax.

Error → Step 6

4 Branch according to the following commands:

VARY CPU ONLINE'	See Diagram, "Varying a CPU Online" (IEEVCPU)
VARY CPU OFFLINE.	See Diagram, "Varying a CPU Offline" (IEEVCPU)
VARY CHANNEL ONLINE.	See Diagram, "Varying a CPU Online" (IEEVCPU)
VARY CHANNEL OFFLINE.	See Diagram, "Varying a CPU Offline" (IEEVCPU)

5 Write SMF records.

6 Issue appropriate messages to operator.

Output

Work Area

Storage Pointers

Error flags

SMF Record(s)

Types	Units
9, 11	Devices
22	CPU and Channels

Diagram 2-47. Varying a CPU (V CPU) or Channel (V CH) Offline or Online (Overview) (IEEVCPU) (Part 2 of 2)

Extended Description	Module	Label
0 This routine enables an operator to logically add or remove a CPU and/or a channel from the operating system. However, under certain conditions (for example, last path situations), an offline command may be rejected.		
1 Input for this command authority can be internally-issued commands, readers with the same authorization as the master console, or the master console itself.	IEEVCPU	IEEVCPU
2 This work area will contain error flags, storage area pointers, and other information that may be needed by cleanup functions.		IEEVCPU
3 The command text is scanned to determine the component (unit) to be varied and the operational state in which it is to be placed. In addition, the following checks are made: For V CPU, test for uniprocessor. For V CH, test if the specified CPU is currently online.		SYNTXCHK
4 Processing to be performed depends on the variation to be accomplished.		
5 SMF record type(s) 9, 11, and/or 22 are written to indicate the devices, CPU, and/or channel that has (have) been varied. The routine uses the recorder block information for this.		
6 The routine issues a WTO macro instruction to indicate the success (or failure) status of the processing.	IEECLEAN	WTORTN

Diagram 2-48. Varying a CPU (V CPU) Online (IEEVCPU) (Part 1 of 2)

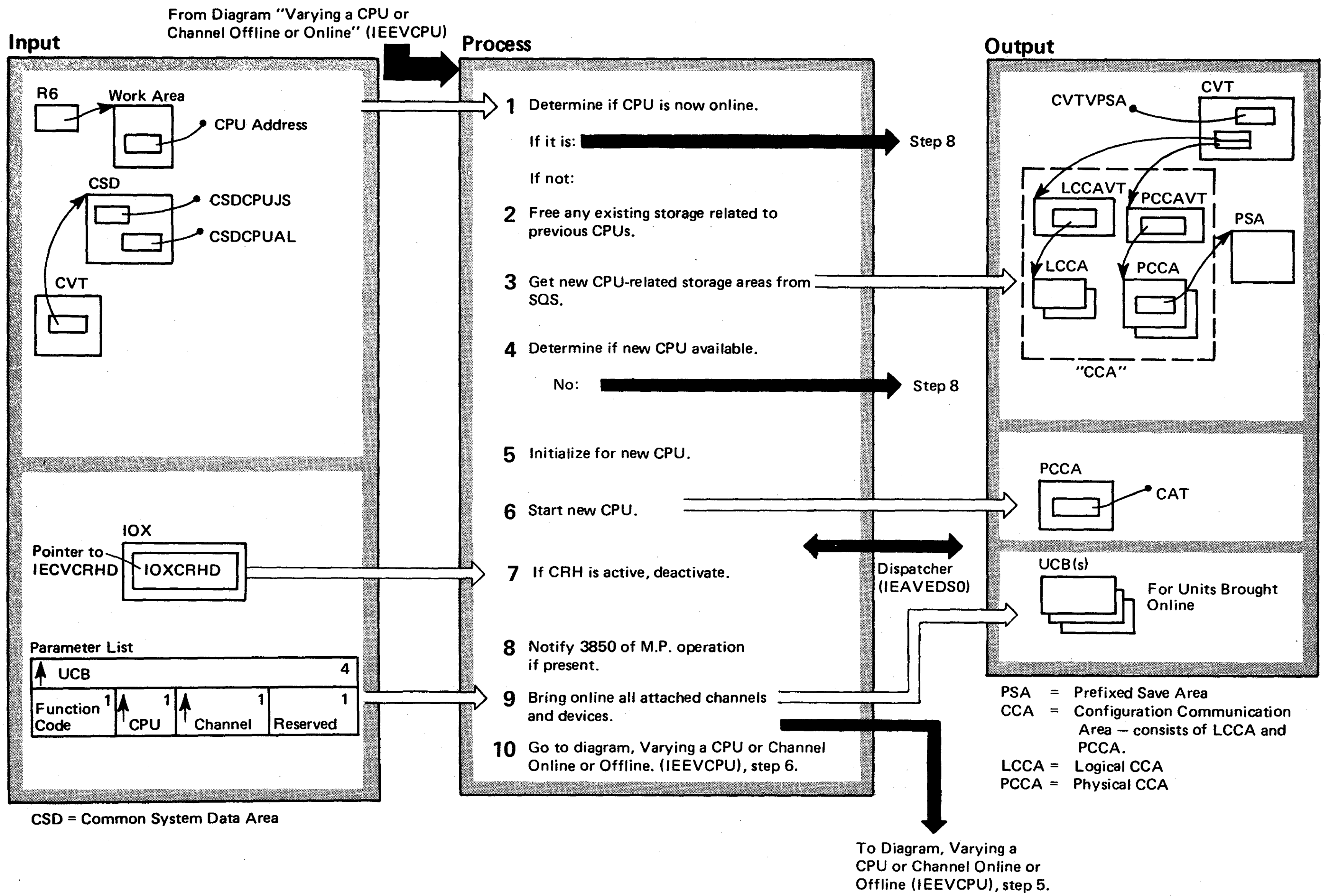


Diagram 2-48. Varying a CPU (V CPU) Online (IEEVCPU) (Part 2 of 2)

Extended Description	Module	Label
This routine makes a specified CPU available to the system. All operational channels attached to the CPU are also brought online (made available) with the CPU.		
1 If the CPU is already online, issue an appropriate message (via the 'cleanup' function). The two CSD masks are checked to determine if the CPU is already online for both the job scheduler and the supervisor. (Previous CPU affinity conditions may have left a CPU offline for only the job scheduler.)	IEEVCPU	IEEVCPU
2 The routine releases storage areas related to previously online CPUs.		
3 The routine gets storage areas from subpool 245 of the system queue area (SQA). Other system components (for example, GTF, RTM, and RMS) provide routines to get and initialize other CPU-related work areas. All storage requests are chained out of the work area.		PREPARE
4 The routine uses the signal processor reset (SIGP RESET) instruction to verify that the target CPU (the one scheduled to come online) exists, and if so, to set the CPU's prefix register. If there is no available CPU, the routine issues an appropriate message (via the 'cleanup' function).		
5 The routine sets the parameters for the target CPU in the 0-4K (absolute) area of storage. The new (or arriving) CPU uses the 0-4K area for its PSA. The routine then sets the restart new PSW so that it points to the ('wakeup') routine for initializing the CPU. The wakeup routine will give control to the dispatcher.	IEEVCPU	PREPARE
6 The processor routine uses a SIGP RESTART instruction to activate the wakeup routine for processing on the target CPU. The main processor routine (IEEVCPU) goes through a flag-checking loop waiting for the wakeup routine to indicate that the target CPU has come online. After the target CPU is online, the wakeup routine completes its own initialization in the following manner: It	IEEVWKUP	IEEVWKUP
• Puts the new PSA address in a prefix register.	IEEVWKUP	
• Loads control registers 0 and 1.		
• Turns on the dynamic address translation (DAT) function.		

Extended Description	Module	Label
• Calls the RMS component to initialize control registers 14 and 15.		
• Stores channel information in the CAT (channel availability table).		
• Calls timer and 'clock' routines to set the timer functions. (The TOD clock is synchronized with the controlling CPU.)	IEAVRTOD	IEAVRSSC
7 Call IOS (at IECVCRHD) to deactivate the Channel Reconfiguration Hardware if it is active.	IEEVCPU	CKCOMPLT
8 Issue Initialize, and Associate orders to 3850, if present, to prepare for M.P. operation.		
9 The routine opens new paths to the devices associated with this CPU.		DEVPREP
IOS checks for operational paths to a device before a device is made online with a CPU.		DEVCHK
The routine checks all 'hierarchical-offline' devices (devices forced offline by virtue of being attached to an offline channel) for presence of a currrent operational path.	IEEVDEV	
The routine also builds recorder (control) blocks and marks these blocks "online" for the devices that are brought online. The 8-byte internal recorder blocks contain information used when writing the SMF records.		
If a recorder block already exists for a UCB, an additional recorder block is not built. All recorder blocks are chained together.		
10 Routine issues appropriate message.		

Diagram 2-49. Varying a CPU (V CPU) Offline (IEEVCPU) (Part 1 of 2)

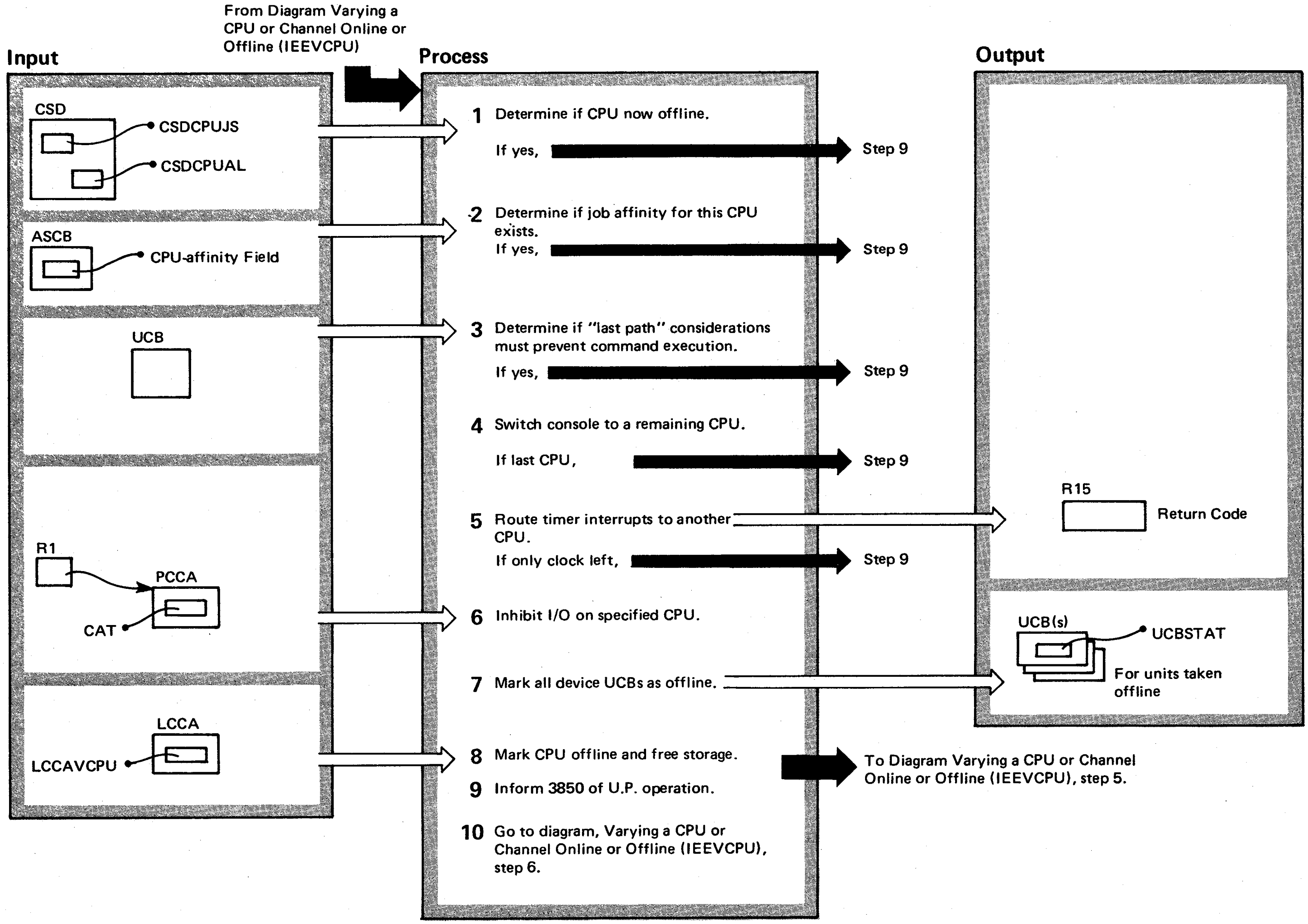


Diagram 2-49. Varying a CPU (V CPU) Offline (IEEVCPU) (Part 2 of 2)

Extended Description	Module	Label
This routine makes a specified CPU unavailable to the system. Before taking a CPU offline, the routine first checks reserved and last path (to a device) considerations and CPU-affinity considerations. The routine also takes offline all devices and channels associated with the specified CPU.		
1 The flags CSDCPUJS (for job scheduler) and CSDCPUAL (for supervisor) indicate degrees of offline status for a CPU. A CPU may be offline to the scheduler (no further affinity scheduling allowed) but online to the supervisor (actively running a program). The supervisor flag is tested here.	IEEVCPU	CPUSRCH
In addition, this routine also tests for other online CPUs in the system.		
2 An active job requiring the indicated CPU will inhibit the execution of this request. A WTOR macro instruction notifies the operator of this situation, and waits for a response. The routine tests a 'CPU-available' flag against a 'CPU-affinity' field in each active ASCB.		AFFINSRC
3 The routine checks for any online or allocated device that may be dependent on the specified CPU for an access path. A device marked as online and unallocated, coupled with the UNCOND operand on the command will override an attempt to reject this command because of a lack of available paths, through other CPUs, to the device.		DEVCHECK
All devices are checked for last path considerations. The command is rejected if either • a path is the last path to an allocated device or • a path is the last path to an online unallocated device without the UNCOND parameter specified.	IEEVDEV	
Recorder (control) blocks are built for all devices going offline with the CPU. (See the diagram Varying a CPU Online, step 7.)		
4 At least one console with input and output capability must remain or the command will be rejected.		
5 If the departing CPU has the only operative clock in the system, the command is rejected by the system. To make this determination, IEEVCPU links to the module IEAVRTOD. The latter module contains, at entry point IEAVRNOT, a subroutine that determines if varying the CPU offline will remove the last usable TOD clock, clock comparator, or CPU timer from the system. If so, the VARY command is cancelled. At entry point IEAVRCAN in the same module, there is a subroutine that restores the clocks to the system if the command is subsequently cancelled for other reasons.	IEEVCPU	
6 Flags in the CAT of the PCCA are set to indicate to IOS that associated channels are offline, and that I/O is not to be started on the channel.		IOQUIT
If I/O activity is present on the departing CPU, the operator gets a chance to cancel the command after 3 minutes have elapsed. The routine waits until all I/O activity ceases.		
7 The routine uses the recorder blocks to supply the information. The actual offline indicator in the UCBSTAT field is the bit UCBONLI.		UCBMARK
8 This action occurs only when there is no activity dependent on the specified CPU. The main line of the processor routine sets a flag in the LCCA. It then switches tasks and the dispatcher gets control on the target CPU (the one scheduled to go offline). The dispatcher checks the flag and when the flag is set, the dispatcher gives control to the 'quiet' code entry point in the module IEEVWKUP. This code then marks the CPU as offline. The following actions occur:		
• Call an RMS routine to reset control registers 14 and 15.	IEEVWKUP	IEEVQUIT
• Clear the prefix register.		
• Set the indicator CSDCPUAL to indicate the offline CPU.	IEEVWKUP	
• Issue a SIGP STOP instruction to place the CPU in a manual (stopped) state.	IEEVSTOP*	IEEVSTOP
The mainline routine then frees the prefixed storage area (PSA) and other related areas for the CPU taken offline. (The PSA is released only when it is not being referenced by another CPU.) Recovery termination management (RTM) and recovery management support (RMS) components (and any other components) will free any of their areas related to the now-offline CPU.		
9 Issue Disassociate and Purge orders to 3850 to allow U.P. operation.		
10 Issue message to operator.		

*This module resides in the nucleus.

Diagram 2-50. Varying a Channel (V CH) Online (IEEVCPU) (Part 1 of 2)

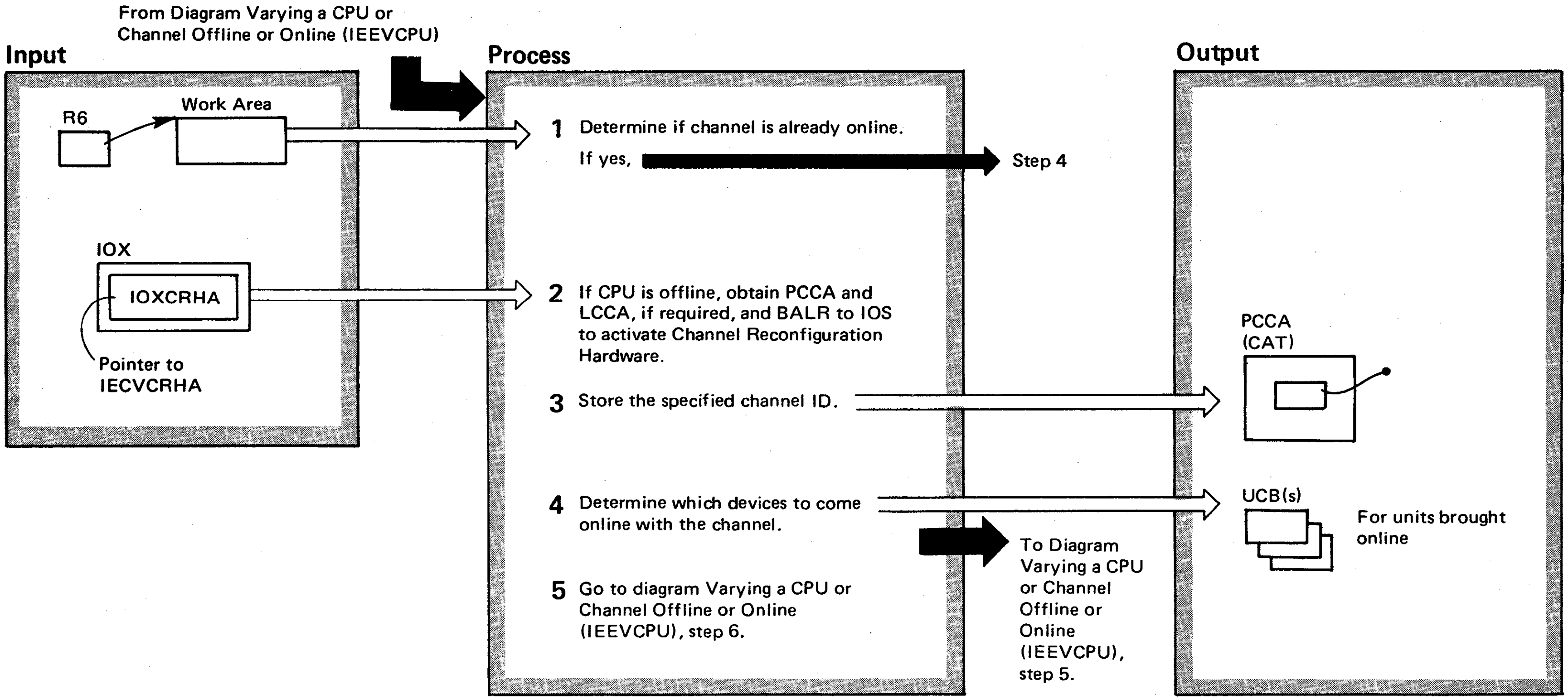


Diagram 2-50. Varying a Channel (V CH) Online (IEEVCPU) (Part 2 of 2)

Extended Description	Module	Label
This routine makes a specified, operational channel available to the specified CPU.		
1 This routine tests the operational status of the incoming channel. The CPU connected to the arriving channel must make the test.	IEEVCPU	IEEVCH
2 If no LCCAs and PCCAs exists for specified CPU, obtain new LCCAs and PCCAs and activate Channel Reconfiguration Hardware (CRH). (Branch to IOS at entry point IECVCRHA to activate CRH.)		
3 The routine stores this ID in the PCCA (CAT).		
4 The routine will test devices attached to the arriving channel to see if they have operational paths. (See the diagram Varying a CPU Online, step 7.)		DEVPREP
All devices that have operational paths and that are offline because of hierarchy considerations will be marked online. The routine then creates recorder (control) blocks for any devices to be brought online with the channel.		
All UCBs represented by the recorder blocks are then marked as having the associated devices online.		UCBMARK

Diagram 2-51. Varying a Channel (V CH) Offline (IEEVCPU) (Part 1 of 2)

From Diagram Varying a CPU or Channel Online or Offline (IEEVCPU)

Input

R6

Work Area

IOX

IOXCRHD

Pointer to IECVCRHD

Process

1 Determine if channel is offline now.

If yes, Step 6

2 Determine if there exist any devices currently dependent on this channel for access.

3 Determine which devices are to go offline with the channel.

4 Switch consoles from the departing channel.

5 Indicate devices are offline.

6 Deactivate CRH (Channel Reconfiguration Hardware) if required.

7 Go to diagram "Varying a CPU or Channel Online or Offline", step 6.

To Diagram Varying a CPU or Channel Online or Offline (IEEVCPU), Step 5

Output

UCB(s)

For units taken offline

SMF Records

11 Devices

22 Channels

Diagram 2-51. Varying a Channel (V CH) Offline (IEEVCPU) (Part 2 of 2)

Extended Description	Module	Label
Dependent on last path conditions and on master console considerations, this routine makes a specified channel unavailable to a specified CPU.		
1 A message will be issued, if this channel is already offline.	IEEVCPU	IEEVCH
2 A 'last path' device can be placed offline only if it is unallocated and if UNCOND was specified on the command. That is, unless specifically indicated to be made offline, a device must have an alternate path other than the one through the channel being placed offline. Searching for last paths involves the IOS macro instruction, IOSGEN, which indicates all online paths to a particular device.		DEVPREP
3 The command is rejected if the routine is unable to switch the active consoles to another channel.		IGC0407B
4 The routine sets indicators in the PCCA (CAT) to inhibit new I/O to the offline channel. If I/O activity on the channel is still going on, the operator has a chance to ask for more time for the I/O to cease before the command is rejected due to the I/O activity.		UCBMARK CATMARK IOQUIT
5 SMF record types 11 and 22 are issued for the devices and channel made offline.		
6 If VARY was for an offline CPU, determines if any channel that belongs to the offline CPU is still online. If no such channel remains online, branches to IOS (at IECVCRHD) to deactivate Channel Reconfiguration Hardware (CRH).		CKCOMPLT
7 The routine issues an appropriate message to the operator.		

Diagram 2-52. Varying the Path (V PATH) to a Device (IEEVPTH) (Part 1 of 4)

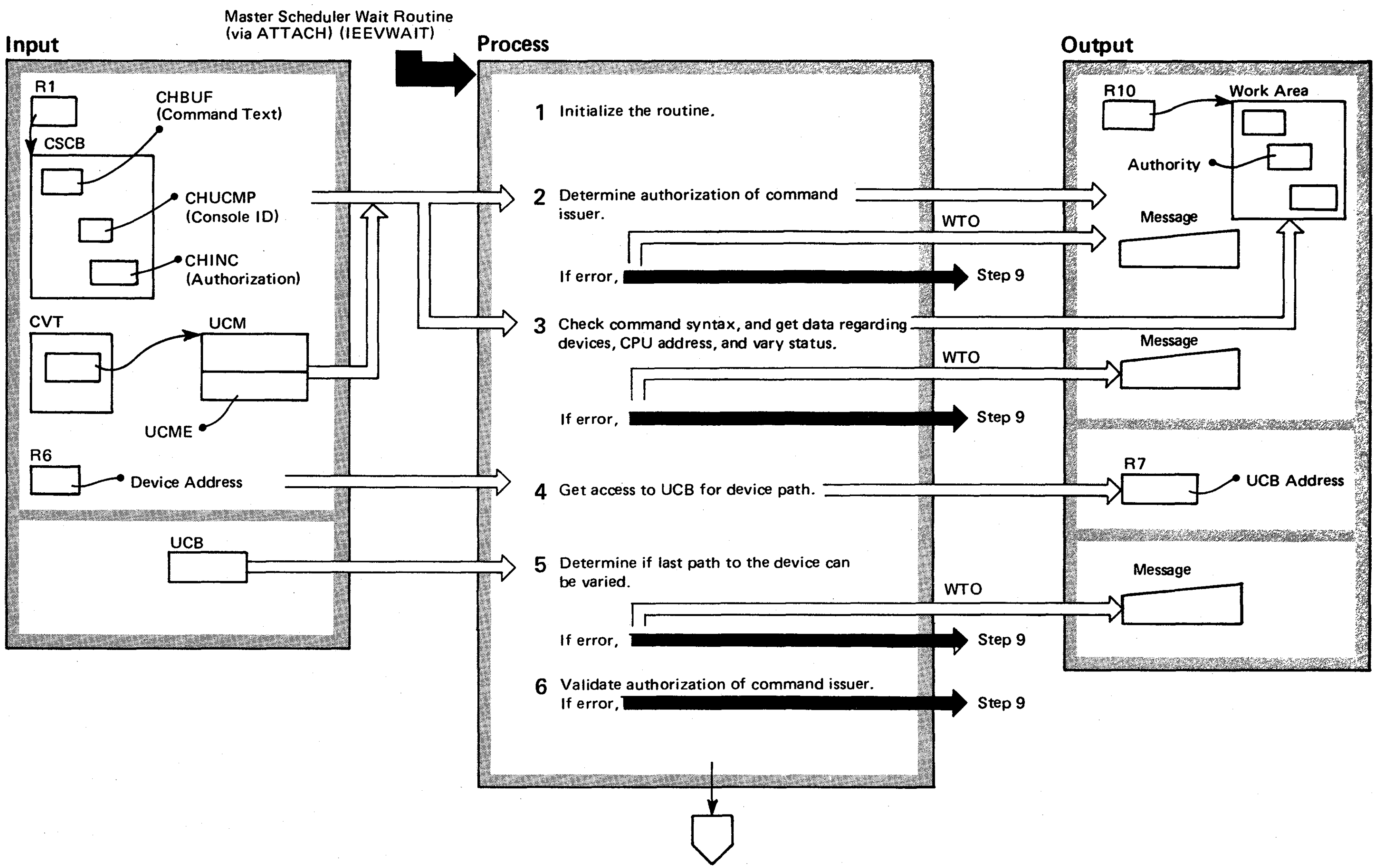


Diagram 2-52. Varying the Path (V PATH) to a Device (IEEVPTH) (Part 2 of 4)

Extended Description	Module	Label
This routine causes a path (a logical connection to a device) to be either brought online for use by the system, or removed from the system.		
1 The routine issues the ESTAE macro instruction to set up the recovery (or ESTAE) exit. In addition, the routine establishes a work area.	IEEVPTH	IEEVPTH
2 Determine the authority of the issuing console. The issuer may be restricted to only varying the path to I/O devices and not to consoles or he may have global authority to vary any path.		CMDAUTH
3 The command text contains the information needed here.		SYNTXCHK
4 The routine issues an ENQ macro instruction on the system resource SYSIEFSD,Q4. This serializes the UCBs for use by this routine.		
For the path to be varied, the address of the UCB that contains path status information comes from the use of the UCBLOOK function (operand) of the IOSGEN macro instruction.		
5 The routine examines the UCB. The last path to an allocated device may not be placed offline.		UCBCKOUT
6 The routine compares the device type (console or I/O device) to the issuer's authorization.		

Diagram 2-52. Varying the Path (V PATH) to a Device (IEEVPTH) (Part 3 of 4)

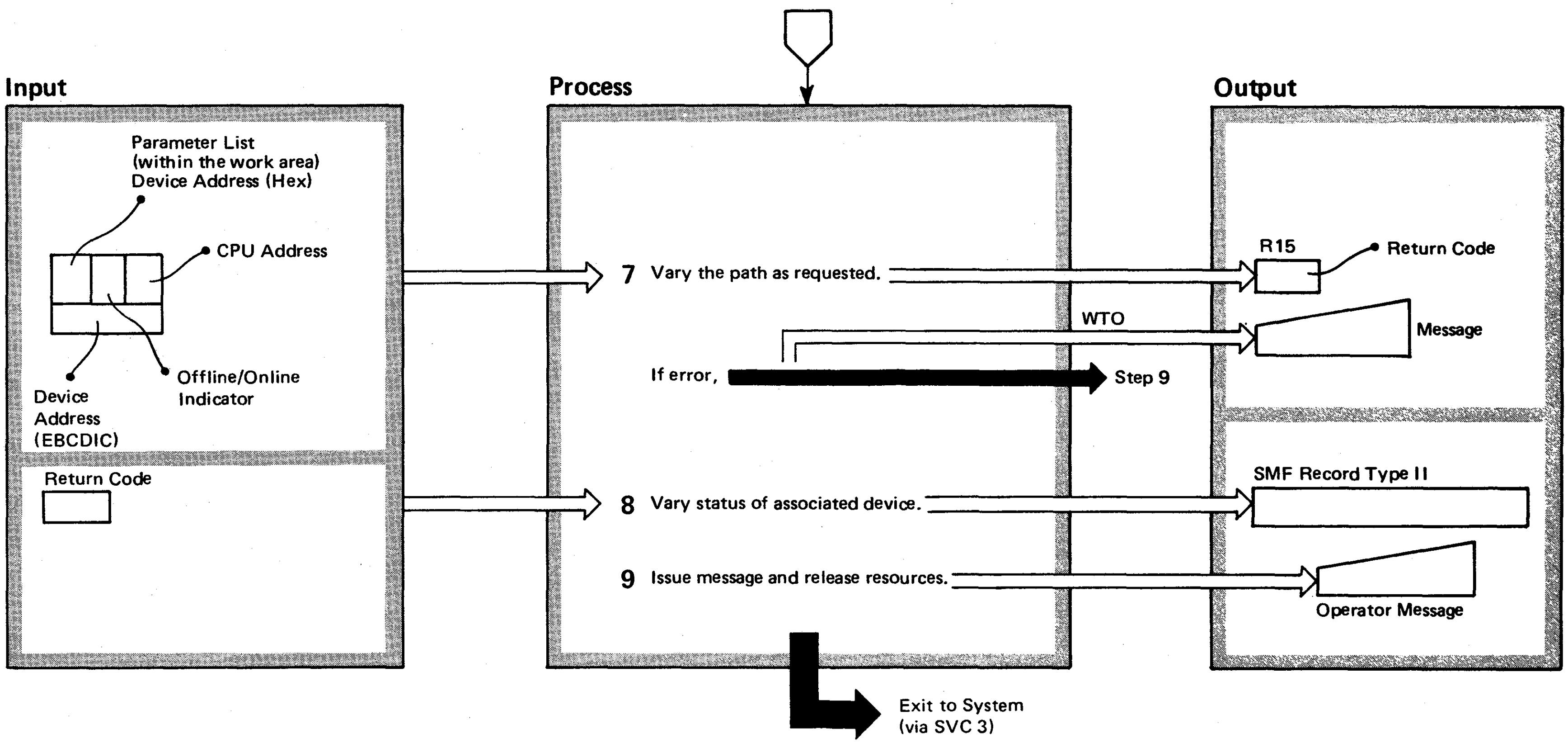


Diagram 2-52. Varying the Path (V PATH) to a Device (IEEVPTH) (Part 4 of 4)

Extended Description	Module	Label
7 The VARY function (operand) of the IOSGEN macro instruction is used to alter the path status bits (in the UCB) thus changing the path. A return code of 0 indicates success.	IEEVPTH	VARYOFF
8 The processing at this point depends on the type (offline or online) of request and the return code from the VARY function. For the request indicated below, the following conditions and responses are as shown: Vary online: • If the associated device is offline for hierarchy reasons, and • If an operational path to the device exists, then the device is varied online. VARY offline: • If the associated device is online, and • If the last path is to be varied offline, then the device is varied offline.	IEEVPTH	
If the device (above) is varied as indicated, the routine writes an SMF record.		SMF
9 The routine uses a WTO macro instruction to inform the operator of the results of the processing.		CLEANUP

Diagram 2-53. Varying the Status of Real Storage (V STOR) (IEEMPVST) (Part 1 of 2)

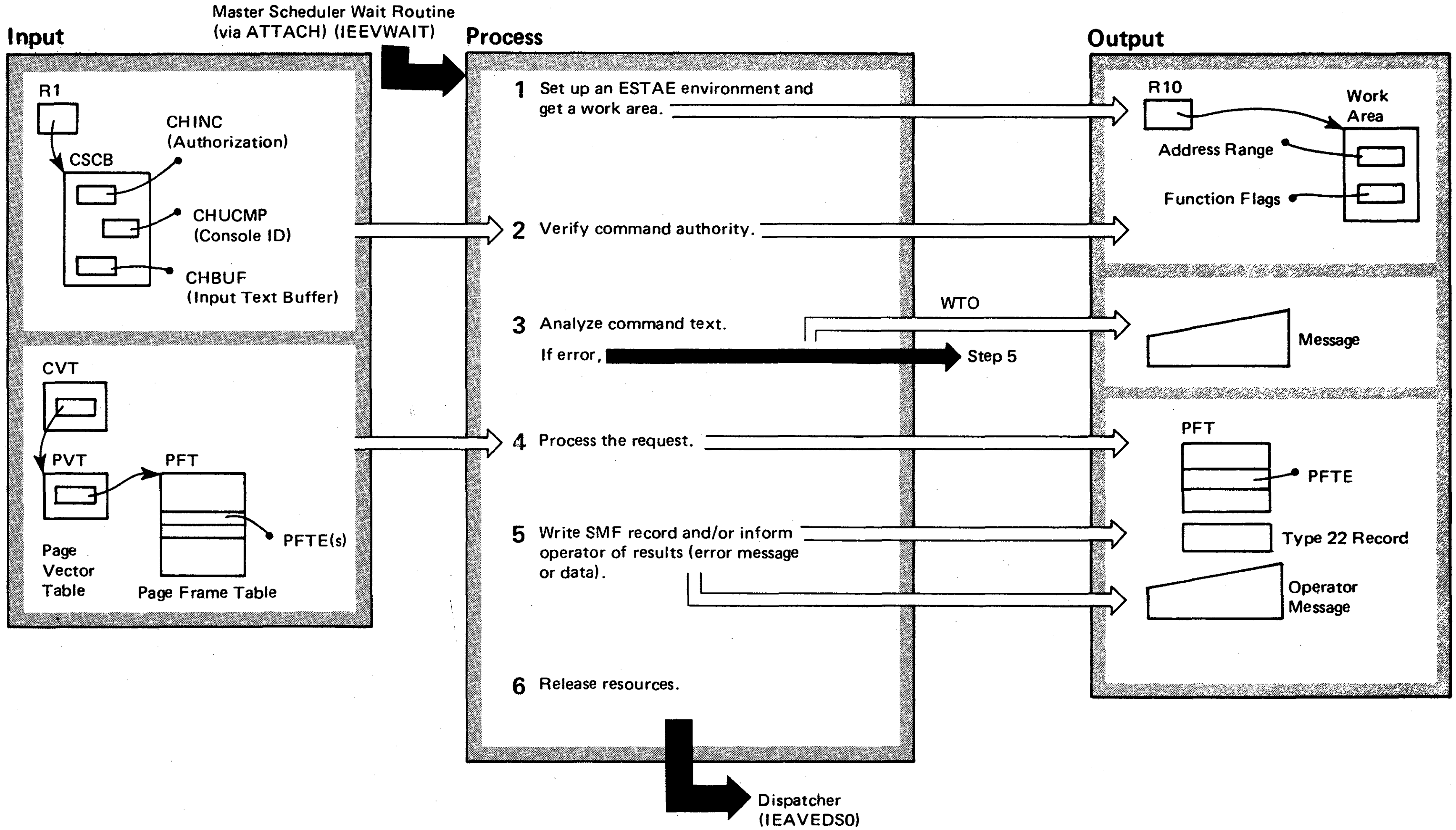


Diagram 2-53. Varying the Status of Real Storage (V STOR) (IEEMPVST) (Part 2 of 2)

Extended Description	Module	Label
This routine changes the status of a specified area of real storage to either an online or an offline condition.		
1 This environment traps any ABEND situations that the routine encounters. For a storage error, the ESTAE routine records the status and schedules a retry for the next storage page. The ESTAE routine frees all resources when an unrecoverable internal error occurs.	IEEMPVST	
2 The command issuer must be either the master console, an internal (system) issuer, or a reader with the same authority as the master console.		SYNCHK
3 The routine scans the command text to validate the text, and it stores the specified address range and function in the work area.		SYNCHK
4 If the storage is being varied online, the routine verifies the current status of the storage areas by checking the page status bytes that are returned by a real storage reconfiguration (RSR) routine. This procedure locates storage that is already online or that has associated storage errors.		VSTORON
If any of the requested storage has errors, the routine prompts the operator (via a WTOR macro instruction) to decide which error-free storage area(s), or if no storage, is to come online. A real storage reconfiguration routine is given control to change the storage status.		VBLCKON
If the storage is being varied offline, the routine determines if the storage is already offline. The routine places offline any storage not already offline.		VSTOROFF
RSR routines are used to accomplish the change in storage status.	IEEMPVST	VSTOROFF
If there is activity within the storage range, the routine waits for the storage to become inactive before placing it offline. A force-page-offline subroutine attempts to expedite this process. Each 4K page of storage placed offline is set to zero.	IEEVALST	FORCEOFF IEEVALST
While zeroing the storage area, the reconfiguration commands are serialized through the use of the ENQ macro instruction for the SYSZVARY,VALIDATE resource.		PROCOFF
5 The routine writes one or more type 22 SMF records to indicate the areas of storage that are placed offline or online.	IEEMPVST	SMFRCRD
6 The routine dequeues off (removes its hold on) the command resource, frees the work area, and releases the CSCB.		

Diagram 2-54. Teleprocessing (TP) Commands

Command
Translator
(IEE5403D)

Process

1 Command Verb *

If Verb is:	Give Control to:
DISPLAY△	IEE3503D IED1303D
HALT△	IEE1403D IED1303D
HOLD△	IEE0803D IED1303D
RELEASE△	IEE0803D IED1303D

△Command Operand is "TP".

*For further details, see the diagrams HALT (Z), SWITCH (I), and TRACE (TRACE) Command Initialization; Holding and Releasing Teleprocessing; and see the publication, *OS/VS2 TCAM Program Logic Manual,* SY30-2040.

Diagram 2-55. Holding (H) and Releasing (A) Teleprocessing Messages (IEE0803D) (Part 1 of 2)

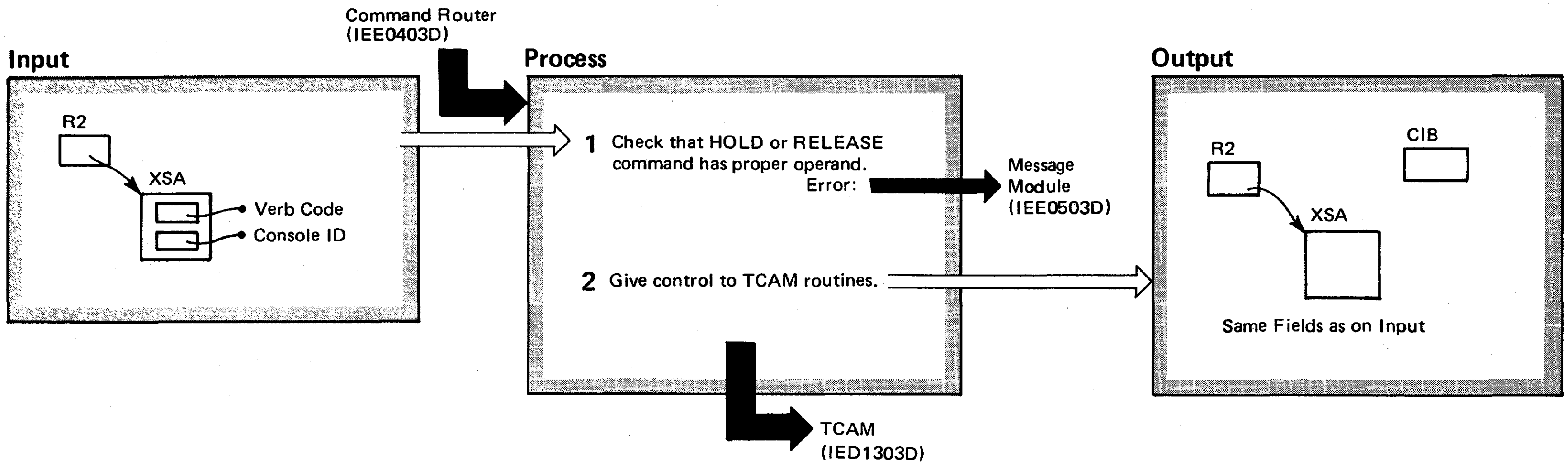


Diagram 2-55. Holding (H) and Releasing (A) Teleprocessing Messages (IEE0803D) (Part 2 of 2)

Extended Description	Module	Label
This routine determines if a valid TP command exists. If so, the master scheduler gives control to a TCAM routine to continue processing.		
1 The only valid operand for a HOLD or RELEASE command is "TP=."	IEE0803D	
2 Control passes to a TCAM routine to continue processing the command.	IED1303D	

Diagram 2-56. Processing Commands With the "NET" Operand

Command Translator (IEE5403D)

Process

1 Command Verb*.

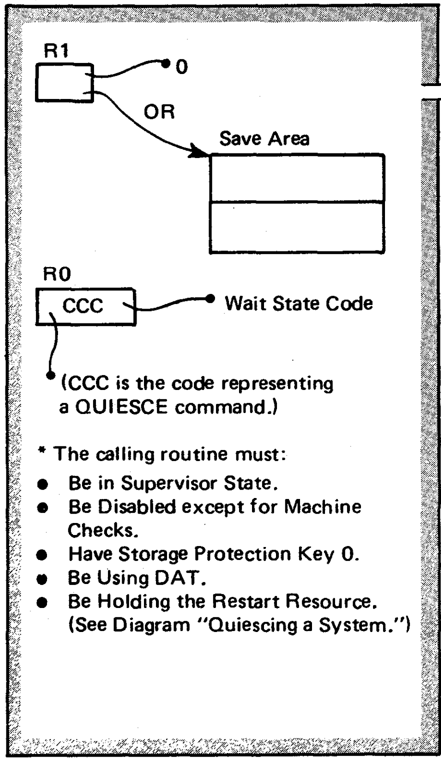
If Verb is:	**Control Goes to:**
HALT$^{\triangle}$	IEE1403D
	ISTCFF3D
DISPLAY$^{\triangle}$	IEE3503D
	ISTCFF3D

Command Operand is "NET."

For further details, see *OS/VS2 VTAM Logic,* SY28-0621.

Diagram 2-57. Stopping and Restarting (QUIESCE) (via an Interrupt) the System (IEESTPRS) (Part 1 of 4)

Input



Caller of IEESTPRS

Process

1 Determine caller of the routine.

2 Setup recovery routine.

3 Inform system resources manager of the intent to quiesce the system.

4 Set up a work area.

5 Store the TOD clock.

Output

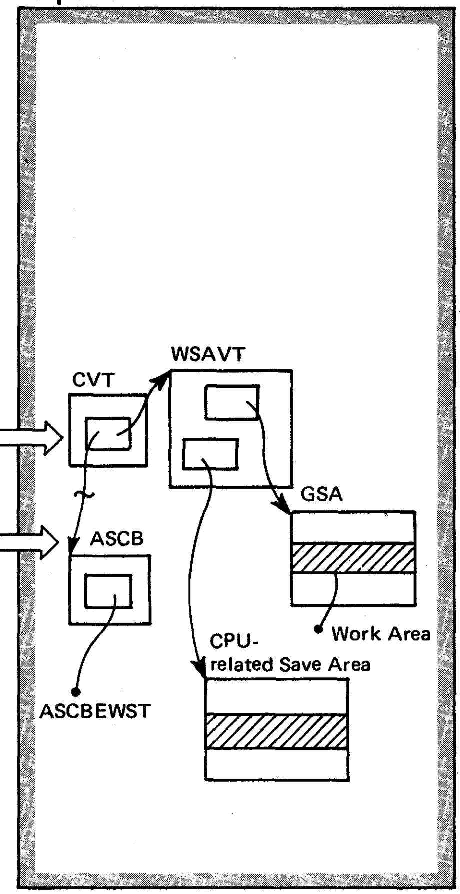


Diagram 2-57. Stopping and Restarting (QUIESCE) (via an Interrupt) the System (IEESTPRS) (Part 2 of 4)

Extended Description	Module	Label
This routine places CPUs in a stopped state and, on a restart signal from an operator, causes them to resume operation at the point of program interruption.		
1 When this routine is entered from the quiesce routine (IEEMPS03), register one contains the address of a save area that contains 208 bytes for each active CPU. (For any other entry, register one has a zero value, and the processing is different in some respects from that described for this figure.)	IEESTPRS	IEESTPRS
This routine uses the save area to store the status of the CPUs before they are placed in a stopped mode.		
2 An FRR is established to handle unexpected errors. In case of an error, the FRR will attempt to restore the system to the state in which it was operating prior to the call to the stop/restart routine. This restoration includes an attempt to restart all CPUs that may have been stopped by this routine prior to the occurrence of the error. The recovery routine returns control to the caller.		

Extended Description	Module	Label
3 The routine uses the SYSEVENT, SYSQSCST macro instruction for this purpose.		
4 The work save area vector table (WSAVT) points to the global save area (GSA) and to the CPU-related save areas that contain the work areas for this routine.		
Note: The GSA is a set of work areas for resident routines. It resides in the nucleus and is mapped by the IHAWSAVT macro instruction.		
5 For each active address space, the routine stores the time at which the CPU(s) entered the stopped state. When a CPU is eventually restarted, the system 'down' time is subtracted from the job's execution time. Thus, job step timing (JST) is preserved across the stopped state. (See also, step 8.)	IEESTPRS	

Diagram 2-57. Stopping and Restarting (QUIESCE) (via an Interrupt) the System (IEESTPRS) (Part 3 of 4)

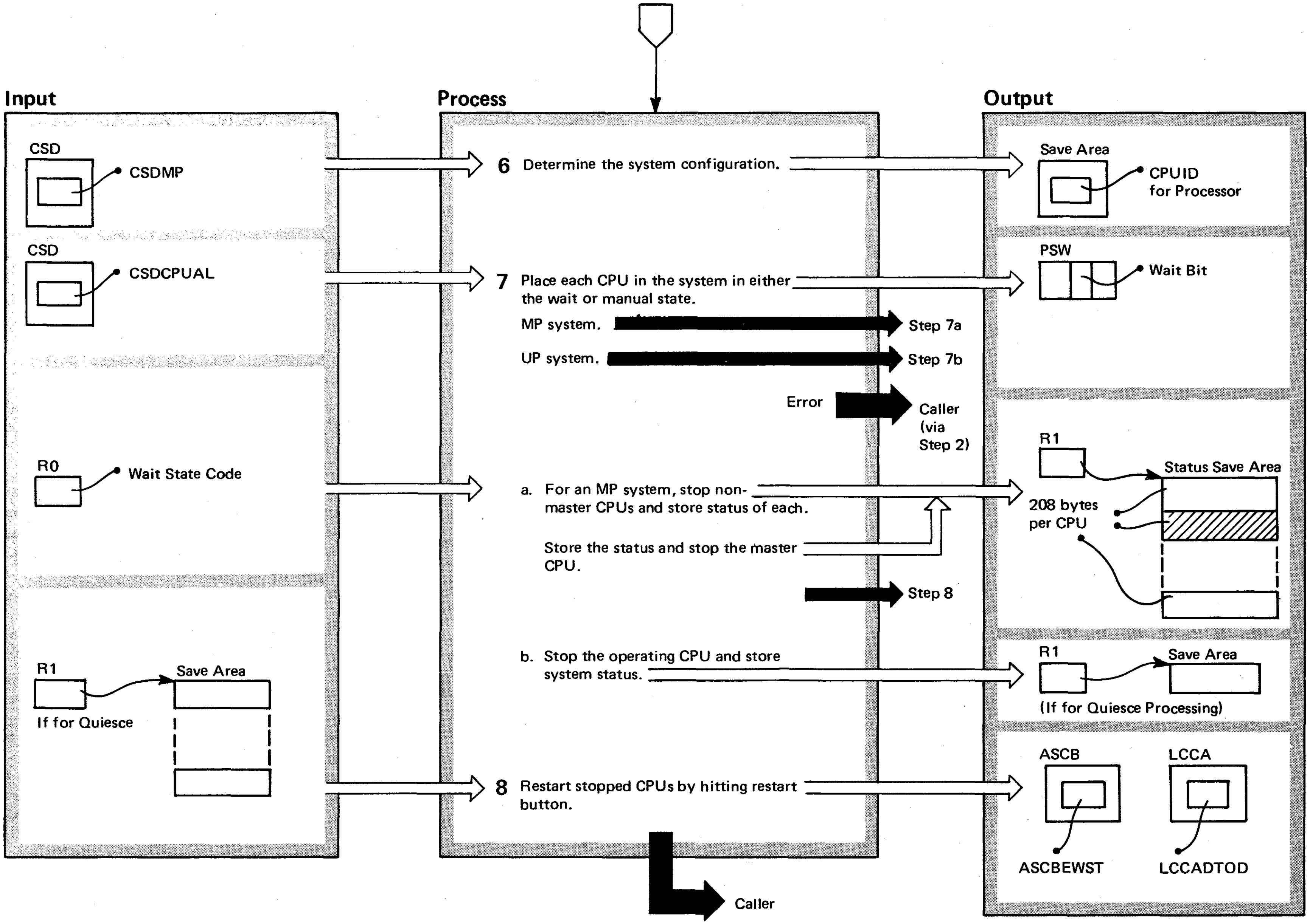
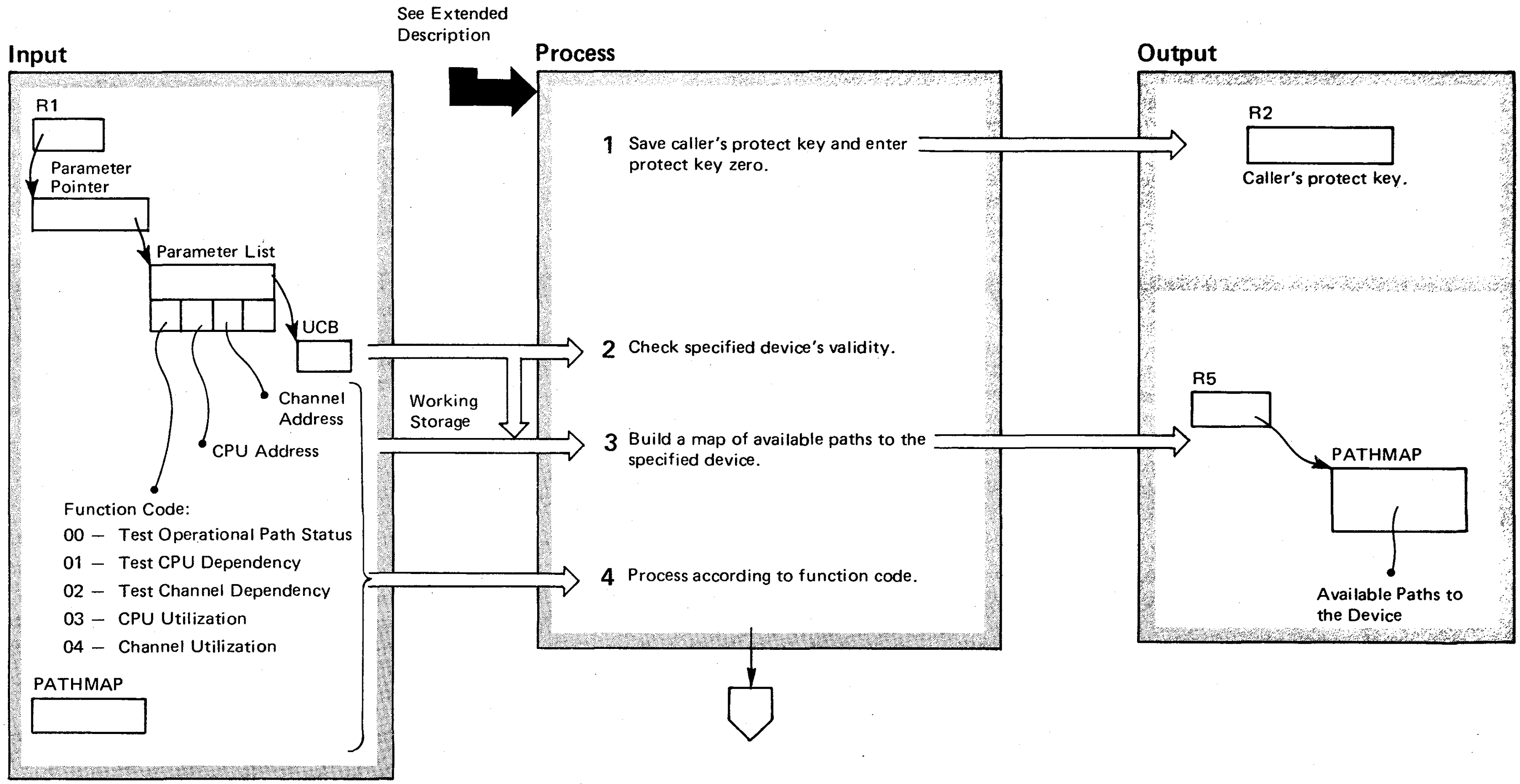


Diagram 2-57. Stopping and Restarting (QUIESCE) (via an Interrupt) the System (IEESTPRS) (Part 4 of 4)

Extended Description	Module	Label
6 For a uni-processor (UP) system, the routine bypasses much processing that is used for a multiprocessing (MP) system. The common system data field CSDMP contains the indicator flag that specifies the processor configuration. For an MP system, the routine stores the identification of the CPU which is executing this routine (IEESTPRS).	IEESTPRS	
7 For an MP system, all CPUs but the master CPU (the CPU on which this routine is being executed) are brought to a stop. Then the master CPU stops itself.		CPUSCAN (STOPSTOR) LASTCPU
For a UP system, the CPU places itself in the wait state.		
7a For each CPU, as determined by the CPU-alive mask, CSDCPUAL, the routine uses the inter-processor communicator (IPC) to issue the signal processor instruction SIGP SSS to halt the CPU and store the CPU status. The following are among the status items stored: • Timer and clock comparator data. • General, floating, and control registers. • Interrupt PSWs. • Current PSW for the given CPU. The routine also uses the SIGP SENSE instruction to determine if a CPU has stopped.	IEESTPRS	CPUSCAN
Normally, all CPUs other than the master CPU stop in the manual state. The routine then constructs a restart new PSW with the 'wait' bit on (1) and uses the IPC to issue a SIGP RESTART instruction. This procedure places the target CPU in the wait state with the wait state code passed by the caller.		STOPSTOR
Subroutine STOPSTOR then changes the target CPU's restart new PSW to point to the restart first level interruption handler routine in module IEESTPRS.		
After the non-master CPUs have been stopped, this routine proceeds to stop the master CPU. It sets the resume PSW (located in the prefix storage area (PSA)) to point to the cleanup routine of module IEESTPRS. It places the general registers and the other status information given previously (for the master CPU) in this step into the save area (addressed by register one). It then sets the restart new PSWs (that is, the new program check, the new SVC, and the new machine check PSWs) to point to the first level interruption handler code in module IEESTPRS. This causes machine check, program check, and SVC interruptions to be disregarded until the system has been restarted. The routine then uses the IPC to issue the SIGP STOP instruction for the master CPU to place itself (that is, the master CPU) in the manual stopped state.	IEESTPRS	LASTCPU
7b For a UP system, the processing is similar to that for the master CPU, as discussed in step 7a. The difference is that instead of issuing a SIGP STOP instruction, the routine loads a disabled wait state PSW and places the CPU in a wait state. A single CPU operating with the MP feature enters a stopped manual state without the wait state code being set in the PSW.		LASTCPU
8 When an operator desires to restart the system after a quiesced state, he hits the Restart button on any one of the CPUs. That CPU becomes the master CPU and executes the first level interruption handler routine of module IEESTPRS (since the restart PSW points to that routine).		RESTFLIH
The master CPU then adjusts the job-step timing for all address spaces and issues a SIGP RESTART instruction to each of the other CPUs indicated in the CPU-restart mask. Each of the CPUs uses a portion of this (RESTFLIH) code to restore its own status conditions from its 208-byte save area. The master CPU then issues a SYSEVENT SYQSCCMP macro instruction to the system resources manager to indicate that the CPUs have restarted. Each CPU then reloads its resume PSW to continue program processing at the point where it was when the stop function was implemented.	IEESTPRS	RESTFLIH
The master CPU first executes a small routine to verify that all CPUs have successfully been brought on line (alive). It also informs dynamic system support (DSS) routines that the CPUs are alive. It then proceeds to load its resume PSW and begin program execution from the stop point.		CLEANUP

Diagram 2-58. Device Information Subroutine (V PATH, * CH, V CPU) (IEEVDEV) (Part 1 of 4)



Extended Description	Module	Label
This routine obtains information regarding the condition of available paths to a device and provides that information to the caller of the routine.		
1 This routine may be entered from one of the following callers:	IEEVDEV	
• Vary CPU or Vary Channel: to check function codes 00, 01, 02, 03, and 04.		
• Vary Path: to check function code 00.		
• Allocation Recovery: to check function code 00.		
• Dynamic Device Reconfiguration: to check function code 00.		
• Volume Attribute Processing: to check function code 00.		
• MP VARY Command Pre-Processor (path availability checker): to check function code 00.		
The caller's protect key is saved and the device subroutine enters a key of zero so it can manipulate UCB hierarchy or operational reason indicators.		
2 The routine determines if the UCB (device) ID is valid. The first word of the parameter list contains the UCB address.		
3 The routine uses the IOSGEN MAP macro function to obtain a table of available paths to the specified device. The routine passes a work area location to contain the path map.		

Extended Description	Module	Label
4 Depending on the function code (FC) specified as an input parameter, processing and results occur as indicated below.	IEEVDEV	
• FC = 00: Device should be offline. The routine determines if the specified device is already online. If so, the routine will return a code of 20. Otherwise, the routine determines if a system function (such as OLTEP) is using the device. If so, the routine will return a code of 16. Otherwise, the routine determines if a logical path to the device exists. If a path is unavailable, the routine returns a code of 4. In this case, the UCB hierarchy bit (UCBVHRSN) has been set to 1.		
Other return codes from this module are: 0, if an operational path to the device is available; 8, if working storage is unavailable – in this case, no path checks are made; 12, a logical path to the device exists, but an operational (physical) path is unavailable. In this case, the UCB operator bit (UCBVORSN) has been set to 1.	IECVIOPM (See Note)	
• On the basis of this determination, the routine issues a return code of either 0, if available paths other than the specified unit exist, or 4, if the CPU or channel represents the last available path.	IECVIOPM	
Note: IEEVDEV links to module IECVIOPM only if function code 00 is in effect and if the path map table indicates that device paths are available. IECVIOPM's return code indicates the operational or non-operational status of the path(s).		
FC = 01 or 02: Device may be offline or online. The routine determines if the CPU address and, in the case of FC = 02, the channel address are valid. (An invalid address results in an abend situation.) Then the routine determines if the specified CPU (for FC = 01) or if the specified channel on the specified CPU (for FC = 02) represents the last available path to the device.	IEEVDEV	CHNLEVEL CPULEVEL
• FC = 03 or 04: Device may be offline or online. The routine determines either if the CPU (for FC = 03) is used in any path to that device or if the channel (for FC = 04) is used in any path to that device.		

Diagram 2-58. Device Information Subroutine (V PATH, * CH, V CPU) (IEEVDEV) (Part 3 of 4)

Input

R15

Return Code from IECVIOPM.
Code 0 or 20 Indicates Existence of Path.

Process

6 Set return code for caller.

7 Reset caller's protect key.

Caller (see extended description, step 1)

Output

R15

Return Code

Diagram 2-58. Device Information Subroutine (V PATH, * CH, V CPU) (IEEVDEV) (Part 4 of 4)

Extended Description	Module	Label
6 The status code returned from the IECVIOPM subroutine is converted to a return code from the device subroutine and returned to the calling routine.	IEEVDEV	
7 The calling routine's protect key is restored.		

Additional Considerations for This Diagram:

1. All system components that bring a device online should use the IEEVDEV subroutine with function code 00 to ensure that the device has an operational path.
2. If a device is offline due to operator reasons (bit UCBVORSN = 1), the IEEVDEV subroutine caller should have the authority to bring the device back online — and the caller should set the UCBVORSN bit equal to 0 when this is done.
3. The operational path status test routine IECVIOPM receives control via a LINK macro instruction and requires a UCB address as input.
4. In addition to building a map (table) of available paths for a specified device, the IOSGEN MAP macro function interrogates the online/offline status of each CPU and channel that comprises the path and indicates this status in the path status field of the map.

Diagram 2-59. Deleting a Virtual Memory (IEAVEMDL) (Part 1 of 2)

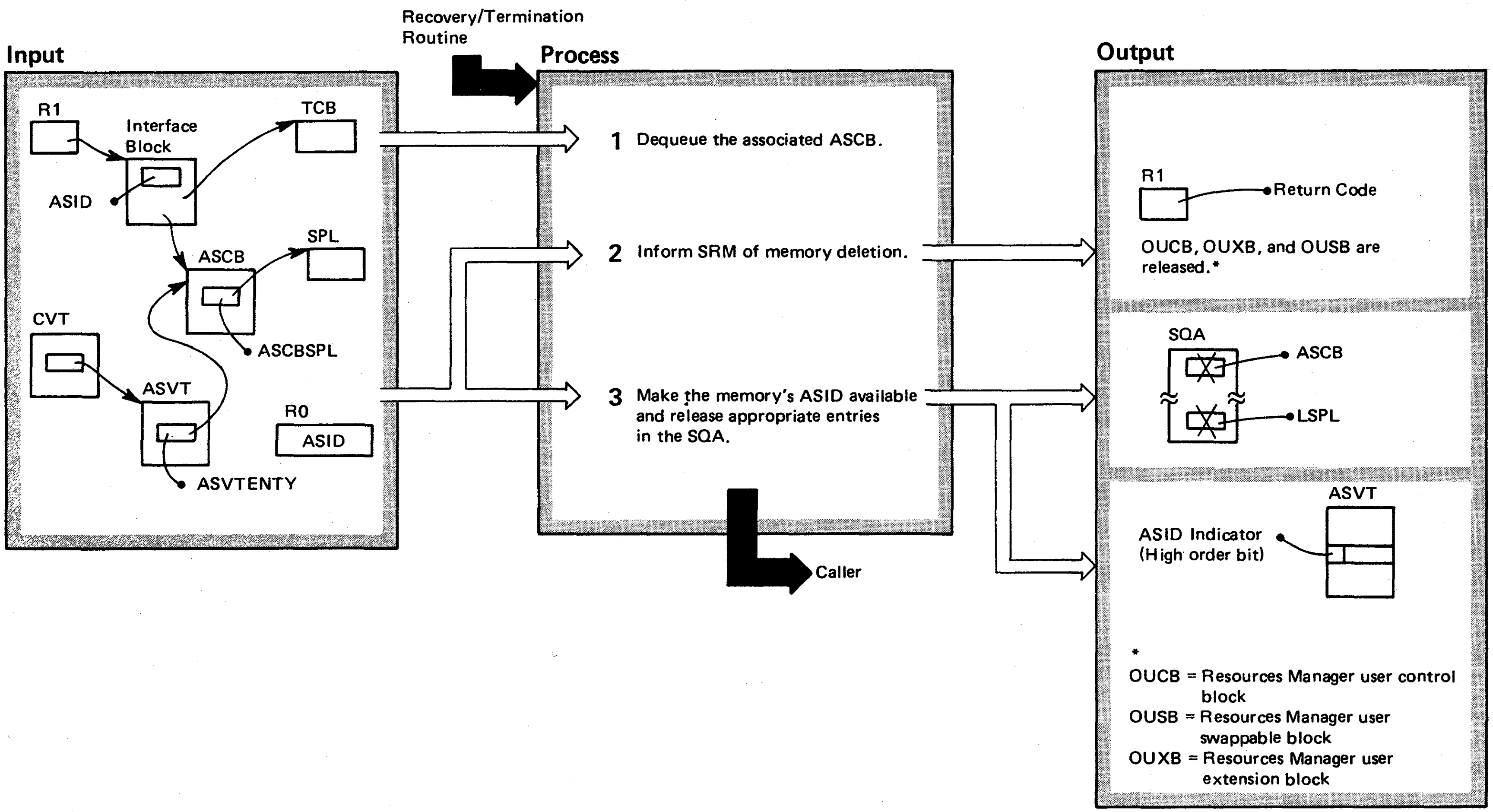


Diagram 2-59. Deleting a Virtual Memory (IEAVEMDL) (Part 2 of 2)

Extended Description	Module	Label
This process makes available the ASID of the given virtual memory that is being deleted. If another routine is asynchronously referencing the ASCB queue, that routine must complete before the given memory's ASCB is freed.		
1 This routine uses the ASCBCHAP macro instruction to dequeue the ASCB from the ASCB ready queue.	IEAVEMDL	
2 The routine uses the SYSEVENT MEMDEL macro instruction (via SVC 95) to inform the SRM. This allows the SRM to release the resources that the memory used. If the SRM is unable to allow the occurrence of memory deletion, module IEAVEMDL waits until SRM posts the ECB for the associated ASCB to indicate that deletion may occur.	IRARMINT	
3 The ASID release occurs within the environment of a page fix, a global lock, and a functional recovery routine. This permits serialized alterations to the ASVT. The release is indicated by setting the memory's ASVT entry for the ASID. The ASCB and SPL resources are then released from SQA.	IEAVEMDL	
Note: This processing occurs in the master scheduler's memory and under an ESTAE environment.	IEAVEMDL	MDLESTAE

Diagram 2-60. SETDMN Command Processing (IEE8603D) (Part 1 of 4)

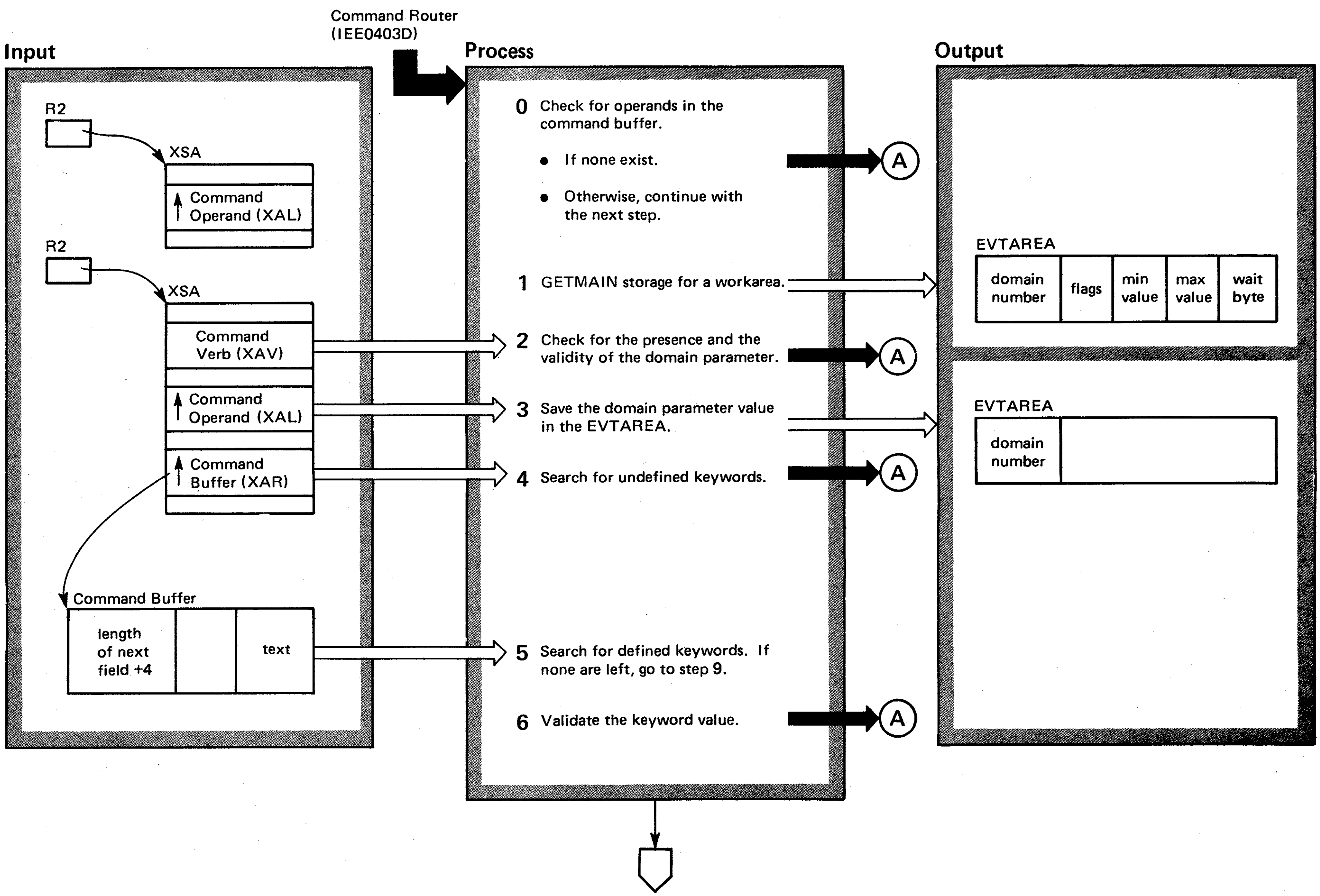
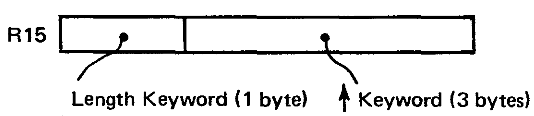


Diagram 2-60. SETDMN Command Processing (IEE8603D) (Part 2 of 4)

Extended Description	Module	Label
This routine changes values in the Domain Descriptor table (DMDT).		
0 If the XAL field is zero, no operands exist. If there are no operands, issue error message 'IEE311I SETDMN PARAMETER MISSING'.	IEE8603D IEE0503D	
1 Storage for EVTAREA comes from subpool 245 (SQA). This area is passed as a parameter to the sysevent processor.	IEE8603D	GETSTOR
2 The XAL field contains zeros if no operands are present. Error conditions are:		VALDMN
a) No operands: issue msg: 'IEE311I SETDMN PARAMETER MISSING'.	IEE0503D	
b) Domain number not in range 1-128: issue message 'IEE535I SETDMN INVALID PARAMETER'.	IEE0503D	
3 The domain number is translated to binary and stored in the EVTAREA.	IEE8603D	TRANSLAT VALDMN
4 The buffer is searched for keywords other than 'MIN', 'MAX', and 'WT'. If any are found,		BADKEYW
issue error message 'IEE309I SETDMN UNIDENTIFIABLE KEYWORD'.	IEE0503D	
5 The scan macro IEEBUFSC is used to locate keywords. Input to IEEBUFSC is:	IEE8603D	SCANKEYW

R0 (points to the last byte of the buffer +1)

R1 (points to the beginning of the buffer)



keywords are: 'MIN', 'MAX', and 'WT'.

Extended Description	Module	Label
OUTPUT from IEEBUFSC is: R1 (length of the keyword value) R14 (pointer to the first byte of the keyword value) R15 return codes: 0 keyword and value found 4 keyword value is invalid 8 keyword not found in buffer		
For RC 4 issue message 'IEE708I KEYWD KEYWORD, VALUE INVALID'.	IEE0503D	
For RC 8, the next keyword is search for (step 5).		
6 The keyword value is checked against the proper range. MIN (0-255) MAX (0-255) WT (1-255)	IEE8603D	VALKEYWV
If the keyword value is not in the valid range, issue message 'IEE708I KEYWD KEYWORD, VALUE INVALID'.	IEE0503D	

Diagram 2-60. SETDMN Command Processing (IEE8603D) (Part 3 of 4)

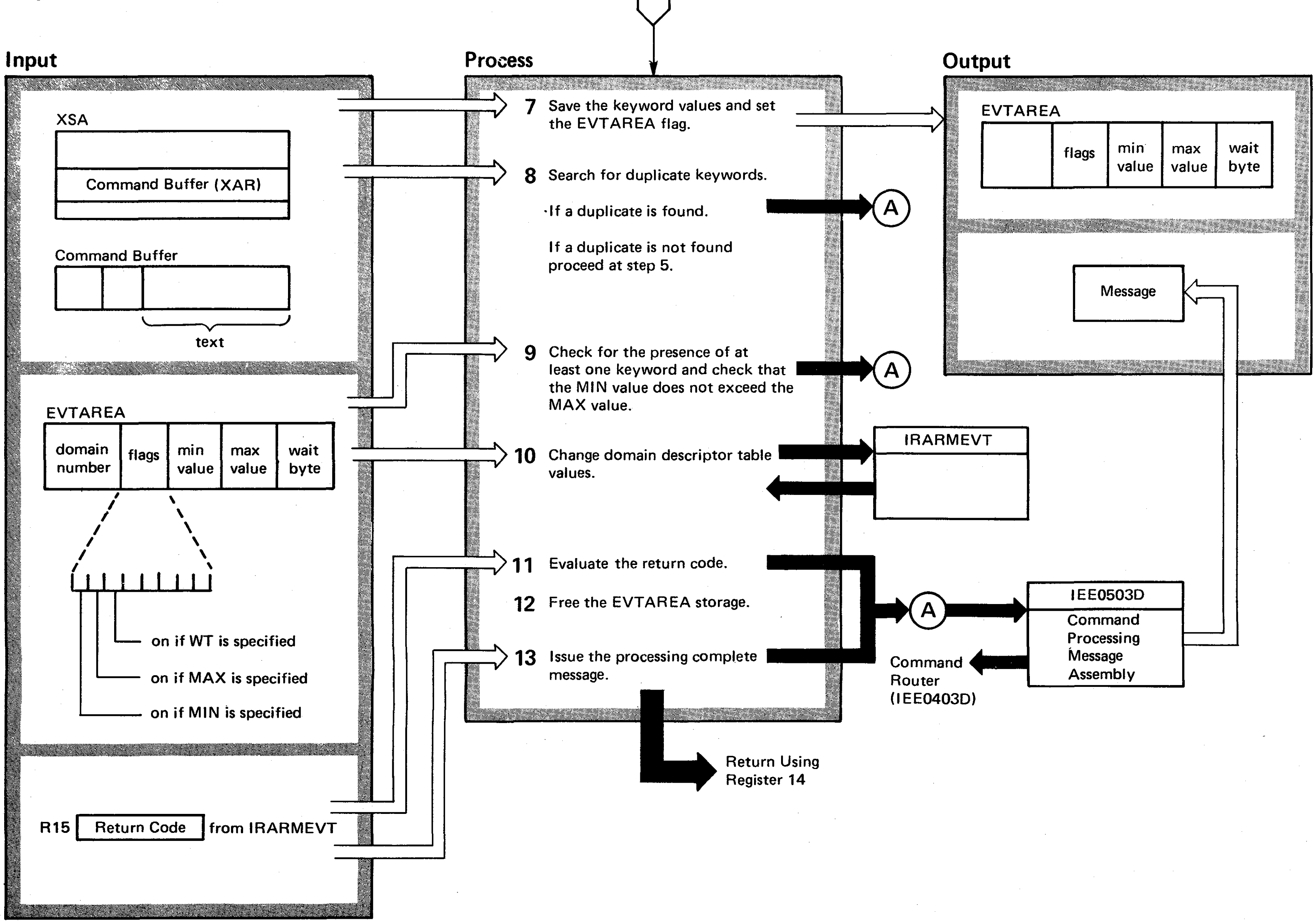


Diagram 2-60. SETDMN Command Processing (IEE8603D) (Part 4 of 4)

Extended Description	Module	Label
7 SETDMN translates the value to binary and saves it in EVTAREA. Also, the appropriate flag signalling the presence of this keyword is set.	IEE8603D	VALKEYWV
8 SETDMN issues a IEEBUFSC macro to look for duplicates.		DUPKEYW
R1 (pointer to the byte after the domain value)		
If RC is 0 or 4, issue error message 'IEE312I SETDMN PARAMETERS CONFLICT'.	IEE0503D	
If RC is 8, Step 5 executes next.		
9 If the MIN value exceeds the MAX value, issue error message 'IEE312I SETDMN PARAMETERS CONFLICT'.	IEE0503D	
If no keywords were specified, issue error message 'IEE310I SETDMN KEYWORD MISSING'.		
10 A branch entry to IRARMEVT is made via sysevent number 37 to set values in the domain descriptor table.	IRARMEVT	
11 The return code is evaluated. For return code 4: 'IEE797I DMN NNN NOT DEFINED IN DOMAIN TABLE' is the error message.	IEE0503D	
For return code 8: 'IEE798I MIN VALUE EXCEEDS MAX VALUE IN DOMAIN TABLE' is the error message.	IEE0503D	

Extended Description	Module	Label
12 Storage is freed from SQA for EVTAREA.		
13 Issue message 'IEE712I SETDMN PROCESSING COMPLETE'.	IEE0503D	

Note: a) Input to IEE0503D

XAV	command name or keyword name
XAE	message index
XAU	UCMI for WTO
XAA	ASID for TPUT

b) SETDMN command processing returns using the contents of register 14 passed on entry by IEE0403D.

c) IEE0503D is invoked to produce a message to the issuer of the SETDMN command. IEE0503D always returns to the step that called it, then IEE8603D returns to its caller.

Error Processing:

Unexpected errors occurring during SETDMN command processing are handled by the caller's ESTAE.

The Region Control Task (RCT) is the highest priority task in an address space and is swapped with the user's tasks. The RCT functions are:

- To prepare an address space to be swapped out.
- To prepare an address space for execution after it has been swapped in.
- To ensure proper scheduling of a user attention exit.

When a new user starts a job, RCT's Initialization routine receives control from Address Space Create to perform initialization functions like attaching the dump task and the Started Task Control task.

When the System Resources Manager determines that an address space should be swapped out, it posts the RCT Quiesce routine. The Quiesce routine sets all quiescable SRBs and tasks under the RCT nondispatchable, purges all I/O requests, and calls the RSM Swap-Out routine to initiate the swap-out. It also performs address space activity checking, to determine whether there is any work to be processed in the address space, and notifies the System Resources Manager of the result.

When an address space is swapped in, Quiesce receives control, sets an indicator requesting Restore processing and passes control to RCT Common Processing. RCT Common processing passes control to the Restore routine. The Restore routine prepares the address space so that it can execute again by rescheduling purged I/O, resetting all tasks under the RCT dispatchable, and notifying the System Resources Manager if the address space is in long-wait condition, having no work to be processed. Restore also handles Quiesce backout, restoration of an address space after Quiesce has failed.

When a user requests an attention exit, RCT routines ensure that it is properly scheduled and executed.

When the Initiator, MOUNT Processor, or TSO session has ended, the RCT Termination routine gets control from RCT Common Processing. The Termination routine performs housekeeping functions and returns control to allow the address space to be freed.

Functional recovery routines are incorporated with the following routines:

- Quiesce
- Restore
- STAX
- Attention Exit Scheduler
- Attention Exit Prolog and Epilog
- Attention Exit Purge

ESTAE processing is performed by code residing in the RCT Initialization/Termination module which routes control to individual modules for specific error processing.

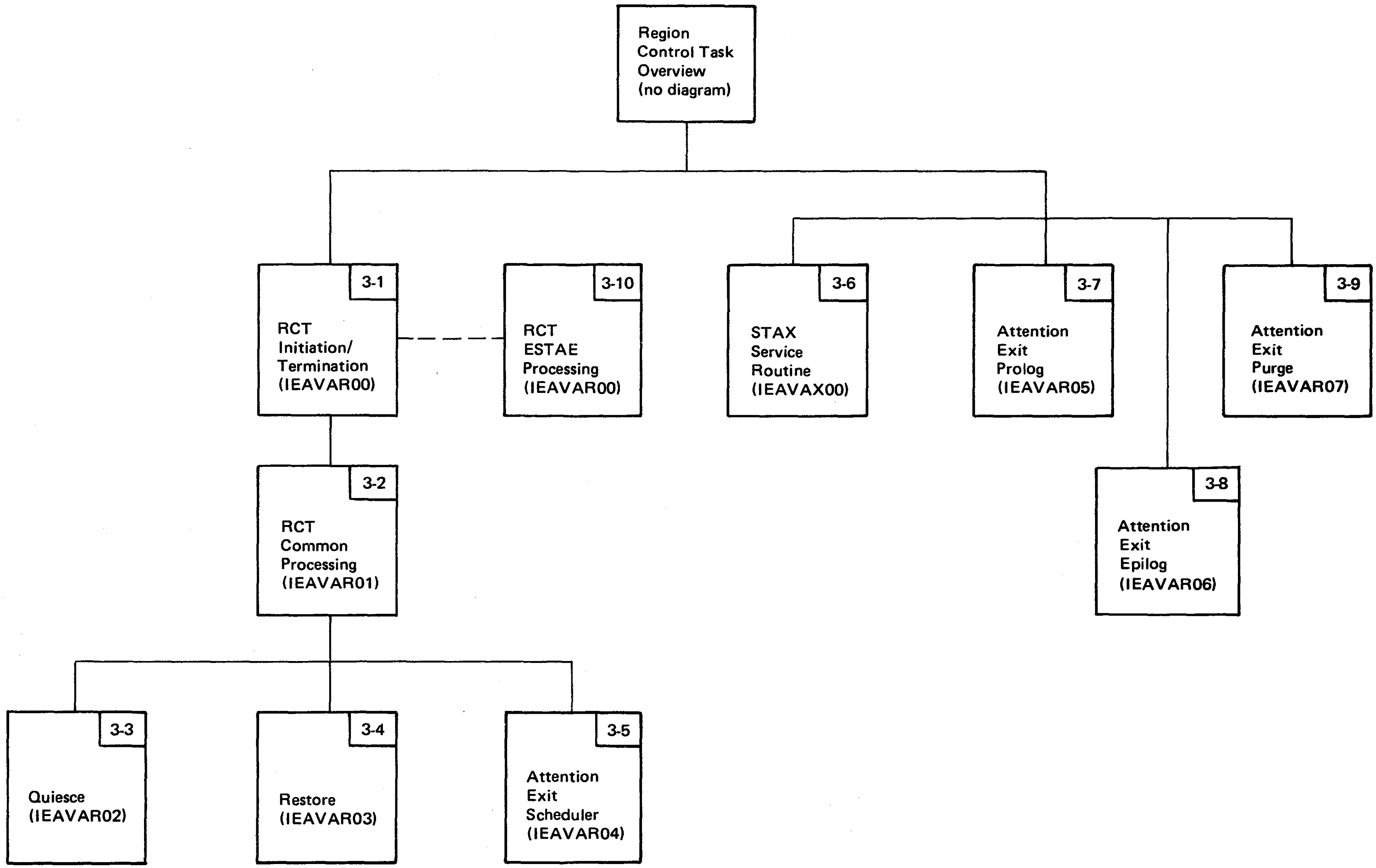


Figure 2-6. Region Control Task Visual Contents

Diagram 3-1. RCT Initialization/Termination Routine (IEAVAR00) (Part 1 of 2)

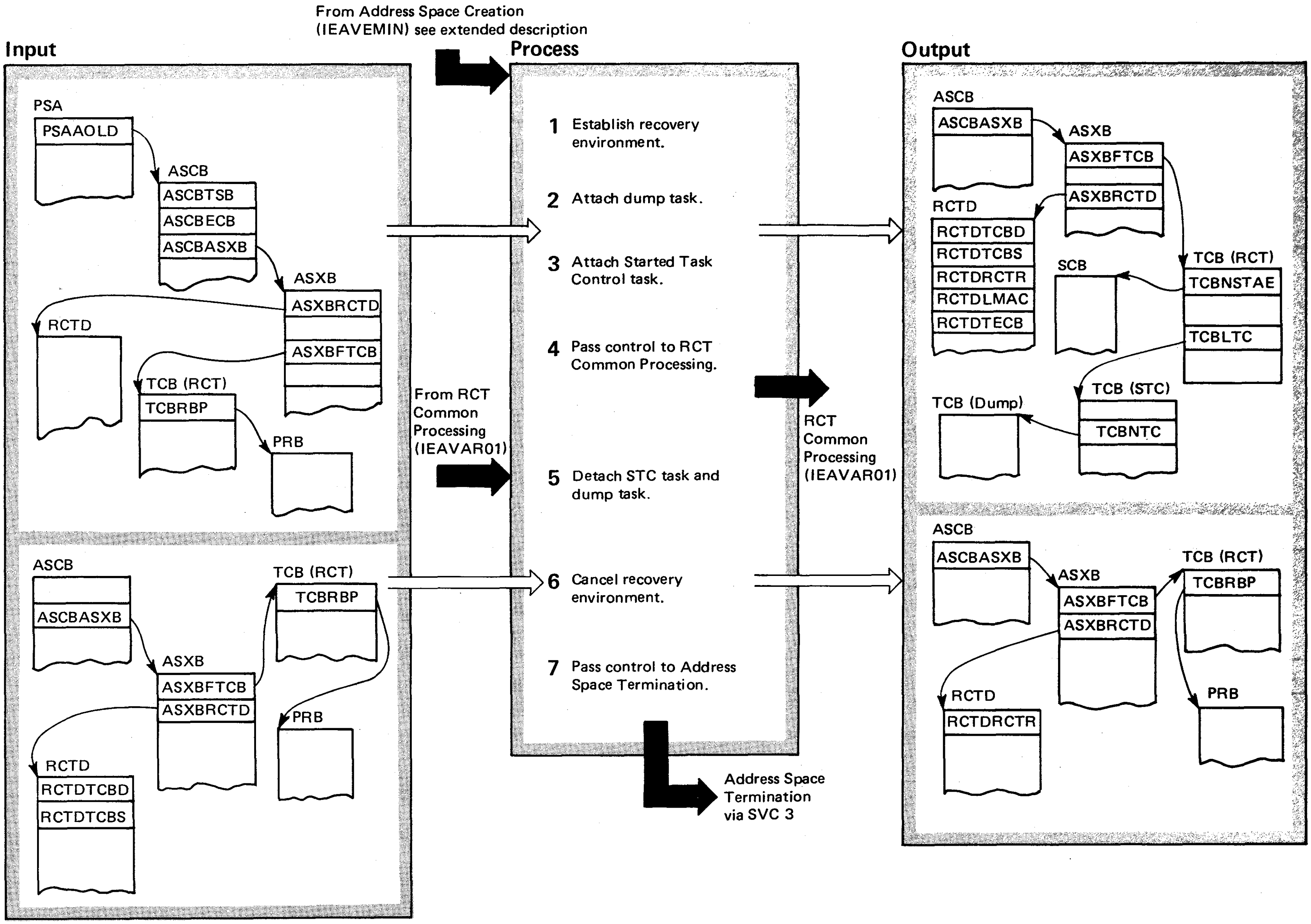


Diagram 3-1. RCT Initialization/Termination Routine (IEAVAR00) (Part 2 of 2)

Extended Description	Module	Label
The RCT Initialization/Termination routine (IEAVAR00) prepares an address space for use when called by Address Space Creation. Memory Initialization (IEAVEMIN) initializes the first TCB and SVRB to enter program manager to route control to IEAVAR00. When the address space is terminating, this function releases any attached tasks before allowing the Address Space Termination function to take control.		
1 RCT Initialization first issues an ESTAE macro instruction to set up a recovery environment. If the ESTAE sets a non-zero return code to indicate failure, RCT Initialization causes an error message to be issued to the operator and to the terminal user, if one exists.	IEAVAR00	IEAVAR00
2 RCT Initialization sets status flags in the RCT Data Area (RCTD) and then attaches the dump task. During the attaching process, RCT Initialization requests that an ECB be posted at termination of the dump task and puts the dump task TCB address in the RCTD. If an error occurs, indicated by a non-zero return code, RCT Initialization causes an error message to be issued and then issues ABEND (code 078) to invoke R/TM.	IEAVEATO	
3 RCT Initialization attaches the STC (Started Task Control) task, requesting an ECB to be posted at STC termination, and then puts the STC task's TCB address in the RCTD. If an error occurs, indicated by a non-zero return code, RCT Initialization causes an error message to be issued before issuing an ABEND (code 078) to invoke RTM.	IEAVEATO	
4 When all ATTACH processing is complete, RCT Initialization passes control to RCT Common Processing.		
5 When RCT Common Processing returns control at termination, RCT Termination frees resources associated with the STC task and with the dump task.	IEAVEEDO	
6 RCT Termination issues an ESTAE macro instruction to cancel the recovery environment.		
7 RCT Termination passes control to Address Space Termination for further termination processing. (Refer to the figure in Recovery Termination Management, The Process of Normal Task Termination, for more detailed information on the termination process.)		
Error Processing: If an error occurs while RCT Initialization/Termination is in control, RTM passes control to RCT's ESTAE (IEAVAR00). The ESTAE routine determines if: • RCT has had the error. • Percolation did not occur. • The previous STA exit had the error. If any of these conditions exist, a SVC DUMP of LSQA is taken. Then the ESTAE indicates for RTM to continue with termination.	IEAVAR00	IEAVAERO

Diagram 3-2. RCT Common Processing Routine (IEAVAR01) (Part 1 of 2)

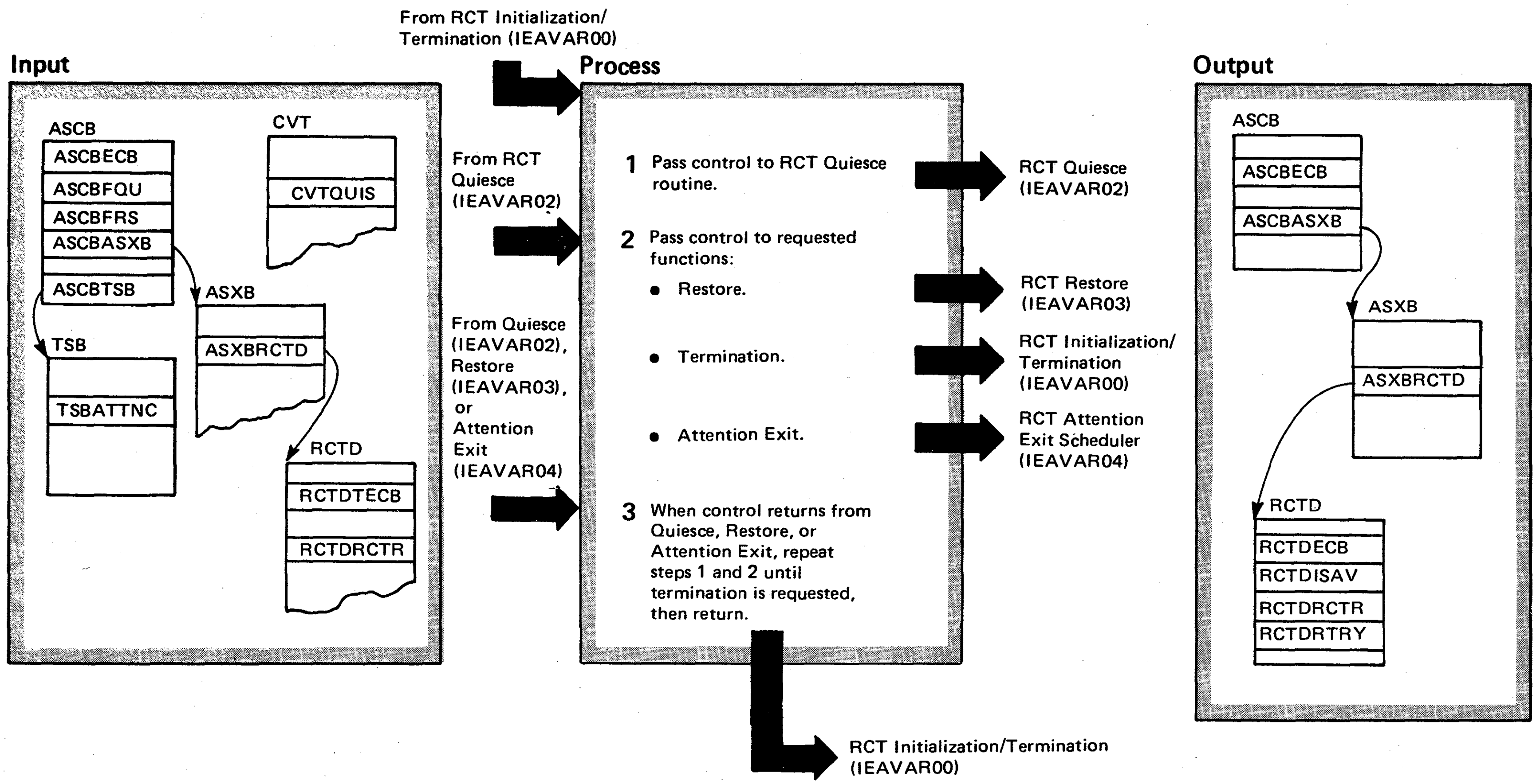
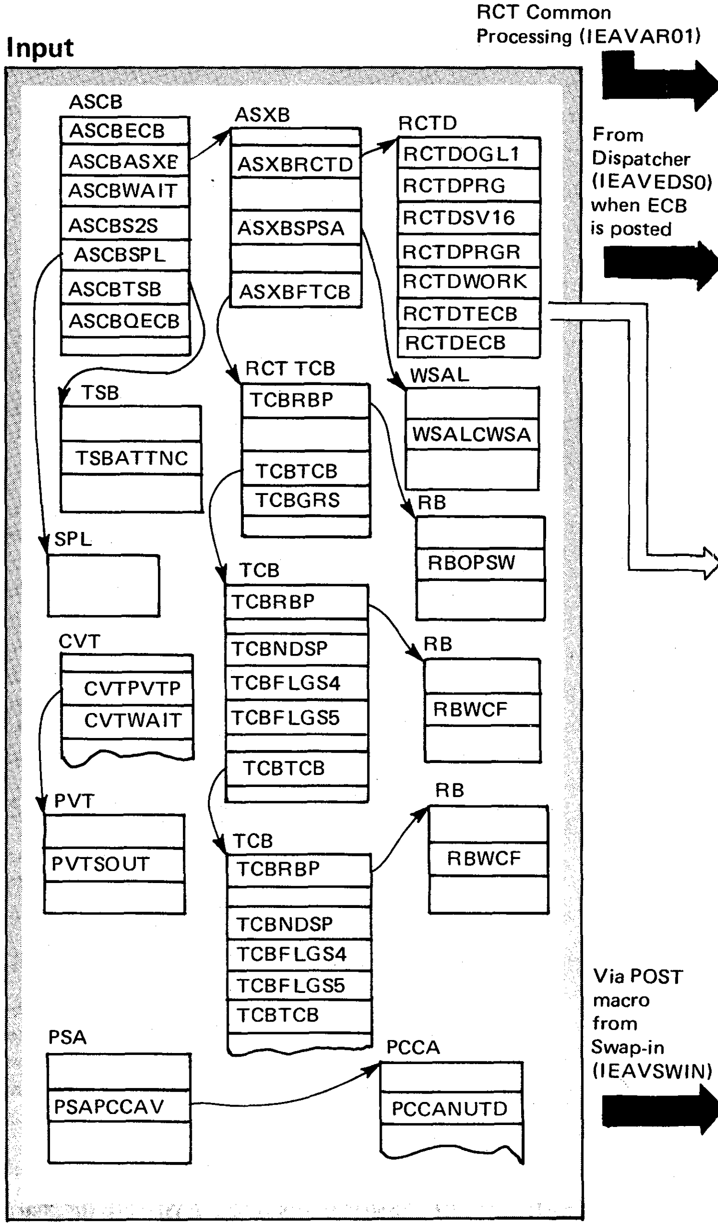


Diagram 3-2. RCT Common Processing Routine (IEAVAR01) (Part 2 of 2)

Extended Description	Module	Label
The RCT Common Processing Routine (IEAVAR01) routes control within the RCT modules to wait for a functional request in the Quiesce module who passes control to the other requested functions: Termination, Restore, or Attention Scheduling.		
1 RCT Common Processing initializes an ECB list consisting of a termination ECB and a work ECB. Then it sets a status flag in the RCTD and passes control to the Quiesce module that will enter the wait state until one of the ECBs is posted.	IEAVAR01	IEAVAR01
2 Control is returned to RCT Common Processing after one of the ECBs is posted in Quiesce. If SRM posts the work ECB, Quiesce will handle this request prior to returning to Common Processing. Otherwise, if Task Termination posts the termination ECB for termination processing, or if Terminal I/O Control posts the work ECB for Attention Scheduler processing, control is immediately returned.		
RCT Common Processing checks for a Restore request in the ASCB (ASCBFRS). If one exists, it sets a status flag in the RCTD and routes control to the Restore function to satisfy the request.		

Extended Description	Module	Label
Next, RCT Common Processing checks the termination ECB; if it is posted, RCT Common Processing sets a status flag in the RCTD and passes control to RCT Initialization/Termination. RCT Common Processing checks for a TSB (Terminal Status Block). If none exists, RCT Common Processing re-invokes Quiesce to enter a wait state. If one exists, RCT Common Processing checks for attention requests (TSBATTNC). If none exist, RCT Common Processing re-invokes Quiesce to enter the wait state; otherwise, RCT Common Processing sets a status flag in the RCTD and passes control to the RCT Attention Scheduler.		
3 When Quiesce, Restore, or Attention Scheduler processing returns control to RCT Common Processing with no other requests to honor, it invokes Quiesce to wait until an ECB is posted. If termination has been requested, RCT Common Processing passes control for termination and will not be re-entered.		

Diagram 3-3. Quiesce Routine (IEAVAR02) (Part 1 of 3)



Process

1 Wait for ECB to be posted.

2 Handle requested function:
- Termination, go to step 15.
- Quiesce, continue with step 3.
- Attention Exit, go to step 15.

3 Set up recovery environment.

4 Stop RCT subtasks.

5 Check for ready tasks for the address space.

6 Notify SRM that Quiesce has started.

7 Purge user I/O activity.

8 Stop the dispatching of SRBs.

9 Check for address space activity.

10 Notify SRM that Quiesce is complete.

11 If swap-out is possible, call RSM to initiate swap-out. If not, go to step 13.

12 Wait for ECB to be posted, indicating restore processing. Then go to step 14.

13 If no swap is to take place, restart SRB dispatching.

14 Prepare for Restore processing.

15 Return.

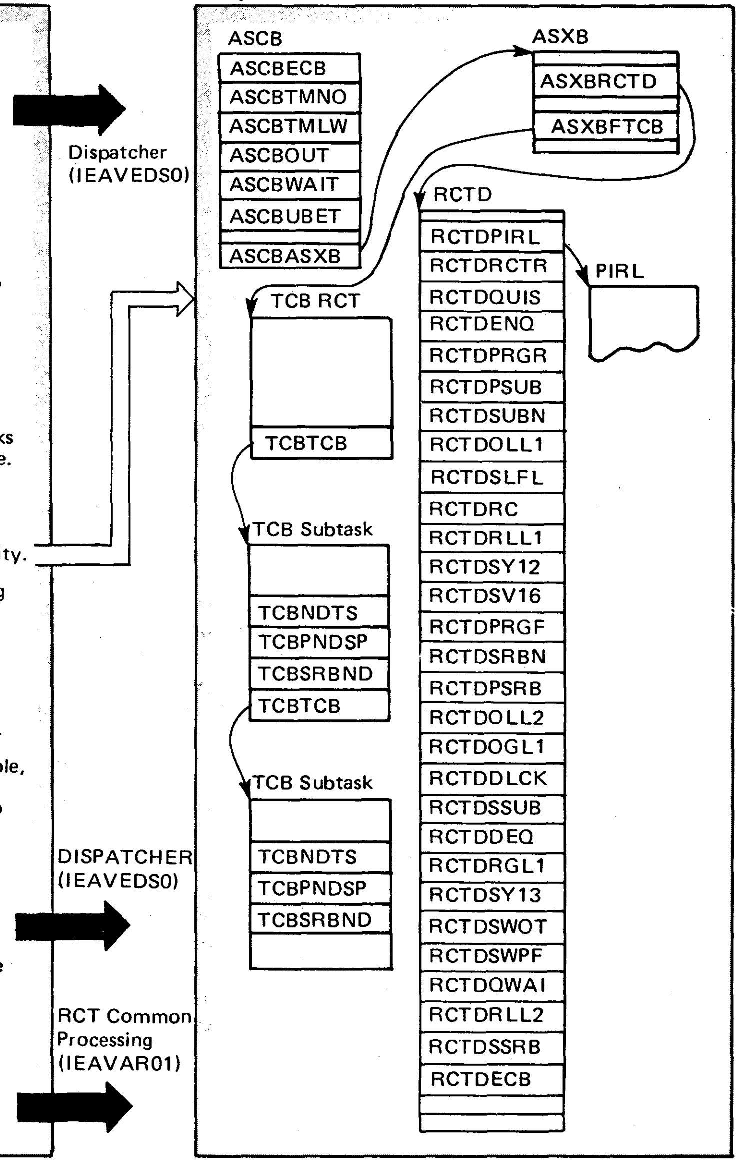


Diagram 3-3. Quiesce Routine (IEAVAR02) (Part 2 of 3)

The Quiesce routine (IEAVAR02) waits for functional requests. If posted by SRM, it prepares an address space for swap-out by stopping address space activity and checking for long wait requests. Either when swap-in is ready for Restore processing or when posted for a Termination or Attention Exit request, Quiesce routes control back to RCT Common Processing for further action.

Extended Description	Module	Label
1 Quiesce issues a WAIT macro instruction, passing an ECB list consisting of a termination ECB and a work ECB. If neither of the ECBs has been preposted, Quiesce enters the wait state.	IEAVAR02	IEAVAR02
2 Quiesce is entered when one of the ECBs is posted. Task Termination posts the termination ECB when the STC task terminates. The SRM or Terminal I/O control posts the work ECB when it requires Quiesce or Attention Scheduler processing.		
First, Quiesce sets the work ECB to zero. Then it checks the termination ECB; if it is posted, Quiesce returns control to Common Processing. Next, Quiesce checks for a Quiesce request in the ASCB (ASCBFQU); if requested, processing continues in the Quiesce module. Otherwise, the request was for Attention Exit processing, and control is returned to Common Processing to satisfy the request.		
3 Quiesce sets recovery flags in the RCTD and enqueues on the Purge Resource (SYSZEC16) before performing any quiescing.	IEAVAR02	IEAVAR02
4 Quiesce invokes STATUS which halts RCT subtask processing by setting nondispatchability flags in the subtask TCBs.		
5 Quiesce gets the local lock; if an error occurs, Quiesce issues an ABEND to route control to R/TM for error recording and action determination. If no error occurs, Quiesce sets up the FRR and checks to see if the address space is in a long wait situation. If so, Quiesce sets the high-order bit of register 1. Then Quiesce cancels the FRR and releases the local lock; if the release fails, QUIESCE issues an ABEND to route control to R/TM to get the error recorded and the appropriate action taken.		

Extended Description	Module	Label
6 Quiesce issues SYSEVENT 12 to notify SRM that Quiesce has started and to indicate whether the address space is in long wait. Quiesce checks the return code from SRM; if it is non-zero, Quiesce restarts the RCT subtasks, dequeues the Purge Resource, resets the ASCB Quiesce flag and returns to RCT Common Processing.		
7 If processing is to continue, Quiesce purges all I/O activity in the system by use of SVC 16. If the purge operation fails, Quiesce issues an ABEND to route control to R/TM for error recording and action determination.		
8 If the purge is successful, Quiesce issues CALLDISP to enter the Dispatcher and invoke the STATUS routine to stop SRB processing.		
9 Quiesce gets the dispatcher lock, checking the return code from SETLOCK for a non-zero (error) value. If an error is detected, Quiesce issues an ABEND to route control to R/TM to record the error and to take the appropriate action. Then Quiesce performs the following tests to determine whether the address space is in a long-wait condition, without work to be performed: • Checks for quiescable SRBs; if found, address space is not in long wait. • Checks to see if any I/O had been purged; if I/O was purged, the address space is not in long wait. • Checks for asynchronous exits (IQEs or RQEs) that are queued but have not been processed; if any are found, the address space is not in long wait. • Checks the TSB for any attention requests; if any exist, the address space is not in long wait. • Checks the TCB priority queue for ready tasks. Quiesce then releases the dispatcher lock and checks the return code from SETLOCK. If the return code is non-zero (error), Quiesce issues an ABEND to route control to R/TM to record the error and to take the appropriate action.		
10 Quiesce issues SYSEVENT 13 to indicate to SRM that Quiesce has completed and whether the address space is in long wait.		

Diagram 3-3. Quiesce Routine (IEAVAR02) (Part 3 of 3)

Extended Description	Module	Label
11 Quiesce checks the output from SRM to determine whether the address space is still in long wait and whether it should be swapped out. If swap-out is possible, Quiesce performs wait limit support; if the address space is not in long wait and if the TOD clock is usable, Quiesce deletes the FRR and then invokes Swap-Out to initiate the actual swapping. If the swap is unsuccessful, Quiesce issues an ABEND to route control to R/TM to get the error recorded and to get the appropriate action taken.		
12 If successful, Quiesce sets up the FRR, and issues a WAIT, waiting for Swap-in to post an ECB to call for Restore processing.		
13 If SRM determined that no swap should occur, Quiesce cancels the FRR, releases the local lock, and invokes STATUS to restart SRB processing. If the SETLOCK return code is non-zero, Quiesce issues an ABEND to route control to R/TM to get the error recorded and the appropriate action taken.		
14 When the restore ECB is posted, RCT will reset the recovery footprint to indicate that SRB's are no longer stopped, zero the restore ECB, and set a flag in the ASCB (ASCBFRS) to indicate that restore is being requested. Then it will branch to the common processing routine.		
Error Processing		
When an error in Quiesce locked code occurs, R/TM passes control to the FRR for Quiesce. The FRR checks for the type and location of the error. If the error was in Quiesce's address space, the FRR records information in the SDWA. If the error occurred after Swap-Out was called, the FRR issues CALLRTM to terminate the address space. Otherwise, it issues SETRP to record the error, free any locks held, and returns control to R/TM which will percolate control via SYNCH to the Quiesce ESTAE routine for further error processing. If error was not in Quiesce's address space, the FRR issues SETRP to route control to R/TM to continue with termination without recording.	IEAVAR02	IEAVAFR2

Diagram 3-4. Restore Routine (IEAVAR03) (Part 1 of 2)

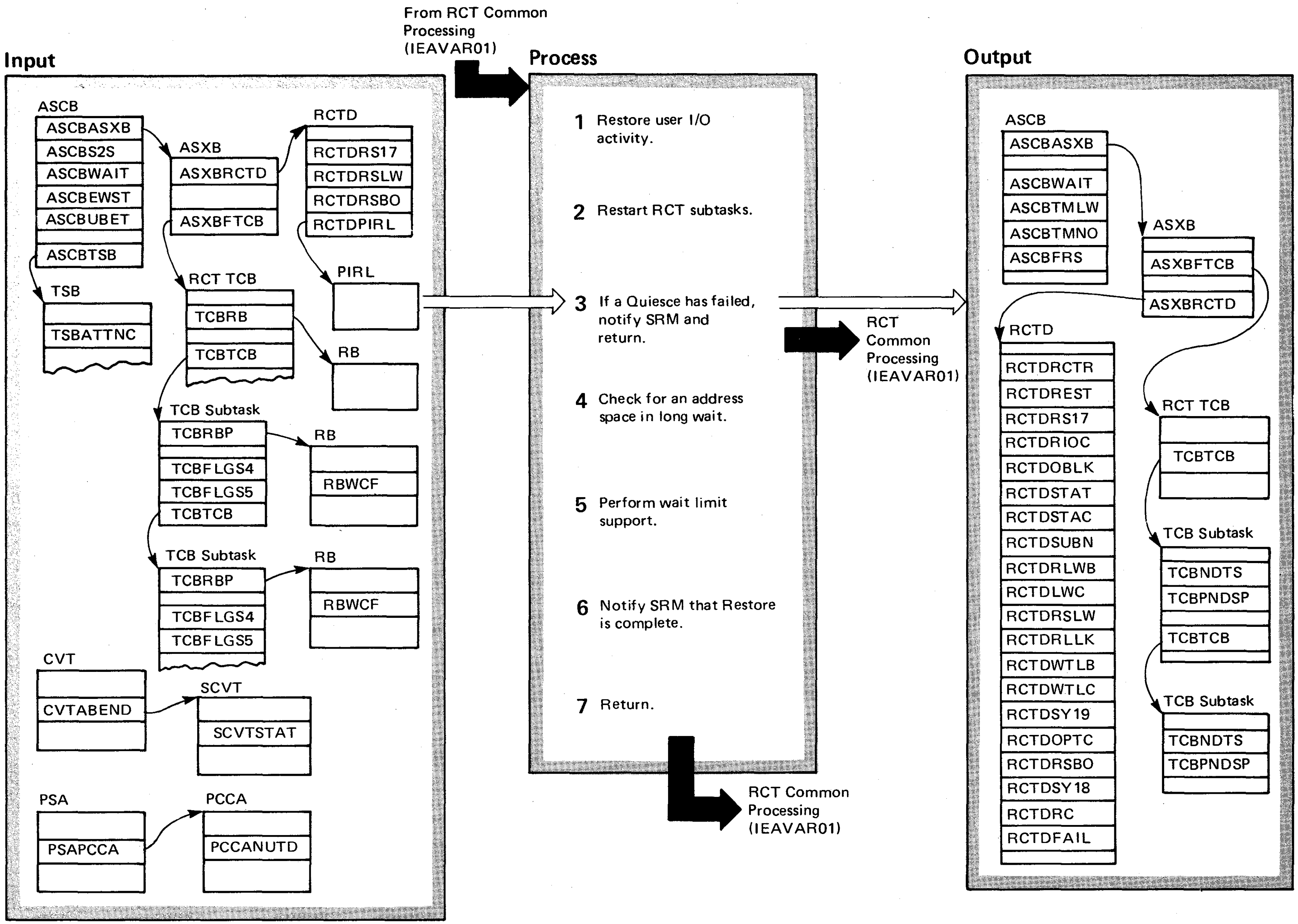


Diagram 3-4. Restore Routine (IEAVAR03) (Part 2 of 2)

Extended Description	Module	Label
The Restore routine (IEAVAR03) is called by Swap-in to prepare an address space for operation after it has been swapped in. Restore is also called to recover from a Quiesce operation that has failed.		
1 Restore sets recovery flags in the RCTD and then, if I/O was purged during quiescing issues SVC 17 to restore the I/O activity.	IEAVAR03	IEAVAR03
2 Restore gets the local lock; in case of a non-zero return code, Restore issues an ABEND to route control to R/TM to get the error recorded and the appropriate action taken. After establishing the FRR, Restore issues STATUS to restart the RCT's subtasks.		
3 If Restore has been entered to recover from a Quiesce that failed, Restore cancels the FRR and releases the local lock, checking the return code from SETLOCK and issuing an ABEND for a failure so that control is routed to R/TM to record the error and take appropriate action. Then it issues a SYSEVENT 18 to notify SRM that Quiesce has failed. Then it resets the indicators in the ASCB and returns control to RCT Common Processing.		

Extended Description	Module	Label
4 Restore checks to see if the address space is in long-wait status; it then deletes the FRR and releases the local lock, checking the return code from SETLOCK and issuing an ABEND for a failure so that control is routed to R/TM to record the error and take the appropriate action.		
5 If the address space is in long-wait status during the Restore process – or if the CPU clock is bad, Restore bypasses wait limit support. Otherwise, Restore performs wait limit support by noting when the address space entered the wait state.		
6 Restore issues SYSEVENT 19 to notify SRM that the Restore process has completed and whether the address space is in long wait.		
7 Restore resets the ASCB indicators and returns conrol to RCT Common Processing.		
Error Processing		
If an error occurs in Restore's locked code, R/TM passes control to the Restore FRR. The FRR checks the cause and location of the error and determines whether retry is possible. If the error was in Restore's address space, the FRR records error information in the SDWA and, if necessary, requests a dump. Then the FRR issues SETRP to free the local lock, record the error information, and return control to R/TM, requesting termination so that control is percolated to the Restore ESTAE routine. If the error was not in Restore's address space, the FRR issues SETRP to route control to R/TM to continue with termination without recording.	IEAVAR03	IEAVAFR3

Diagram 3-5. Attention Exit Scheduler Routine (IEAVAR04) (Part 1 of 2)

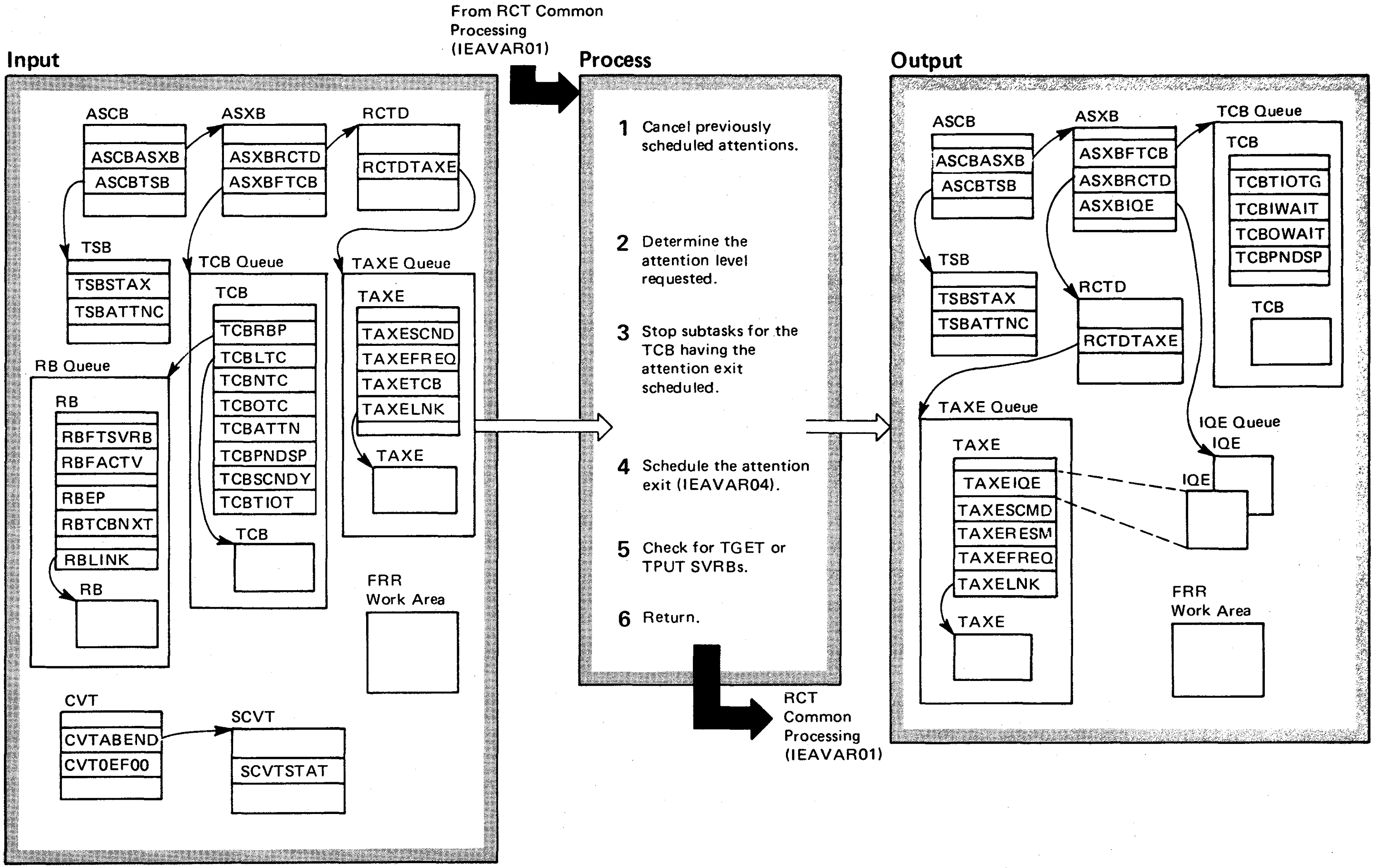


Diagram 3-5. Attention Exit Scheduler Routine (IEAVAR04) (Part 2 of 2)

Extended Description	Module	Label
The Attention Exit Scheduler (IEAVAR04) schedules the processing of user attention exits.		
1 The Scheduler routine gets the local lock and sets up the FRR. If the SETLOCK macro fails, Scheduler issues an ABEND to route control to R/TM to have the error recorded and the appropriate action taken. Otherwise, Scheduler finds any TAXEs scheduled but not executing and cancels them by setting a flag (TAXERESM) in the TAXE.	IEAVAR04	IEAVAR04
2 Scheduler finds the available TAXE as indicated by the requested attention level recorded in the TSB (TSBATTNC). The user attention count and the STAX count in the TSB are decreased by one for every TAXE marked unavailable during the search.		
3 Scheduler invokes Status to stop all subtasks under the TCB that are having the attention exit scheduled.		
4 Scheduler performs the scheduling of the attention exit by calling the Stage 2 Exit Effector.		

Extended Description	Module	Label
5 When control returns from the Stage 2 Exit Effector, Scheduler checks for any TGET/TPUT SVRBs. If it finds any, Scheduler invokes Status to reset that TCB dispatchable.		
6 Scheduler cancels the FRR and releases the local lock. If the release fails, Scheduler issues an ABEND to route control to R/TM to have the error recorded and the appropriate action taken. If no error occurs, Scheduler returns control to RCT Common Processing.		
Error Processing		
When an error occurs in Scheduler's locked code, R/TM passes control to the Scheduler FRR. The FRR determines whether the error is in Scheduler's address space; if not, the FRR returns control to R/TM to continue with termination. If the error was in Scheduler's address space, the FRR determines the type of error, indicates a dump or retry if necessary, tries to reset resources, requests that TPUT issue an error message, frees the local lock and records information in the SDWA. Then the FRR returns control to R/TM to record and to continue with termination. This percolation causes Scheduler's ESTAE routine to get control.	IEAVAR04	IEAVAFR4

Diagram 3-6. STAX Service Routine (IEAVAX00) (Part 1 of 2)

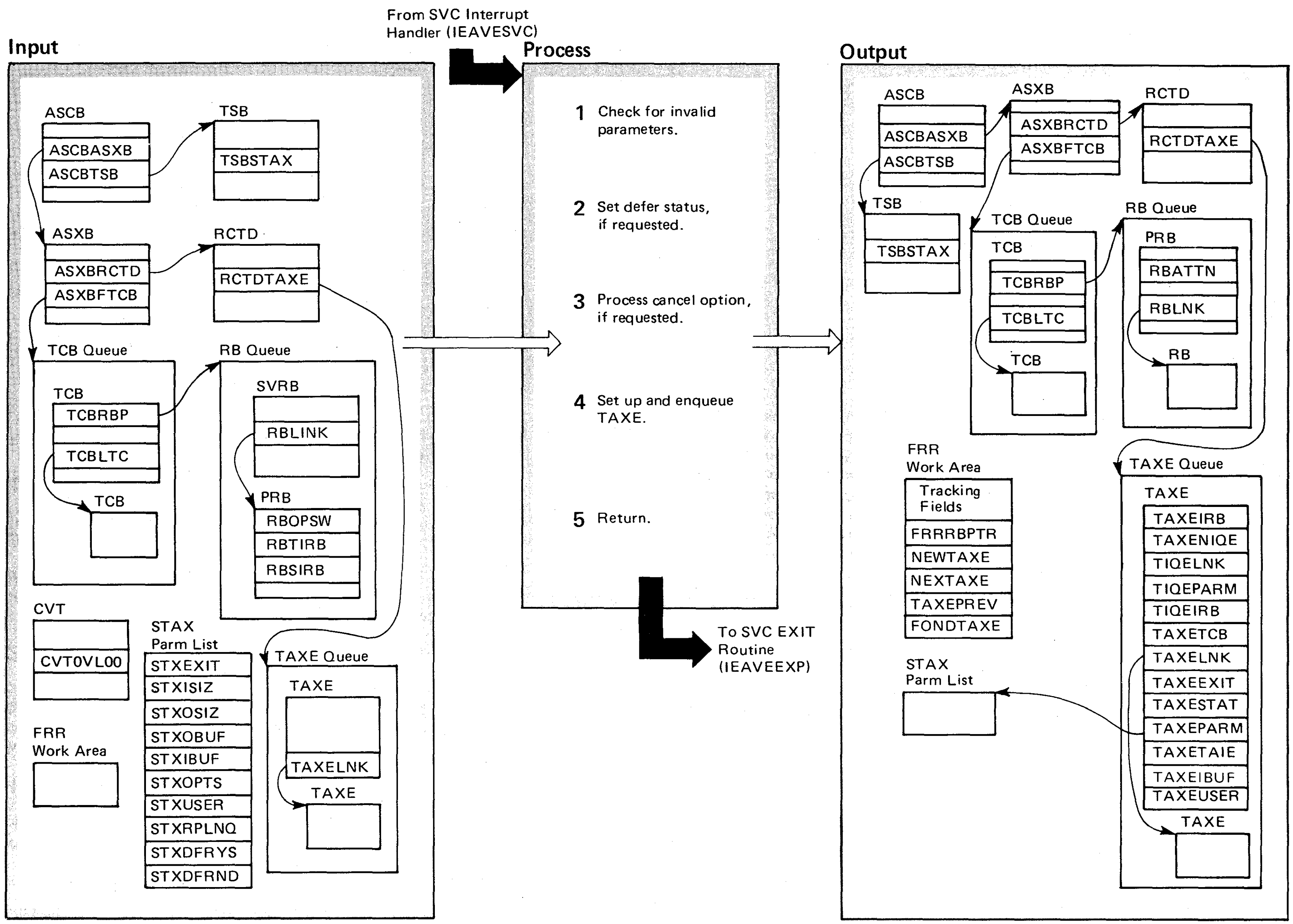


Diagram 3-6. STAX Service Routine (IEAVAX00) (Part 2 of 2)

Extended Description	Module	Label
The STAX Service routine (IEAVAX00) processes requests for user attention exits made with the STAX macro instruction. On entry to this routine, the local lock is held.		
1 After setting up the FRR, STAX verifies the parameters passed by the user. If a parameter is invalid, STAX cancels the FRR and issues an ABEND.	IEAVAX00	IGC0009F
2 If the defer option is requested, STAX indicates the option by setting the RBATTN flag.		
3 If the cancel option is requested, STAX finds the TAXE for the caller's TCB and determines whether it is active. If the TAXE is not active, STAX updates the STAX count, dequeues the TAXE, and calls FREEMAIN to free the virtual storage for the TAXE and Problem Program Save Area. Then STAX cancels the FRR and passes control to Exit. If the TAXE is active, STAX marks the TAXE for freeing by Exit, cancels the FRR, and passes control to Exit.		

Extended Description	Module	Label
4 If the cancel option is not chosen, STAX either creates a new TAXE, calling the Stage 1 Exit Effector, or replaces the values in the old TAXE with values from the STAX Parameter List.		
Then STAX initializes the TAXE fields, using the STAX Parameter List values as a source. Finally STAX enqueues the TAXE, enqueuing it at the lowest possible attention level on the TAXE queue but higher than any of the TAXES for subtasks under that TCB.		
5 After increasing by one the STAX count in the TSB (if no higher level TAXE is active), STAX cancels the FRR and passes control to the SVC EXIT routine (IEAVEEXP).		
Error Recovery		
If an error occurs in STAX's locked code, R/TM passes control to the FRR. The FRR resets the status bits to their settings before the request (for defer status, bits are set according to the request), resets the TAXE queue, updates the STAX count in the TSB, records error information, and passes control back to R/TM to continue with termination processing. If the error did not occur in this address space, the FRR returns control to R/TM to continue termination without error recording.	IEAVAX00	STXFRR

Diagram 3-7. Attention Exit Prolog Routine (IEAVAR05) (Part 1 of 2)

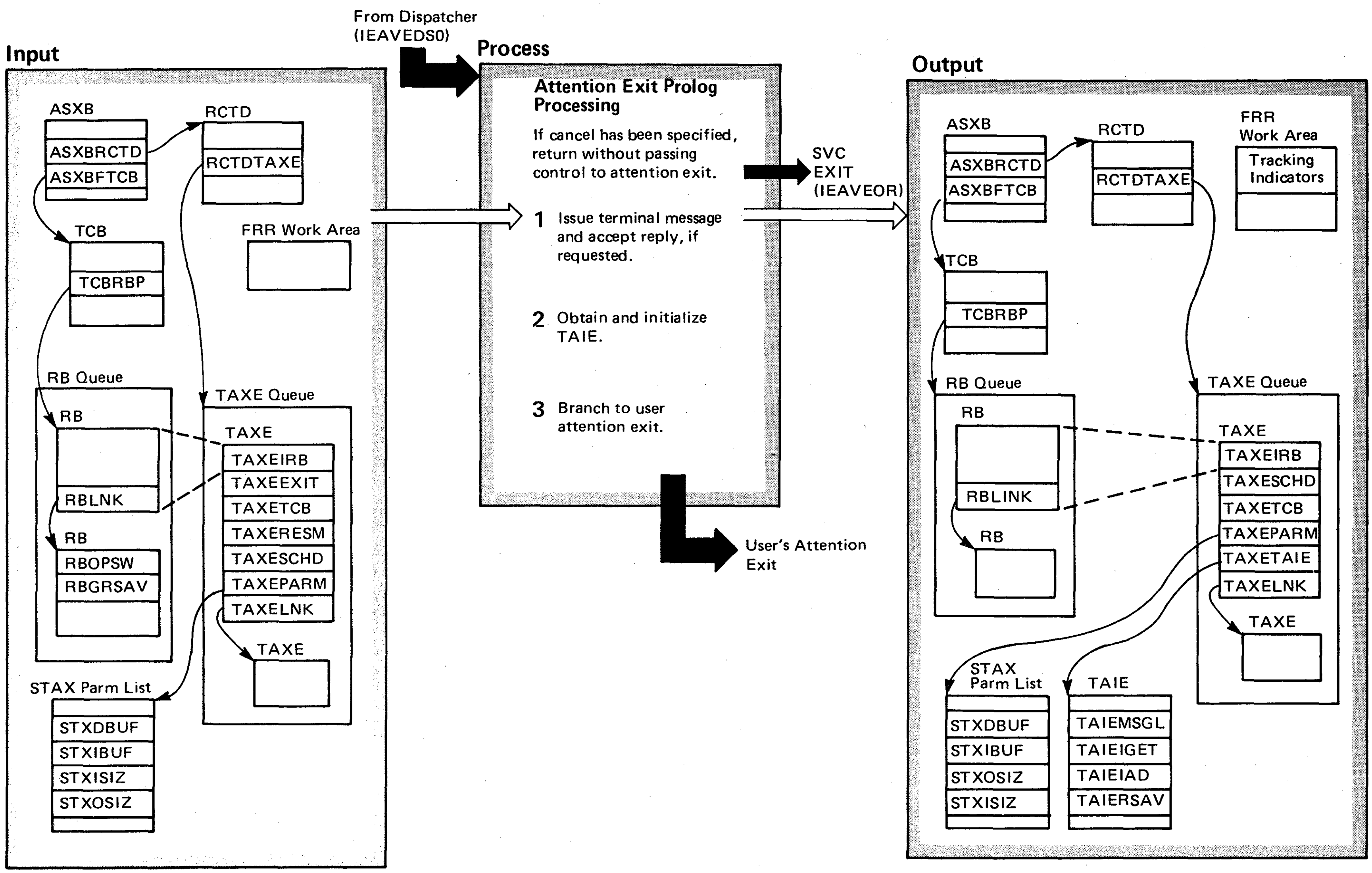


Diagram 3-7. Attention Exit Prolog Routine (IEAVAR05) (Part 2 of 2)

Extended Description	Module	Label
The Attention Exit Prolog (IEAVAR05) and Epilog (IEAVAR06) routines handle Terminal Attention Interrupt Element (TAIE) creation and housekeeping for the user attention exit routine. On entry to the Epilog routine, the local lock is held.		
1 Prolog issues a MODESET to take itself out of key 0 state and then, if specified, issues a TPUT and a TGET to issue a message to the terminal and accept the reply. After the TPUT/TGET processing, Prolog issues a second MODESET to reenter key 0 state.	IEAVAR05	IEAVAR05
2 Prolog gets the local lock and sets up the FRR. Then it calls GETMAIN to get space for the TAIE from user storage (subpool 250). If the GETMAIN fails, Prolog cancels the FRR, releases the local lock, issues a TPUT with an error message for the terminal, and returns control to Exit. If the GETMAIN is sucessful, Prolog initializes fields in the TAIE.		
3 If cancel has not been specified, Prolog cancels the FRR, releases the local lock, and issues MODESET so that the user's attention exit receives control in the proper key and state. Then Prolog branches to the user's attention exit.		

Diagram 3-8. Attention Epilog Routine (IEAVAR06) (Part 1 of 2)

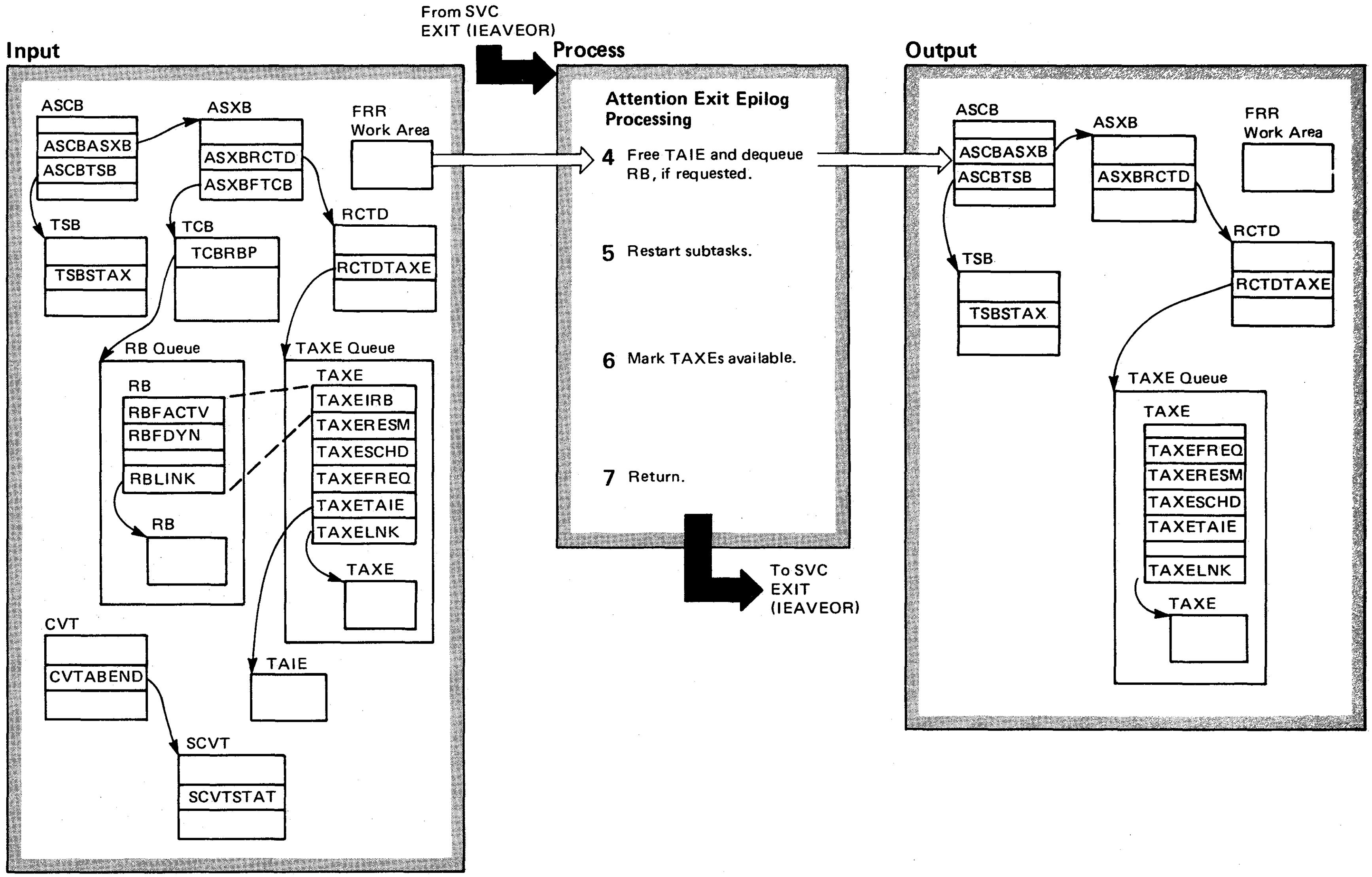


Diagram 3-8. Attention Epilog Routine (IEAVAR06) (Part 2 of 2)

Extended Description	Module	Label
4 Epilog gets control from Exit after the user's attention exit has completed. Epilog first sets up the FRR and resets flags in the TAXE to indicate that the TAXE is no longer scheduled. Then Epilog checks for a TAIE and, if one exists, uses FREEMAIN to free it. Epilog checks to see if the RB is to be freed. If it is (RBFDYN set to one), Epilog dequeues the TAXE.	IEAVAR06	IEAVAR06
5 Epilog invokes Status to restart the subtasks of the TCB that has the completing attention exit.		
6 Epilog searches the TAXE queue for the next lower attention level scheduled. If an active TAXE is found, indicated by the RBFACTV or TAXESCHD flag set to one, Epilog marks the TAXEs between the exiting TAXE and the active TAXE as available by resetting the TAXEFREQ flag to zero. Then Epilog increases the STAX count in the TSB by the number of TAXEs marked available.		
7 Epilog cancels the FRR and passes control to the Exit routine.		

Extended Description	Module	Label
Error Processing		
When an error occurs in Prolog- or Epilog-locked code, R/TM passes control to the Prolog or Epilog FRR.		
The Prolog FRR checks to see if the error occurred in the address space in which Prolog was running.	IEAVAR05	IEAVAFR5
If so, the FRR records in the SDWA and returns to R/TM requesting recording and retry. R/TM then reenters Prolog to cancel the FRR, release the local lock, and issue an error message via TPUT. If the error is not in the same address space, the FRR returns control to R/TM to continue with termination.	IEAVAR05	IEAVART5
The Epilog FRR checks to see whether the error occurred in Epilog's address space. If it did, the FRR dequeues the TAXE, cleans up the TAXE queue, updates the STAX count in the TSB, and, if necessary, restarts subtasks under the TCB with the completing attention exit. Then the FRR passes control to R/TM to record the error and to continue with termination. If the error was not in this address space, the FRR issues SETRP to control return to R/TM to continue with termination without recording.	IEAVAR06	IEAVAFR6

Diagram 3-9. Attention Exit Purge Routine (IEAVAR07) (Part 1 of 2)

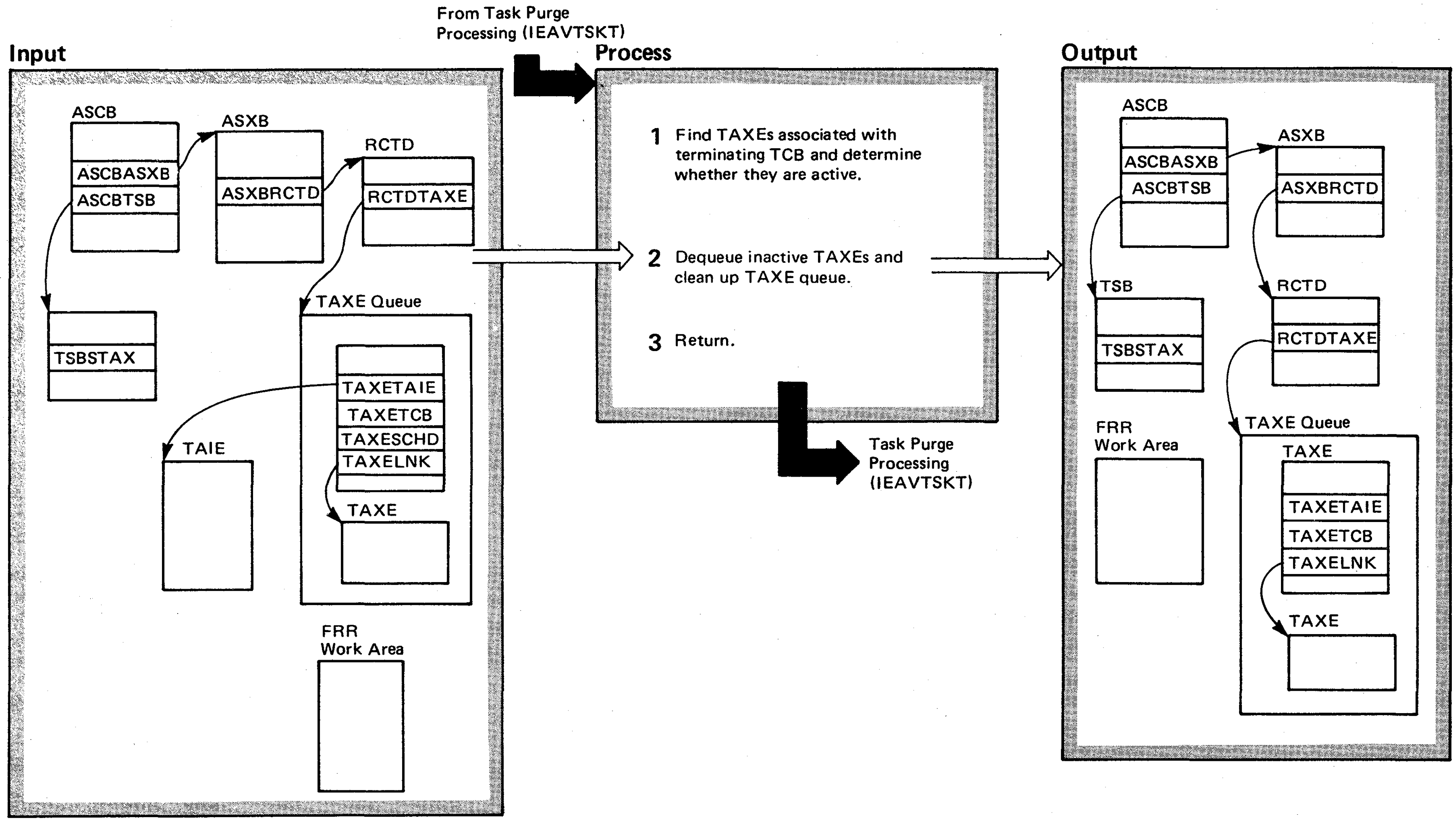


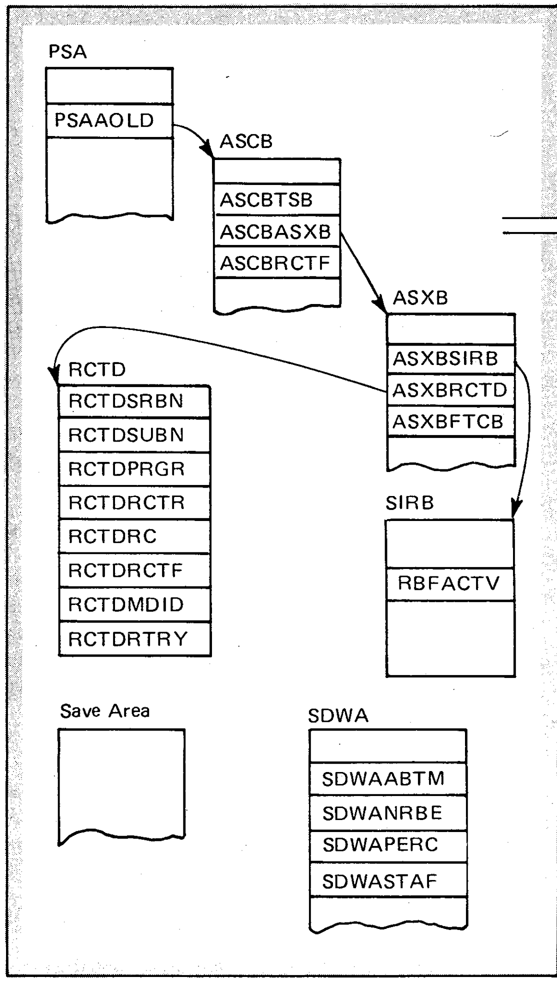
Diagram 3-9. Attention Exit Purge Routine (IEAVAR07) (Part 2 of 2)

Extended Description	Module	Label
Attention Exit Purge routine (IEAVAR07) is called by Task Purge Processing (IEAVTSKT) to eliminate any TAXEs belonging to the TCB being terminated.		
1 Purge gets the local lock and sets up the FRR. Then it finds TAXEs associated with the terminating task by checking the TAXETCB fields. Purge checks the RBFACTV flag in the TAXE to determine if the TAXE is active; if it is, Purge sets the RBFDYN flag to ensure that Exit will free the RB.	IEAVAR07	IEAVAR07
2 Purge dequeues the TAXE by moving the TAXELNK field value of the terminating TAXEs to the TAXELNK field of the previous TAXE on the queue. Then Purge marks any TAXEs on the TAXE queue between the highest active attention level (the lowest element on the queue) and the end of the queue available and increases the STAX count in the TSB by the number of available TAXEs.		
3 Purge cancels the FRR, releases the local lock, and returns control to Task Purge Processing (IEAVTSKT).		
Error Processing		
If an error occurs in Purge's locked code, R/TM passes control to the Purge FRR. The FRR checks to see if the error is in Purge's address space. If it is, the FRR clears the TAXE queue and the STAX and attention level counts in the TSB. Then the FRR records in the SDWA and passes control to R/TM, via the SETRP macro instruction, to record the error and to continue with termination. If the error is not in Purge's address space, the FRR returns control to R/TM to continue with termination.	IEAVAR07	IEAVARF7

Diagram 3-10. RCT ESTAE Processing (IEAVAR00) (Part 1 of 2)

From Recover Task Processing (IEAVTAS1)

Input



Process

1 If SIRB error, request address space termination.

2 Restart SRBs stopped by Quiesce.

3 Record error information in SDWA.

4 Check for abnormal termination for RCT. If abnormal termination, go to step 7.

5 Diagnose error and determine whether to retry.

6 If retry, check for recursion.

7 If recursion or abnormal termination, dump LSQA.

8 Clean up resources.

9 Return.

To Recover Task Processing (IEAVTAS1)

Output

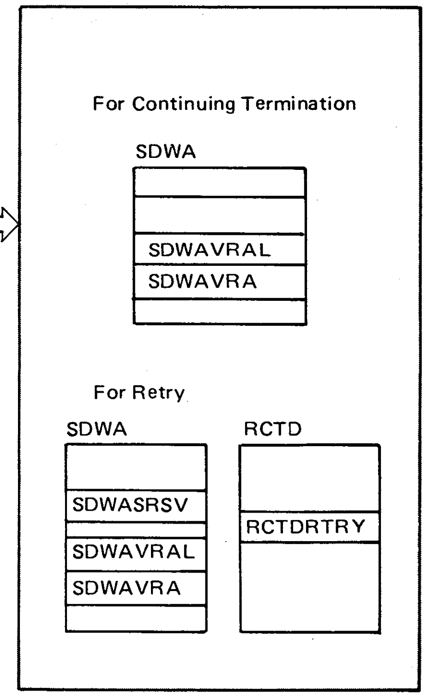


Diagram 3-10. RCT ESTAE Processing (IEAVAR00) (Part 2 of 2)

Extended Description	Module	Label
RCT ESTAE Processing is performed in all RCT modules when an error occurs in unlocked RCT code.		
1 If the local SIRB is active, RCT ESTAE issues CALLRTM for address space termination and then returns to R/TM to continue with termination.	IEAVAR00	
2 If the local SIRB is not active, RCT ESTAE invokes Status to start all "quiescable" SRBs that were stopped during Quiesce processing.		
3 If an SDWA exists, RCT ESTAE moves error information from the RCTD and ASCB to the SDWA. Otherwise, RCT ESTAE ensures that all subtasks are dispatchable and that RCT is not enqueued on the Purge Resource. Then it returns to R/TM to continue with termination.		
4 If RCT is being abnormally terminated, go to step 7.		

Extended Description	Module	Label
5 If RCT is not being abnormally terminated, RCT ESTAE calls a diagnostic subroutine for the function in error. The diagnostic subroutine checks the type of error, determines whether or not to retry the failing routine, and cleans up resources. It may also dump LSQA storage or issue an error message for terminal users.		
6 If the diagnostic subroutine indicates retry, RCT ESTAE checks for a possible recursion, a second entry into the ESTAE from the same routine. If no recursion is indicated, RCT ESTAE returns control to R/TM for recording and retry.		
7 If recursion or abnormal termination of RCT has occurred, RCT ESTAE issues an SDUMP macro instruction to dump LSQA storage to the SYS1.DUMP data set.		
8 RCT ESTAE performs the same resource cleanup functions as those described in step 3.		
9 RCT ESTAE issues a SETRP macro instruction to pass control to R/TM for recording and to continue with termination.		

The started task control (STC) routines oversee the processing of START, MOUNT, and LOGON commands. Started task control uses the initiator/terminator as a subroutine to complete command processing; the initiator actually takes the command task through execution and termination.

These are the major functions of STC:

- To obtain the region in which STC will run.
- To determine which of the three commands has been specified.
- To build the internal JCL text for the command task.
- To build the control blocks required for initiator processing.
- To free those control blocks after the initiator/terminator has terminated the command task.

STC gets a region for both itself and the initiator subroutine.

STC next determines what command has been specified and invokes the appropriate STC for the LOGON routine to check the command and its parameters for correct syntax. Started task control uses the command and its parameters to build internal JCL text for the task. This is done to enable the initiator to process the task as though it were any job identified by JOB, EXECUTE, and DD statements. The STC builds the control blocks required to invoke the initiator, the IEL, the SSIB, and the SSOB. STC writes the newly created JCL text into an appropriate subsystem data set. It also creates a SWA structure for the task that includes some skeletal scheduler control blocks: JSCB, JCT, SCT, and ACT. It then ends preparations by initializing the initiator entrance list (IEL) and invoking the initiator.

Once the initiator/terminator has completed processing, control returns to the STC routines. STC simply deletes the SWA structure it previously created and frees the CSCB that identified the command. At that point, started task control is finished so it returns to its caller, RCT.

Diagram 4-1. Started Task Control Processing (Part 1 of 8)

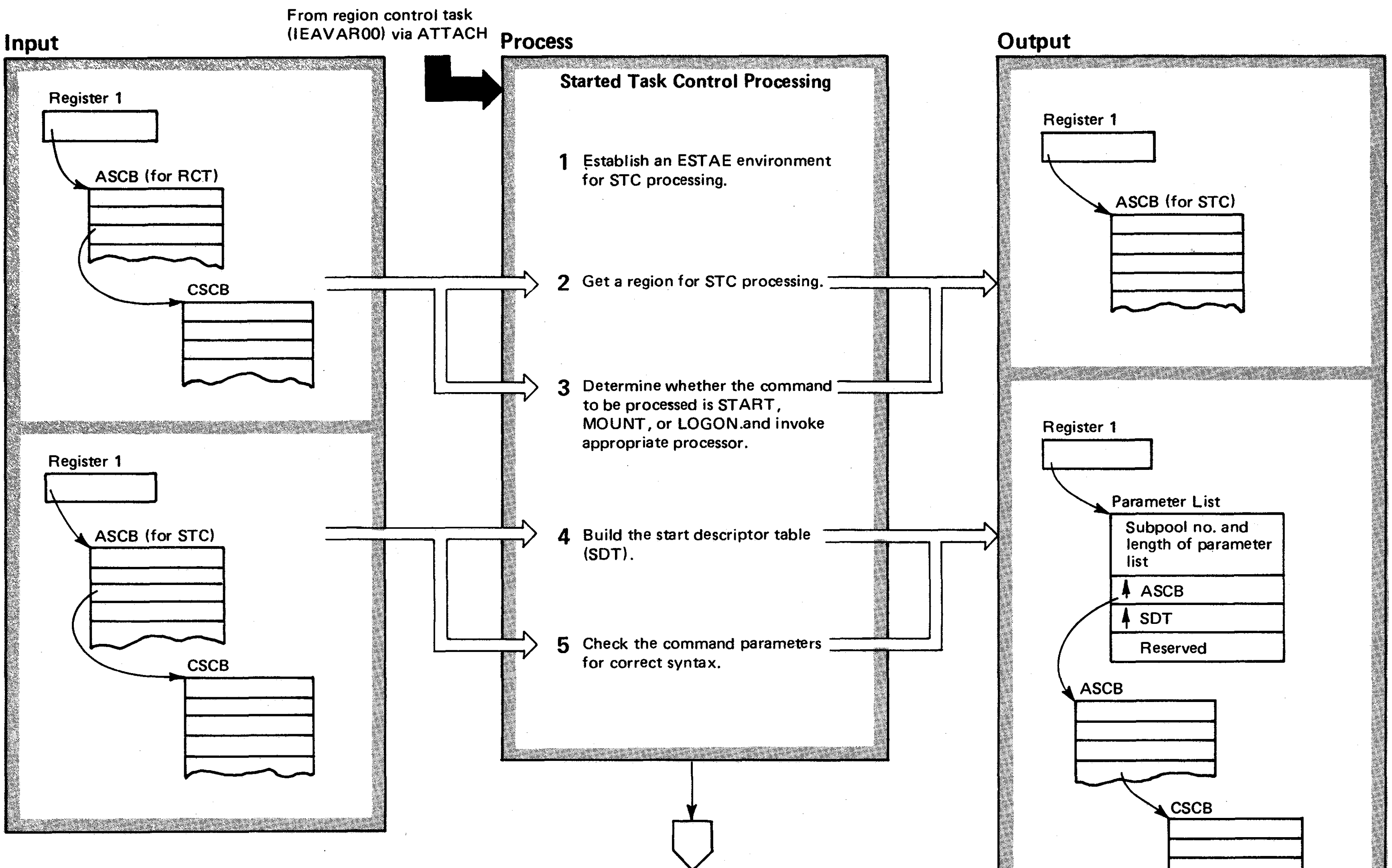


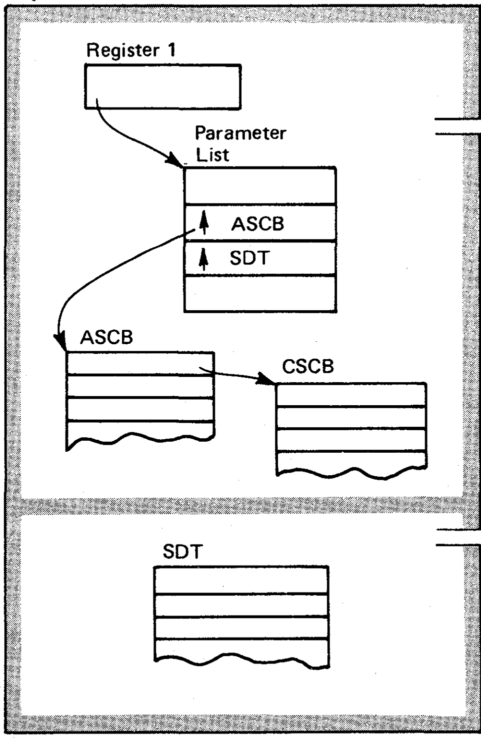
Diagram 4-1. Started Task Control Processing (Part 2 of 8)

Extended Description	Module	Label
The Started Task Control (STC) routines oversee the processing of START, MOUNT, and LOGON commands.		
1 The first STC module that receives control is IEEPRWI2. This routine establishes an ESTAE environment for STC by creating an ESTAE parameter list and loading module IEESB665, the STC recovery exit routine.	IEEPRWI2 IEESB665	
The ESTAE environment ensures that the recovery/termination management(R/TM) routines will receive control in the event of an STC error. R/TM will, in turn, invoke IEESB665, which will then attempt to recover.		
If R/TM provides an SDWA (system diagnostic work area) containing the necessary data, IEESB665 will schedule a retry by terminating the current STC processing and issuing a SETRP macro instruction.		
If no SDWA exists, IEESB665 will simply continue ABEND processing.		
In either case, the recovery routine will record the error in the SYS1.LOGREC data set and then return to R/TM.		
2 Once the ESTAE environment is complete, IEEPRWI2 issues a GETMAIN macro instruction to obtain its own region from subpool 247.	IEEPRWI2	

Extended Description	Module	Label
3 IEEPRWI2 checks an indicator in the CSCB to determine which command, START, MOUNT, or LOGON, is to be processed.		
If the command is START, IEEPRWI2 issues an XCTL macro instruction to IEEVSTAR.	IEEVSTAR	
For a MOUNT command, the XCTL macro instruction invokes IEEVMNT1, and for a LOGON, IKJEFLA.	IEEVMNT1 IKJEFLA	
Both IEEVSTAR and IEEVMNT1 are part of STC; IKJEFLA is part of LOGON processing.	IEEVSTAR and IEEVMNT1	
4 Both routines, IEEVSTAR and IEEVNMT1 begin processing by creating a start descriptor table (SDT) and initializing it with blanks and zeroes.		
5 IEEVSTAR and IEEVMNT1 check the commands and associated parameters for correct syntax. When either routine finds an error, it places the command name in an extended save area and invokes module IEE0503D to write a message using that name.		
IEEVSTAR and IEEVMNT1 continue error processing by freeing the SDT and issuing an XCTL macro instruction to the last routine of STC, IEEPRTN2. This module cleans up the data areas and ends the task that was begun for a START or MOUNT command.		

Diagram 4-1. Started Task Control Processing (Part 3 of 8)

Input



Process

6 Build JCL for the command and put it into the SDT.

7 Build a command input buffer (CIB).

8 Update appropriate control blocks.

9 Build the job scheduling tables:
Job scheduling entrance list (JSEL).
Jos scheduling options list (JSOL).
Job scheduling exit list (JSXL).

Output

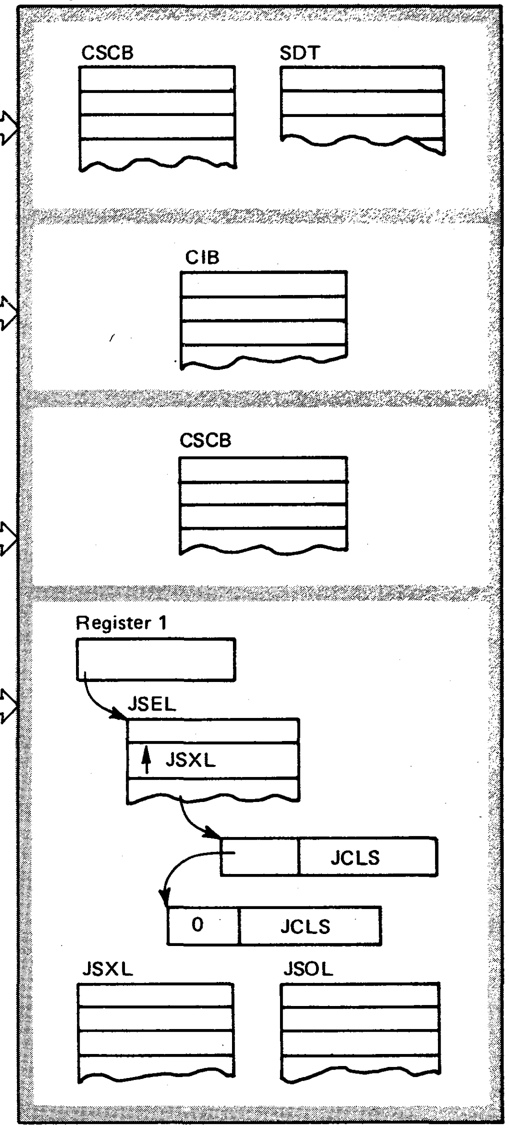


Diagram 4-1. Started Task Control Processing (Part 4 of 8)

Extended Description	Module	Label
6 Every time a MOUNT command is specified, JCL that identifies the device to be mounted is supplied by the user. When a START command is specified, the JCL must be created for it.		
IEEVSTAR uses the START command parameter to build internal JCL statements.	IEEVSTAR	
The procedure name that was specified with the START command becomes the JCL jobname. The name is placed in the CSCB and a pointer to it in the ASCB.		
When an ID was included on the START command, it becomes the stepname for an EXEC card. It too is saved in the CSCB. If no ID was specified, the stepname used is "STARTING".		
If a unit parameter and volume serial number were entered with START, they are used for a DD statement.		
As each JCL statement is generated, it is moved into the SDT.		
7 Both IEEVSTAR and IEEVMNT1 create a command input buffer using storage from subpool 245.	IEEVSTAR and IEEVMNT1	

Extended Description	Module	Label
8 IEEVMNT1 stores JCL related information, which was provided with the MOUNT command, in the SDT.	IEEVMNT1	
IEEVMNT1 and IEEVSTAR both update the CSCB with information related to the command parameters. Then, before passing control to IEEVJCL, they build a parameter list for it.	IEEVMNT1 and IEEVSTAR	
9 IEEVJCL is the JCL build routine. For each newly created JCL statement, IEEVJCL gets an 88-byte area of storage space called a JCLS. It moves each statement into a JCLS (job control language string) and chains each JCLS to another.	IEEVJCL	
After the JCLS chain is done, IEEVJCL issues a GETMAIN macro instruction for space for the JSXL. It places these pointers in the JSEL: • A pointer to the JSXL. • A pointer to the JCLS chain. • A pointer to the CSCB. • A pointer to the ASCB.		
IEEVJCL issues another GETMAIN macro instruction, this time for the JSOL, which is initialized with the command's jobname, EXEC name, and procedure name.		
IEEVJCL completes processing by freeing the SDT and invoking the job scheduling subroutine, IEESB605.		

Diagram 4-1. Started Task Control Processing (Part 5 of 8)

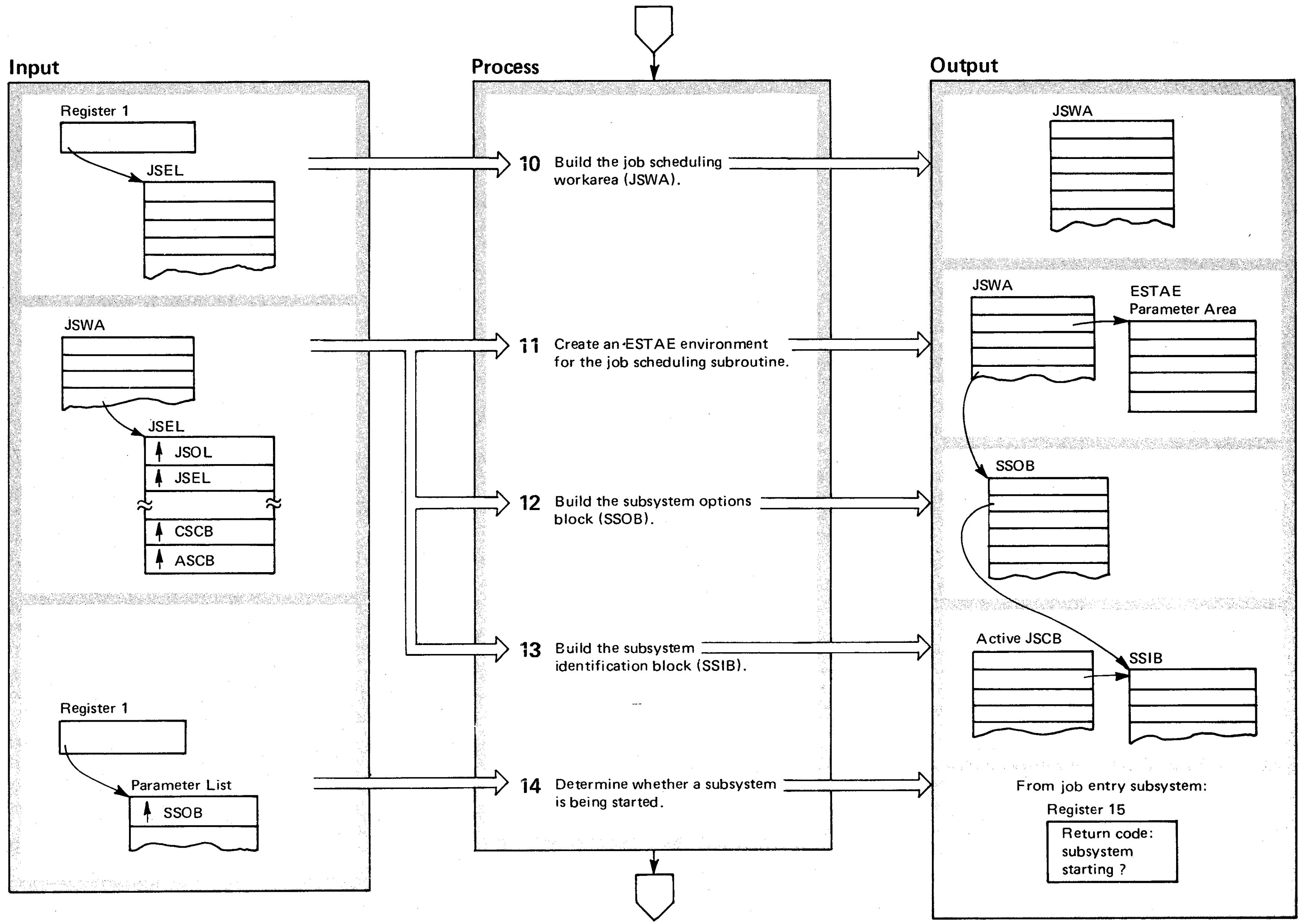


Diagram 4-1. Started Task Control Processing (Part 6 of 8)

Extended Description	Module	Label
10 IEESB605 creates the environment needed for initiator processing. It begins by obtaining storage space for its own work area, called the job scheduling work area (JSWA).	IEESB605	
11 IEESB605 creates its own ESTAE environment. It builds an ESTAE parameter list and issues the ESTAE macro instruction, then loads IEESB670, its own recovery exit routine. Then, if an error occurs in IEESB605, the R/TM routines will receive control and after preliminary processing, pass control on to IEESB670.	IEESB605	
If R/TM provides an SDWA containing the necessary data, IEESB670 will schedule a retry of IEESB605 that terminates current STC processing by issuing a SETRP macro instruction.	IEESB670	
If no SDWA exists, IEESB670 will continue ABEND processing.		
In either case, the recovery routine will record the error in the SYS1.LOGREC data set and return to R/TM.		
12 IEESB605 builds an SSOB to represent the command as a job. It places, in the JSWA, an SSOB pointer and an indicator that the SSOB exists.	IEESB605	
13 IEESB605 also builds an SSIB for the command and places pointers to it in the current JSCB and the SSOB.		
14 IEESB605 determines whether a subsystem is being started by issuing the IEFSSREQ macro instruction to invoke the master subsystem. When the master subsystem returns control, IEESB605 checks the return code in register 15.		
If STC is starting a subsystem, IEESB605 places a pointer to the JCLS chain in the SSIB. This will allow the master subsystem access to the JCLS.		

Diagram 4-1. Started Task Control Processing (Part 7 of 8)

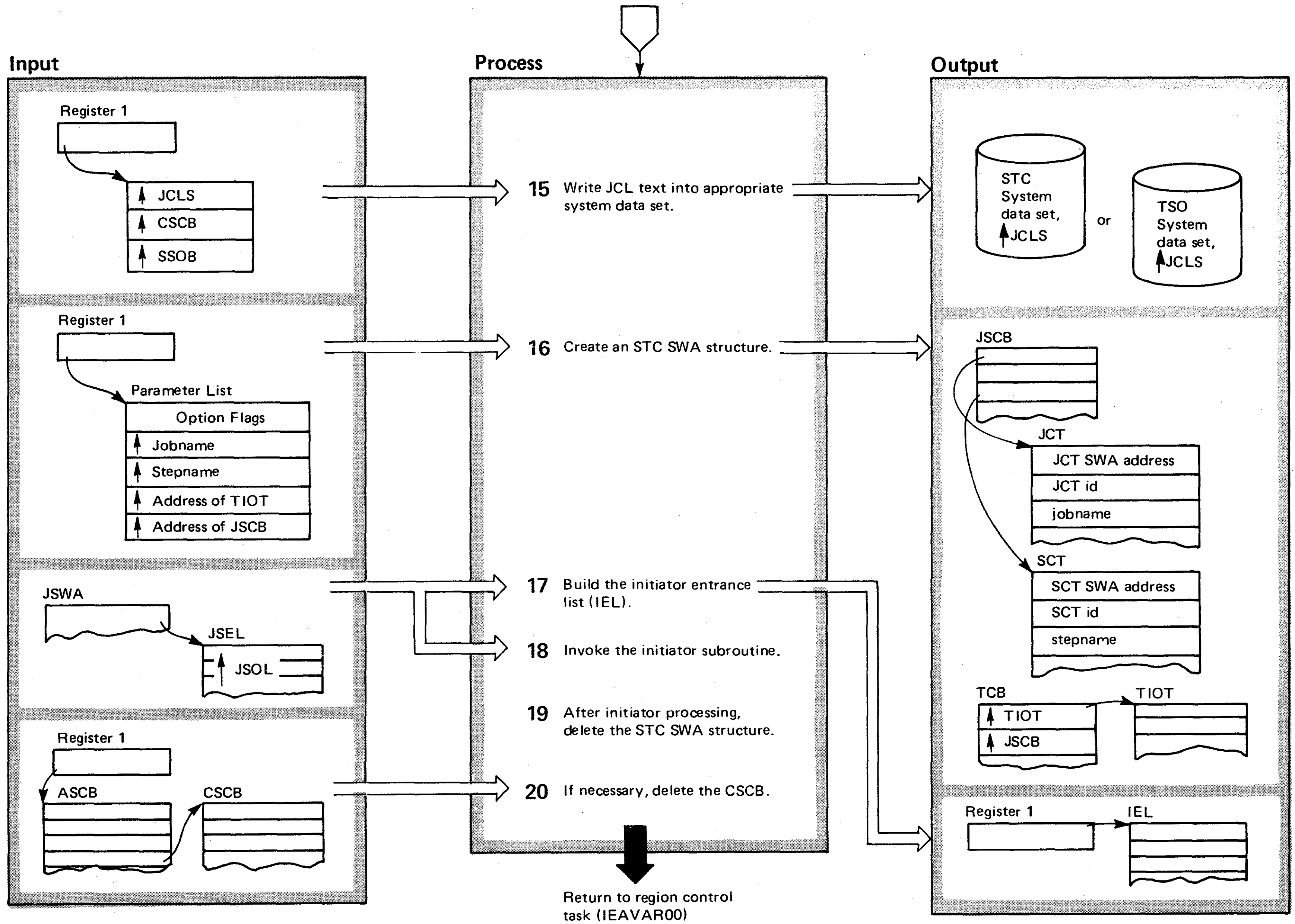


Diagram 4-1. Started Task Control Processing (Part 8 of 8)

Extended Description	Module	Label
15 When STC is not starting a subsystem, IEESB605 calls IEFJSWT, STC's internal JCL write routine. IEFJSWT first initializes a request parameter list (RPL) and an access control block (ACB), the control blocks associated with the system data set into which the JCLS will be written.	IEFJSWT	
IEFJSWT checks an indicator in the CSCB to determine which command is in progress. For a LOGON, the system data set used in TSOINRDR; for the other commands, it is STCINRDR. For every command, IEFJSWT opens the appropriate data set and writes each JCLS record into it. When the writing is done, IEFJSWT returns to IEESB605.		
16 IEESB605 clears all the existing JCLS pointers and then calls the STC SWA initialization routine, IEESB601. This initialization routine builds a skeletal SWA structure in preparation for initiator processing. The SWA structure includes these control blocks: • JSCB. • QMPA (queue manager parameter area). • JCT. • ACT • SCT. • ACT. • TIOT.	IEESB605 IEESB601	
17 Once the SWA structure is complete, IEESB601 returns control to IEESB605, which initializes an initiator entrance list (IEL) with the following information: • Options from the JSOL. • A pointer to the JSEL. • A pointer to the initiator option list. • A pointer to the initiator exit list.	IEESB605	
18 IEESB605 clears pointers to the JSOL, which is no longer needed, and then issues a LINK macro instruction invoking an initiator subroutine, IEFSD060. From that routine, initiator processing proceeds normally until the command task has been executed and is in termination. At that point, IEESB605 again receives control. (During initiator processing of a MOUNT command, the initiator ATTACH routine, IEFSD263, attaches IEEVMNT2, the MOUNT command processor. IEEVMNT2 returns control to IEFSD263.)	IEFSD060	
19 IEESB605 performs STC clean-up functions by freeing the IEL, the JSWA, and the SSIB and SSOB. It invokes IEESB601 once again, this time to delete the SWA structure it previously created. When control returns to IEESB605, it issues an XCTL macro instruction to IEEPRTN2.	IEESB605	
20 The STC free region routine, IEEPRTN2, does not free storage space but simply checks for the existence of CSCB in the ASCB and frees it if it still exists. IEEPRTN2 returns to region control task.	IEEPRTN2	

Started Task Control (STC), passes control to LOGON Initialization, IKJEFLA. Here, the various control blocks required for LOGON and the terminal session are initialized, the ESTAE recovery routine, IKJEFLS, is established, Master Scheduler JCL, MSTRJCL, is searched to ensure that SYSLBC, System Braodcast Dataset, and SYSUADS, System User Attribute Dataset, are available to LOGON and the subsequent terminal ession, and then LOGON Scheduler, IKJEFLB, is called.

IKJEFLB receives control from IKJEFLA during a LOGON, and receives control from the Job Scheduling Subroutine, JSS, during a re-LOGON and a LOGOFF. IKJEFLB invokes the LOGON Prompting Monitor, IKJEFLC, and then waits for notification to either continue with the LOGON by passing control to JSS, or in the case of a LOGOFF, IKJEFLB will terminate and pass control back to STC.

IKJEFLC passes control to the LOGOFF processor, IKJEFLL, in the case of a LOGOFF or a re-LOGON. Then IKJEFLC passes control to LOGON Verification, IKJEFLE, who parses the command to obtain the LOGON data and verify this data against the UADS, User Attribute Dataset. In the case of a LOGOFF, IKJEFLC, notifies IKJEFLB that LOGON should terminate and then IKJEFLC terminates. For a LOGON or a re-LOGON, IKJEFLC notifies IKJEFLB that it should pass control to JSS and then IKJEFLC passes control to IKJEFLH, the routine that invokes LISTBC, List Broadcast Dataset.

IKJEFLB passes control to JSS for the LOGON or the re-LOGON, and JSS eventually passes control to the Pre-TMP Exit, IKJEFLJ. IKJEFLJ notifies IKJEFLH that once LISTBC has completed, IKJEFLH and then IKJEFLC should terminate. After IKJEFLJ terminates, the TMP is invoked for the users terminal session.

When a LOGON command, referred to as re-LOGON, or a LOGOFF command is entered, the TMP terminates. JSS then passes control to the Post-TMP Exit, IKJEFLK, for some housekeeping. After JSS has completed its work it passes control to IKJEFLB who inturn invokes IKJEFLC to handle the LOGOFF or the re-LOGON.

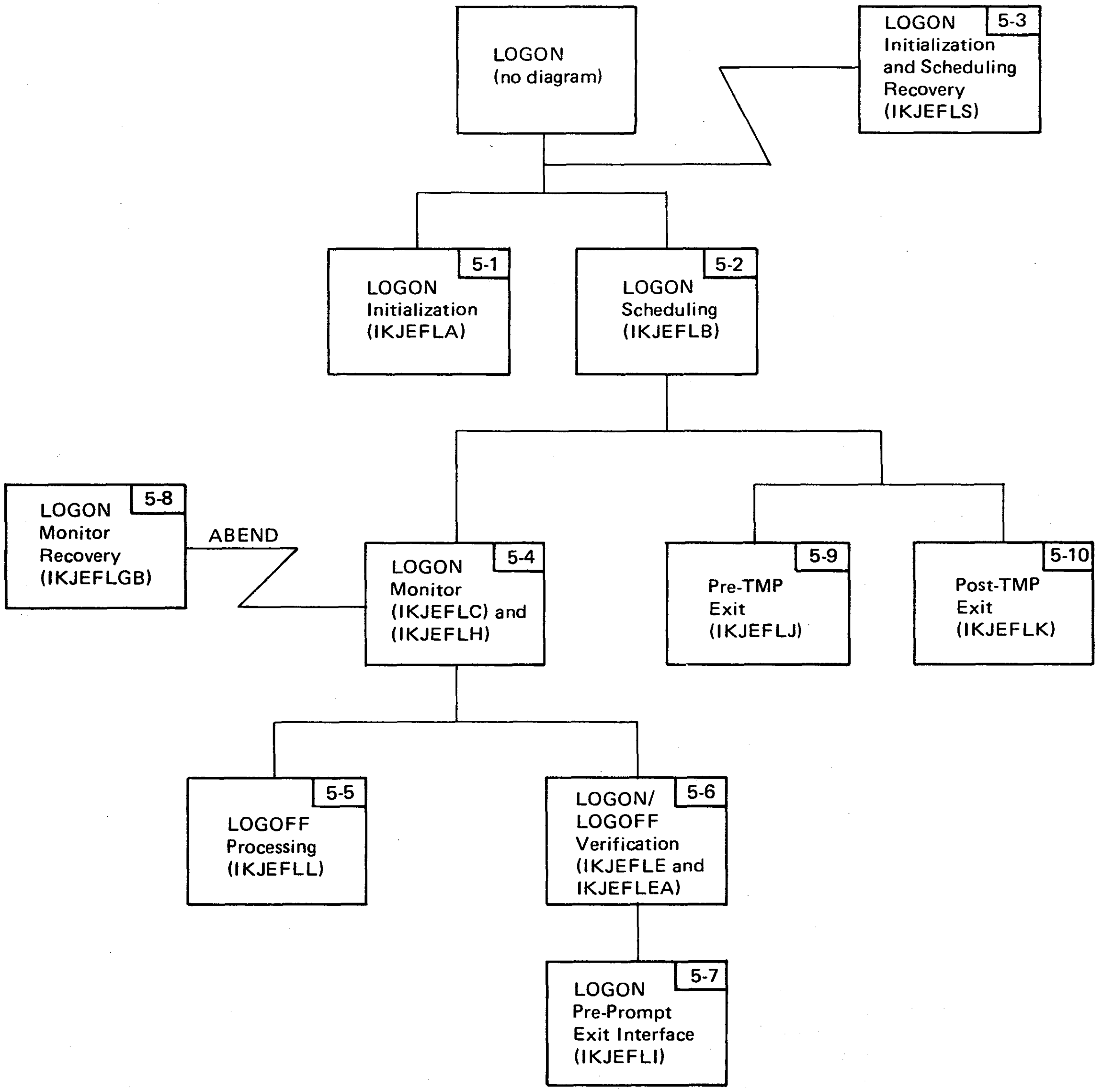


Figure 2-7. LOGON Scheduling Visual Contents

Diagram 5-1. LOGON Initialization (IKJEFLA) (Part 1 of 2)

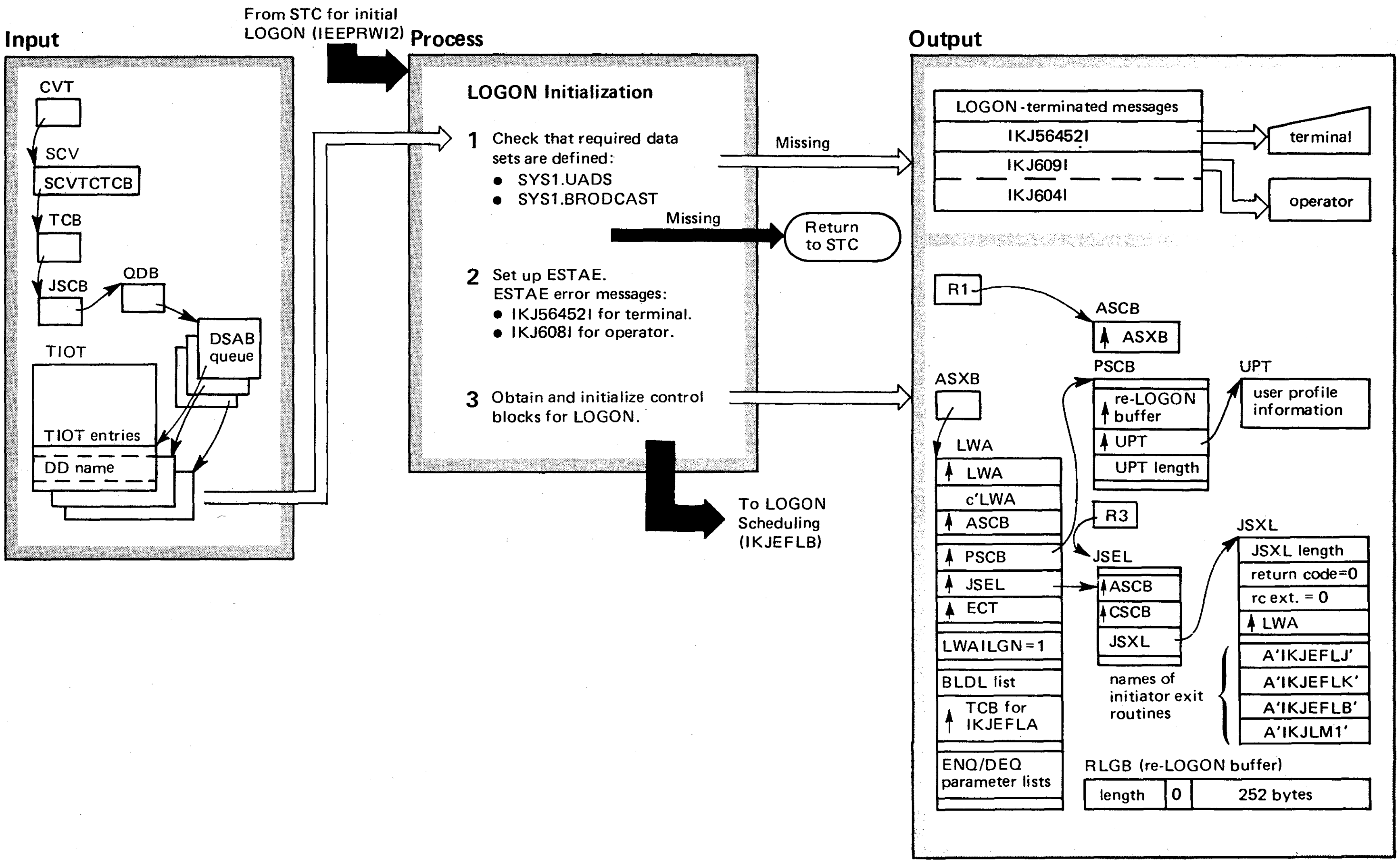


Diagram 5-1. LOGON Initialization (IKJEFLA) (Part 2 of 2)

Extended Description	Module	Label
LOGON initialization receives control from started task control (STC) to process an initial LOGON command from a terminal. The initialization functions are bypassed for a LOGOFF or reLOGON.	IKJEFLA	
1 Two TSO data sets—SYS1.UADS and SYS1.BRODCAST—must have been defined by master scheduler's JCL (MSTRJCL member of SYS1.LINKLIB). LOGON initialization checks for these data sets by searching the master scheduler's TIOT for the DD names SYSUADS and SYSLBC. If either of the names is missing, error messages are issued and LOGON is terminated.	IKJEFLA	
2 IKJEFLS is used as the ESTAE routine to protect IKJEFLA and IKJEFLB.		
3 LOGON initialization creates the control blocks that contain LOGON information needed by the various LOGON routines. (In the "Data Areas" section of this publication, is an overview chart showing the chaining and function of the LOGON control blocks. See Figure 5-5.) LOGON initialization turns on the initial-LOGON bit (LWAILGN) to indicate that this is the first LOGON command to be processed for the current address space.	IKJEFLA	

Diagram 5-2. LOGON Scheduling (IKJEFLB) (Part 1 of 2)

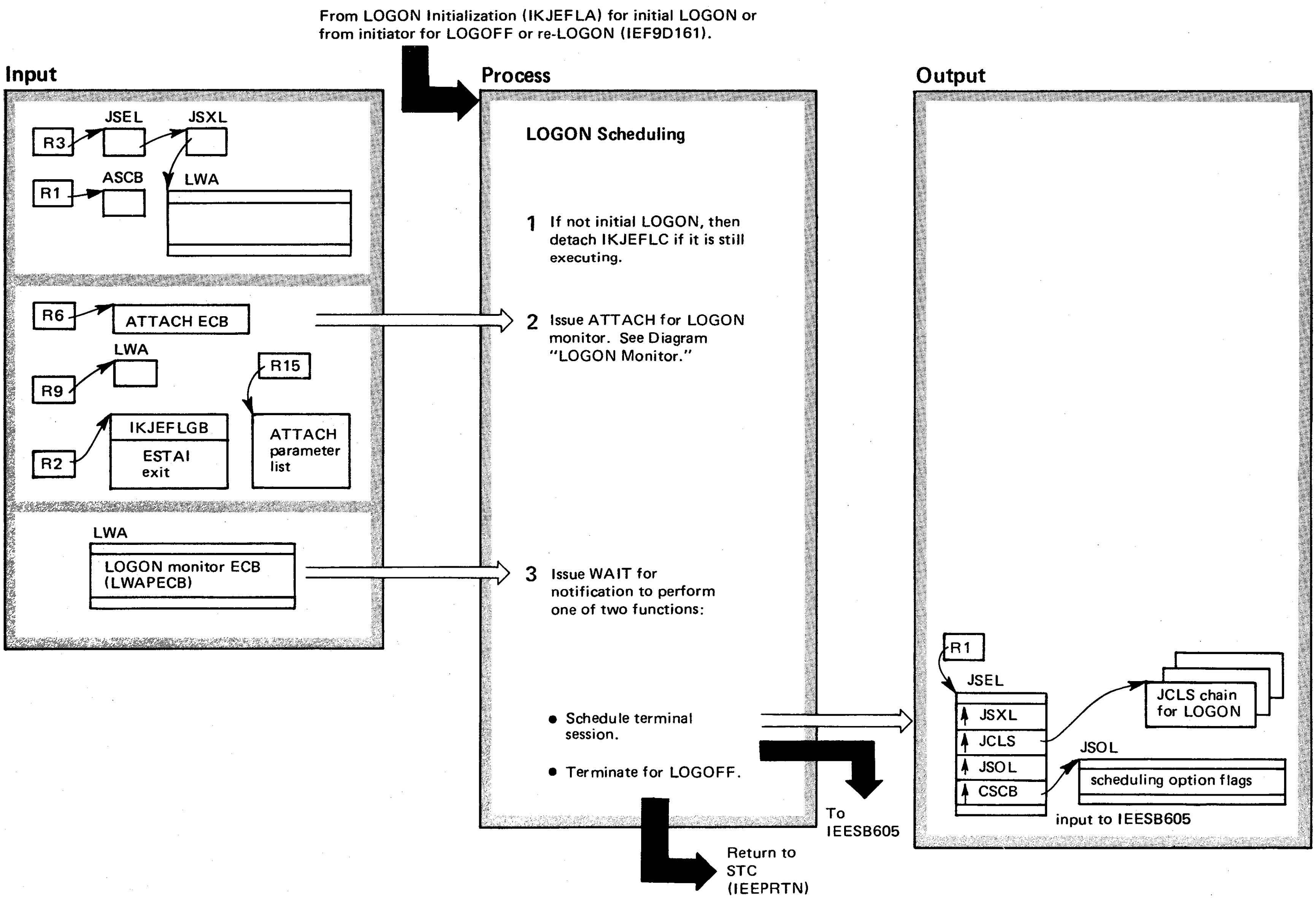


Diagram 5-2. LOGON Scheduling (IKJEFLB) (Part 2 of 2)

Extended Description	Module	Label
LOGON scheduling receives control from LOGON initialization or from the initiator at the end of the terminal session (for LOGOFF or re-LOGON). The new terminal session that is scheduled following a re-LOGON operates in the same address space as the initial terminal session.		
LOGON scheduling invokes the job scheduling subroutine. This subroutine interprets the JCL card images that define the terminal session and attaches the terminal monitor program (TMP), which processes commands from the terminal. The TMP remains active until it intercepts a LOGOFF or a re-LOGON command from the terminal. At that time, the TMP terminates and the initiator passes control back to LOGON scheduling to process the command.		
1 Upon receiving control from STC for a LOGOFF or re-LOGON, LOGON scheduling ensures that the LOGON monitor has already terminated. If the monitor is yet active, LOGON scheduling notifies the monitor (ILWASECB-post code 20) to terminate. Once the monitor has terminated (LWAPECB-post code 24), LOGON scheduling detaches it and sets the attach ECB (LWAAECB) to zero. LOGON scheduling then performs the attach of the LOGON monitor (Step 2) as usual.	IKJEFLB	
If the LOGON monitor posts LWAPECB with an invalid post code (other than 16 and 24), LOGON scheduling terminates as follows:	IKJEFLB	WAITUST
• Detaches the LOGON monitor.		
• Cancels the ESTAE environment.		BEXIT
• Places the address of the ASCB in register 1.		
• Returns to STC (IEEPRTN) for CSCB clean-up.		
But, if the LOGON monitor has caused an ABEND and recovery is to be attempted (LWABEND=1), LOGON scheduling does not terminate; it reissues the ATTACH of the LOGON monitor (returns to Step 2).		LCRESTRT

Extended Description	Module	Label
2 LOGON scheduling handles the initial LOGON, a LOGOFF, or a re-LOGON. First, it issues an ATTACH macro instruction to invoke the LOGON monitor (see Diagram "LOGON Monitor"). The monitor routine executes until it requires a function that LOGON scheduling performs. At that time, the monitor notifies LOGON scheduling via the LOGON monitor ECB (LWAPECB).	IKJEFLB	
3 When notified by the LOGON monitor, LOGON scheduling performs one of two functions; the function performed is determined by the post code located in the monitor's ECB: (LWAPECB).	IKJEFLB	WAITLIST

post code	function performed by LOGON scheduling	Module	Label
16	Schedules a terminal session as follows: • Notifies the LOGON monitor (LWASECB—post code 16) to invoke the LOGON information routine IKJEFLH. • Creates the job scheduling option list (JSOL) and chains it to the JSEL. The JSOL contains option flags that affect the scheduling of this terminal session. • Moves the JCL card image chain (created by either the LOGON monitor or the preprompt exit) from subpool 1 to subpool 253. • Invokes the initiator routine IEESB605 to schedule the terminal session.	IKJEFLB	
24	Terminates LOGON scheduling as follows (performed following a LOGOFF command): • Notifies the LOGON monitor to terminate (LWASECB—post code 24). • Issues a DETACH macro instruction for the LOGON monitor. • Cancels the ESTAE environment protecting LOGON scheduling. • Transfers control to STC routine IEEPRTN for CSCB clean-up.	IKJEFLB	ENDJOB

Diagram 5-3. LOGON Initialization and Scheduling Recovery Routine (IKJEFLS) (Part 1 of 2)

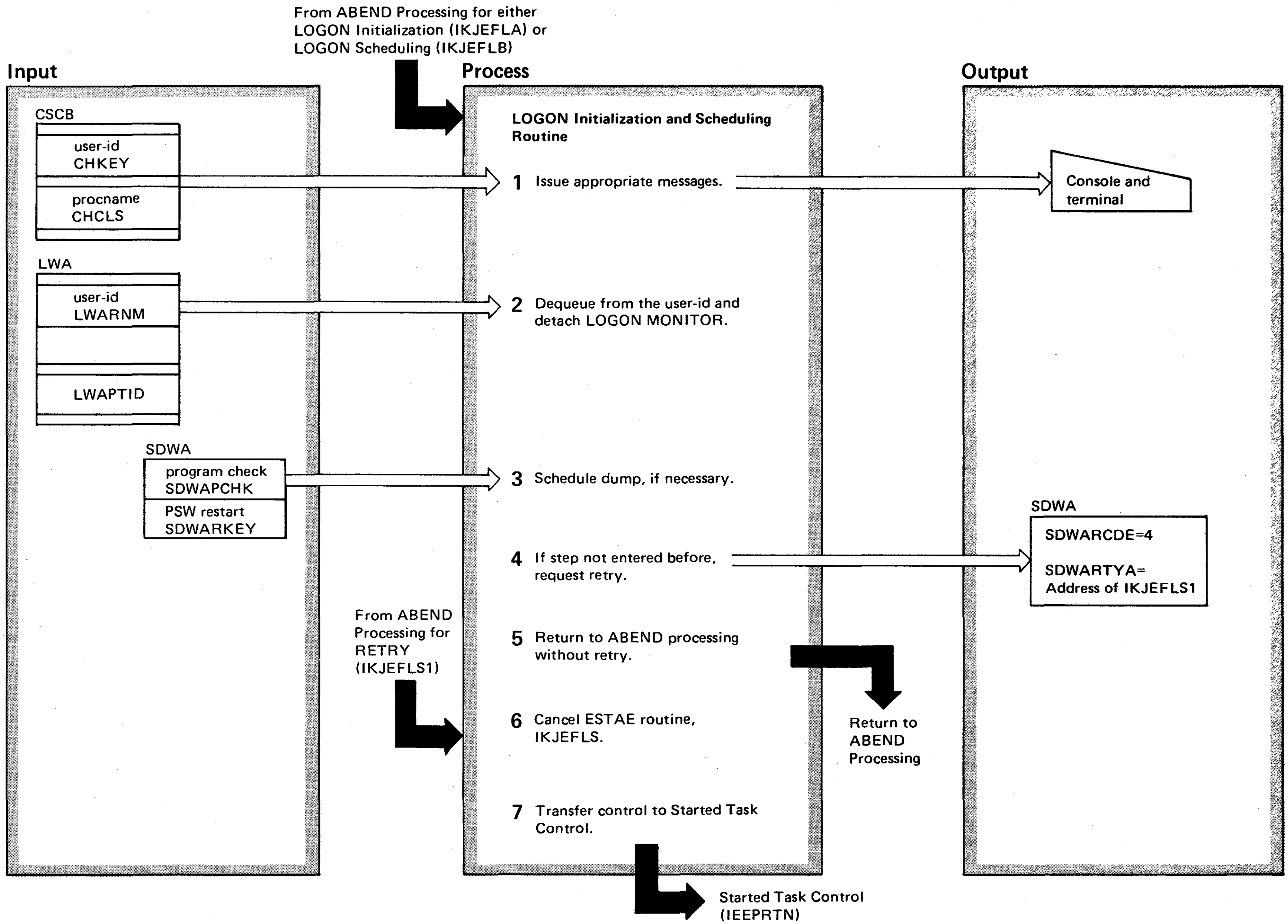


Diagram 5-3. LOGON Initialization and Scheduling Recovery Routine (IKJEFLS) (Part 2 of 2)

Extended Description	Module	Label
LOGON Initialization creates an ESTAE environment that handles abends that can occur during initialization and scheduling.	IKJEFLA	
1 Message IKJ601I is sent to the operator and message IKJ56452I is sent to the terminal.	IKJEFLS	
2 Dequeue from the user-id and detach the LOGON MONITOR. (The LWAPTID is the LOGON monitor TCB pointer.)		
3 Obtain a dump for a program check or PSW restart.		
4 If not a recursive abend, then indicate "RETRY" in the SDWA with the retry routine, IKJEFLS.		
5 Return to ABEND processing (IKJEFLS1) to possibly schedule a retry (see step 4).		
6 Cancel the ESTAE environment.	IKJEFLS1	
7 Transfer control to started task control, IEEPRTN, by using XCTL.		

Diagram 5-4. LOGON Monitor (IKJEFLC) (Part 1 of 4)

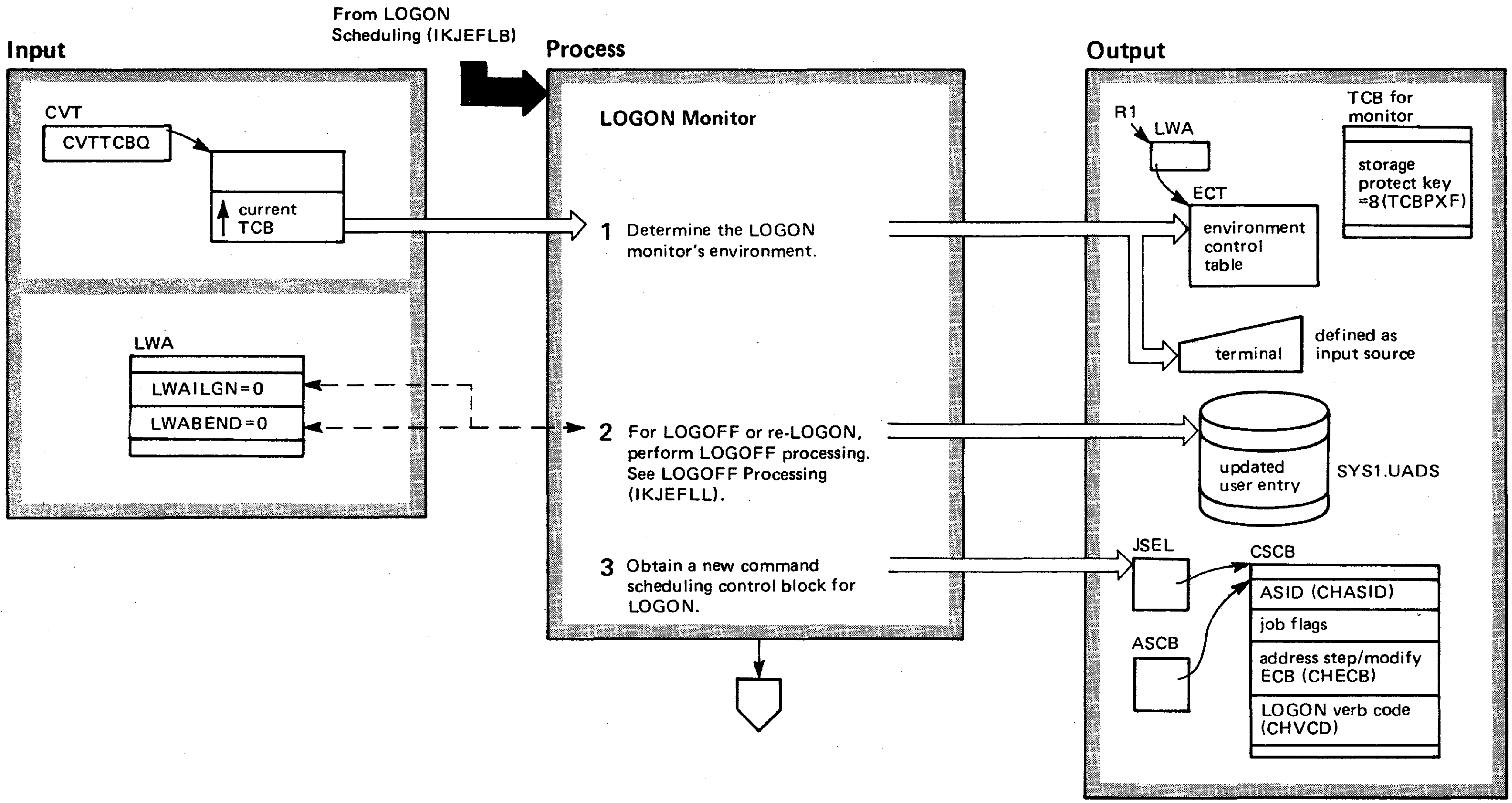


Diagram 5-4. LOGON Monitor (IKJEFLC) (Part 2 of 4)

Extended Description	Module	Label
The LOGON monitor controls the processing that verifies the LOGON or LOGOFF command, and the processing that issues informational and prompting messages to the terminal. It notifies LOGON scheduling to schedule a terminal session or, in the case of a LOGOFF, to terminate the LOGON scheduling task. Some of the informational messages (that is, mail, notices, and LOGON-proceeding messages) are issued in parallel with the scheduling of the terminal session. All LOGON monitor messages are issued by the message handler IKJEFLGM.	IKJEFLC	
1 The LOGON monitor creates the environment control table (ECT), which contains information about I/O service routines the monitor will use. Also, the monitor sets its own storage protection key to 8. This allows the storage obtained by the monitor to be referenced by programs not executing in privileged state (for example, LISTBC and the pre-prompt exit). Finally, the monitor issues a STACK macro instruction to define the terminal as the first source of input for time-sharing commands.	IKJEFLC	INITWKAR STACK
2 LOGOFF processing updates the terminal user's entry in SYS1.UADS and analyzes the return codes from the job scheduling subroutine and from the terminal session. LOGOFF processing is not performed for an initial LOGON (LWAILGN=1) or for recovery processing (LWABEND=1). For more detail, refer to Diagram "LOGOFF Processing."	IKJEFLL	
3 The LOGON monitor builds a new CSCB that contains the verb code for the LOGON command. This new CSCB replaces the one built for address space creation processing (START/LOGON/MOUNT) or, if this LOGON is a re-LOGON, replaces the CSCB previously created by the LOGON monitor. (It is important that LOGON establish a full size CSCB for all logons and re-logons before passing it to the initiator. The initiator, assuming the full size CSCB is passed, frees the second portion and uses only the first portion of the CSCB.)	IKJEFLC	CSCBINIT
4 The LOGON monitor issues a STAX macro instruction to establish a routine (IKJEFLG) that receives control when the terminal user causes an attention interruption by pressing the terminal's attention key. After causing the interruption, the terminal user may enter a question mark (?) to request second-level messages or may enter a new LOGON command to replace the one currently being processed.	IKJEFLC	TERMINAL

Diagram 5-4. LOGON Monitor (IKJEFLC) (Part 3 of 4)

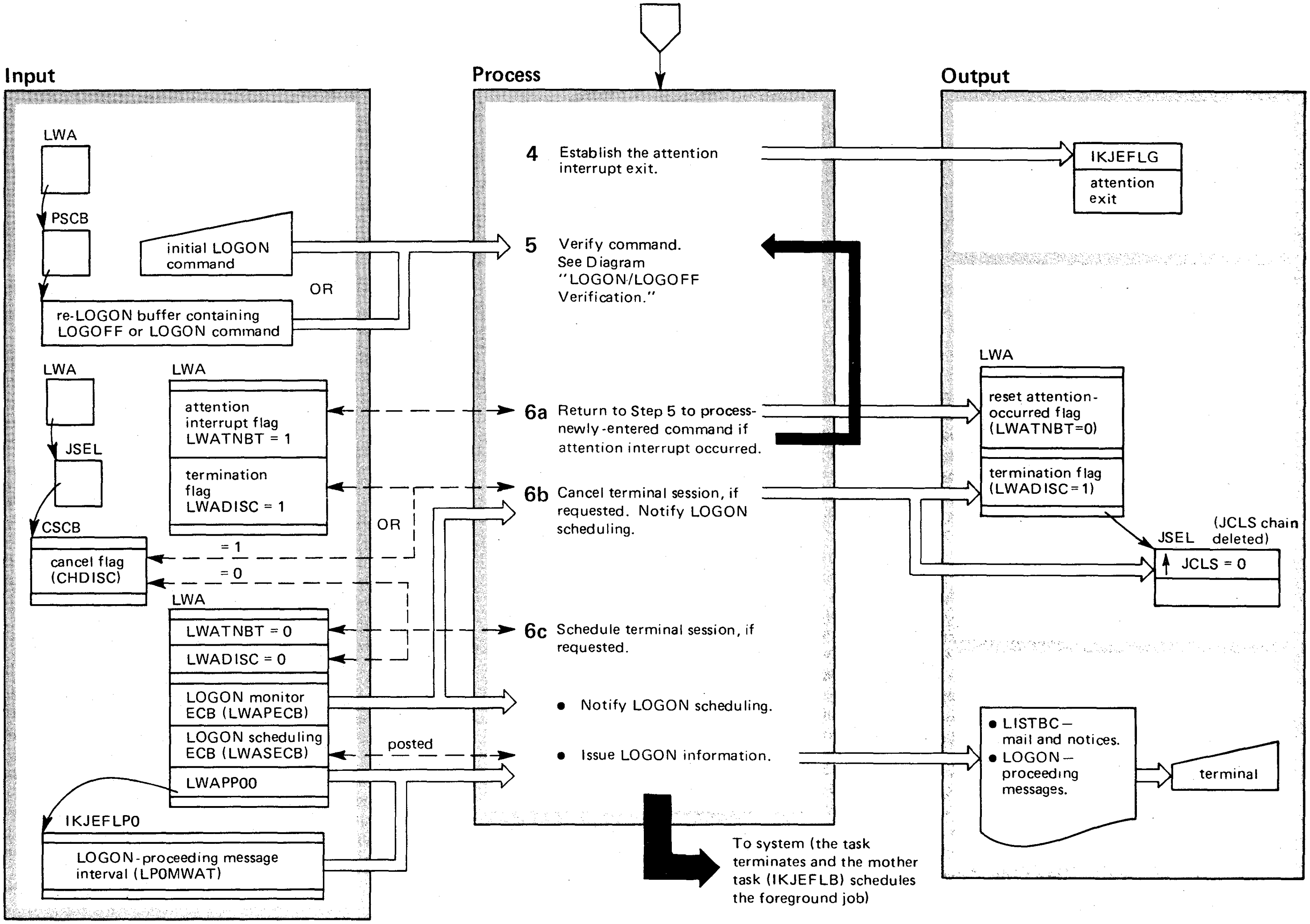


Diagram 5-4. LOGON Monitor (IKJEFLC) (Part 4 of 4)

Extended Description	Module	Label
5 The LOGON monitor invokes LOGON/LOGOFF verification (IKJEFLE) to scan and parse the LOGON or LOGOFF command. For a LOGOFF or a re-LOGON, the command text is found in the re-LOGON buffer; otherwise, the command is obtained from the terminal. LOGON verification checks the user's authorization and LOGON parameters against the user information in SYS1.UADS (user attribute data set) and prompts the user to replace invalid or missing information. See Diagram "LOGON/LOGOFF Verification."	IKJEFLE IKJEFLEA	
6a If the user presses the terminal's attention key during LOGON processing, he may re-enter the LOGON command. In this case, the LOGON monitor re-invokes LOGON verification to analyze the newly-entered command. The attention interrupt flag is reset to zero to indicate that the interrupt has been completely processed.	IKJEFLC	GOTOLE
6b If the system operator cancels the terminal user, if the user has entered a LOGOFF command, or if the user has failed to enter a valid LOGON command, the LOGON monitor ends the terminal session as follows:	IKJEFLC	GOTOLE
• Issues an error messages (IKJ56453I) to the terminal for an operator cancel.	IKJEFLGM	
• Issues a null STAX macro instruction to cancel the LOGON attention exit.	IKJEFLC	
• Frees the environment control table (ECT).		
• Notifies LOGON scheduling to terminate (LWAPECB—post code 24).		
• Waits for notification from LOGON scheduling to terminate (LWASECB—post code 24).		
• Returns to the operating system via SVC 3.		
6c • After LOGON verification has processed a valid LOGON command, the LOGON monitor notifies LOGON scheduling to schedule the terminal session (LWAPECB—post code 16). LOGON scheduling invokes the job scheduling subroutine of the initiator which attaches the terminal monitor program (TMP).	IKJEFLC	
• When LOGON scheduling is ready to invoke the job scheduling subroutine, it notifies the LOGON monitor to continue its operation. (LWASECB—post code 16). At that time, the LOGON monitor calls the LOGON information routine, allowing it to execute in parallel with the scheduling of the terminal session. The information routine attaches the LISTBC processor to issue mail and notices to the terminal user. Then the routine sets the timer to expire at the interval specified in the module IKJEFLP0. The LOGON-proceeding message is issued repeatedly to the terminal at this timed interval until the initiator is ready to attach the TMP. At that time, the pre-TMP exit (IKJEFLJ) notifies the information routine (LWASECB—post code 20) that the LOGON scheduling process is complete. The routine then cancels the timer and notifies the pre-TMP exit that LISTBC processing is completed (LWAPECB—post code 20).	IKJEFLH	
Finally, the LOGON monitor terminates as follows: —Issues a null STAX macro instruction to cancel the LOGON attention exit. (Pressing the terminal attention key no longer has any effect on LOGON processing.) —Deletes the environment control table (ECT). —Returns to the operating system via SVC 3.	IKJEFLC	CLEANUP
Error Processing		
LOGON scheduling establishes the LOGON monitor's ESTAI environment via a parameter on the ATTACH macro instruction. Since the LISTBC command processor is attached by the LOGON monitor task, it too is protected by the ESTAI environment. If the LOGON monitor task or the LISTBC task terminates abnormally, the ESTAI routine IKJEFLGB receives control. See Diagram "LOGON Monitor Recovery.	IKJEFLB IKJEFLGB IKJEFLC	
The LOGON monitor issues the STACK macro instruction to initialize the terminal as the source of input for commands. If this process encounters any errors, the LOGON monitor invokes the message handler to issue appropriate error messages to the terminal (IKJ56454I) or to the operator (IKJ608I). Also, the monitor turns on the LOGON-termination bit (LWADISC).	IKJEFLGM	
The LOGON monitor issues the MGCR macro instruction to chain a new CSCB. If this routine passes back a non-zero return code, the monitor issues error messages (IKJ56454I) to the terminal via the message handler. If the cancel bit is on (CHDISC field of the CSCB), a session-cancelled message (IKJ56453I) is issued by the message handler. In any case, the monitor ends the terminal session as in Step 6b of this diagram.	IKJEFLC IKJEFLGM	

Diagram 5-5. LOGOFF Processing (IKJEFLL) (Part 1 of 2)

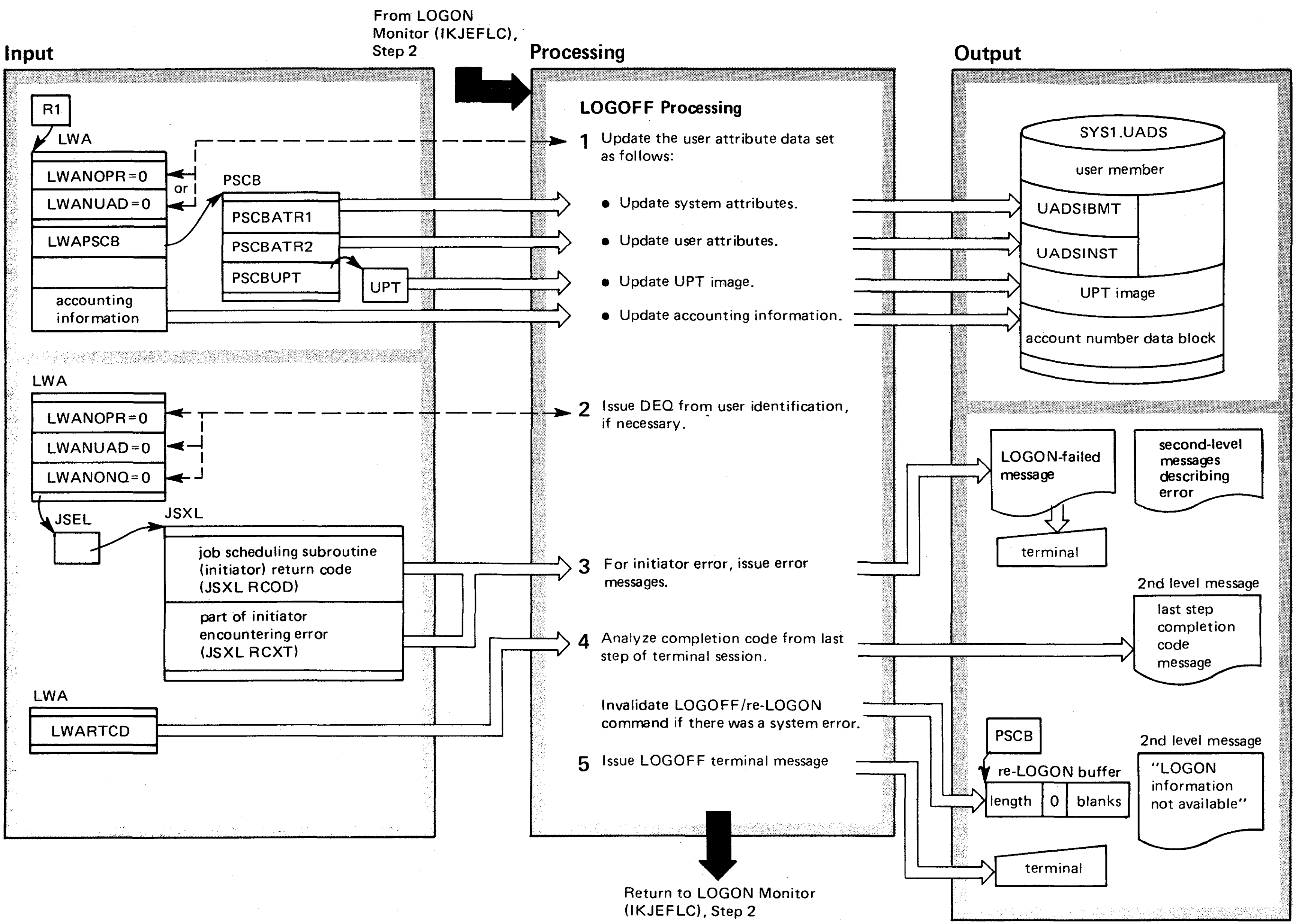


Diagram 5-5. LOGOFF Processing (IKJEFLL) (Part 2 of 2)

Extended Description	Module	Label
LOGOFF processing updates the terminal user's entry in SYS1.UADS and analyzes the return codes from the job scheduling subroutine (initiator) and from the last step of the terminal session. LOGOFF processing is performed for a LOGOFF command and for a re-LOGON. It is not performed for an initial LOGON (LWAILGN=1) or for recovery processing (LWABEND=1).	IKJEFLL	
1 Using the PROFILE command, the terminal user is able to change the attributes associated with his user identification. These attributes are supplied by a member of SYS1.UADS. LOGOFF processing must update this member at the end of the terminal session to reflect the changes made by the user. If the installation has supplied all of the LOGON information normally supplied by SYS1.UADS (LWANOPR=1 and LWANUAD=1), it is not necessary to update the user's member of SYS1.UADS. If any of the three bits LWAATR1, LWAATR2, and LWABUPT are off, the corresponding information (system attributes, user attributes, and the user profile, respectively) was not supplied by the installation. The information not supplied by the installation (and, therefore, subject to changes made via the PROFILE command) is updated by LOGOFF processing. If LWAACCT≠0, the user's accounting information in SYS1.UADS is also updated. Accounting information consists of the following items: the length of the terminal session, the amount of CPU time used, and the number of service units used.	IKJEFLL	UPDTUADS
2 LOGOFF processing must release the user identification resource that was obtained during LOGON verification. LOGOFF issues the DEQ macro instruction. If the three bits LWANOPR, LWANUAD, and LWANONQ are turned off, an ENQ was never issued on the user identification. In this case, a DEQ is not necessary.	IKJEFLL	DEQUSER
3 If the job scheduling subroutine encountered an error (LWARTCD≠0), LOGOFF processing examines the field JSXLRCXT to determine what part of job scheduling failed. Next, it examines the fields JSXLRCOD and LWARCDE to determine the nature of the error. Finally, LOGOFF informs the message handler (IKJEFLGM) to build the appropriate second-level message (IKJ56457I to terminal).	IKJEFLL	
4 LOGOFF analyzes the return code from the last step of the terminal session (LWARTCD) and builds an appropriate second-level message (IKJ56470I to terminal) via the message handler. If the code is a system return code, the re-LOGON buffer is considered to be unusable and is filled with blanks. In this case, LOGON/LOGOFF verification must prompt the user for a LOGON or LOGOFF command. (See Diagram "LOGON/LOGOFF Verification.") The exception is a system return code that was generated by attention exit processing (indicated by LWATNBT=1). The attention exit posts the cancel ECB in the CSCB with a system code of 622, so that the job scheduling subroutine terminates in the same way as for an operator cancel. In this case, there is no reason why the re-LOGON buffer would be unusable; therefore, the contents of the buffer are retained.	IKJEFLL	
5 LOGOFF calls the LOGON time and date processor (IKJEFLPA) to set up the date and time-of-day buffers for the logged-off message. Then LOGOFF invokes the message handler to issue the logged-off message to the terminal (IKJ56470I).	IKJEFLL	LGMSETUP
Error Processing If, at any time, LOGOFF processing encounters an I/O error, an OPEN error, or a service routine error, it issues an error message (IKJ56454I) to the terminal via the message handler and turns on the LOGON-termination bit.	IKJEFLL	

Diagram 5-6. LOGON/LOGOFF Verification (IKJEFLE and IKJEFLES) (Part 1 of 4)

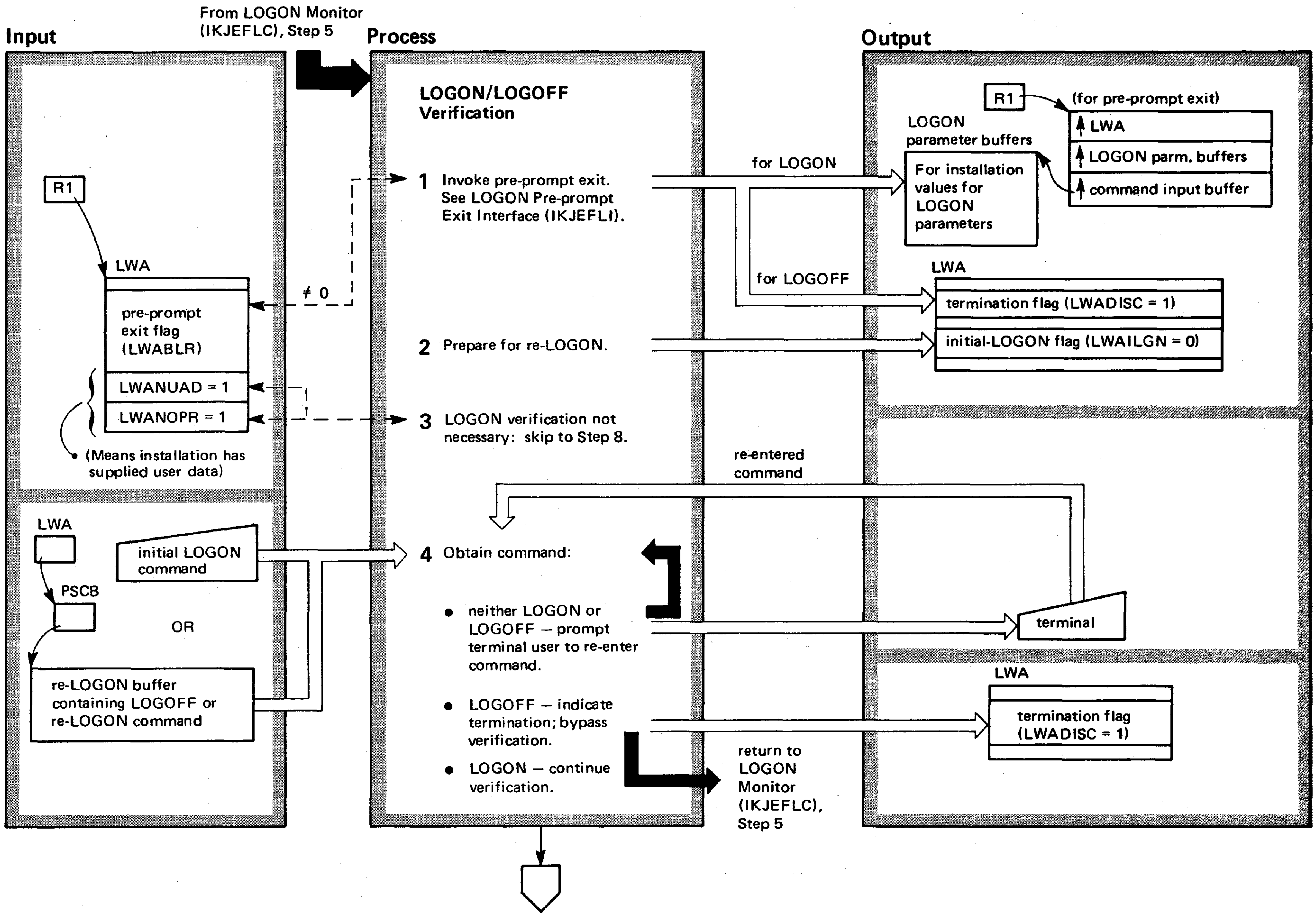


Diagram 5-6. LOGON/LOGOFF Verification (IKJEFLE and IKJEFLES) (Part 2 of 4)

Extended Description	Module	Label
LOGON/LOGOFF verification scans the LOGON or LOGOFF command and checks the LOGON parameters against the information in the user's member of the SYS1.UADS data set. As the verification process is checking LOGON parameters, it records valid LOGON information in various control blocks. (See Figure 5-5.) An optional installation exit (pre-prompt exit IKJEFLD) can replace any part or all of the verification processing. If the LOGON is valid, JCL card images (JOB and EXEC) that define the terminal session are built.	IKJEFLE	
1 If the VCON for the installation exit (IKJEFLD) is non zero (indicating an installation exit is present and link-edited into the LOGON load module), the interface routine IKJEFLI is invoked to initialize a parameter list for the exit. (See Diagram "LOGON Pre-prompt Exit Interface.") The interface does not pass control to the pre-prompt exit (IKJEFLD) if the command is a LOGOFF.	IKJEFLE	GOTOIER
2 The initial-LOGON flag is turned off following the first GETLINE macro instruction issued by LOGON/LOGOFF verification. Any subsequent LOGON command entered by the terminal user for the current address space is considered to be a re-LOGON.	IKJEFLE	
3 LOGON/LOGOFF verification returns to the LOGON monitor if the termination flag is on (LWADISC) or if the cancel flag is on (CHDISC). If the pre-prompt exit has supplied all the LOGON information and indicates that no verification is necessary, the normal verification is bypassed.	IKJEFLE	
4 After the command scan service routine (IKJSCAN) scans the command for LOGON or LOGOFF, the verification process continues as follows:	IKJEFLEA	
• If neither command was found, the terminal user is prompted to enter LOGON or LOGOFF and the scan is repeated.		LOGONOFF
• If the command was a LOGOFF, the verification process returns control to the caller, the LOGON monitor. For a LOGOFF HOLD (TSBHLDL=1), terminal input/output control (TIOC) keeps a line open to the terminal. If at any time a terminal line is accidentally disconnected, TIOC retains, for a time specified in IKJPRM00 of SYS1.PARMLIB, the control blocks and the address space used for the current terminal session. If the terminal user then enters a LOGON RECONNECT command with the same user identification as the retained address space, TIOC reinstates the user in that address space.		
• If the command was a LOGON, the verification process continues (see Step 5).		

Diagram 5-6. LOGON/LOGOFF Verification (IKJEFLE and IKJEFLES) (Part 3 of 4)

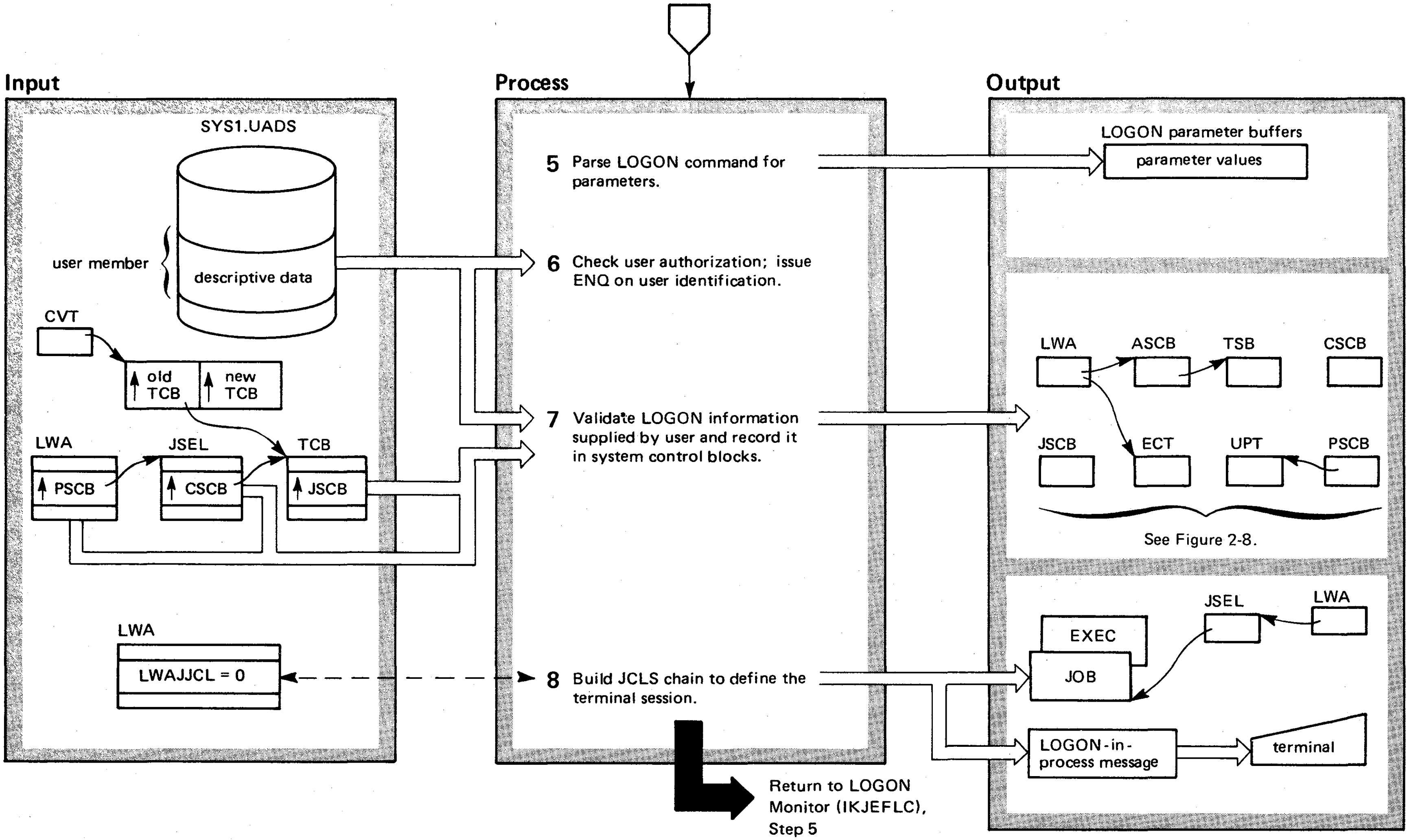


Diagram 5-6. LOGON/LOGOFF Verification (IKJEFLE and IKJEFLES) (Part 4 of 4)

Extended Description	Module	Label
5 The verification process invokes the parse service routine (IKJPARSE) to check the syntax of the LOGON command. If the command contains the RECONNECT parameter, TIOC determines whether the user identification is already assigned to an address space (one that TIOC retained following a disconnected line). If the user identification has an address space assigned to it, LOGON verification terminates; TIOC reinstates the user in the retained address space. If the user identification has no address space assigned to it, the LOGON RECONNECT is rejected.	IKJEFLE	
6 LOGON verification opens the SYS1.UADS data set (user attribute data set) and copies into real storage the member associated with the user identification on the LOGON command and then ensures that the user identification is authorized. The user identification and its length are stored in the PSCB (protected step control block). Then LOGON issues an ENQ on the user identification resource. If the resource has already been obtained, LOGON verification reinvokes the pre-prompt exit if it exists. The installation can choose to authorize the user or to cancel the LOGON process.	IKJEFLE	OPEN

Extended Description	Module	Label
7 LOGON verification compares the LOGON parameter values with the user information in SYS1.UADS to check for the validity of the LOGON parameters. If parameters are invalid or missing, LOGON verification prompts the user for correct parameters. The user's reply is re-parsed and verified. Verification checks the user's password, account number, procedure name, region size, and performance group. The system resources manager checks that the performance group is defined to the system and that the group can be used at this time. The job entry subsystem verifies that the destination choice (DEST parameter) defines a valid device for SYSOUT data sets. See Figure 2-8 for a list of the data areas that LOGON initializes with user information.	IKJEFLE	

8 If LWAJJCL=1, the pre-prompt exit has supplied the JCL card images that define the terminal session. Otherwise, LOGON processing constructs the JCL card images as follows:
 	
//userid JOB 'account #',REGION=region size
//procname EXEC procname,PERFORM=performance group
 	
where the userid (user identification), account #, region size, and performance group are obtained from the LOGON parameters, from the user's member of SYS1.UADS, or from the pre-prompt exit.
	IKJEFLEA	BUILDJCL

Error Processing		
If the LOGON is an initial LOGON (LWAILGN=1), and the address of the terminal input line is zero, LOGON verification obtains a line from the terminal (issues a GETLINE for the terminal).	IKJEFLE	
LOGON verification is part of the LOGON monitor task and, therefore, is protected by the monitor's ESTAI environment in case of an ABEND.	IKJEFLGB	

The following data areas contain TSO user information supplied by the SYS1.UADS data set, by the installation, or by the LOGON parameters:

Data Area Name	Field Name	Contents
ASCB	ASCBJBNS	Address of user identification.
CSCB	CHCLS	Procedure name for this LOGON.
	CHKEY	User identification.
ECT	ECT	Flags that control LISTBC processing.
EXEC card image		Procedure name for this LOGON.
		Performance group number.
JOB card image		Account number.
		Region size.
JSEL	JSEL	Address of JCL card images.
JSOL	JSOLDEST	Default destination for SYSOUT data sets.
LWA	LWACTLS	Control switches set by the installation exit.
	LWADEST2	Default destination for SYSOUT data sets.
	LWAACCT	Offset of accounting information in SYS1.UADS.
	LWATCPU	Total CPU time used.
	LWATSRU	Total service units used.
	LWATCON	Total time connected to the system.
	LWARTCD	Completion code for the last step of the terminal session.
PSCB	PSCBUSER	User identification.
	PSCBUSRL	Length of user identification.
	PSCBATR1	System attributes: switches that control use of OPERATOR, ACCOUNT, and SUBMIT commands, that indicate volume and mount authorization, and that define the attention key as the line-delete key.
	PSCBATR2	User attributes – reserved for installation use.
	PSCBGPNM	Generic unit name.
	PSCBRSZ	Region size.
TSB	TSBPSWD	Password.
UPT	UPTSWS	Environmental switches.
	UPTNPRM	No-prompting switch.
	UPTMID	Switch that controls printing of message identifiers.
	UPTNCOM	Switch that controls SEND command authorization.
	UPTPAUS	Switch that indicates whether to pause for a "?".
	UPTALD	Switch that defines the attention key as the line-delete key.
	UPTMODE	Switch that controls printing of mode messages.
	UPTWTP	Switch that allows the user to receive WTP messages.
	UPTCDEL	Character-delete character.
	UPTLDEL	Line-delete character.
	UPTPREFX	Data set name prefix.
	UPTPREFL	Length of data set name prefix.

Figure 2-8. Data Areas Containing LOGON User Information

Diagram 5-7. LOGON Pre-prompt Exit Interface (IKJEFLI) (Part 1 of 2)

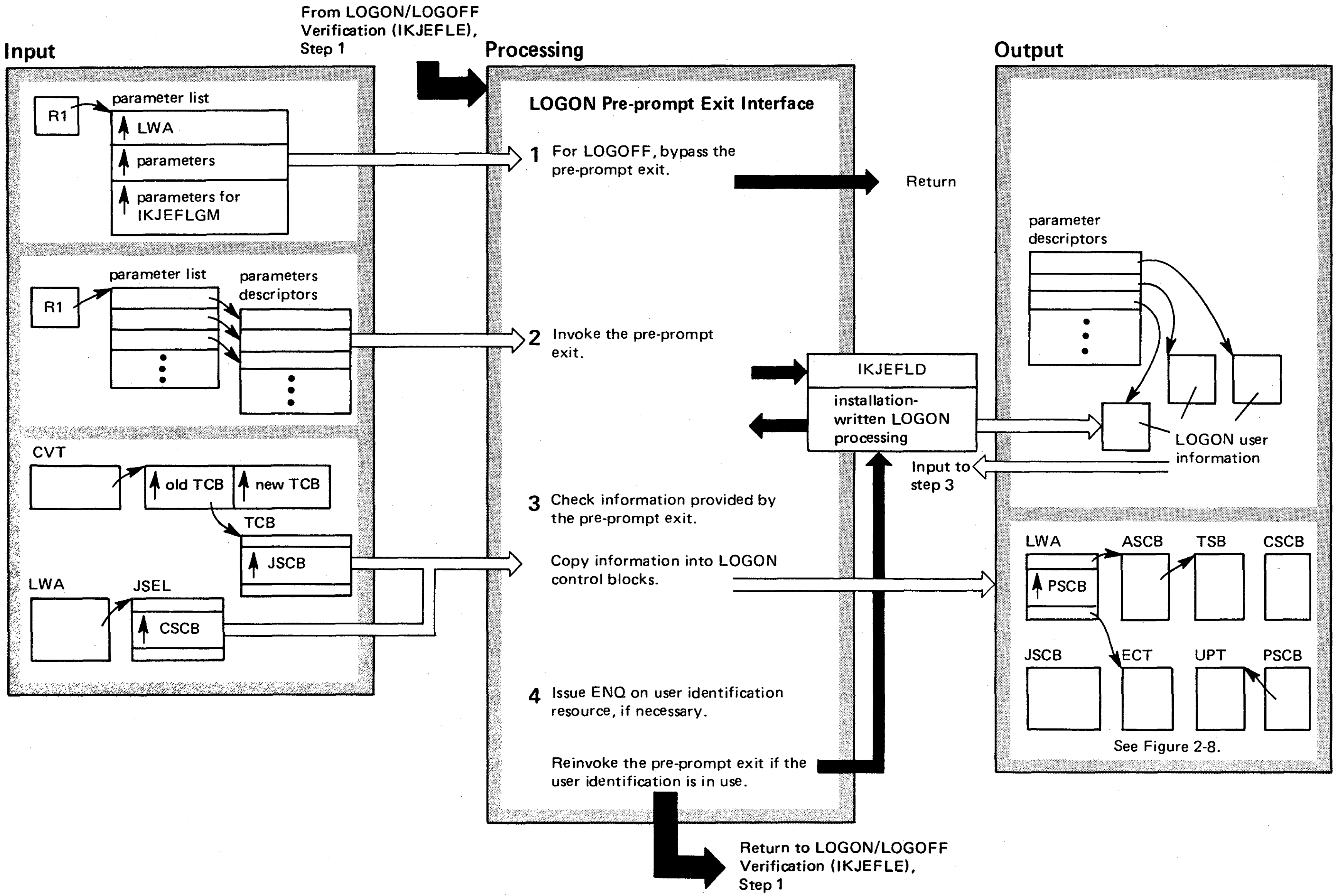


Diagram 5-7. LOGON Pre-prompt Exit Interface (IKJEFLI) (Part 2 of 2)

Extended Description	Module	Label
The LOGON pre-prompt exit interface invokes the LOGON pre-prompt exit which is a routine written by the installation. The pre-prompt exit can provide LOGON information on behalf of the terminal user, verify the user's LOGON command, and collect accounting information. Any user information provided by the pre-prompt exit overrides the information stored in the user's member of the SYS1.UADS data set. In fact, an installation can, if it wishes, replace all of the normal LOGON verification processing. For directions on writing the exit routine, refer to the topic "Writing a LOGON Pre-prompt Exit" in the publication *OS/VS2 System Programming Library: TSO,* GC28-0629.	IKJEFLI	
1 The pre-prompt exit interface uses the command scan service routine (IKJSCAN) to determine if the command is a LOGON or LOGOFF. If it is a LOGOFF, the interface does not invoke the pre-prompt exit. Instead, it returns to its caller.	IKJEFLI	
2 The interface builds and passes to the pre-prompt exit a parameter list that defines those parameters the pre-prompt exit needs to verify the LOGON command and to provide LOGON information. Most of the addresses in the parameter list point to two-word descriptors. The first word of the descriptor contains the address of the actual parameter. The second word contains both the maximum length for the parameter and the actual length.		L10100

Extended Description	Module	Label
3 After invoking the pre-prompt exit, the interface routine checks the parameter list for validity:	IKJEFLI	
• Ensures the parameter list is unchanged.		
• Ensures the parameter descriptors are unchanged, except for the field containing the actual length of the parameter.		
• Checks that the actual length of each parameter does not exceed the maximum length for the parameter.		
If errors are discovered, the interface invokes the message handler (IKJEFLGM) to issue error messages and terminates the terminal session (LWADISC=1). If no errors are found, the interface copies into the appropriate control blocks all user information provided by the pre-prompt exit. See Figure 2-8. A control field in the LOGON work area (LWACTLS) contains bits that indicate what information the installation has provided.		L1800
4 If the pre-prompt exit has specified in the LOGON work area that the terminal user is not to be prompted (LWANOPR=1), that all LOGON information has been verified (LWANUAD=1), and that an ENQ is to be issued (LWANONQ=0), then the interface issues an ENQ on the user identification resource. If the resource is already in use, the pre-prompt exit is re-invoked to determine a course of action. The installation may choose to allow more than one user with the same user identification to be logged-on simultaneously (LWANONQ=1). In this case, the interface does not issue an ENQ on the user identification resource. Or, the installation may, instead, choose to terminate the session (LWADISC=1).	IKJEFLI	
Error Processing		
If either the LOGON pre-prompt exit interface (IKJEFLI) or the pre-prompt exit (IKJEFLD) cause an ABEND, the LOGON monitor's ESTAI routine IKJEFLGB is invoked by ABEND processing. In certain cases, the ESTAI routine schedules a re-attach of the LOGON monitor task. See Diagram "LOGON Monitor Recovery."	IKJEFLGB	

Diagram 5-8. LOGON Monitor Recovery (IKJEFLGB) (Part 1 of 2)

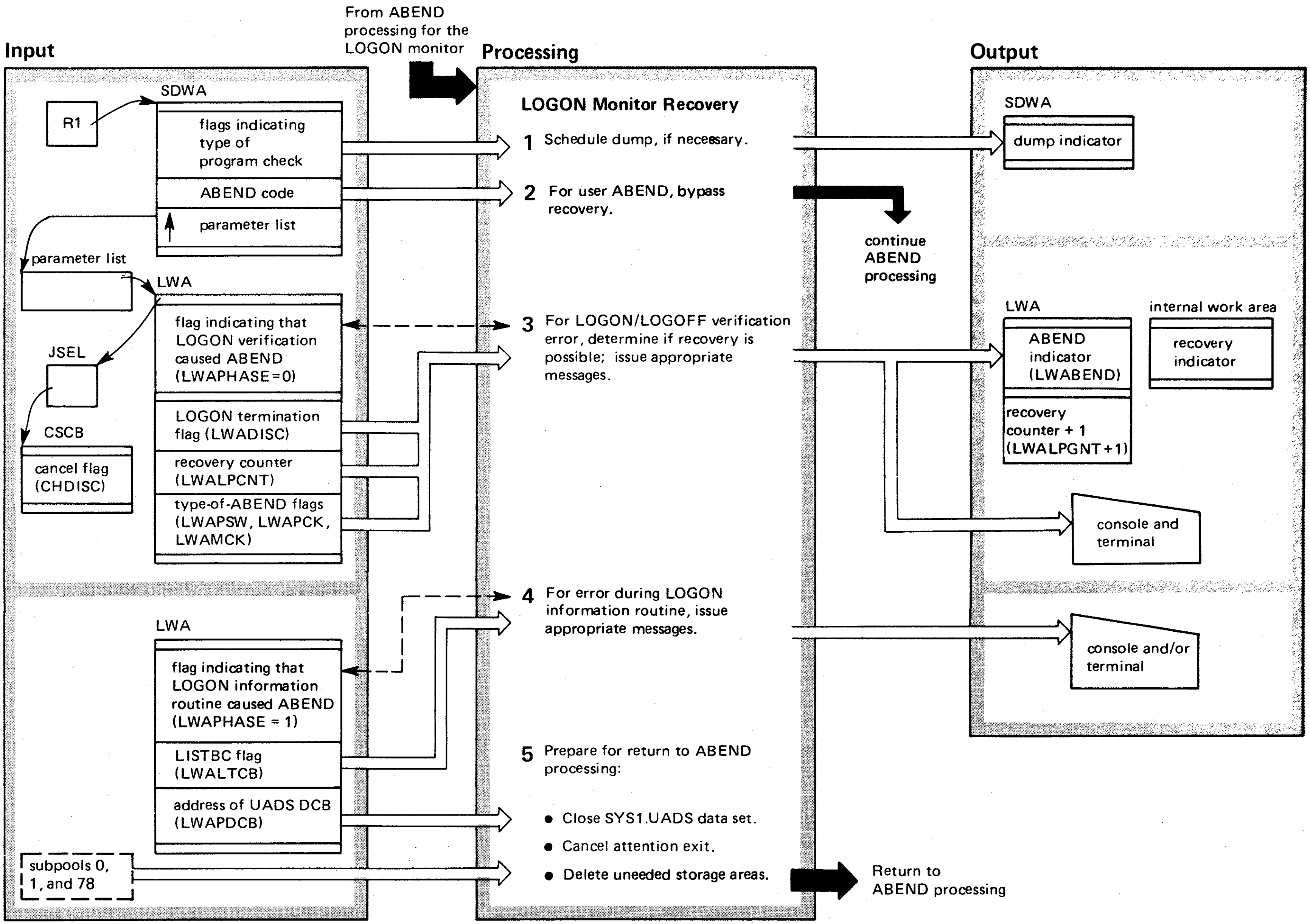


Diagram 5-8. LOGON Monitor Recovery (IKJEFLGB) (Part 2 of 2)

Extended Description	Module	Label
The LOGON monitor recovery routine receives control from ABEND processing following the abnormal termination of the LOGON monitor task. LOGON monitor recovery is an ESTAI routine that was specified on the ATTACH macro instruction when the LOGON monitor was attached by the LOGON scheduling task. If possible, a retry of the LOGON monitor is attempted by informing the LOGON scheduling task to re-attach the LOGON monitor (LWABEND = '1' B).	IKJEFLGB	
1 A dump is scheduled if the abnormal termination was the result of a program check or a PSW restart (an external interrupt from the operator).	IKJEFLGB	
2 If the ABEND code represents a user completion code, then recovery of the LOGON monitor task is not attempted. LOGON monitor recovery issues no error messages and passes control back to ABEND processing to continue the abnormal termination.	IKJEFLGB	
3 If the LOGON monitor abnormally terminated during LOGON/LOGOFF verification, recovery of the LOGON monitor task is scheduled (LWABEND=1).	IKJEFLGB	PHASE1
Recovery is not attempted in the following cases:		
• The system or the operator has canceled the terminal session (CHDISC=1).		
• The terminal session is scheduled for termination (LWADISC=1).		
• Four recoveries have already been attempted (LWALPCNT=4).		
• The current ABEND is the same type as the previous one (determined by checking bit settings in the LOGON work area: fields LWAPSW, LWAPCK, and LWAMCHK).		
LOGON monitor recovery builds and issues appropriate messages to the terminal and to the system operator. One set (IKJ56451I for the terminal and IKJ603I for the operator) is issued if the LOGON pre-prompt exit terminated abnormally (LWAINX1=1). Another set (IKJ56452I for the terminal and IKJ601I for the operator) is issued if LOGON/LOGOFF verification itself terminated abnormally (LWAINX1=0).		MSGINIT

Extended Description	Module	Label
4 If the ABEND occurred after the user's LOGON information has been processed and the terminal session has been scheduled (that is, LWAPHASE=1), recovery may not be necessary. If LWAPHASE=1, the ABEND occurred either during LISTBC command processing or during the issuing of the LOGON-proceeding messages (issued by LOGON module IKJEFLH). If LISTBC caused the ABEND (LWALTCB=1), LOGON monitor recovery issues an error message to the terminal (IKJ56406I) and the LISTBC task terminates. In this case, the scheduling of the terminal session proceeds normally. If the LOGON module IKJEFLH caused the ABEND, LOGON monitor recovery does not schedule a re-attach of the monitor (LWABEND=0) but does issue error messages to the terminal (IKJ56452) and to the operator (IKJ601).	IKJEFLGB	PHASE2
5 LOGON monitor recovery performs exit processing as follows:	IKJEFLGB	
• Closes the SYS1.UADS data set using the DCB address in the LOGON work area. If this address is zero, recovery does not issue the CLOSE macro instruction. Recovery also issues a DEQ on the SYS1.UADS directory resource.		CLOSUADS
• Issues a null STAX macro instruction to cancel the attention exit. Pressing the terminal attention key no longer has any effect on LOGON processing.		
• Frees the storage allocated to subpools 0, 1, and 78.		FREECORE

Diagram 5-9. Pre-TMP Exit (IKJEFLJ) (Part 1 of 2)

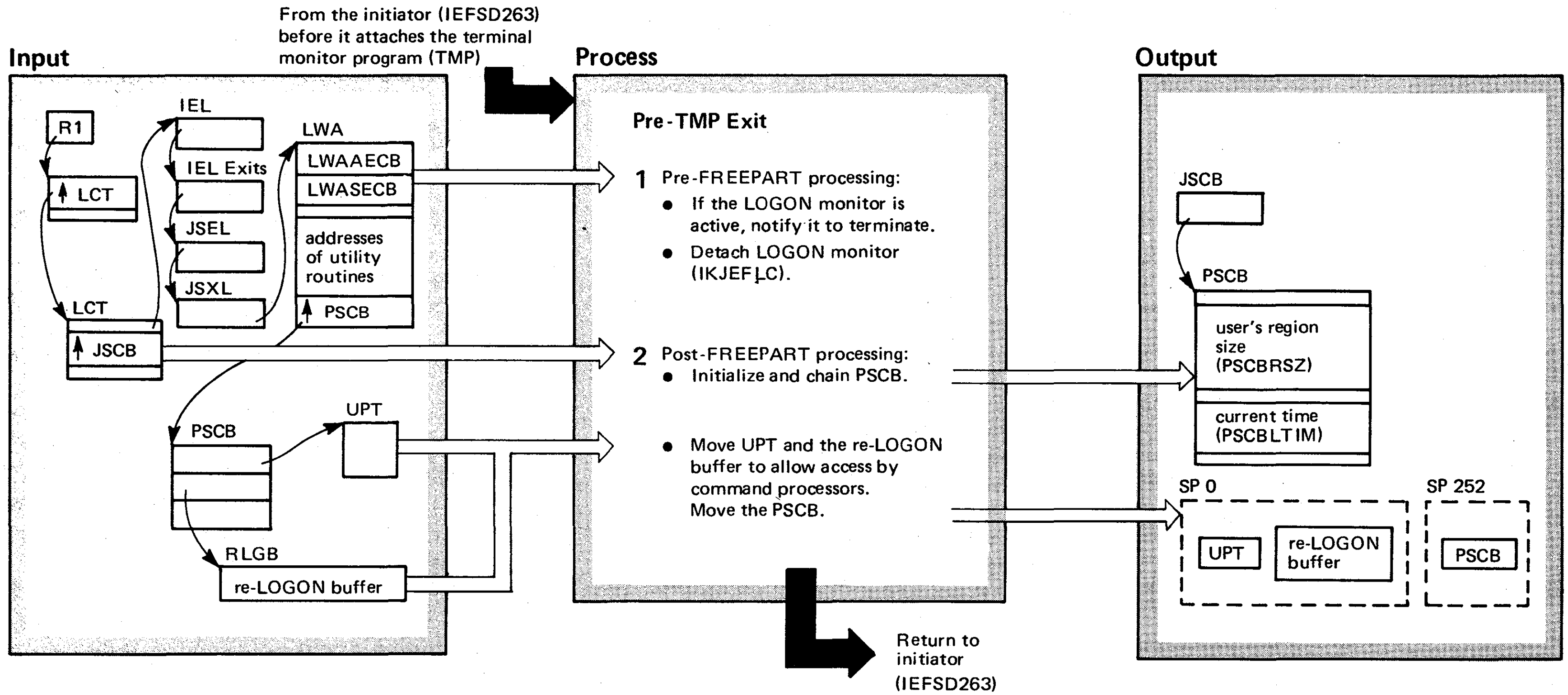


Diagram 5-9. Pre-TMP Exit (IKJEFLJ) (Part 2 of 2)

Extended Description	Module	Label
The initiator (IEFSD263) invokes the pre-TMP exit before attaching the terminal monitor program (TMP); it invokes the post-TMP exit after the TMP terminates. The pre-TMP exit prepares for the terminal session to begin by notifying the LOGON monitor task to terminate. The pre-TMP exit has two parts; an entry point name is assigned to each part. The first part is invoked before the initiator issues the FREEPART macro instruction (pre-FREEPART processing). The second part is invoked following the FREEPART (post-FREEPART processing).	IKJEFLJ	
1 This step represents pre-FREEPART processing. It is performed before the initiator issues the FREEPART macro instruction. Since the LOGON monitor task may still be active, the data areas it uses must not be deleted (by FREEPART) until the task is notified to terminate.	IKJEFLJ	IKJLM1
• Pre-FREEPART processing notifies the LOGON monitor task to terminate (LWASECB—post code 20). When the monitor task terminates, it notifies pre-FREEPART processing to continue (LWAPECB—post code 20). See LOGON Monitor (IKJEFLC), Step 6c.		
• The System Initiated Cancel (SIC) is notified that the TMP was executing when the line dropped or the user canceled. SIC will then notify the Post-TMP exit to free other users who are waiting on this memory. For example, SEND W/WAIT option sent to a canceled memory can cause the sender to wait forever unless the Post-TMP exit frees the sender.		

Extended Description	Module	Label
2 This step represents post-FREEPART processing. It is performed after the initiator issues the FREEPART macro instruction. Post-FREEPART processing now can move the UPT and the re-LOGON buffer to subpool 0 (which is deleted by the FREEPART).	IKJEFLJ	IKJLJ1
• Post-FREEPART processing invokes the SWA manager to obtain the user's region size from the step control block (SCB). The region size is stored in the protected step control block (PSCB). If the SCT indicates that the terminal session is a job with more than one step, post-FREEPART processing passes a non-zero return code back to the initiator, which then terminates the job. The current time of day is also stored in the PSCB for later use in computing the length of the terminal session.		
• The UPT and the re-LOGON buffer are moved to subpool 0 (a non-protected subpool) so that the command processors may alter them during the terminal session. The PSCB is moved to subpool 252; the command processors cannot alter data areas in subpool 252.		

Diagram 5-10. Post-TMP Exit (IKJEFLK) (Part 1 of 2)

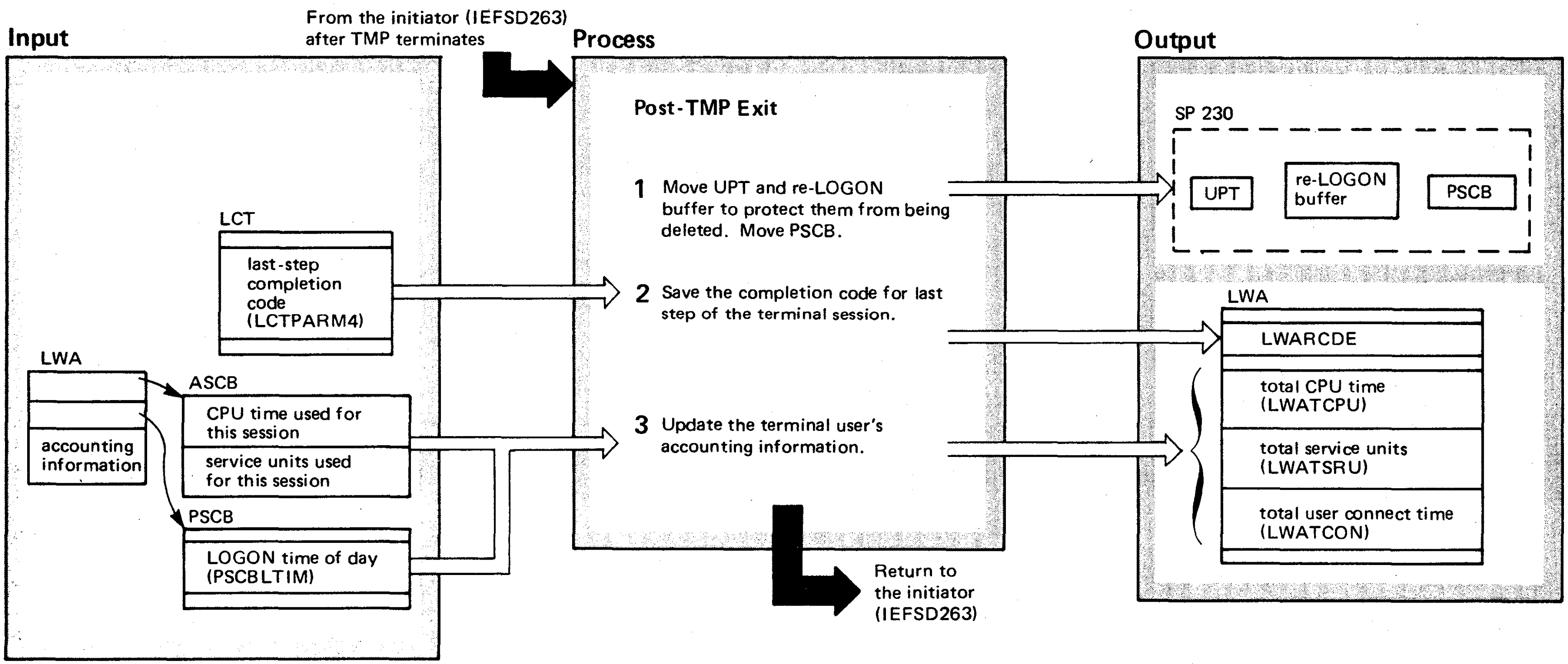


Diagram 5-10. Post-TMP Exit (IKJEFLK) (Part 2 of 2)

Extended Description	Module	Label
The initiator (IEFSD263) invokes the post-TMP exit after the TMP terminates. The post-TMP exit saves the completion code from the last step of the terminal session and updates the user's accounting information in the LOGON work area. Then, the initiator performs termination processing and passes control back to the LOGON scheduling task.		
1 The post-TMP exit moves the UPT and the re-LOGON buffer from subpool 0 to subpool 230 to prevent job scheduling from deleting them during job termination. The PSCB is also moved to subpool 230.	IKJEFLK	IKJLK1
2 The post-TMP exit saves the completion code from the last step of the terminal session, obtaining it from the linkage control table (LCT). The completion code is later analyzed by LOGOFF processing to determine if the terminal session terminated abnormally. See Diagram "LOGOFF Processing."	IKJEFLK	IKJLK1
3 The post-TMP exit updates the accounting information in the LOGON work area to account for the system resources used during the terminal session that is now terminating.	IKJEFLK	
Error Processing If either the pre-TMP exit or the post-TMP exit causes an ABEND, LOGON scheduling's ESTAE routine IKJEFLS is invoked by ABEND processing. The function of this ESTAE routine is described under "Error Processing" in the diagram "LOGON Initialization and Scheduling."	IKJEFLJ,K	

Index

INDEX

SY28-0762-0

OS/VS2 System Logic Library Volume 2 (S370-36) Printed in U.S.A. SY28-0762-0

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

OS/VS2
System Logic Library
Volume 2
SY28-0762-0

READER'S COMMENT FORM

Your views about this publication may help improve its usefulness; this form will be sent to the author's department for appropriate action. Using this form to request system assistance or additional publications will delay response, however. *For more direct handling of such requests, please contact your IBM representative or the IBM Branch Office serving your locality.*

Possible topics for comment are:

Clarity Accuracy Completeness Organization Index Figures Examples Legibility

Cut or Fold Along Line

What is your occupation? ______________________________

Number of latest Technical Newsletter (if any) concerning this publication: ____________

Please indicate your name and address in the space below if you wish a reply.

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.)

SY28-0762-0

Cut or Fold Along Line

Your comments, please . . .

This manual is part of a library that serves as a reference source for system analysts, programmers, and operators of IBM systems. Your comments on the other side of this form will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Fold Fold

First Class
Permit 81
Poughkeepsie
New York

Business Reply Mail
No postage stamp necessary if mailed in the U.S.A.

Postage will be paid by:

International Business Machines Corporation
Department D58, Building 706-2
PO Box 390
Poughkeepsie, New York 12602

Fold Fold

IBM®

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)