SY33-8577-1
File No. S370-36

Systems

DOS/VS POWER/VS

Logic Part 2

Program Number 5745-SC-PWR
Release 34

IBM

Second Edition (April 1977)

This edition includes the support of the 3790 Communication System – Remote Job Entry (RJE) facility and various quality improvements. Changes and additions to the text and illustrations are indicated by a vertical line to the left of the change.

This edition, as amended by Technical Newsletter SN33-9241, applies to version 5, release 34, of the Disk Operating System/Virtual Storage, DOS/VS, and to any subsequent versions and releases 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.

Note: *For the availability dates of features and programming support described in this manual, please contact your IBM representative or the IBM branch office serving your locality.*

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 reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to *IBM Germany, Publications Dept., Schwertstrasse 58, D-7032 Sindelfingen, Germany.* Comments become the property of IBM.

IBM® Technical Newsletter

This Newsletter No.	SN33-9241
Date	November 28, 1977
Base Publication No.	SY33-8577-1
File No.	S370-36
Prerequisite Newsletters	None

DOS/VS POWER/VS Logic, Part 2

This Technical Newsletter, a part of the independent component release (ICR) of support for the IBM 3800 Printing Subsystem under Release 34 of the IBM Disk Operating System/Virtual Storage, DOS/VS, and DOS/VS POWER/VS, provides replacement pages for your publication. Information contained on these pages applies only if the ICRs are installed on your system. You need not insert the pages if they are not installed. These replacement pages remain in effect for subsequent DOS/VS releases unless specifically altered. Pages to be replaced are:

Title page-Charts
13,14
17-24
27-32.1(32.1 added)
33-38
41,42
69,70
77-78.1(78.1 added)
105, 106
155-158.2 (158.1,158.2 added)
197,198
207-210
225,226
231-234
237-244
247-248.2(248.1,248.2 added)
275,276
283-286.2(286.1,286.2 added)
287-288.1(288.1 added)
327,328
333,336.1(336.1 added)
337,342.1(342.1 added)
343-346.4(346.1-346.4 added)
347-352.1(352.1 added)
355,356
359,360.3(360.1-360.3 added)
393-394.6(394.1-394.6 added)
437-442.2(442.1,442.2 added)
443-444.1(444.1 added)
445-446.4(446.1-446.4 added)
447,448
453-470
473-482.1(482.1 added)
485-486.1(486.1 added)
487-490.2(490.1,490.2 added)
493,494
497-498.1(498.1 added)
509-518.10(518.1-518.10 added)
519-532
569-582.1(582.1 added)
583-596.1(596.1 added)
660.1 (added)
663,664

A technical change to the text or to an illustration is indicated by a vertical line to the left of the change.

Summary of Amendments

Changes to the system are summarized under "Summary of Amendments" preceding the Contents.

For a complete list of publications that support the DOS/VS IBM 3800 Printing Subsystem ICR, and the DOS/VS POWER/VS IBM 3800 ICR, see the *DOS/VS IBM 3800 Printing Subsystem Programmer's Guide*, GC26-3900.

Note: *Please insert this page in your publication to provide a record of changes.*

IBM Germany, Publications Department, Schwertstrasse 58, D-7032 Sindelfingen, Germany

Printed in U.S.A.

which should be used together with *DOS/VS POWER/VS Logic Part 1,* SY33-8576, contains information about the internal logic of POWER/VS. To use it effectively, the reader should be familiar with the concepts and facilities of POWER/VS as they are described in the following IBM DOS/VS manuals:

Introduction to DOS/VS, GC33-5370
DOS/VS POWER/VS Installation Guide and Reference, GC33-6048
DOS/VS POWER/VS Work Station User's Guide, GC33-6049

RJE,SNA users should also be familiar with the VTAM concepts and facilities as they are described in:

VTAM Concepts and Planning, GC27-6998
VTAM Macro Language Reference, GC27-6995

Summary of Amendments

INDEPENDENT COMPONENT RELEASE OF IBM 3800 PRINTING SUBSYSTEM SUPPORT

Technical Newsletter SN33-9241 documents changes to support the IBM 3800 Printing Subsystem under DOS/VS POWER/VS.

RELEASE 34

This revised edition of the Logic Manual documents extended support for Remote Job Entry with SNA terminal support (RJE,SNA) using the 3790 Communication System with RJE Facility, plus quality improvement items.

RELEASE 33

Technical Newsletter SN33-9208 documents new support for the internal reader/writer (PUTSPOOL, GETSPOOL, and CTLSPOOL) and improvements in performance, handling, and accounting.

RELEASE 32 (RJE,SNA)

This edition of the Logic Manual documents new support for Remote Job Entry with SNA terminal support (RJE,SNA).

Manual usability has been improved through dividing the Logic Manual into two parts. Part 1 (SY33-8576) describes the overall logic, data area layouts, a directory of the modules and macros and messages, and other general logic information. Part 2 contains the detailed logic description of each POWER/VS module.

Contents

Figures

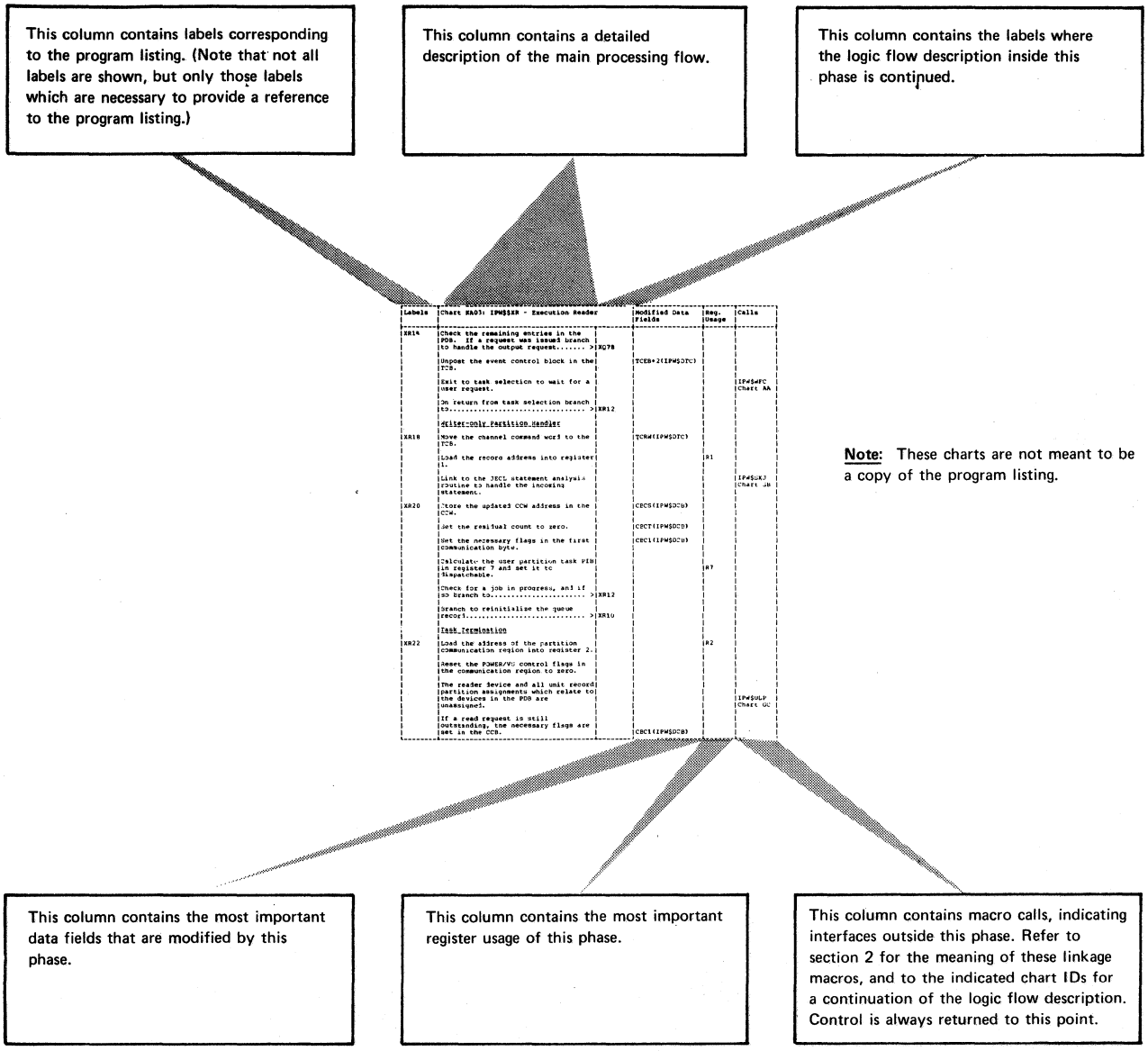
Charts

For a directory of the contents of this manual, see Section 4: Directory of the DOS/VS POWER/VS Logic Part 1, SY33-8576.

This manual describes in detail the phases of the POWER/VS program. An introduction describes linkage conventions between routines. The phase descriptions are preceded by a general overview chart.

The command processing phases, and the SNA processing phases are described using HIPO; all other phases are described by using a combination of text, flowcharts, listings and references. The latter method can be used to locate a piece of coding in the listing by going through the logic flow and scanning data fields and register references and comparing them with their actual contents as they appear in a dump. The contents of these charts is explained below.

The module IPW$$MD is not documented, due to its simplicity. It contains only POWER/VS message definitions, and otherwise no executable code.



POWER/VS LINKAGE CONVENTIONS

This section begins with a description of the conventions used in the hierarchic structure of the POWER/VS program, including the following linkages (see Figure 1) and register usage.

- Register conventions which define the general usage of registers within the POWER/VS program.
- Interface linkage, when an external routine passes control to an internal routine, or vice versa.
- Function linkage, when an internal routine invokes a POWER/VS function.
- Service linkage, when any POWER/VS routine invokes a POWER/VS service.

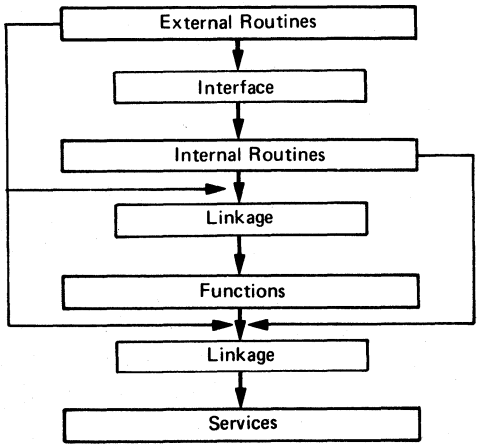


Figure 1. Hierarchic Stucture of POWER/VS

REGISTER CONVENTIONS

This section describes the standard functions and uses assigned to certain of the general purpose registers throughout POWER/VS. The POWER/VS registers are conveniently regarded as running from register 10 to register 9.

Register 10 - Partition Base Register

Register 10 is used to contain the address of the first byte of the POWER/VS partition at all times during POWER/VS execution, and thus secures addressability for the control address table (CAT), task management and task services contained in pages 00 and 01 of the partition. The register is not available for other use.

Register 11 - Task Control Address Register

Register 11 is used to contain the address of the first byte of the TCB for the task currently in control of the central processor, and thus secures addressability for the task parameters and task work space contained in the TCB. The register is not available for other use.

Register 12 - Asynchronous Address Register

Register 12 is used by the task management and page fault appendage routine to retain the return address of a task entering task selection. Since the register contents are liable to asynchronous change, the register is not available for other use.

Register 13 - Save Area Register

Register 13 is used to address the current save area, that is, the storage area in which the general purpose registers are to be saved when an entry linkage is next performed.

Register 14 - Linkage Register

Register 14 is used to contain the linkage address, that is, the address to which return is to be made when an exit linkage is next performed. When not required for this purpose, the register is available for general use.

Register 15 - Entry Point Register

Register 15 is used to address the entry point of the routine to be entered when an entry linkage is performed. This address is normally that of the storage descriptor which precedes the routine to be executed. The register may be conveniently used as the base register for the routine to be executed. When not required for this purpose, the register is available for general use.

Register 0 - Parameter and Work Register

Register 0 is used to pass parameters to and from invoked routines. When not required for this purpose, the register is available for general use.

Register 1 - Parameter and Work Register

Register 1 is used to pass parameters or addresses of parameter lists to and from invoked routines, and in particular to pass command control block addresses to the physical IOCS routines of the DOS/VS supervisor. It also has machine usage when a translate and test instruction is executed. When not required for these purposes, the register is available for general task use.

Register 2 - Linkage and Work Register

Register 2 is used by function and service routines to retain the return address of the requesting task. It also has machine usage when a translate and test instruction is executed. When not required for these purposes, the register is available for general task use.

Register 3 - Resource Address Register

Register 3 is used by functions and services to address resource control blocks. When not required for this purpose, the register is available for general task use.

Registers 4-9

Registers 4-9 are available for general task use.

Each external and internal routine of POWER/VS is coded as a unique control section. Control is initially given by task management to the external routine to be associated with a specific task. This external routine must then establish a linkage to the appropriate internal routine or routines by means of the interface linkage.

Open interface (IPW$OLI macro instruction)

The interface is opened by the creation of a dynamic save area, which is associated with the internal routine. The save area associated with the external routine is located in the TCB. Each save area contains in word 2 the address of the other save area (see Figure 2).

Get/Put linkage (IPW$GLR and IPW$PLR macro instructions)

Linkage is done as follows. The calling task must first establish its return address in register 14, and then save the current contents of registers 14 through 9 in its own save area. It must then load register 13 from word 2 of its save area, thus addressing the other save area. Registers 14, 15, and 2 through 9 are then loaded from the second save area, and a branch made to the address contained in register 14. (Registers 0 and 1 are used for passing parameters and are therefore not reloaded at this time.)

Control has now passed across the interface to the called routine. This routine returns control to the calling routine by repeating the sequence of operations described in the preceding paragraph.

Close interface (IPW$CLI macro instruction)

The dynamic save area associated with the internal routine is released.

(Registers 10 through 13 have the special uses described in "Register Conventions", and are therefore neither saved nor restored during interface linkage.)

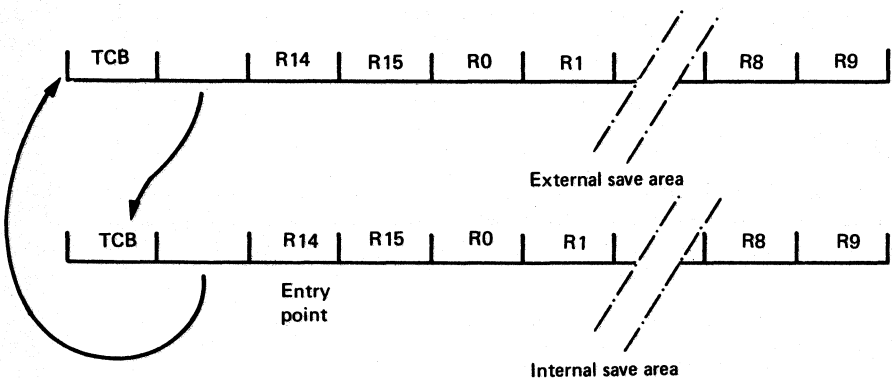


Figure 2. Relationship of Internal and External Save Area

FUNCTION LINKAGE

Each POWER/VS function is coded as a unique control section. The first sixteen bytes of each control section consist of an alphameric control section descriptor. A fullword address constant containing the address of each control section is contained in the control address tables.

Linkage to a function is achieved by loading register 15 with the address of the appropriate control section and then executing a branch and link instruction in the form BAL 14,16(15). Thus, entry is made to the control section at the first byte following the control section descriptor, the task return address being preserved in register 14.

Upon entry, the contents of registers 14 through 9 are saved in words 3 through 14 of the dynamic save area provided by the calling routine and addressed by register 13 (IPW$SAV macro instruction).

On return from a function, registers 14 through 9 are restored from the dynamic save area addressed by register 13. A branch is then made to the return address now contained in register 14 (IPW$RET macro instruction).

(Registers 10 through 13 have the special uses described in "Register Conventions", and are therefore neither saved nor restored during function linkage.)

SERVICE LINKAGE

Each POWER/VS service is coded as a unique routine contained in the nucleus phase (IPW$$NU).

Linkage to a service is based on the use of registers 0 through 3. In most cases register 2 acts as a branch-and-link register.

Registers 0 and 1 are often used to pass parameters between calling routine and the invoked service. Figure 3 shows the various usages of the registers 0 through 3.

The service macros are used to address the services via a service routine branch table located in the CAT in the nucleus phase.

Macro	R0		R1		R2		R3		Other
	Before	After	Before	After	Before	After	Before	After	
IPW$ATT	return		TCB	TCB					
IPW$DET	ECB		TCB						
IPW$WFI									
IPW$WFO									
IPW$WFL							RCB		
IPW$WFM			ECB						
IPW$WFQ			list						R12=return address
IPW$WFC			CCB or						
IPW$WFS			ECB						
IPW$WFD									
IPW$FCH	note [1]								
IPW$RSR					return		RCB		
IPW$RLR					return		RCB		
IPW$RSW	note [2]	real address	size	virtual address	return				
IPW$RLW		zero	virtual address	zero	return				
IPW$WTO					return				
IPW$WTR					return				
IPW$RDQ	return		DRW						
IPW$RDD	return		DRW						
IPW$WTQ	return		DRW						
IPW$WTD	return		DRW						
IPW$WTT	return		TRW						
IPW$RDT	return		TRW						
IPW$CTT	return		TRW						
IPW$RDC				TOD	return				
IPW$VDA					return				R6=PDB R8=CCW address

Figure 3. Contents of registers when a service is invoked

[1] Displacement within DOS/VS task selection 'TRTMK' field of the byte representing the use of the required area (PTA/LTA).

[2] 8 - do not return if no work space can be allocated (IPW$WFL SCB/ECB)
0 - return (R0=zero, R1=ECB in SCB) if no work space available.

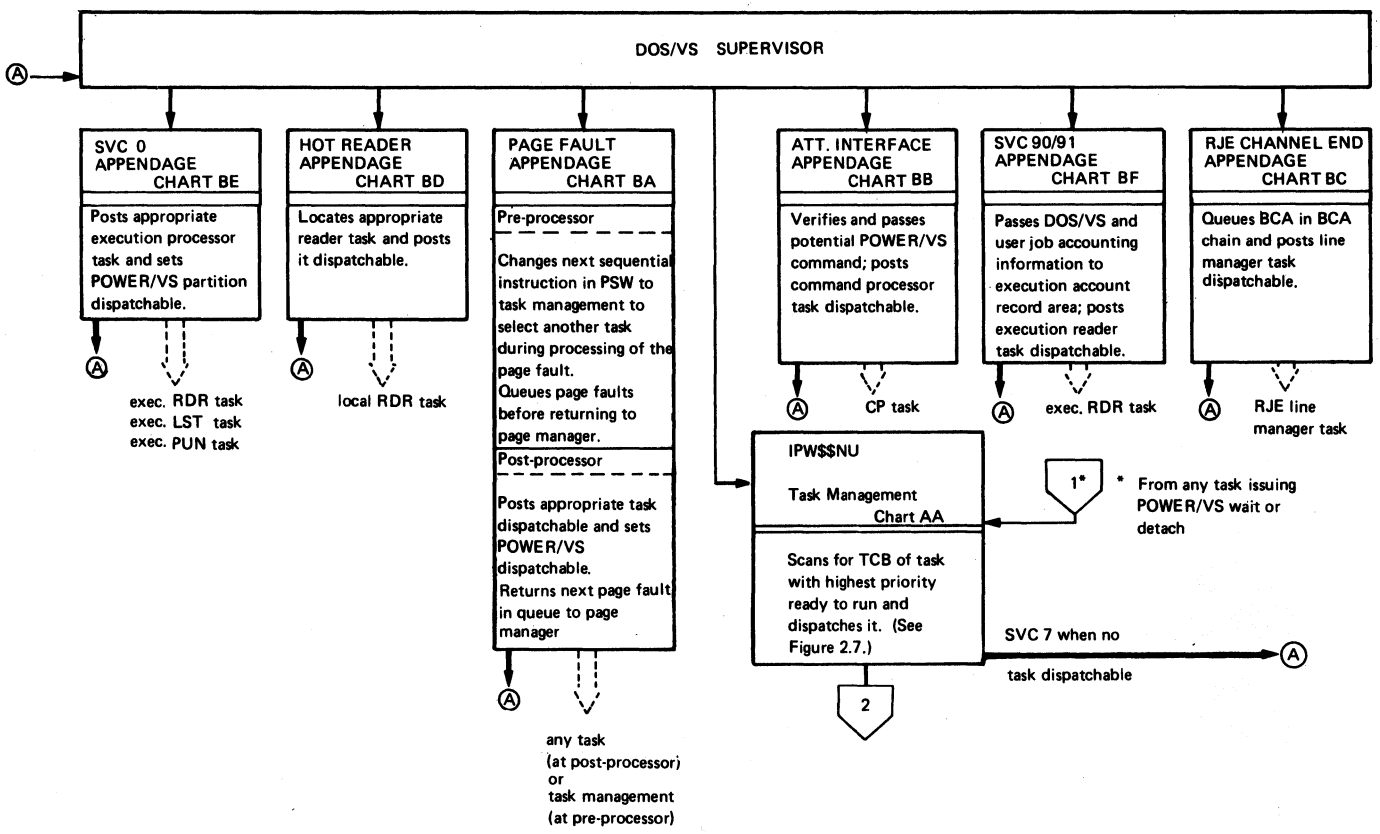


Figure 4. Interfaces and Task Structured References (Part 1 of 5)

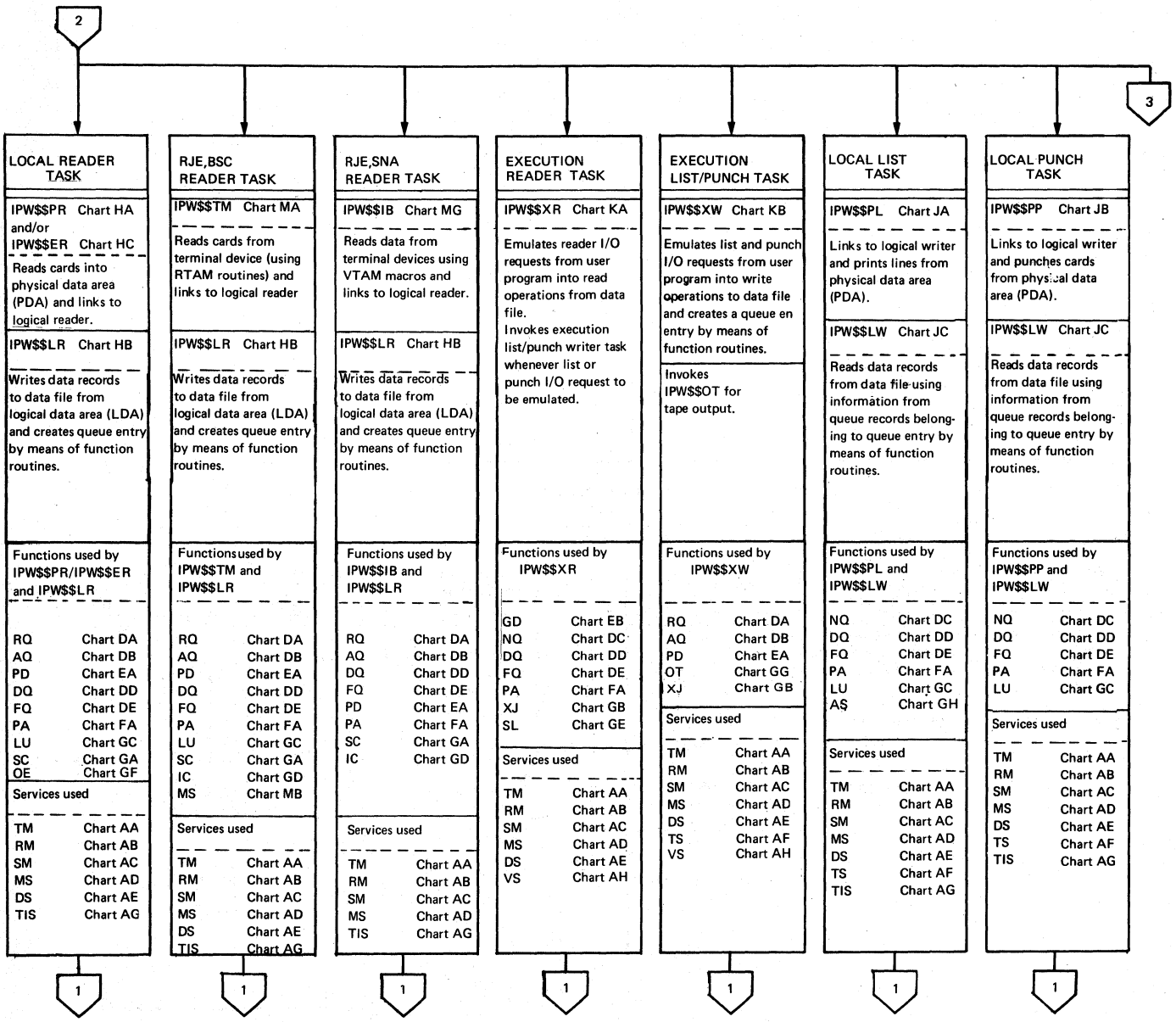


Figure 4. Interfaces and Task Structured References (Part 2 of 5)

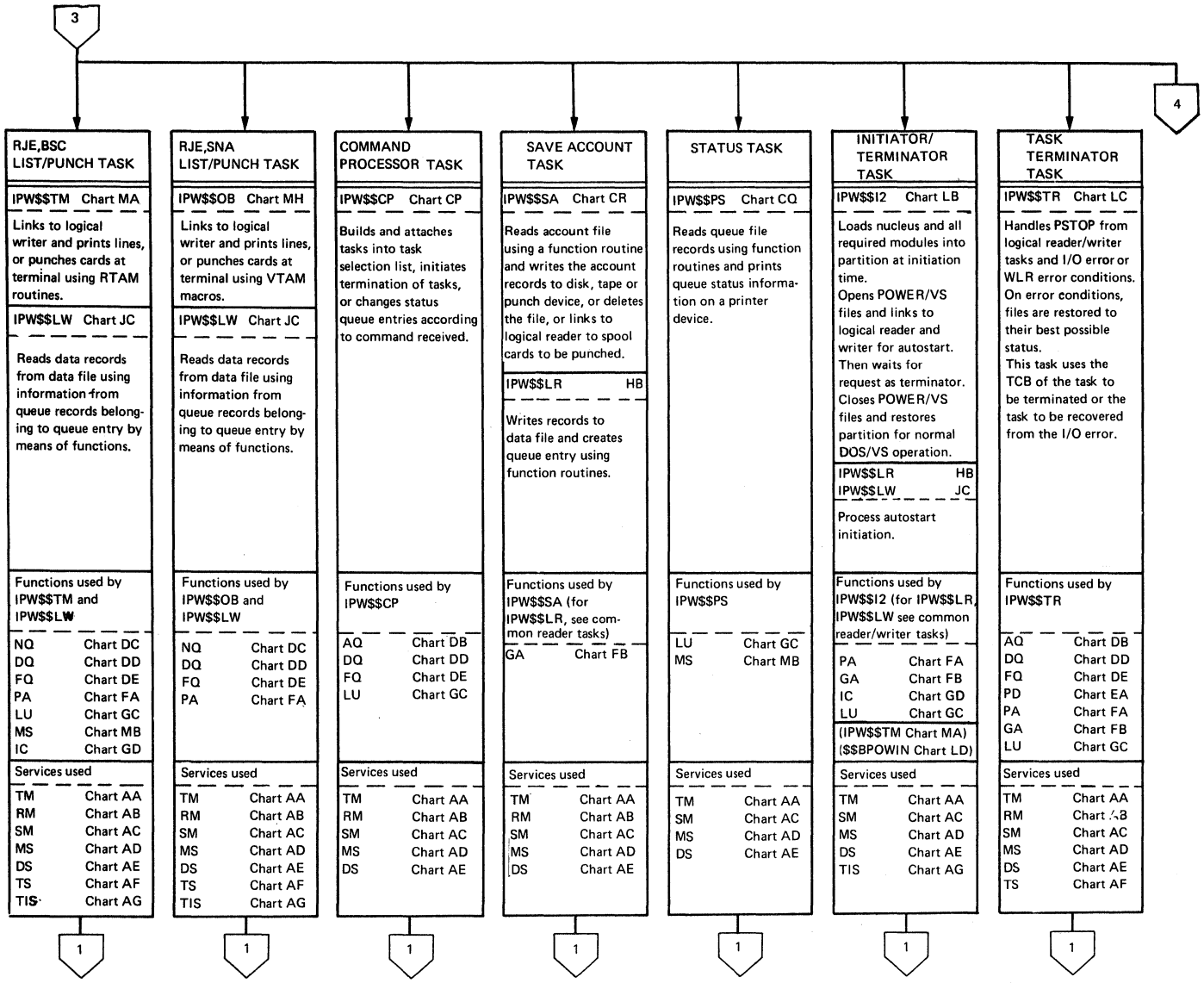


Figure 4. Interfaces and Task Structured References (Part 3 of 5)

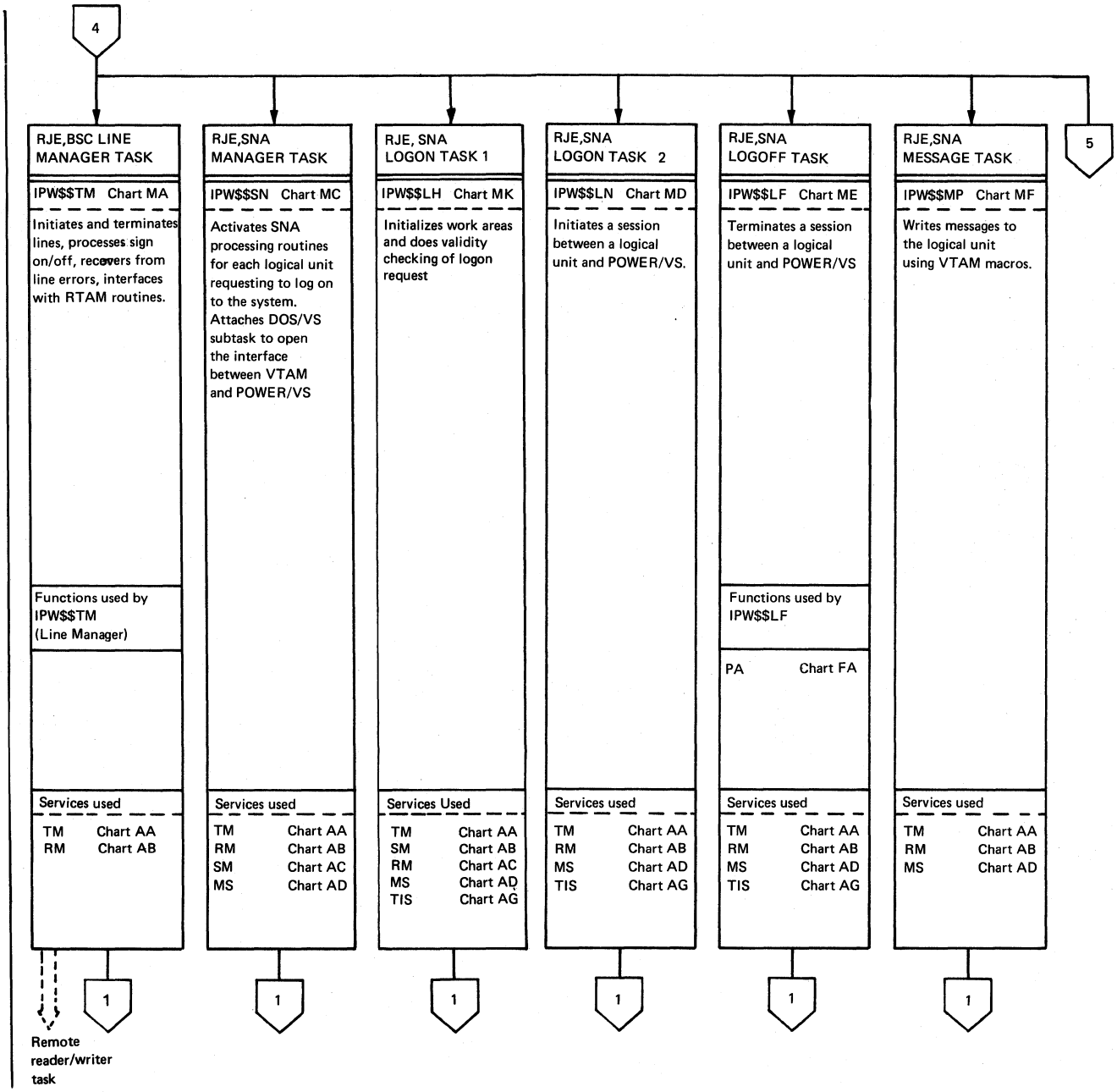


Figure 4. Interfaces and Task Structured References (Part 4 of 5)

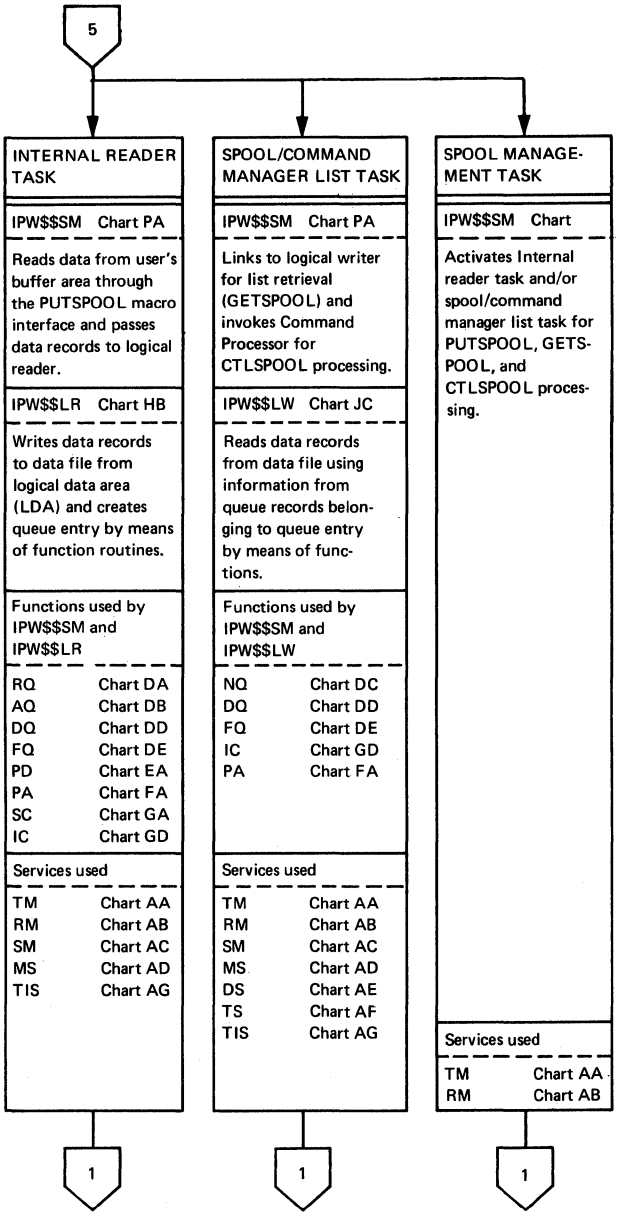
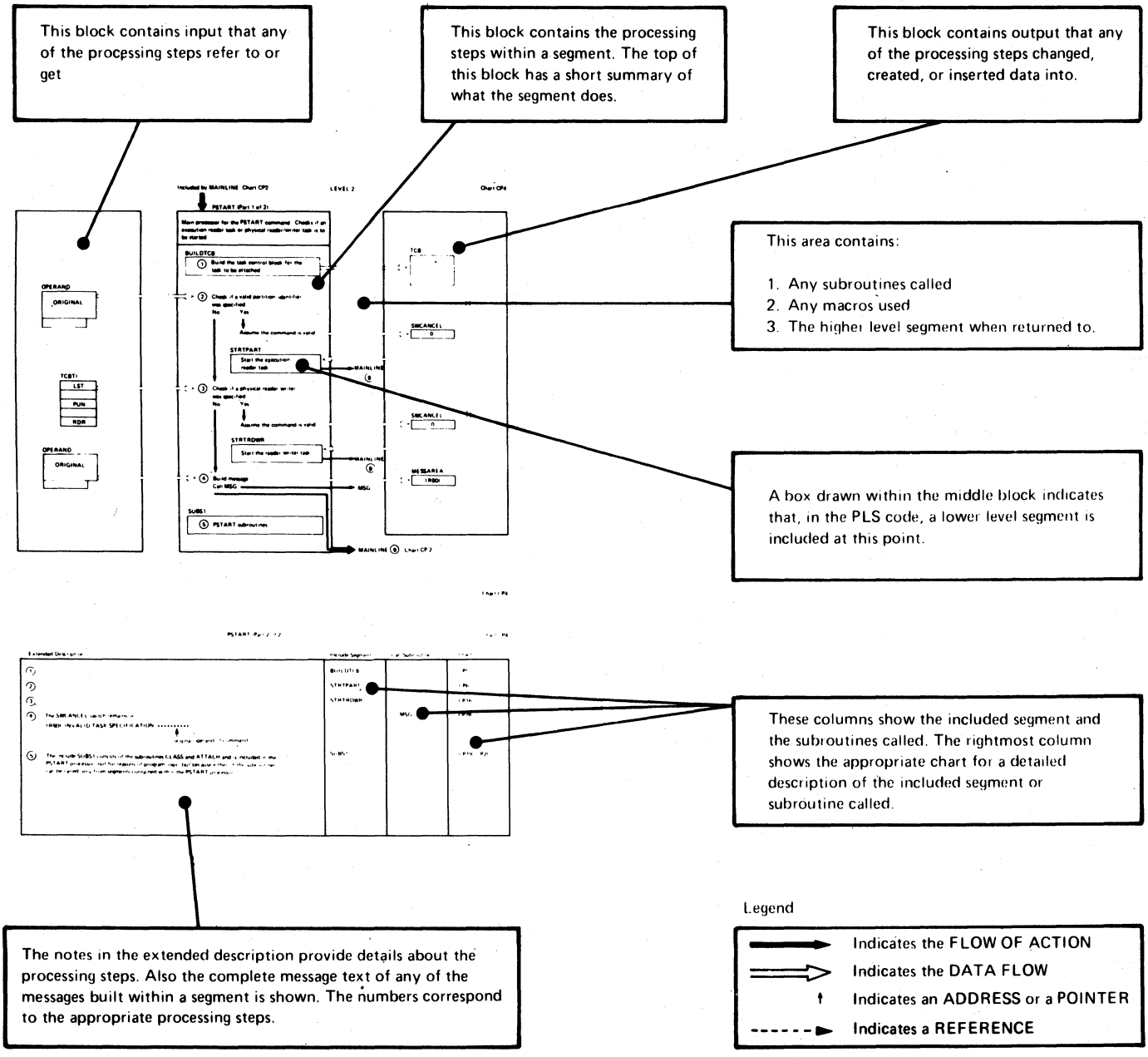


Figure 4. Interfaces and Task Structured References (Part 5 of 5)

UNDERSTANDING POWER/VS HIPO CHARTS

The following routines are shown in form of HIPO charts: Command Processor, Print Queue Status, and RJE/SNA. They are coded in several levels, and each higher-level segment may include one or more lower-level segments. For each phase, the first chart is an overview of all segments contained in the phase. The contents of the charts is explained below.

Each chart has four major blocks: an input block to the left, a processing block in the middle, an output block to the right and an extended description block at the end of the chart. (See figure below).



POWER/VS NUCLEUS (IPW$$NU): SERVICES

Chart AA00: IPW$$NU - POWER/VS Nucleus Services, General Flow and Macro Calls

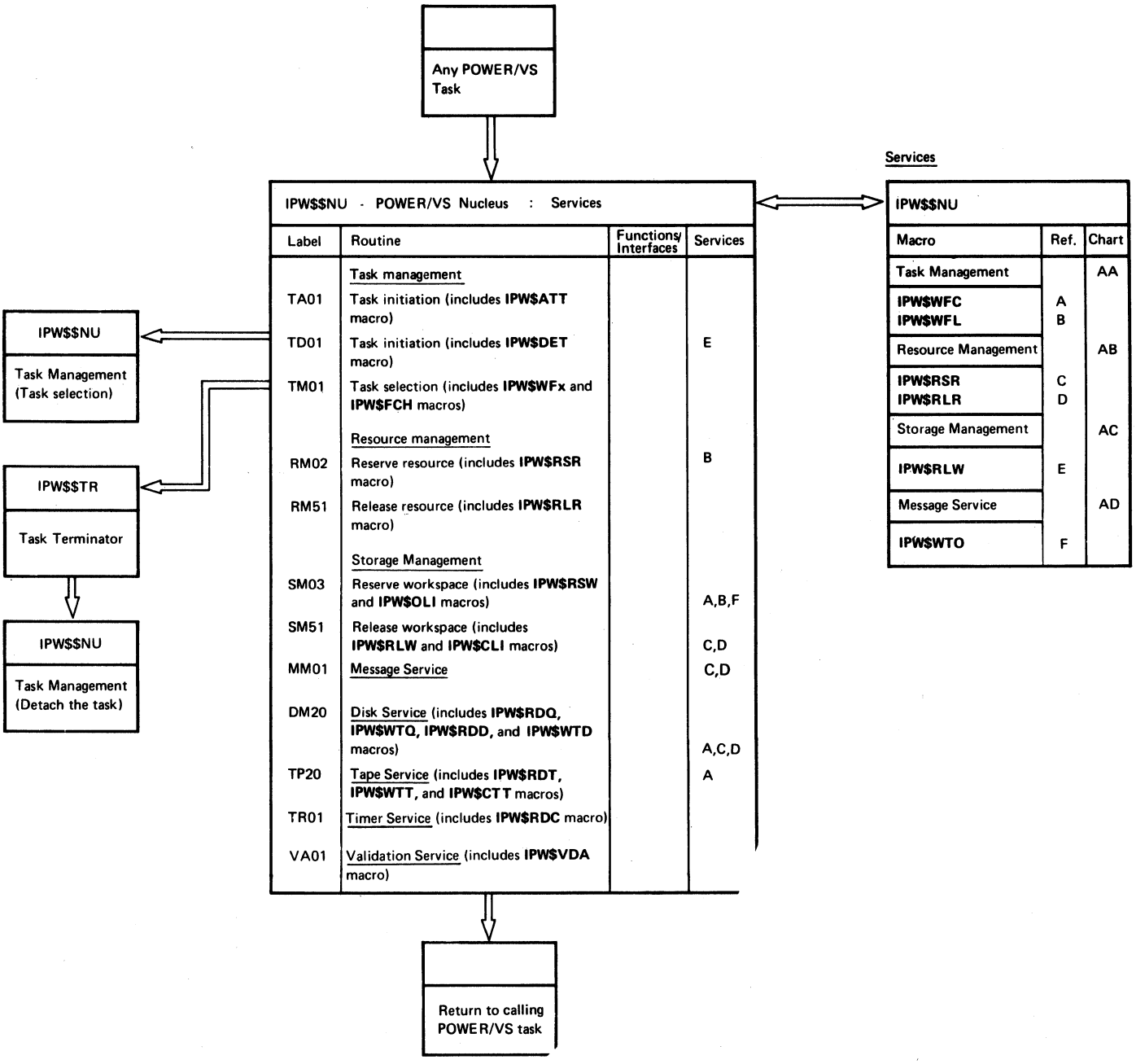


CHART AA: IPW$$NU - TASK MANAGEMENT (3 PARTS)

Labels	Chart AA01: IPW$$NU - POWER/VS Nucleus, Task Management	Modified Data Fields	Reg. Usage	Calls
TMSD	The first 16 bytes constitute the section descriptor:			
	'WCB V10M0 '			
	Task Initiation			
	Registers at entry:			
	0: Return address to caller		R0	
	1: Address of TCB		R1	
	3: Address of routine to be entered (CSECT)		R3	
TA01	Set task dispatchable Initialize registers in task save area	TCSF(IPW$DTC)		
	15: CSECT address	TCRF(IPW$DTC)	R15	
	9: CSECT address	TCR9(IPW$DTC)	R9	
	12: Entry point address	TCRC(IPW$DTC)	R12	
TA02	Scan task control address table in CAT and match task identifier prefix of the new task (first character)			
	3: Task control address table		R3	
	2: Preceding task address.		R2	
TA06	Store the new TCB address in the appropriate entries of the task index table.			
TA10	Retrieve next pointer from preceding task and set task selection list pointers in new TCB.	TCTN(IPW$DTC) TCTP(IPW$DTC)		
	Store address of new TCB in preceding TCB.	TCTN(IPW$DTC)		
	Store address of new TCB in next TCB.	TCTP(IPW$DTC)		
	Increment number of current tasks.	NRTC(IPW$DPA)		
	Update maximum of previous NRTC if higher.	NRTH(IPW$DPA)		
TA20	Return to caller.		R2	
	Task Termination			
	Registers at entry:			
	0: Either zero or address of an ECB.		R0	
	1: Address of TCB		R1	
TD01	Remove TCB from selection list:			
	Store address of next TCB in previous TCB.	TCTN(IPW$DTC)		
	Store address of previous TCB in next TCB.	TCTP(IPW$DTC)		

Labels	Chart AA02: IPW$$NU - POWER/VS Nucleus, Task Management		Modified Data Fields	Reg. Usage	Calls
TD02	Store the previous TCB address in the appropriate entries of the task index table.				
TD05	Post ECB, if available. Decrement the number of tasks.		NRTC(IPW$DPA)		
	If no more than two tasks in existence, check for termination in progress: If so, post terminator ECB.		TCGW(IPW$DTC)		
TD06	Release work space occupied by TCB.				IPW$RLW Chart AC
	Enter task selection..... >	TM02			
	Task Selection				
	Register at entry:				
	12: address of next instruction to be executed when task is next dispatched.			R12	
TM01	If the task is in invalid 'R' state, return to reload the original state value because a page fault has occurred after the I/O interrupt.			R12	
TM01A	Save task registers 12 through 9		TCTR(IPW$DTC)		
TM02	inclusive unpost POWER/VS master ECB.		PAEB(IPW$DPA)		
TM10	Scan the task selection list:				
	11: Addresses TCB 12: Addresses selection routine.			R11 R12	
TM20	1. W state: If POWER/VS master ECB is posted..... >	TM02			
	Otherwise, wait for posting of it (SVC 7).				
TM30	2. L state: If lockword of the resource zero, enter dispatch routine..... >	TM90			
	Otherwise, address next TCB.... >	TM10			
TM50	3. M and Q state: Scan control block list for posting of the traffic or event bit. If found, store address of relevant control block in register 1 in task save area and enter dispatch routine..... >	TM90	TCR1(IPW$DTC)		

Labels	Chart AA03: IPW$$NU - POWER/VS Nucleus, Task Management		Modified Data Fields	Reg. Usage	Calls
TM55	4. F state: If the transient area is available, (selection mask in supervisor selection resource table TRTMASK indicates this) enter dispatch routine..... >	TM90			
	Otherwise, address next TCB.... >	TM10			
TM80	5. C and S state: If traffic or event bit posted in control block:				
	Test if unrecoverable error has occurred. If not, enter dispatch routine..... >	TM90			
	Test for RJE. If line manager, ignore the error.				
	If writer task (LST or PUN), then enter dispatch routine..... >	TM90			
	Test for I/O error during data file access. If so, enter dispatch routine..... >	TM90			
TM82	Set termination indicator (U) in TCB.		TCTT(IPW$DTC)		
	Address terminator phase (IPW$$TR) and set entry point.			R2	
	Store entry point in register 12 (task selection).		TCRC(IPW$DTC)		
	Enter dispatch routine..... >	TM90			
	Otherwise, (no posting) address next TCB..... >	TM10			
	6. I, O, and P state: address next TCB..... >	TM10			
	7. D state: enter dispatch routine..... >	TM90			
TM90	Dispatch the task.				
	Restore task registers.			R12-R9	
	Indicate task is running.		TCSF(IPW$DTC)		
	Restore condition code in PSW.				
	Branch to next instruction (register 12).			R12	

CHART AB: IPW$$NU - RESOURCE MANAGEMENT

Labels	Chart AB01: IPW$$NU - POWER/VS Nucleus, Resource Management		Modified Data Fields	Reg. Usage	Calls
	Registers at entry:				
	2: return address			R2	
	3: address of control block to be reserved.			R3	
RM02	Reserve Resource				
	Wait for the resource.				IPW$WFL Chart AA
	Set ownership (address of TCB) in the control block.				
	Return to caller.			R2	
RM51	Release Resource				
	Reset lockword to zero.				
	Return to caller.			R2	

CHART AC: IPW$$NU - STORAGE MANAGEMENT (3 PARTS)

Labels	Chart AC01: IPW$$NU - POWER/VS Nucleus Storage Management		Modified Data Fields	Reg. Usage	Calls
SCSD	The first 16 bytes constitute the section descriptor: 'SCB V7M0 '				
SM03	Reserve Work Space Registers at entry: 0: return code 1: length of required work space 2: return address Lock SCB (explicitly coded).			R0 R1 R2	IPW$WFL Chart AA
SM01	Save caller registers 14 through 5 inclusive in SCB.		SCTR(IPW$DSC)	R14-R5	
SM10	Address first or next page. If next page pointer not zero..... >	SM40		R3	
SM20	Storage assignment table scan routine: Examine the page control byte: X'00' PFIX this page and format it..... > X'40' X'80' calculate next page address and retry..... > X'C0' no space available..... >	SM30 SM20 SM21			
SM21	Unpost ECB in SCB. Examine the return codes (in register 0): X'00' immediate exit..... > X'04' bypass message and wait..... > X'08' issue message and wait..... >	 SM26 SM24 SM22	SCEB(IPW$DSC)	 R0	
SM22	Issue warning message to operator 1Q59I WAITING FOR REAL STORAGE Reset return code to X'04'. Update count of tasks waiting for storage.		 SCR0(IPW$DSC) NRTW(IPW$DPA)		IPW$WTO Chart AD
SM24	Restore caller registers. Save space length. Unlock SCB. Wait for posting of ECB in SCB. Restore space length and retry..... >	 SM01	 SCLK(IPW$DSC)	R14-R5 R3 R1	 IPW$WFC Chart AA

CHART AD: IPW$$NU - MESSAGE SERVICE (2 PARTS)

Labels	Chart AD01: IPW$$NU - POWER/VS Nucleus, Message Service		Modified Data Fields	Reg. Usage	Calls
MMSD	The first 16 bytes constitute the section descriptor: 'MMB V10M0 '				
	The message control block is defined via the IPW$DMM macro. (Refer to Section 5: "Data Areas" for a description of the layout of the message control block.)				IPW$DMM
	Local Message Service				
MM01	Save the return address.		TCRG(IPW$DTC)	R2	
	Lock the local message control block.				IPW$RSR Chart AB
	Save registers 14 through 9.		MMSV(IPW$DMM)	R14-R9	
	Load the address of the message module (IPW$$MS) into register 15 and branch and link to perform the message service..... >	20(RF)		R15	
	Upon return from IPW$$MS, restore registers 14 through 9.			R14-R9	
	If the hold operand was specified in the message request word (TCMW), branch to..... >	MM25			
	Unlock the local message control block.				IPW$RLR Chart AB
MM25	Restore the return address.			R2	
	Return to calling routine.			R2	
	Remote Message Service				
MM51	Save the return address.		TCRG(IPW$DTC)	R2	
	Check for ADDNRM request. If not, branch to..... >	MM53			
	Check for SNA. If not branch to... >	MM53			
	Lock the SNA control block.				IPW$RSR Chart AB
MM53	Lock the remote message control block.				IPW$RSR Chart AB
	Set up register 3 as a base address for the remote message control block (MSCB).			R3	
	Save registers 14 through 9.		MSSV(IPW$DMS)	R14-R9	

Labels	Chart AD02: IPW$$NU - POWER/VS Nucleus, Message Service		Modified Data Fields	Reg. Usage	Calls
	Load the address of the message module (IPW$$MS) into register 15 and branch and link to perform the message service..... >	16(RF)		R15	
	Upon return from IPW$$MS, restore registers 14 through 9.			R14-R9	
	Check for SNA. If not, branch to.. >	MM55			
	Check if the request is from the command processor of the SNA manager, If so, branch to..... >	MM55			
	Unlock the SNA control block.				IPW$RLR Chart AB
MM55	Unlock the remote message control block.				IPW$RLR Chart AB
	Restore the return address.			R2	
	Return to calling routine.			R2	

CHART AE: IPW$$NU - DISK SERVICE

Labels	Chart AE01: IPW$$NU - POWER/VS Nucleus, Disk Service		Modified Data Fields	Reg. Usage	Calls
	Registers at entry:				
	1: Disk request word (DRW) 2: Return address			R1 R2	
DM20	The first byte of DRW is used to locate MCB (via module control block table in CAT).				
DM21	Lock MCB.		MCLK(IPW$DMC)	R3	IPW$RSR Chart AB
	Save disk request word address		MC$T(IPW$DMC)		
	Address CCB and wait for completion of previous I/O			R1	IPW$WFC Chart AA
	If error has occurred (I/O error, wrong length), branch to..... >	DM25			
	Otherwise branch to continue..... >	DM30			
DM25	Locate the TCB which issued the I/O in error.			R2	
	If TCB cannot be found, branch to.. >	DM30			
	Set termination indicator (U) in TCB.		TCTT(IPW$DTC)		
	Set proper function.		TCFT(IPW$DTC)		
	Track byte (to G or P) address terminator phase (IPW$$TR) and set entry point.			R1	
	Store entry point in register 12 (task selection).			R12	
	Give task ownership of MCB concerned.		MCLK(IPW$DMC)		
	Branch to..... >	DM21			
DM30	Initialize channel program:				
	• Move seek address • Move command code and data address • Move record number to sector • Save virtual address of buffer • Save TCB addresses of owner		MCSA(IPW$DMC) MCRW(IPW$DMC) MCSE(IPW$DMC) MCTV(IPW$DMC) MC$T(IPU$DMC)		
	Execute channel program (SVC0).				
	If this is not the data file, branch to..... >	DM60			
	If the last request was a double buffer request, the buffer is freed:				
	Get buffer address (if any) and release buffer			R1	IPW$RLW Chart AC
	If this is not a double buffer request, branch to..... >	DM50			

CHART AF: IPW$$NU - TAPE SERVICE

Labels	Chart AF01: IPW$$NU - POWER/VS Nucleus, Tape Service		Modified Data Fields	Reg. Usage	Calls
	If writer task (LST or PUN), branch to..... >	DM65			
	If CCB is complete, branch to..... >	DM50			
	Attempt to get 2nd buffer.				IPW$RSW Chart AC
	If no buffer is available, branch to>	DM50			
	Set up new buffer addresses.		TCDA(IPW$DTC)	R0,R1	
	Indicate buffer cleared (N). Save old buffer address. Branch to..... >	DM65	TCDB(IPW$DTC) MCTV(IPW$DMC)		
DM50	Restore registers.			R0,R1, R3	
DM60	Indicate request handled..... >		MC$T(IPW$DMC)		IPW$WFC Chart AA
	Wait for completion.				
	If wrong length, or unrecoverable I/O error occurred, exit to task selection..... >	TM82			
DM65	Unlock MCB.		MCLK(IPW$DMC)	R3	IPW$RLR Chart AB
	Task selection is forced to allow any higher priority task waiting for the resource to get it at that point.				IPW$WFD Chart AA
	Return to calling routine.			R2	
	Registers at entry:				
	0: Return address 1: Address of tape request word (TRW)			R0 R1	
TP20	Retrieve tape control block (TBB) address from TRW.			R3	
	Prepare TBB for execution:				
	• Copy command code and data address from TRW. • Copy data length.		TBCH(IPW$DTB) TBCH+6(IPW$DTB)		
	Set address of TBB in register 1.			R1	
	Execute channel program (SVC 0) and wait for completion.				IPW$WFC Chart AA
	If wrong length, exit to task selection..... >	TM82			
	Return to caller.			R2	

CHART AG: IPW$$NU - TIMER SERVICE

Labels	Chart AG01: IPW$$NU - POWER/VS Nucleus, Timer Service		Modified Data Fields	Reg. Usage	Calls
	Register at entry:				
	0: Return address			R0	
TR01	Read time-of-day clock (GETIME STANDARD) (register 1 contains time in packed decimal format).			R1	
	Locate partition communication region.			R2	
	Address disk management block.			R3	
	Copy date into master record in DMB.		MRDY(IPW$DQC)		
	Return to caller.			R2	

CHART AH: IPW$$NU - VALIDATION SERVICE

Labels	Chart AH01: IPW$$NU - POWER/VS Validation Service	Modified Data Fields	Reg. Usage	Calls
	Registers at entry:			
	2: Return		R2	
	6: Partition control block		R6	
	8: Command control word address		R8	
	Exits:			
	0(R2) error exit 4(R2) normal exit			
VA01	The limits of the user real partition, or if the partition is running virtual, the limits of the virtual partition are retrieved from the boundary box in the DOS/VS supervisor.		R3	
	The CCW address is checked to determine whether it is on a double word boundary and whether it is in the user partition or in the LTA. If the CCW is not at a valid address, return to caller via register 2.		R2	
VA05	The data area address and the length are obtained from the CCW.		R0,R1	
	If the length is higher than 512, return to caller via error exit.		R2	
	Calculate end address of data area.		R1	
	If the length is zero and it is not a TIC operation, return to caller via error exit.		R2	
VA07	If it is a read operation with no data transfer, return to caller via normal exit.		R2	
VA09	The addresses of the user data area are matched against the limits of the partition, and if they are not within the partition, against the limits of the logical transient area.			
VA12	If the addresses of the user data area are higher than the partition limits, they are matched against the limits of the SVA:		R3	
	If it is a read/sense operation and the addresses are outside of the partition or LTA, return to caller via error exit.		R2	

Labels	Chart AH01: IPW$$NU - POWER/VS Validation Service		Modified Data Fields	Reg. Usage	Calls
	If the addresses are invalid, return to caller via register 2 (error exit).			R2	
	If the addresses are valid, return to caller via register 2 with a displacement of 4.			R2	

POWER/VS NUCLEUS (IPW$$NU): APPENDAGES

CHART BA: IPW$$NU - PAGE FAULT APPENDAGE (2 PARTS)

Labels	Chart BA01: IPW$$NU - POWER/VS Nucleus, Page Fault Appendage	Modified Data Fields	Reg. Usage	Calls
	Registers at entry:			
	7: Return address		R7	
	8: Entry address		R8	
	13: Page fault request word		R13	
PF01	Page Fault Pre-processor			
	Calculate base address for nucleus code.		R10	
	Address TCB of the task.		R14	
	The status of the failing task is saved.		R14	
	• Address of next sequential instruction.	TCRC(IPW$DTC)		
	• Task registers register 13 through register 9 inclusive.	TCRD(IPW$DTC)		
	• Condition code.	TCTR(IPW$DTC)		
	Save page fault request in TCB.	TCPF(IPW$DTC)		
	Indicate wait for page-in.	TCSF(IPW$DTC)		
	Modify PSW in partition save area to force entry into POWER/VS task selection.	PSAD(IPW$DPA)		
	If page fault in progress, zero page fault register request word.		R13	
	Otherwise, save current page fault request and TCB address.	PFCF PFCT		
	Return to supervisor.		R7	
	Page Fault Post-processor			
PF03	Post POWER/VS partition dispatchable.			
	Calculate base address for nucleus code.		R10	
	Post POWER/VS master ECB.	PAEB(IPW$DPA)		
	Address TCB of the task (from save area).		R14	
	Clear page fault request in TCB.	TCPF(IPW$DTC)		
	Set the task dispatchable.	TCSF(IPW$DTC)		

Labels	Chart BA02: IPW$$NU - POWER/VS Nucleus, Page Fault Appendage		Modified Data Fields	Reg. Usage	Calls
PF04	Scan task selection list for any other outstanding page fault: • Register 13 loaded with page fault request (if any) or zero. • Register 14 loaded with address of TCB (if found) or zero. The page fault request (register 13) and TCB address (register 14) are saved. Return to supervisor.		 PFCF PFCT 	 R13 R14 R7	

CHART BB: IPW$$NU - ATTENTION INTERFACE APPENDAGE

Labels	Chart BB01: IPW$$NU - POWER/VS Nucleus, Attention Interface Appendage	Modified Data Fields	Reg. Usage	Calls
	Registers at entry:			
	5: Address of command		R5	
	7: Length (minus 1) of command code		R7	
	8: Normal return address		R8	
	12: Error return address		R12	
AI00	Save registers 12 through 11 inclusive.	AISV	R12-R11	
	Set buffer address.		R6	
	Set base register.		R15	
	Calculate base address for nucleus code.		R10	
	If the command code is incorrect:			
AI10	• Restore registers • Take error exit (register 12).		R12-R11	
	Take error exit (register 12).			
AI20	Address command processor TCB.		R11	
	If the command processor is active:			
	• Restore registers • Take 'in use' exit (10(register 12))		R12-R11	
	Address command processor control block:		R8	
	Set RJE-ID to zero (local).	CPID(IPW$DTC)		
	Clear command area.	CPCM(IPW$DTC)		
	Move command code in command area.	CPCM(IPW$DTC)		
	Scan for operand string and move it to the operand area (if any).	CPOP(IPW$DTC)		
AI30	Clear sequence number and remainder.	CPNO(IPW$DTC) CPEA(IPW$DTC)		
	Initialize command processor TCB:			
	• Clear register save area • Store address of CCB • Set base register • Set entry address • Set task dispatchable.	TCTR(IPW$DTC) TCR7(IPW$DTC) TCR8(IPW$DTC) TCRC(IPW$DTC) TCSF(IPW$DTC)		
	Post POWER/VS partition PIB flag dispatchable.			
	Post POWER/VS master ECB.	PAEB(IPW$DPA)		
	Restore registers and exit via register 8.		R12-R11 R8	

CHART BC: IPW$$NU - RJE,BSC CHANNEL END APPENDAGE

Labels	Chart BC01: IPW$$NU - POWER/VS Nucleus,RJE,BSC Channel End Appendage		Modified Data Fields	Reg. Usage	Calls
	Registers at entry:				
	1: Address of BCA			R1	
	8: Entry address			R8	
	7: Exit address			R7	
CE00	Check for channel end. If not, return to supervisor via register 7.			R7	
	Save registers 6 and 7.				
	Calculate base address for nucleus code and address line manager TCB.			R7,R8	
	Copy address next CCW from CSW.		LASTCCW		
	Set the line manager to dispatchable.		TCEB+2(IPW$DTC)		
CE05	Scan the appendage queue for the last CCB in the chain.				
CE08	Store last CCB in chain.				
	Restore registers 6 and 7.			R6,R7	
	Return to supervisor via register 7.			R7	

CHART BD: IPW$$NU - HOT READER APPENDAGE

Labels	Chart BD01: IPW$$NU - POWER/VS Nucleus, Hot Reader Appendage		Modified Data Fields	Reg. Usage	Calls
	Registers at entry:				
	3: PUB address of the device			R3	
	8: Return address			R8	
	15: Entry address			R15	
HR00	Save registers 9 and 10.		HRSV	R9-R10	
HR10	Scan reader tasks TCBs for a matching device:				
	If found indicate reader device end occurred		TNG2 (IPW$DTC)		
	If found and task is inactive -				
	Set task dispatchable		TCSF(IPW$DTC)		
	Post POWER/VS master ECB		PAEB(IPW$DPA)		
	Post POWER/VS partition PIB flag dispatchable				
HR30	Restore registers and return to supervisor via register 8.			R9-R10 R8	

CHART BE: IPW$$NU - SVC 0 APPENDAGE

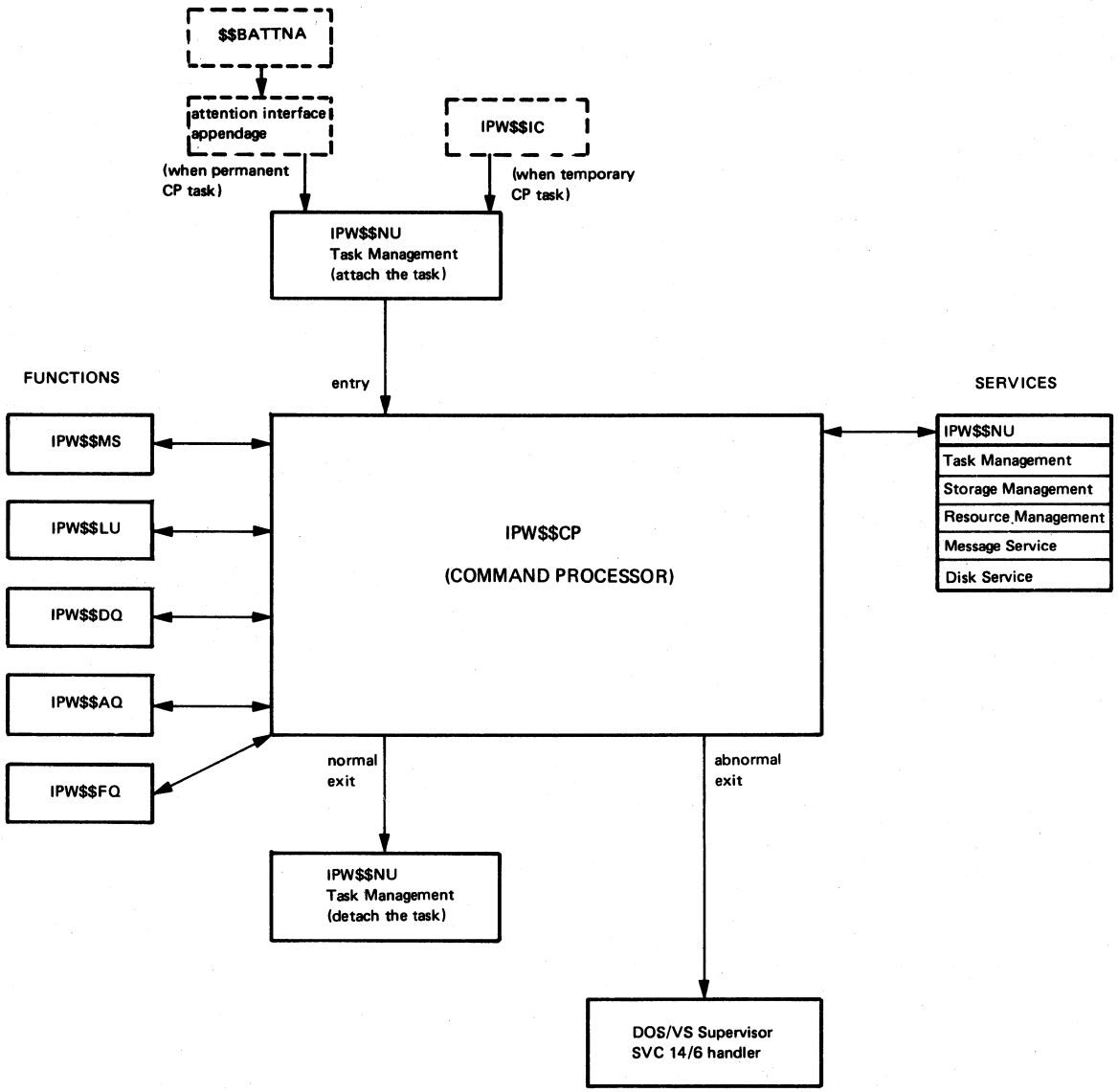
Labels	Chart BE01: IPW$$NU - POWER/VS Nucleus, SVC 0 Appendage		Modified Data Fields	Reg. Usage	Calls
	Registers at entry:				
	0: TIK value			R0	
	1: Address of CCB			R1	
	2: Address of task entry in PDB			R2	
	8: Return address			R8	
	15: Entry address			R15	
SU00	Save register register 10.		SCSA		
	Calculate base address for nucleus code.			R10	
	If not console read operation, branch to..... >	SC12			
SC04	Address message text.			R9	
	If JECL prefix..... >	SC14			
SC11	Restore register register 10, return to supervisor (4(register 8)).			R10	
SC12	If no interception required..... > (disposition N).	SC11			
SC14	If a previous request pending:				
	Restore register 10, return to supervisor (register 8).				
	Store request (address of CCB).		TLCB(IPW$DDE)		
	Store requestor identifier (TIK).		TLRQ(IPW$DDE)		
	Address task control block and post its ECB.		TCEB(IPW$DTC)	R9	
	Post POWER/VS master ECB.		PAEB(IPW$DPA)		
	Post POWER/VS partition dispatchable (PIB flag).				
	Save register 0 through 3 inclusive.		SCS0	R0-R3	
	If job account interface existing, scan the SIO table (if present) in the accounting table for matching device and increment counter.				
	Restore registers and return to supervisor via register 8.			R10 R0-R3	

CHART BF: IPW$$NU - SVC 90/91 APPENDAGE

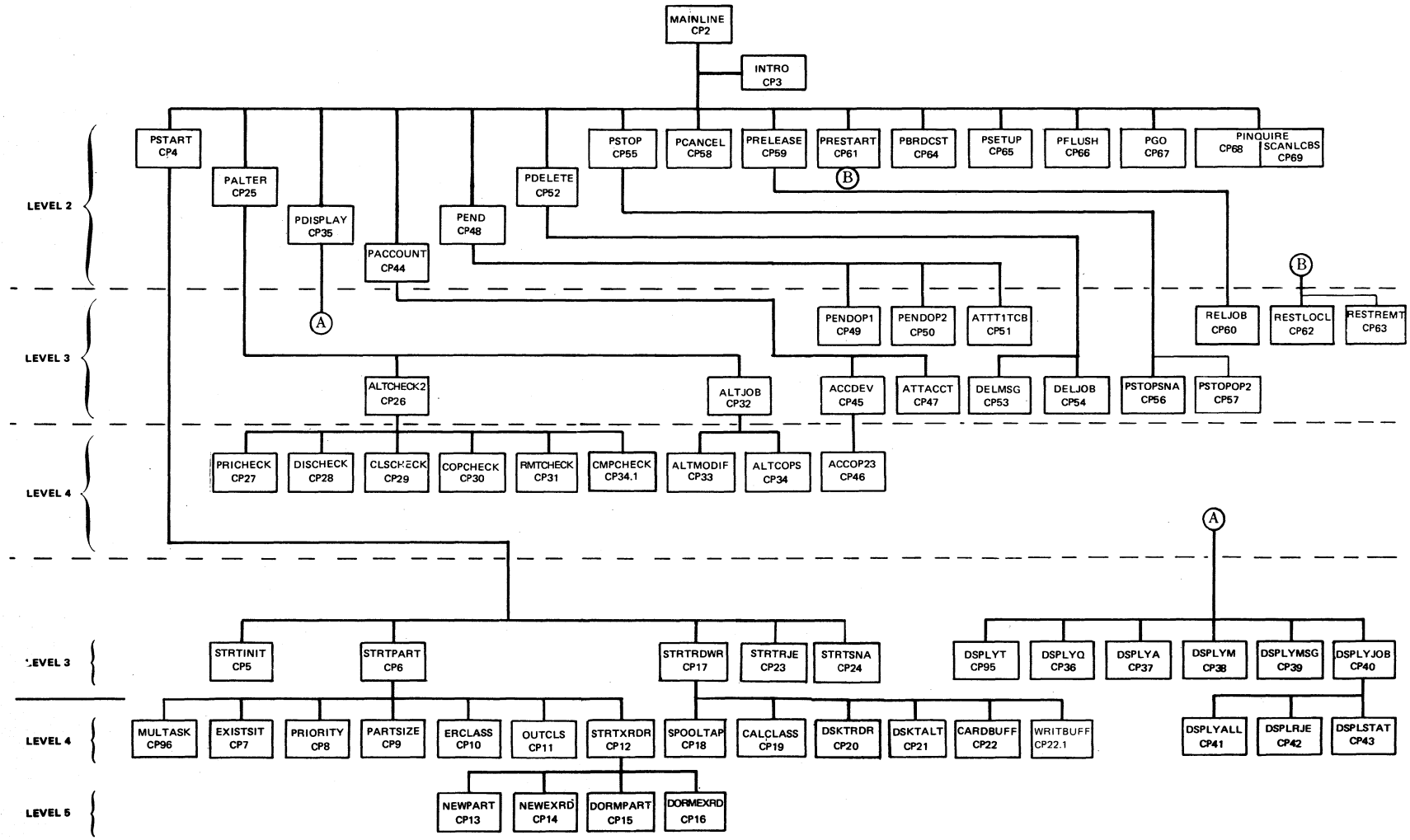
Labels	Chart BF01: IPW$$NU - POWER/VS Nucleus, SVC 90/91 Appendage	Modified Data Fields	Reg. Usage	Calls
	Registers at entry:			
	0: Parameter list (ECB, length and address of account information)		R0	
	8: Return address		R8	
	15: Entry address		R15	
SU90	Save register 10.		R10	
	Calculate base address of nucleus code.		R10	
	Locate PDB via partition communication region.		R1	
	Store address of parameter list in PDB.	TLCB(IPW$DDE)		
	Locate execution reader TCB via PDB.		R1	
	Post its ECB.	TCEB(IPW$DTC)		
	Post POWER/VS master ECB.	PAEB(IPW$DPA)		
	Post POWER/VS partition dispatchable (PIB flag).			
	Restore registers and return to supervisor (register 8).		R10	

CHART CP: IPW$$CP - COMMAND PROCESSOR (185 PARTS)

Chart CP00: IPW$$CP - Command Processor, General Flow and Macro Calls

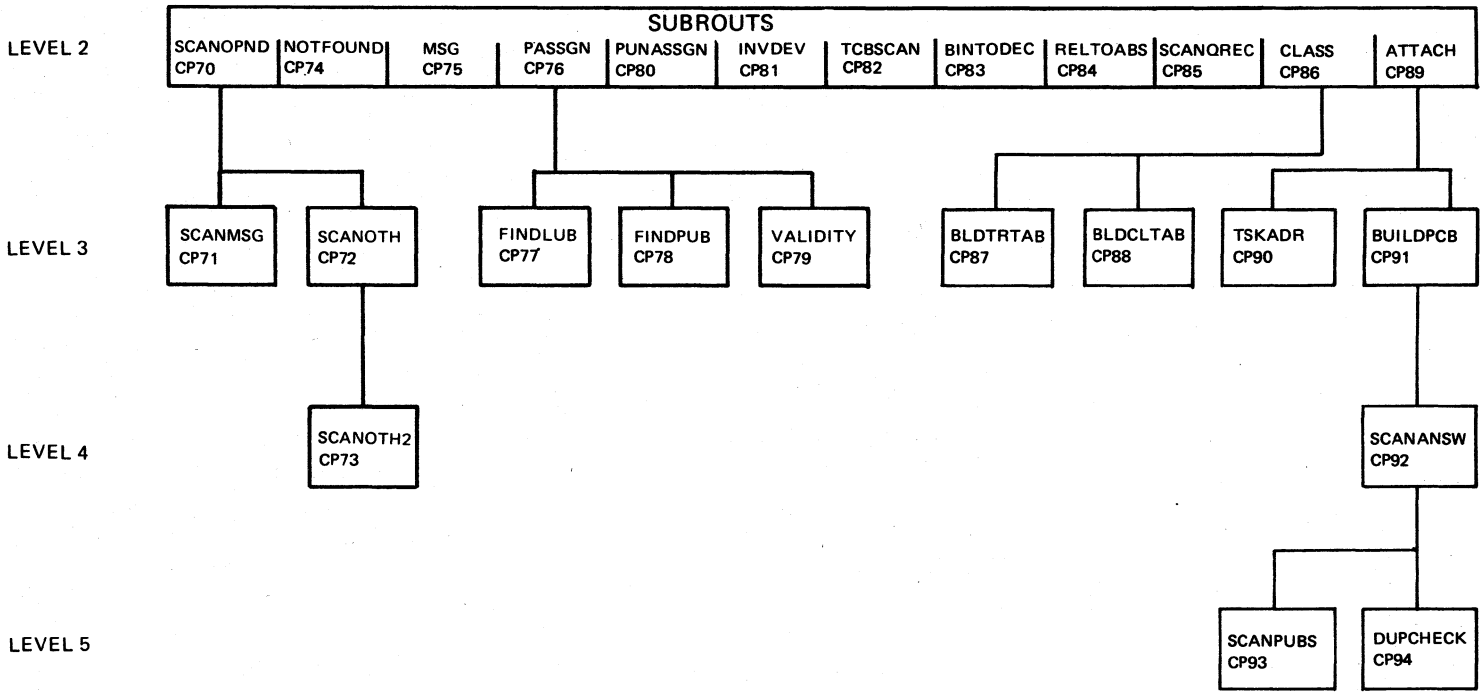


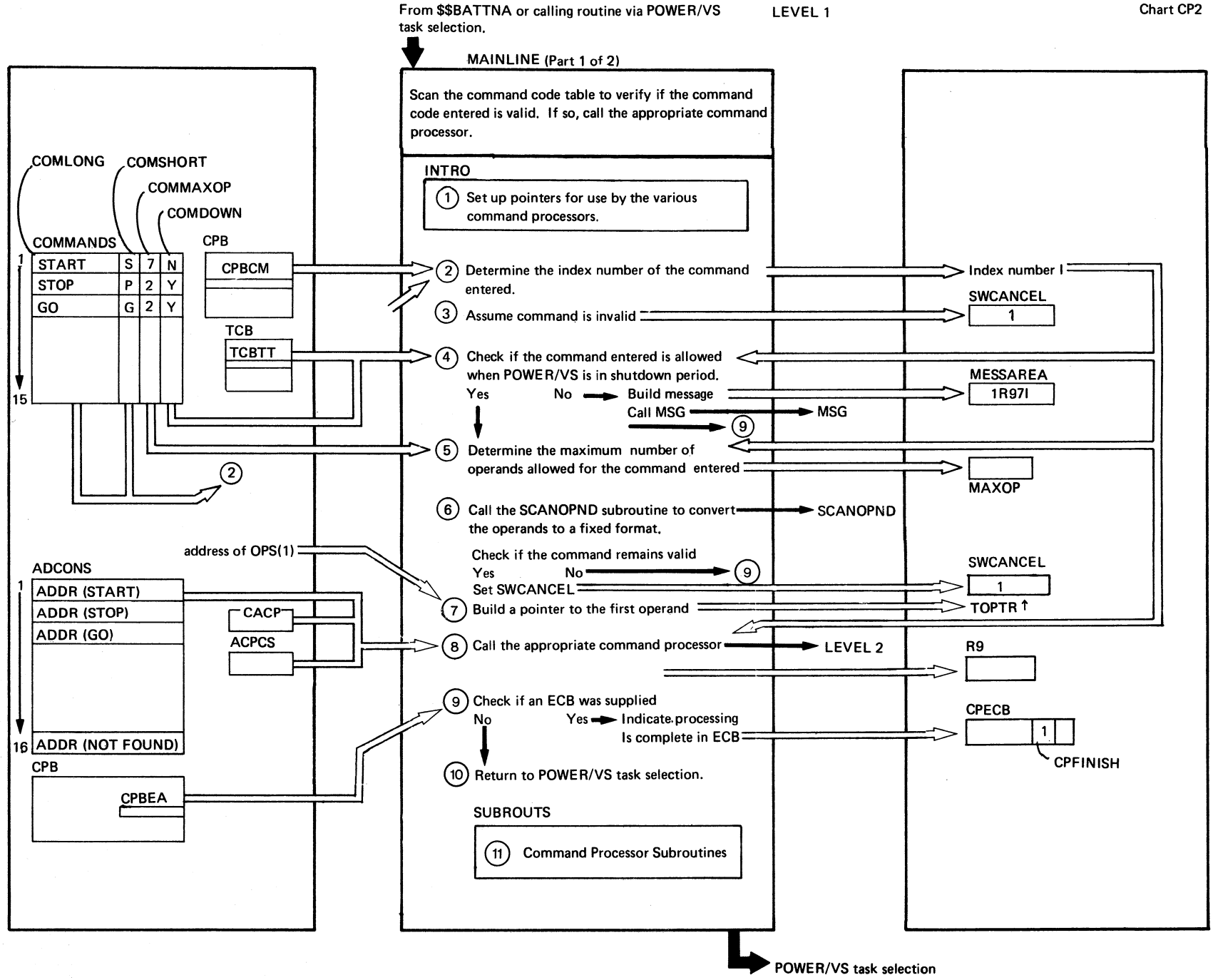
COMMAND PROCESSOR (IPW$$CP) ORGANIZATION (PART 1 OF 2)



Page of SY33-8577-1, Revised November 24, 1977, By TNL SN33-9241

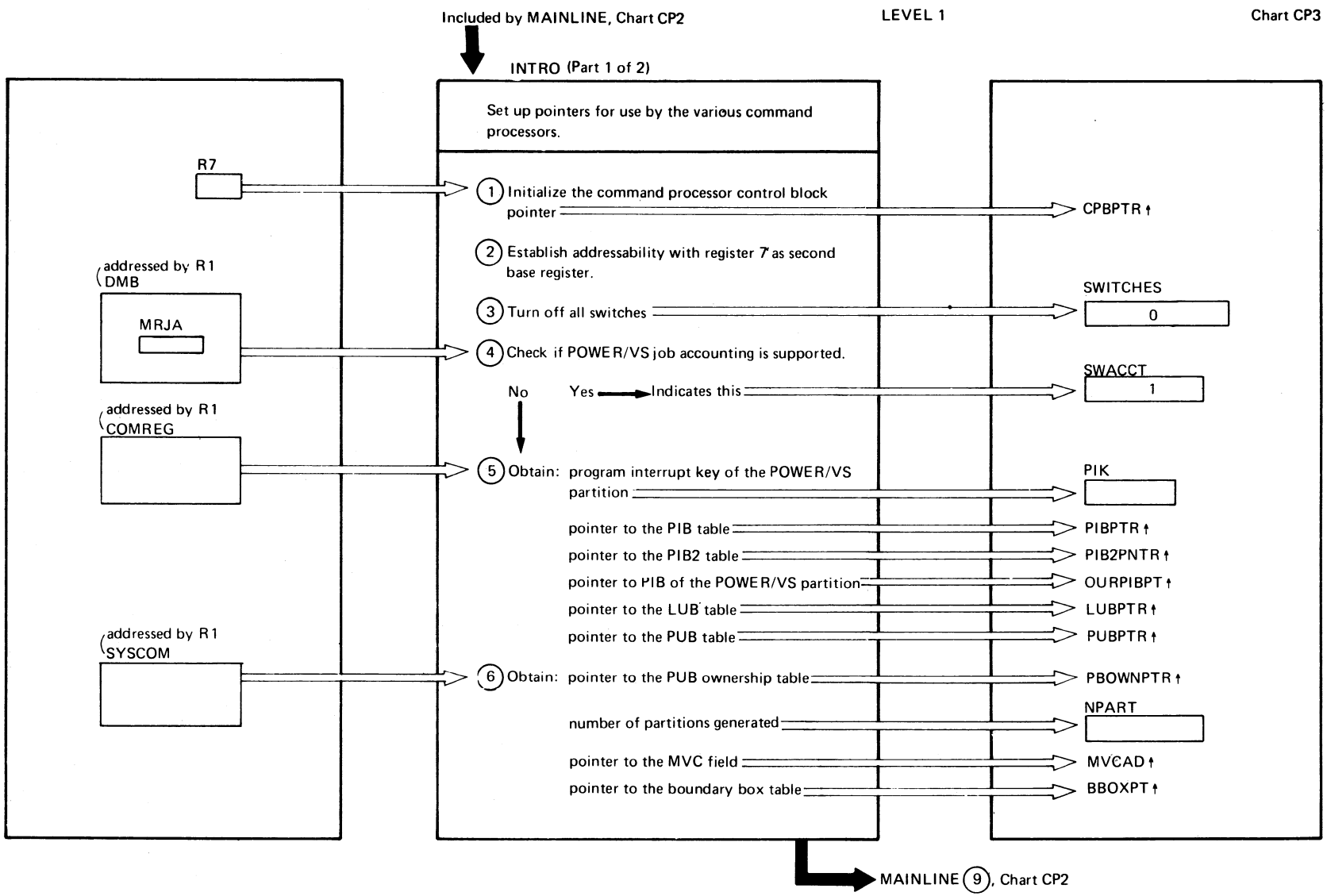
COMMAND PROCESSOR (IPW$$CP) ORGANIZATION, SUBROUTINES (PART 2 OF 2)




From $$BATTNA or calling routine via POWER/VS task selection.
LEVEL 1
Chart CP2
MAINLINE (Part 1 of 2)
Scan the command code table to verify if the command code entered is valid. If so, call the appropriate command processor.
INTRO
1 Set up pointers for use by the various command processors.
COMLONG
COMSHORT
COMMAXOP
COMDOWN
COMMANDS
1
START S 7 N
STOP P 2 Y
GO G 2 Y
15
CPB
CPBCM
TCB
TCBTT
2 Determine the index number of the command entered.
Index number I
3 Assume command is invalid
SWCANCEL
1
4 Check if the command entered is allowed when POWER/VS is in shutdown period.
Yes
No
Build message
Call MSG
MSG
9
MESSAREA
1R97I
5 Determine the maximum number of operands allowed for the command entered
MAXOP
6 Call the SCANOPND subroutine to convert the operands to a fixed format.
SCANOPND
address of OPS(1)
Check if the command remains valid
Yes
No
9
Set SWCANCEL
SWCANCEL
1
ADCONS
1
ADDR (START)
ADDR (STOP)
ADDR (GO)
CACP
ACPCS
7 Build a pointer to the first operand
TOPTR
8 Call the appropriate command processor
LEVEL 2
R9
16
ADDR (NOT FOUND)
CPB
CPBEA
9 Check if an ECB was supplied
No
Yes
Indicate processing Is complete in ECB
CPECB
1
CPFINISH
10 Return to POWER/VS task selection.
SUBROUTS
11 Command Processor Subroutines
POWER/VS task selection
Chart CP2

MAINLINE (Part 2 of 2)

Extended Description	Include Segment	Call Subroutine	Chart
(1) The command processor is a reenterable program coded in PLS, so the PLS/II compiler generates a GETMAIN assembler macro to obtain the dynamic storage required for variables. When the permanent command processor has control, the first page of pageable storage is acquired. When the temporary command processor has control, the IPW$RSW macro is issued.	INTRO		CP3
(2) CPBCM contains a long (COMLONG) or a short (COMSHORT) format command. If a long format command was entered, the preceding 'P' was stripped off by the attention interface. 'I' in output, indicates the index number in the command code table corresponding to the command entered.			
(4) The Termination Type (TCBTT) field in the permanent command processor TCB is addressed via TAOC. When POWER/VS is in a shutdown period, the PEND processor moves an 'E' for stop at end into the TCBTT field. 1R97I COMMAND INVALID DURING SHUTDOWN PERIOD		MSG	CP75
(6) The SCANOPND subroutine converts the operands to fixed format and converts decimal and hexadecimal to binary.		SCANOPND	CP70
(8) CACP points to the start of the command processor at load time. ACPCS points to the start of the command processor at linkage edit time. The difference between the two values is the relocation factor which must be added to the address constant.		LEVEL 2	PSTART CP4 PALTER CP25 PDISPLAY CP35 PACCOUNT CP44 PEND CP48 PDELETE CP52 PSTOP CP55 PCANCEL CP58 PRELEASE CP59 PRESTART CP61 PBRDCST CP64 PSETUP CP65 PFLUSH CP66 PGO CP67 PINQUIRE CP68
(10) The PLS/II compiler generates a FREEMAIN macro. If the permanent command processor has control, the IPW$WFI macro is issued. If the temporary command processor has control, the IPW$RLW macro is issued to release the space obtained by IPW$RSW (see (1)). Then the IPW$DET macro is issued to release the TCB.			
(11)	SUBROUTS		SCANOPND CP70 NOTFOUND CP74 MSG CP75 PASSGN CP76 PUNASSGN CP80 INVDEV CP81 TCBSCAN CP82 BINTODEC CP83 RELTOABS CP84 SCANQREC CP85 CLASS CP86 ATTACH CP89


Included by MAINLINE, Chart CP2
LEVEL 1
Chart CP3
INTRO (Part 1 of 2)
Set up pointers for use by the various command processors.
R7
addressed by R1
DMB
MRJA
addressed by R1
COMREG
addressed by R1
SYSCOM
1 Initialize the command processor control block pointer
CPBPTR
2 Establish addressability with register 7 as second base register.
3 Turn off all switches
SWITCHES
0
4 Check if POWER/VS job accounting is supported.
No
Yes
Indicates this
SWACCT
1
5 Obtain: program interrupt key of the POWER/VS partition
PIK
pointer to the PIB table
PIBPTR
pointer to the PIB2 table
PIB2PNTR
pointer to PIB of the POWER/VS partition
OURPIBPT
pointer to the LUB table
LUBPTR
pointer to the PUB table
PUBPTR
6 Obtain: pointer to the PUB ownership table
PBOWNPTR
number of partitions generated
NPART
pointer to the MVC field
MVCAD
pointer to the boundary box table
BBOXPT
MAINLINE 9, Chart CP2

Extended Description	Include Segment	Call Subroutine	Chart
(4) Register 1 is loaded with the address (CAQC) of the disk management block in IPW$DPA and is then used to address the field in which the POWER/VS job accounting indicator is placed. (5) Register 1 is used as a base register for COMREG to obtain these pointers. (6) Register 1 is used as a base register for SYSCOM to obtain these pointers.			

Included by MAINLINE, Chart CP2

LEVEL 2

PSTART (Part 1 of 2)

Main processor for the PSTART command. Checks if an execution reader task, a physcial reader/writer task, an RJE,BSC initiator task, or an RJE,SNA manager task is to be attached.

STRTINIT

(1) Build the task control block for the task to be attached.

TCB

OPERAND

ORIGINAL

(2) Check if a valid partition identifier was specified.

No Yes

Assume the command is valid

SWCANCEL

0

STRTPART

Start the execution reader task.

(3) Check if a physical reader/writer was specified. (RDR, LST, PRT, or PUN)

No Yes

Assume the command is valid

SWCANCEL

0

STRTRDWR

Start the reader/writer task.

OPERAND

ORIGINAL

(4) Check if RJE is specified.

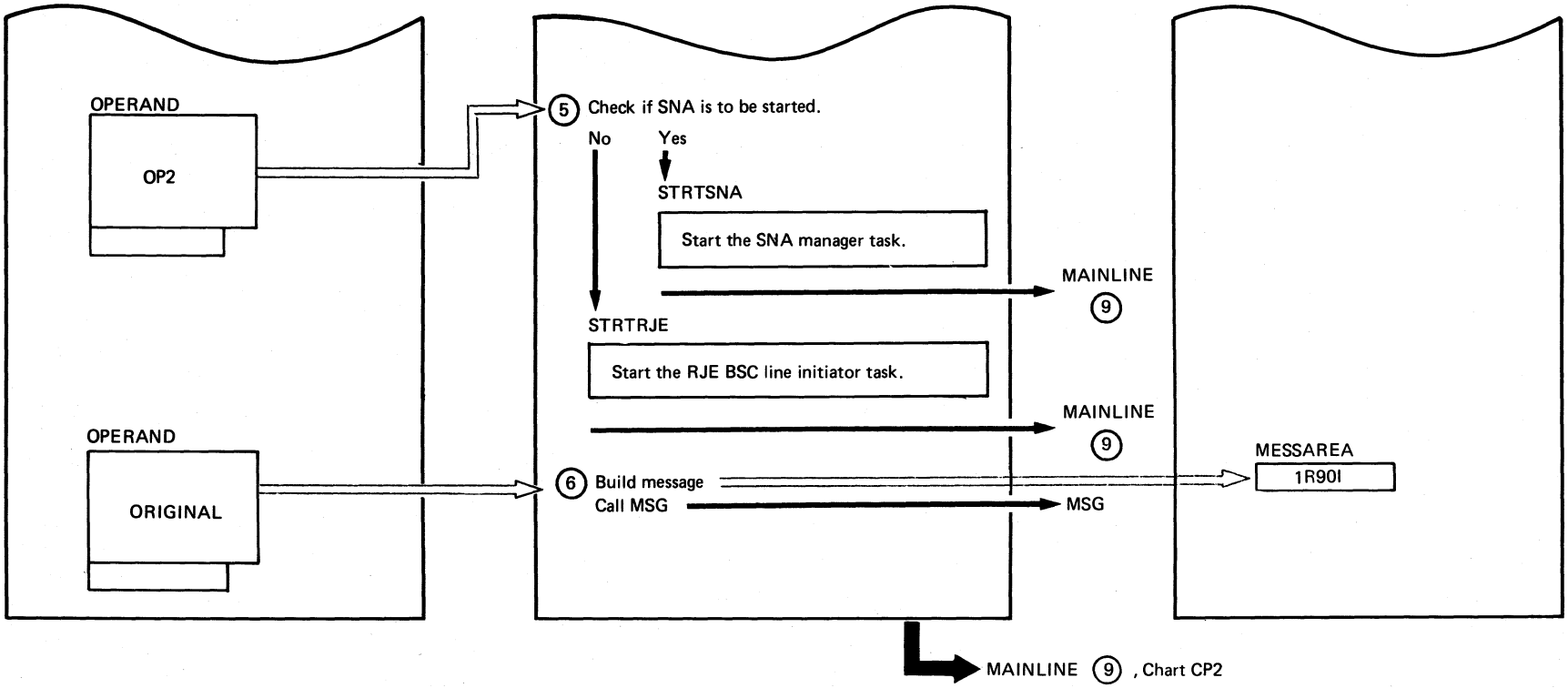
Yes No → (6)

Assume the command is valid

SWCANCEL

0

(Continued on next page)

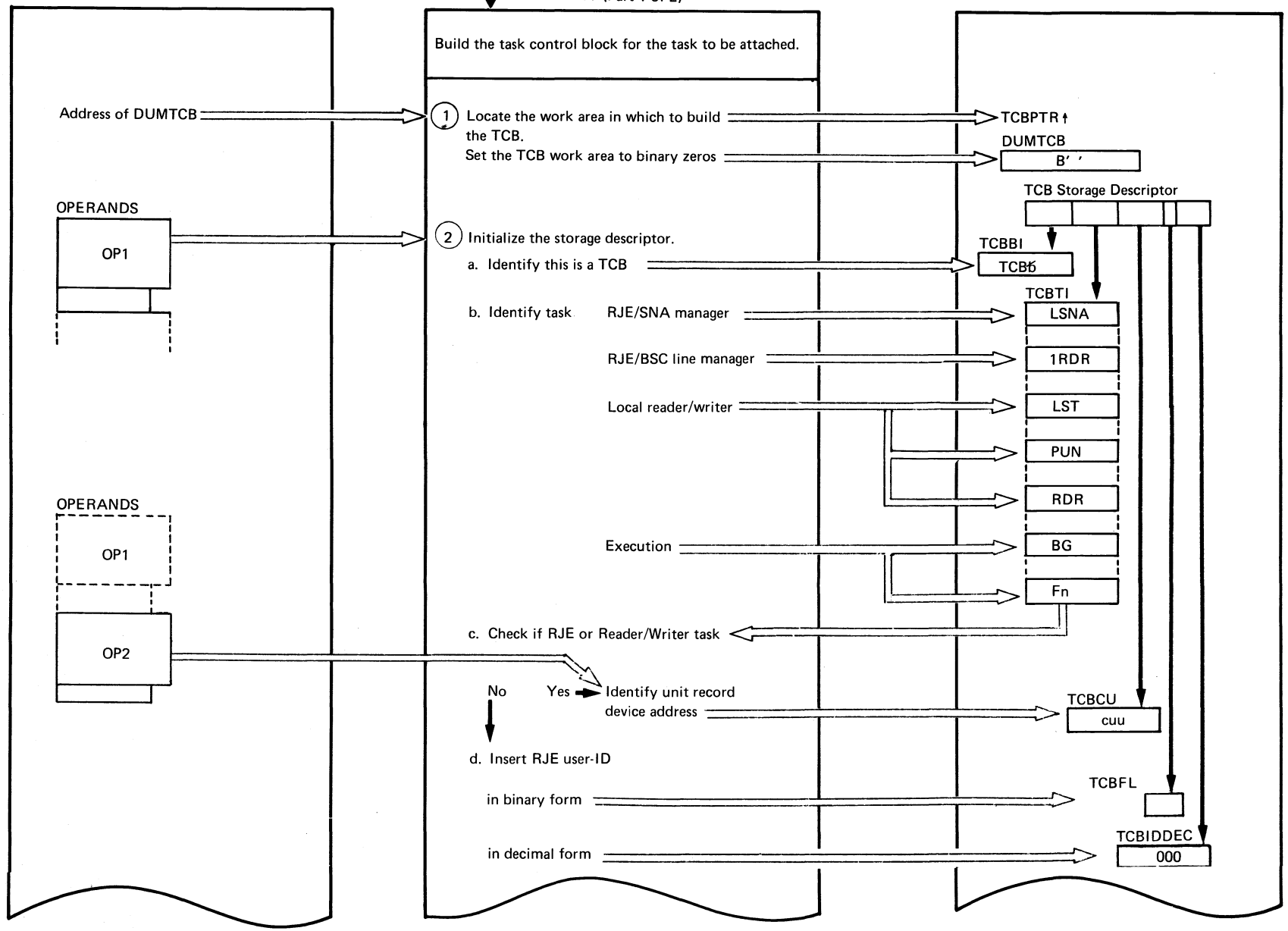


Extended Description		Include Segment	Call Subroutine	Chart
1		STRTINIT		CP5
2	Register 5 is loaded as second base register	STRTPART		CP6
3		STRTRDWR		CP17
5		STRTSNA STRTRJE		CP24 CP23
6	1R90I INVALID TASK SPECIFICATION, xxxxxxxxxx original operand of command		MSG	CP75

Included by PSTART, Chart CP 4

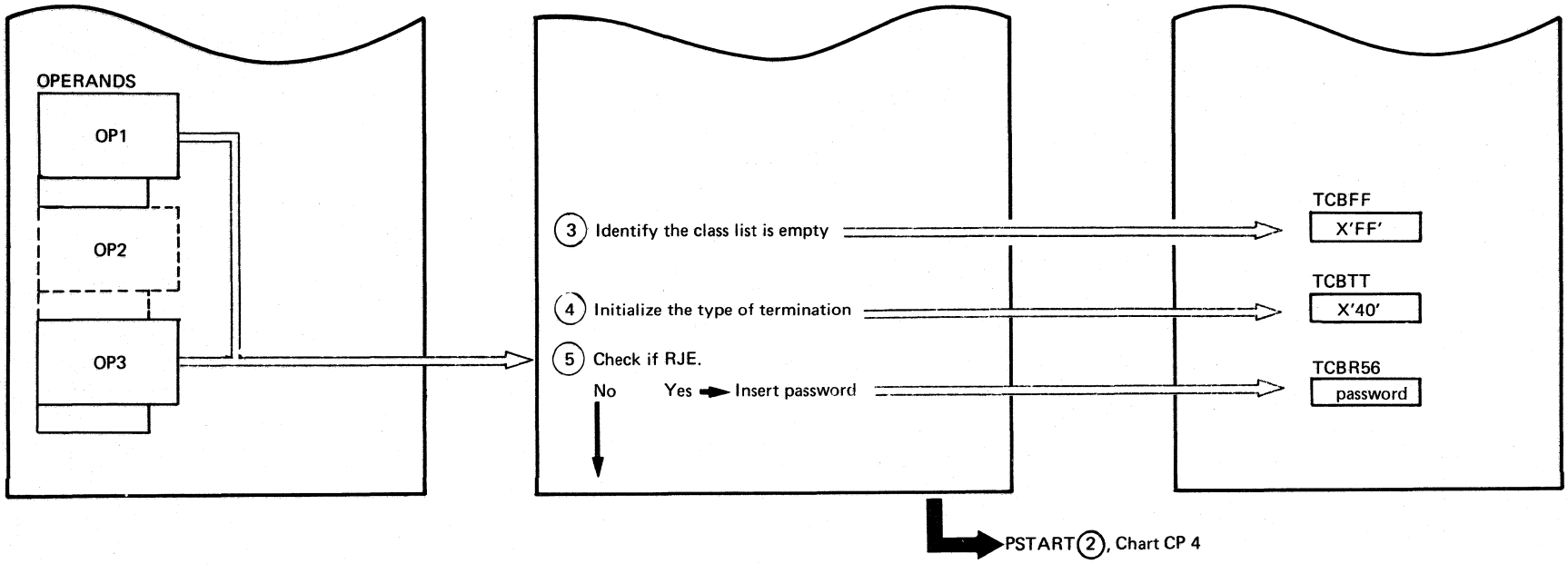
LEVEL 3

STRTINIT (Part 1 of 2)

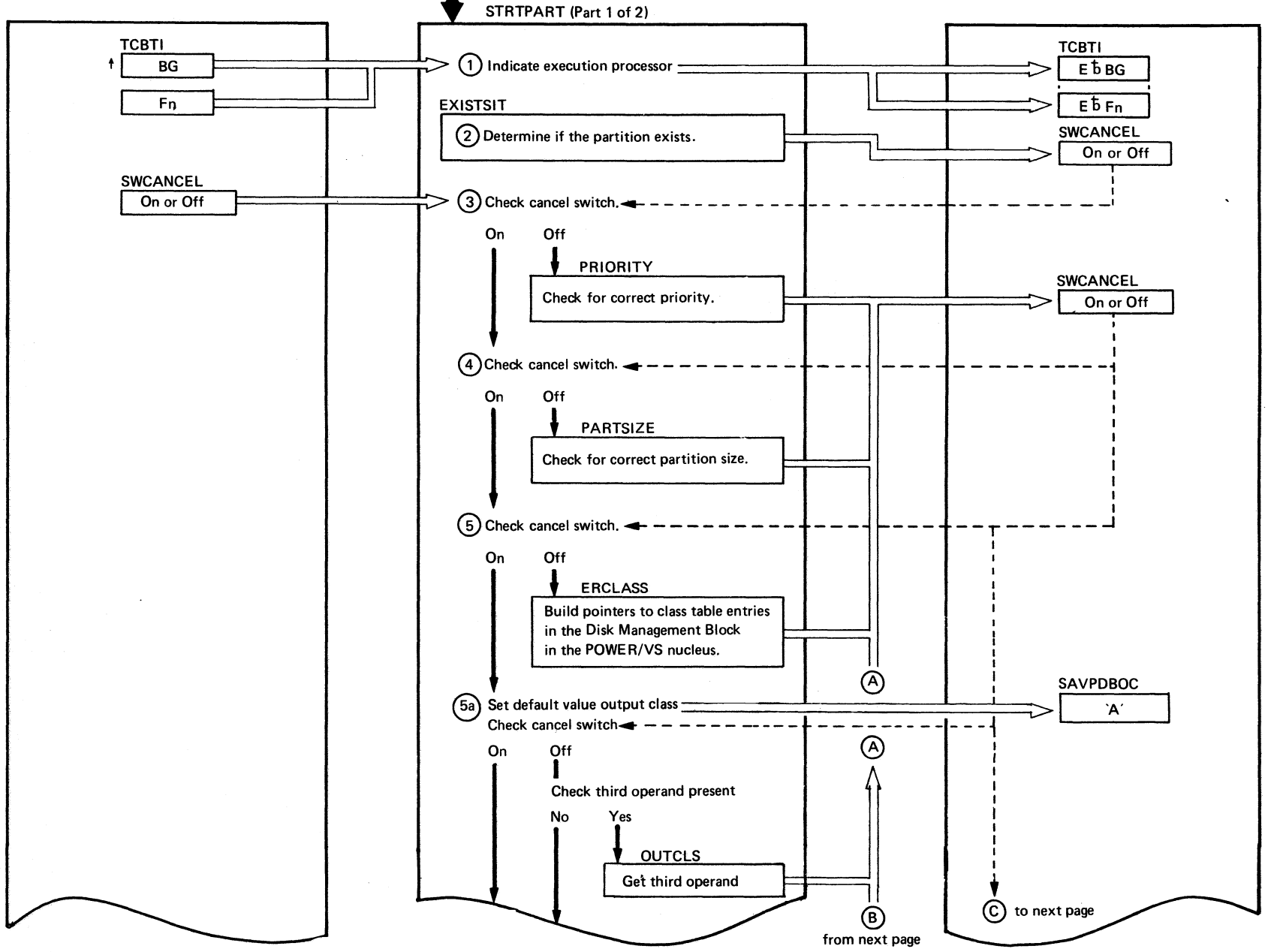


(Continued on next page)

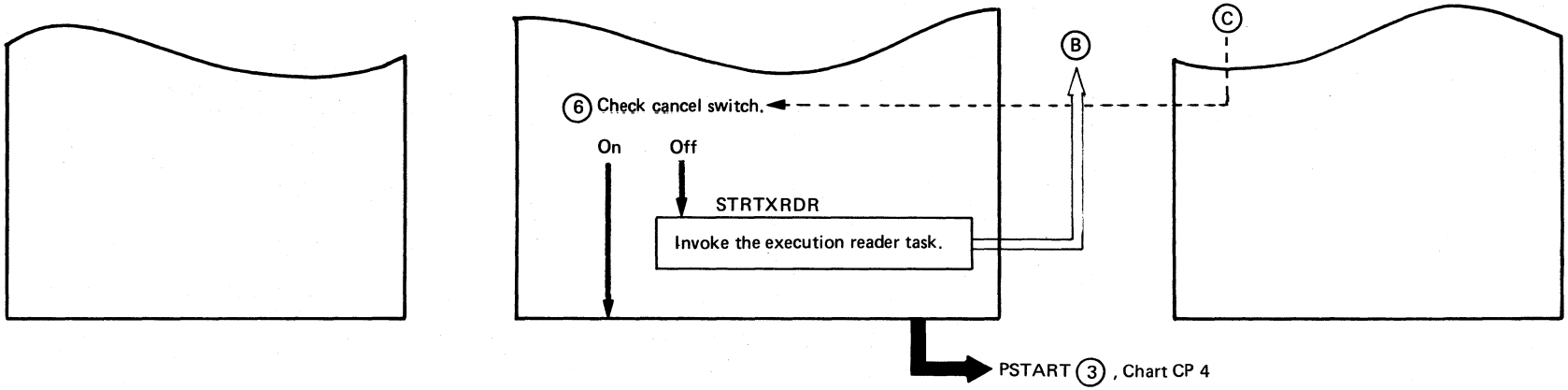
STRTINIT (Part 2 of 2)



Extended Description	Include Segment	Call Subroutine	Chart
(2) d. TCBIDDEC is always zeroed for the central operator. TCBFL is normally X'00' for the central operator. (4) The Task Control Block Termination Type (TCBTT) field in the TCB contains: blank : for running U : for unrecoverable I/O error C : for cancel F : for flush H : for flush with hold option E : for stop at end R or S : for stop immediately			


STRTPART (Part 1 of 2)
TCBTI
BG
Fn
SWCANCEL
On or Off
1 Indicate execution processor
EXISTSIT
2 Determine if the partition exists.
3 Check cancel switch.
On
Off
PRIORITY
Check for correct priority.
4 Check cancel switch.
PARTSIZE
Check for correct partition size.
5 Check cancel switch.
ERCLASS
Build pointers to class table entries in the Disk Management Block in the POWER/VS nucleus.
5a Set default value output class
Check cancel switch
Check third operand present
No
Yes
OUTCLS
Get third operand
TCBTI
E b BG
E b Fn
SWCANCEL
On or Off
SWCANCEL
On or Off
SAVPDBOC
'A'
A
B
from next page
C to next page

STRTPART (Part 2 of 2) LEVEL 3 Chart CP6



Extended description	Include Segment	Call Subroutine	Chart
2	EXISTSIT		CP7
3	PRIORITY		CP8
Steps 2 through 6 : If no errors occur, the cancel switch remains off.			
4	PARTSIZE		CP9
5 The pointers built by ERCLASS are contained in the TCB being built.	ERCLASS		CP10
6	STRTXRDR		CP12

Included by STRTPART, Chart CP6

Chart CP 7

EXISTSIT (Part 1 of 2)

LEVEL 4

Ensure that the partition to be brought under POWER/VS control is supported by this DOS/VS system and is not the POWER/VS partition.

PIBTAB

PIBPTR

AR	
BG	
Fn	

OPERANDS

OP1

① Locate the last entry of the PIB table

LIMIT

PIBTAB

AR	
BG	
Fn	

④

② Locate the PIB entry of the partition to be started

PIBTAB

PIBPOINT

AR	
BG	
Fn	

④

③ Calculate the PIK of the partition to be started

OTHPIK

XX

④ Check if partition to be started is supported by DOS/VS.

④

Yes

No → Turn on cancel switch

SWCANCEL

1

Build error message

MESSAREA

1R60I

Call MSG → MSG

PIK (of POWER/VS partition)

XX

⑤ Check whether the partition to be started is the POWER/VS partition.

No

Yes → Build error message

MESSAREA

1R68I

Call MSG → MSG

SWCANCEL

1

Turn on cancel switch

STRTPART ③, Chart CP 6

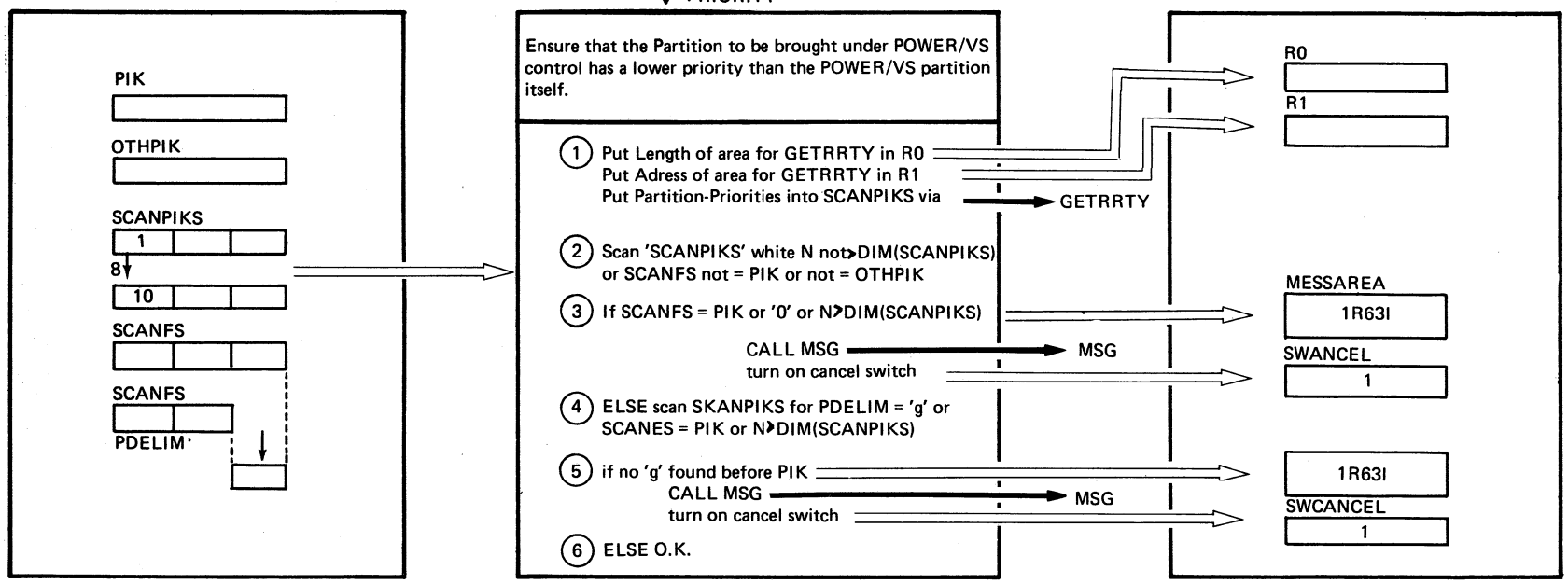
Chart CP7

Extended Description	Include Segment	Call Subroutine	Chart
(1) The address of the last entry of the PIB table is calculated with the use of the IJBNPART (in SYSCOM) value of the DOS/VS supervisor.			
(2) This is done by comparing the partition identifier in the first operand (OP1) of the PSTART command with the PIBLOGID field in the PIB table entries.			
(3) OTHPIK in output contains the PIK of partition to be started.			
(4) 1R60I xx NOT SUPPORTED ↑ partition identifier		MSG	CP75
(5) PIK contains the displacement of the PIB entry found.		MSG	CP75
1R68I xx IS THE POWER PARTITION ↑ partition identifier			

Included by STRTPART, Chart CP6

Chart CP8

PRIORITY

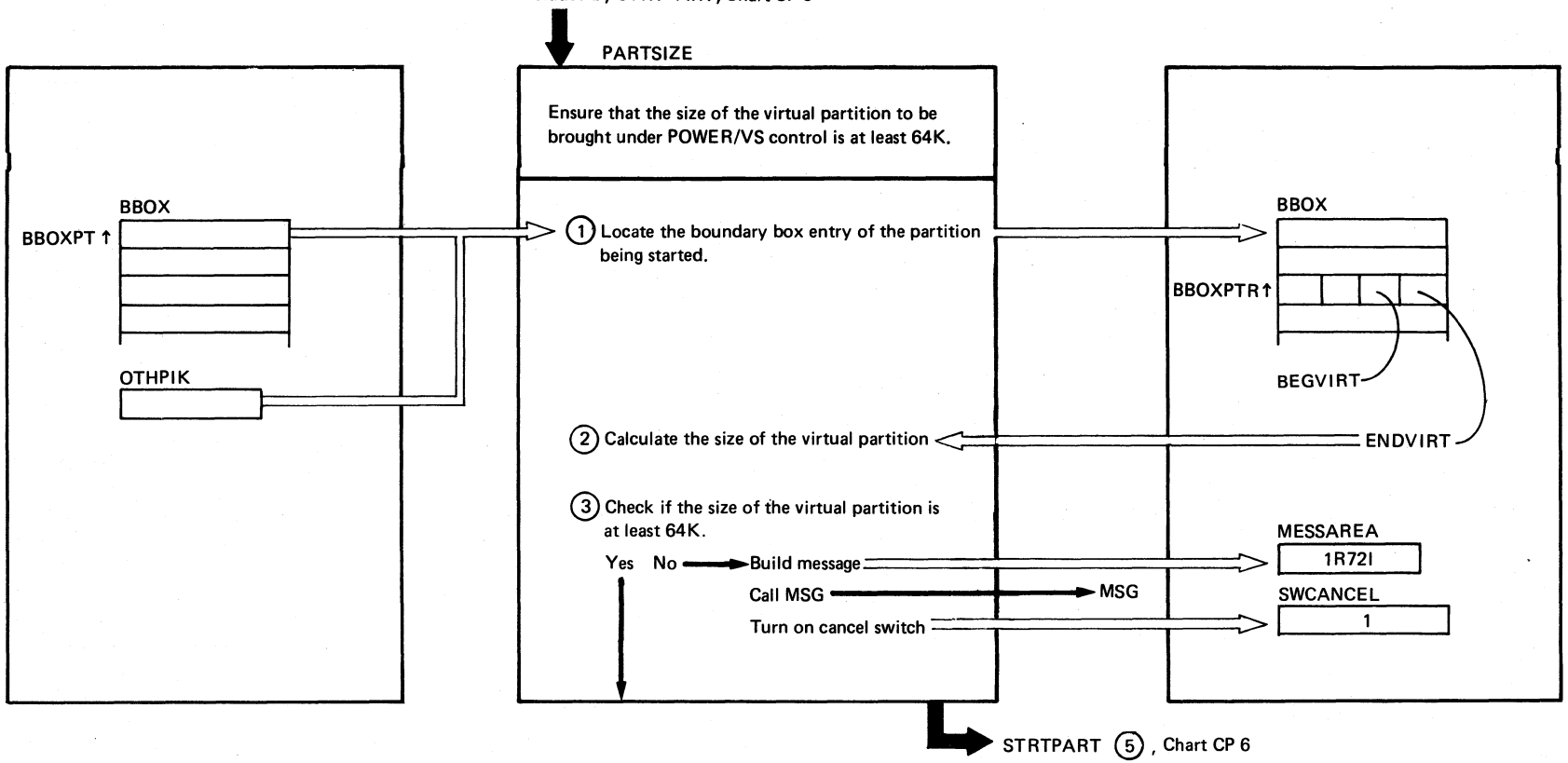


STRTPART (4), Chart CP 6

Extended Description	Include Segment	Call Subroutine	Chart
(3) '1R63I xx PRIORITY TOO HIGH'	MSG		CP75
(5) '1R63I xx PRIORITY TOO HIGH'	MSG		CP75

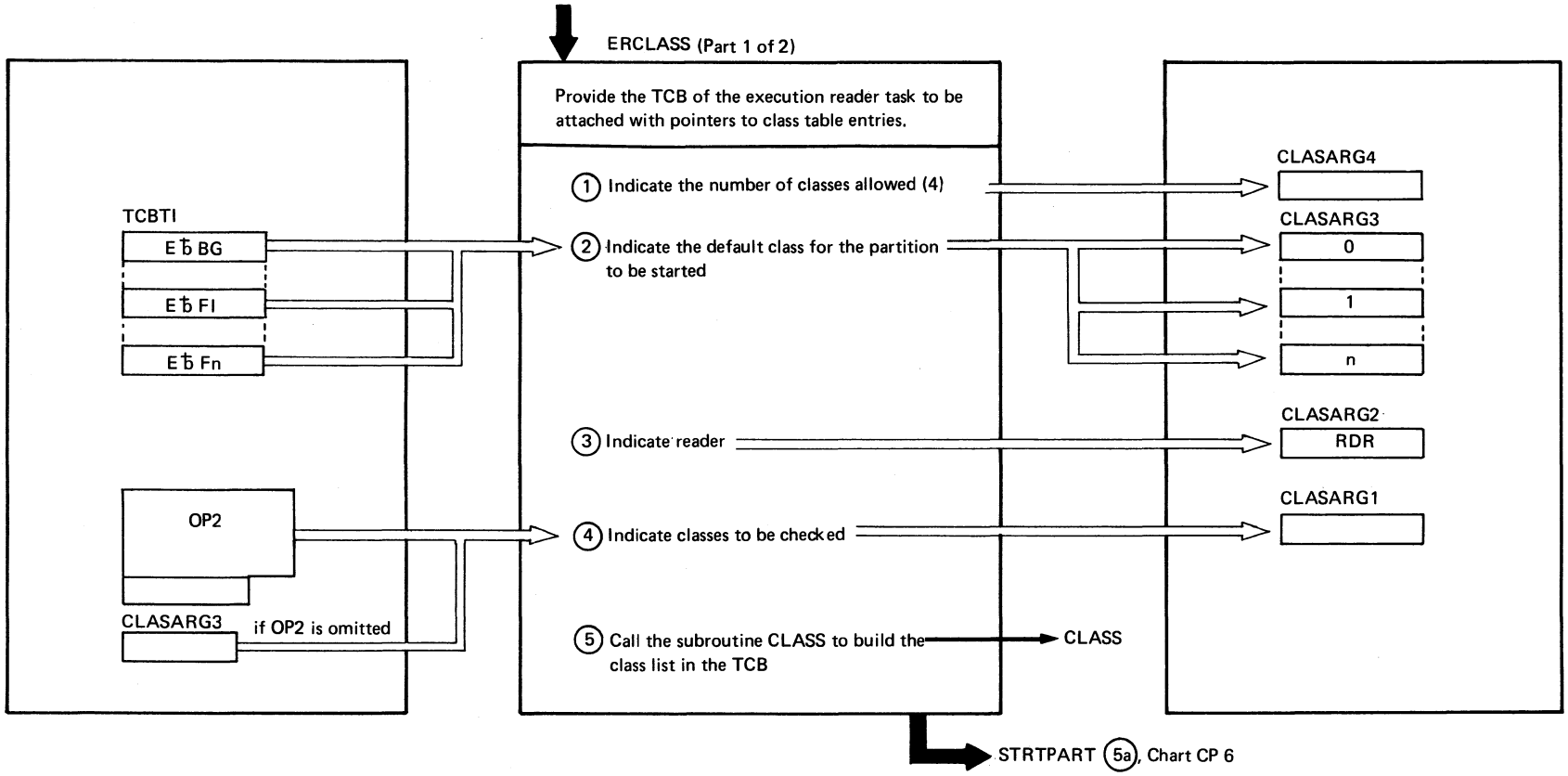
Chart CP8

Included by STRTPART, Chart CP 6 LEVEL 4 Chart CP9



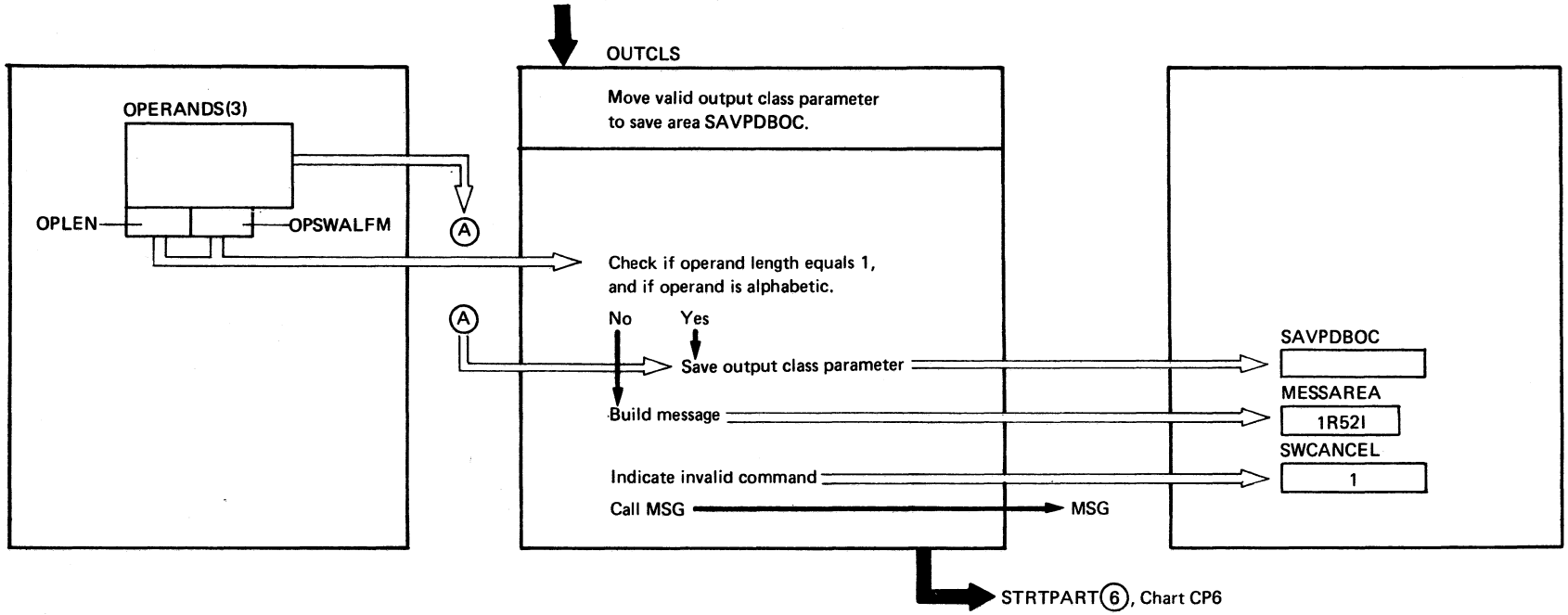
Extended Description	Include Segment	Call Subroutine	Chart
(1) OTHPIK is used to index the BBOX table in the DOS/VS supervisor. (2) The size is calculated by deducting the value in BEGVIRT from the value in ENDVIRT. (3) 1R72I VIRTUAL xx SMALLER THAN 64K ↑ partition identifier		MSG	CP75

Included by STRTPART, Chart CP 6 LEVEL 4 Chart CP 10

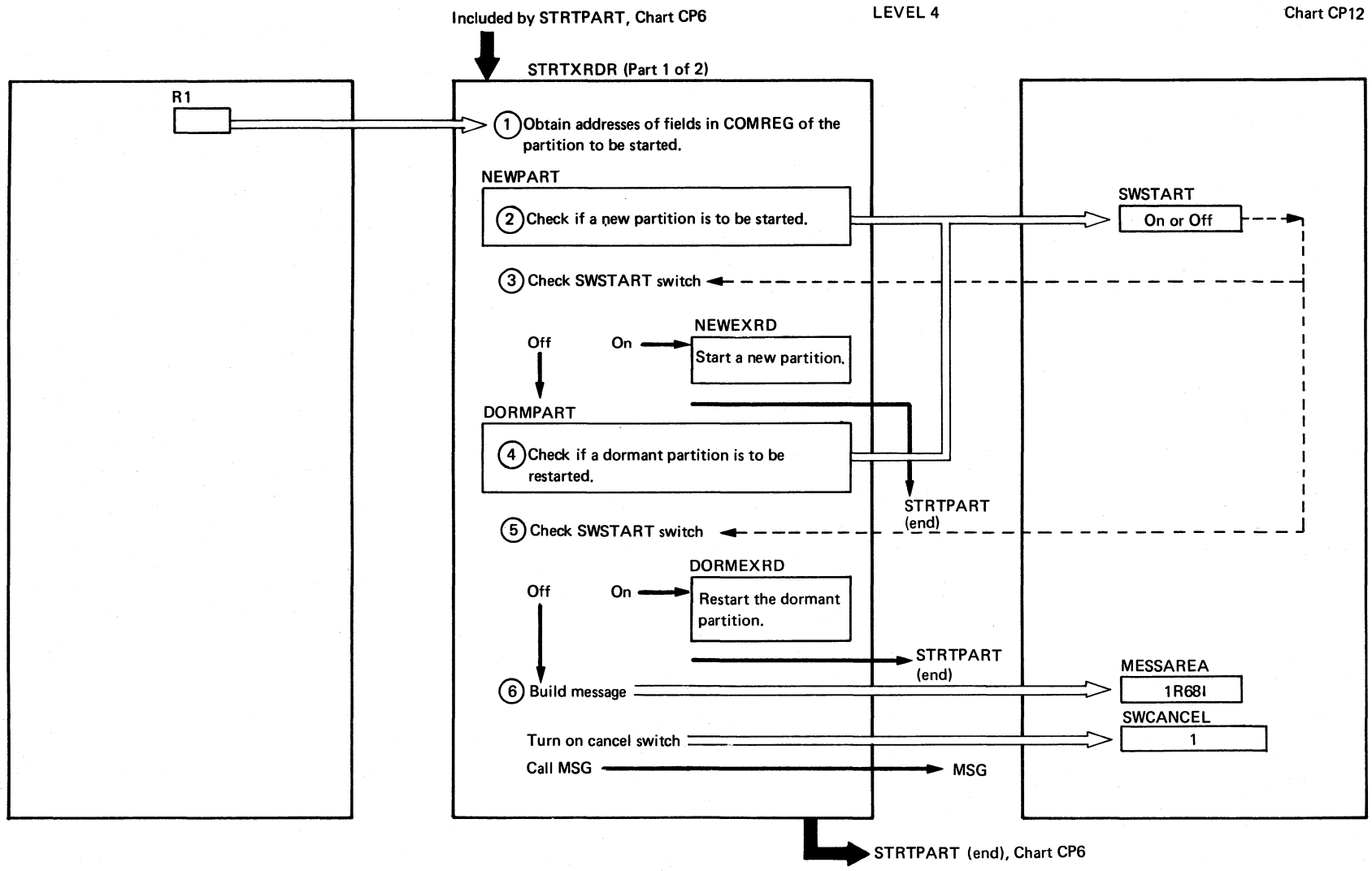


Extended Description	Include Segment	Call Subroutine	Chart
(2) For BG the default is '0' For F1 the default is '1' For F2 the default is '2', etc. (4) If the second operand is not specified the default is taken from CLASARG3. (5)		CLASS	CP86

Included by STRTPART, Chart CP6 LEVEL 4 Chart CP11

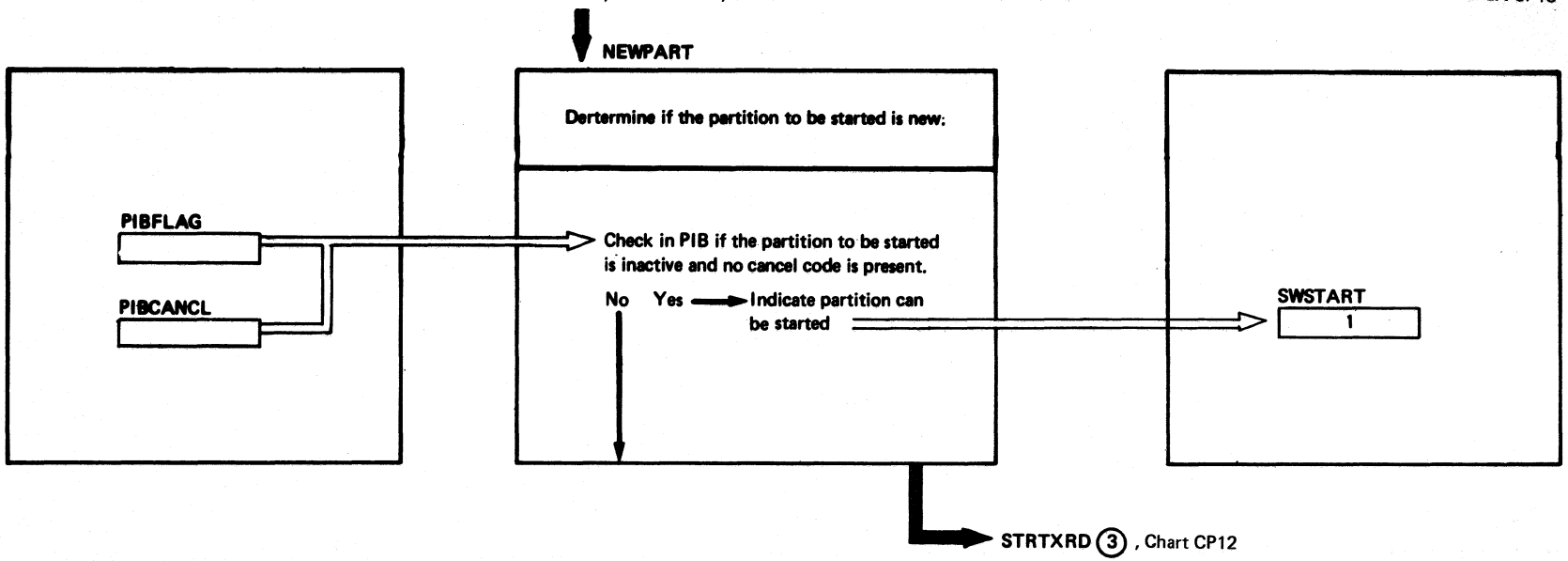


Extended Description	Include Segment	Call Subroutine	Chart
1R52I INVALID OUTPUT–CLASS		MSG	CP75


Included by STRTPART, Chart CP6
LEVEL 4
Chart CP12
STRTXRDR (Part 1 of 2)
R1
1 Obtain addresses of fields in COMREG of the partition to be started.
NEWPART
2 Check if a new partition is to be started.
SWSTART
On or Off
3 Check SWSTART switch
NEWEXRD
Off
On
Start a new partition.
DORMPART
4 Check if a dormant partition is to be restarted.
STRTPART (end)
5 Check SWSTART switch
DORMEXRD
Off
On
Restart the dormant partition.
STRTPART (end)
MESSAREA
1R68I
6 Build message
SWCANCEL
1
Turn on cancel switch
Call MSG
MSG
STRTPART (end), Chart CP6

Extended Description	Include Segment	Call Subroutine	Chart
(1) Using the other partition's PIK, calculate the address of its COMREG and calculate the address of: 1. the POWER/VS flag byte POWFLG1 2. the JCL switch byte JCSW4 3. the job name to be inserted 4. the pointers to the Partition Control Block to be created.			
(2) Register 1 is used to address COMREG of the partition to be started.	NEWPART		CP13
(3)	NEWEXRD		CP14
(4)	DORMPART		CP15
(5)	DORMEXRD		CP16
(6) 1R68I xx PARTITION NOT AVAILABLE ↑ partition identifier		MSG	CP75

Included by STRTXRDR, Chart CP12 LEVEL 5 Chart CP13



Extended Description	Include Segment	Call Subroutine	Chart
PIBLAG = X'80' if the partition is inactive. PIBCANCL = X'00' if no cancel condition is present. When both conditions are met the partition can be started.			

Included by STRTXRD, Chart CP12

LEVEL 5

Chart CP14

NEWEXRD (Part 1 of 2)

Attach the new execution reader task by calling the subroutine ATTACH and simulate the normal 'AR' 'START' command for a DOS/VS partition that has not yet been active.

LUBTAB

BG entries

OTHPIK

(1) Call the subroutine ATTACH to attach a new execution reader task → ATTACH

(2) Copy BG SYSLOG and SYSRES assignments into SYSLOG and SYSRES LUB entry of the partition to be started

LUBTAB

indexed by OTHPIK

(3) Turn on the initialize batch flag for job control in COMREG

JBCSW4

X 1

(4) Set job name in COMREG

JBNAM

NO NAME

(5) Make the partition save area addressable.

(6) Clear the partition save area.

(7) Insert job name

SAVEAREA

NO NAME

SVC 22

Seize the system and disable interrupts.

Last byte of PIB

XXXX XXXX

PIBBAT

PIBCANCL

PIBFTEOJ

(8) Initialize the PIB of the partition to be started

Last byte of PIB

IXXX IIXX

PIBFLAG

81

byte 1

PIBCANCL

10

byte 2

(9) Increase the number of active virtual partitions

SYSCOM

IJBAPND

n+1

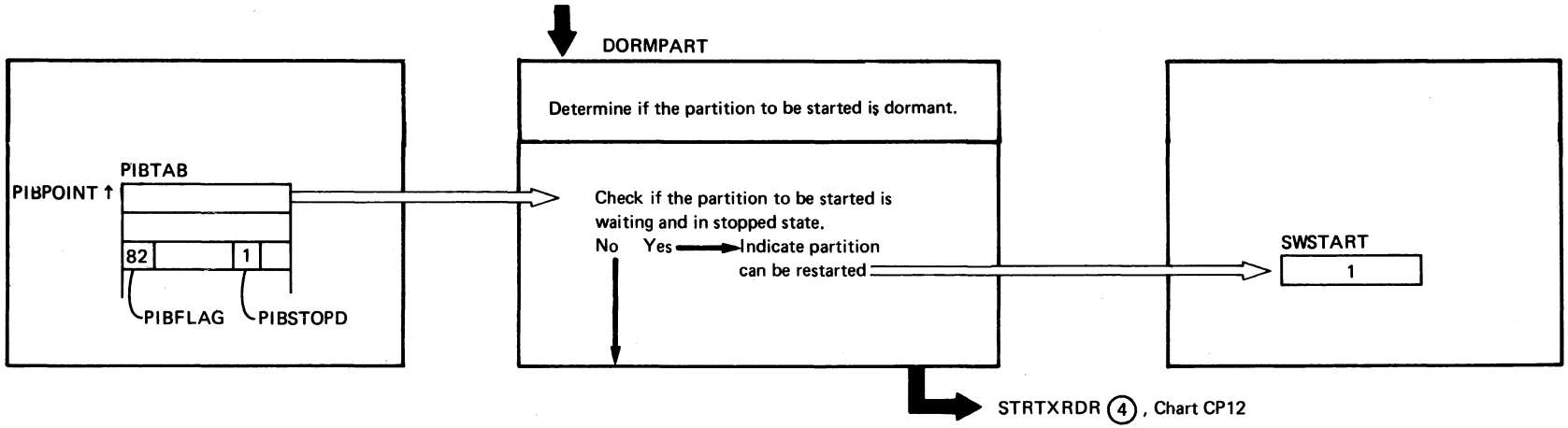
SVC 22

Release the system and enable interrupts.

STRTXRDR (end), Chart CP12

Extended Description	Include Segment	Call Subroutine	Chart
(1) Steps (2) through (7) simulate the normal 'AR' start command for the DOS/VS partition that has not yet been active, or has been unbatched. (8) In this step PIBPOINT is not used but loaded into register 2 to avoid page faults, since this part of the code is disabled for interrupts and the system is seized. For the precise meaning of the bits set, refer to the description of PIB in DOS/VS Supervisor Logic, SY33-8551. Register usage: R0 to disable or enable interrupts R2 to address PIB R1 to address SYSCOM		ATTACH	CP89

Included by STRTXRDR, Chart CP12 LEVEL 5 Chart CP15



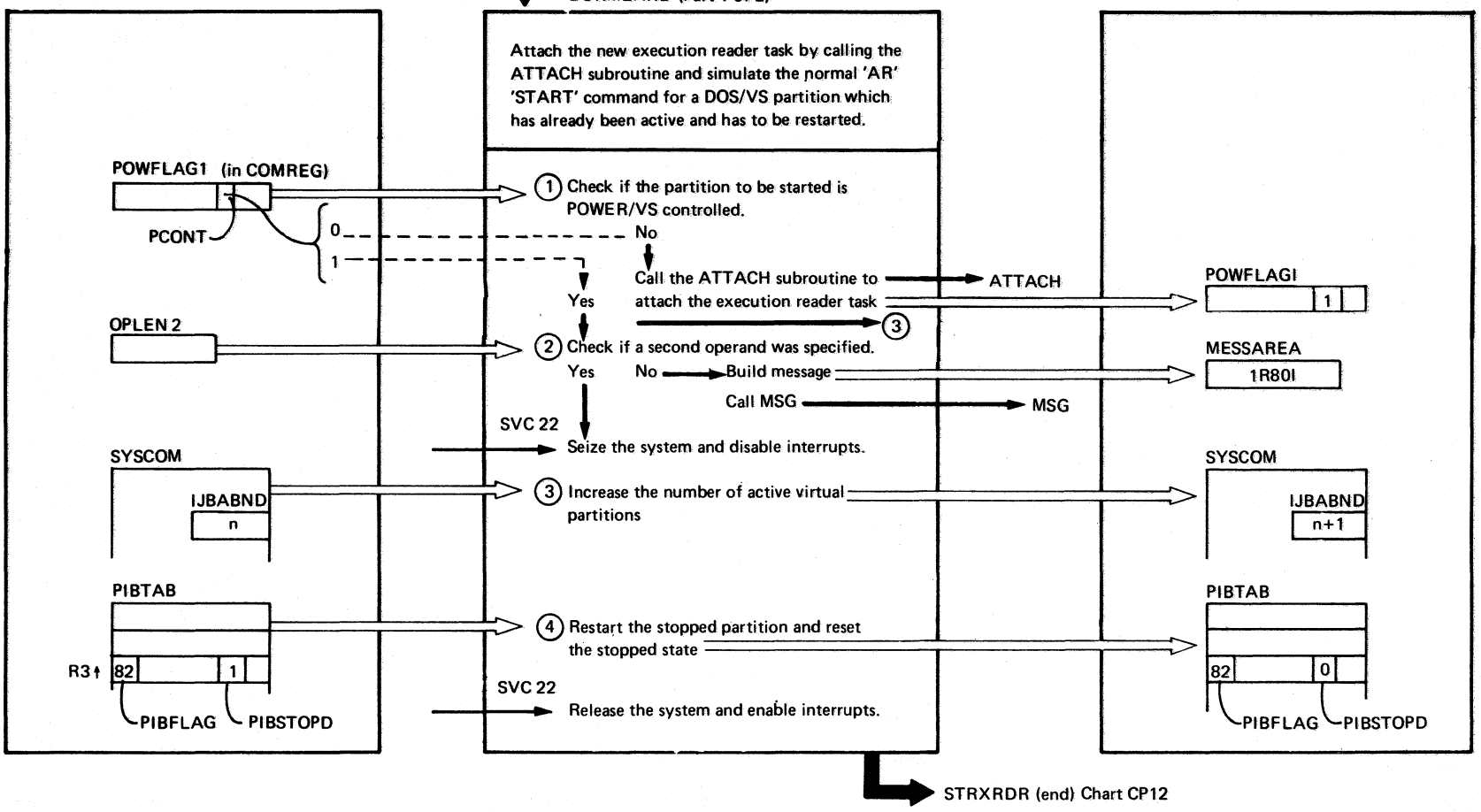
Included by STRXRDR, Chart CP12

LEVEL 5

Chart CP16

DORMEXRD (Part 1 of 2)

Attach the new execution reader task by calling the ATTACH subroutine and simulate the normal 'AR' 'START' command for a DOS/VS partition which has already been active and has to be restarted.



STRXRDR (end) Chart CP12

Chart CP16

Extended Description	Include Segment	Call Subroutine	Chart
(1)		ATTACH	CP89
(2) The class parameter is ignored because the execution reader tasks and their classes exist already. 1R80I WARNING: CLASS SPECIFICATION IGNORED		MSG	CP75
(3) and (4) This code is executed in disabled state so no page fault should occur. The PIB of the partition is, therefore, addressed by a register and not by PIBPOINT. Register usage: R0 to disable or enable interrupts R1 to address SYSCOM R2 to increase the number of virtual partitions R3 to address PIB			

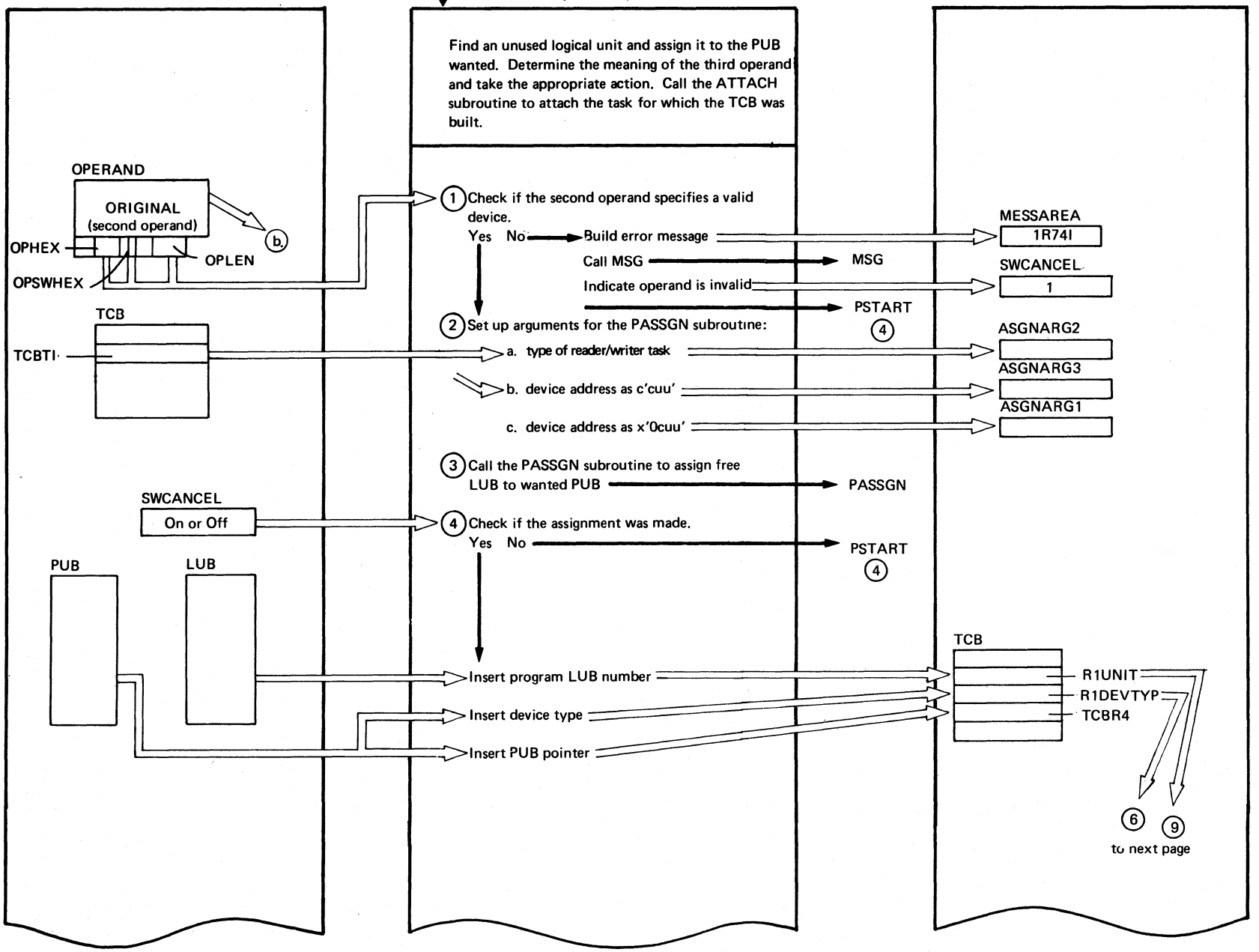
Included by PSTART, Chart CP2

LEVEL 3

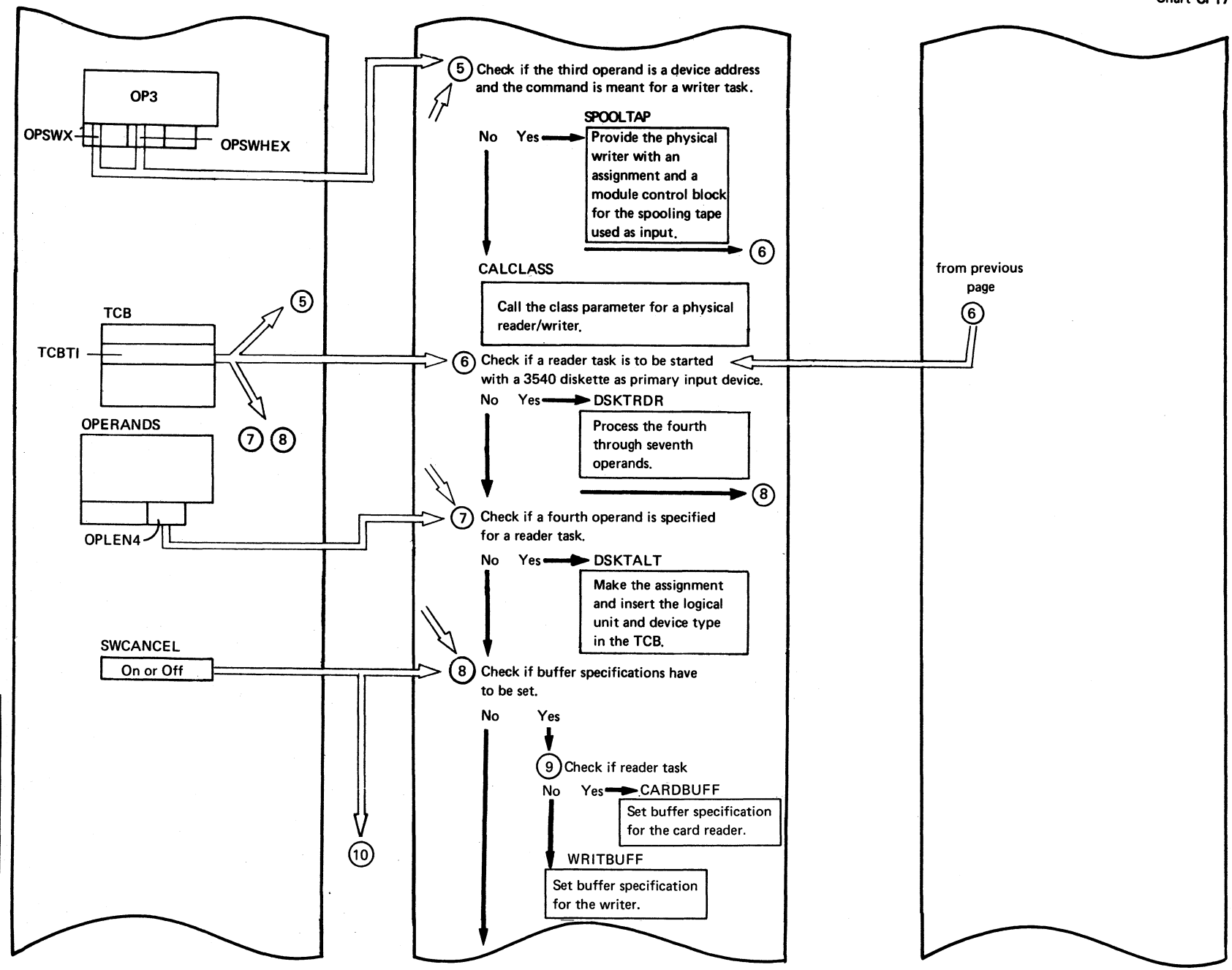
Chart CP17

STRTRDWR (Part 1 of 3)

Find an unused logical unit and assign it to the PUB wanted. Determine the meaning of the third operand and take the appropriate action. Call the ATTACH subroutine to attach the task for which the TCB was built.



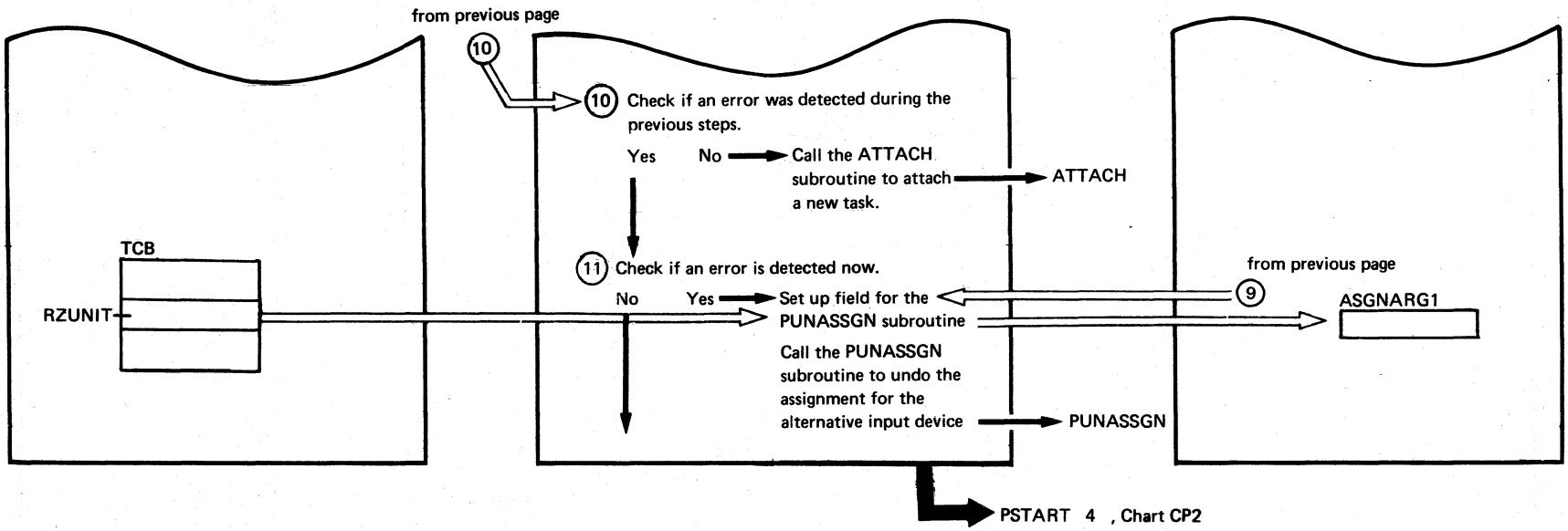
continued on next page



(Continued on next page)

Page of SY33-8577-1, Revised November 24, 1977, By TNL SN33-9241

STRTRDWR (Part 3 of 3)



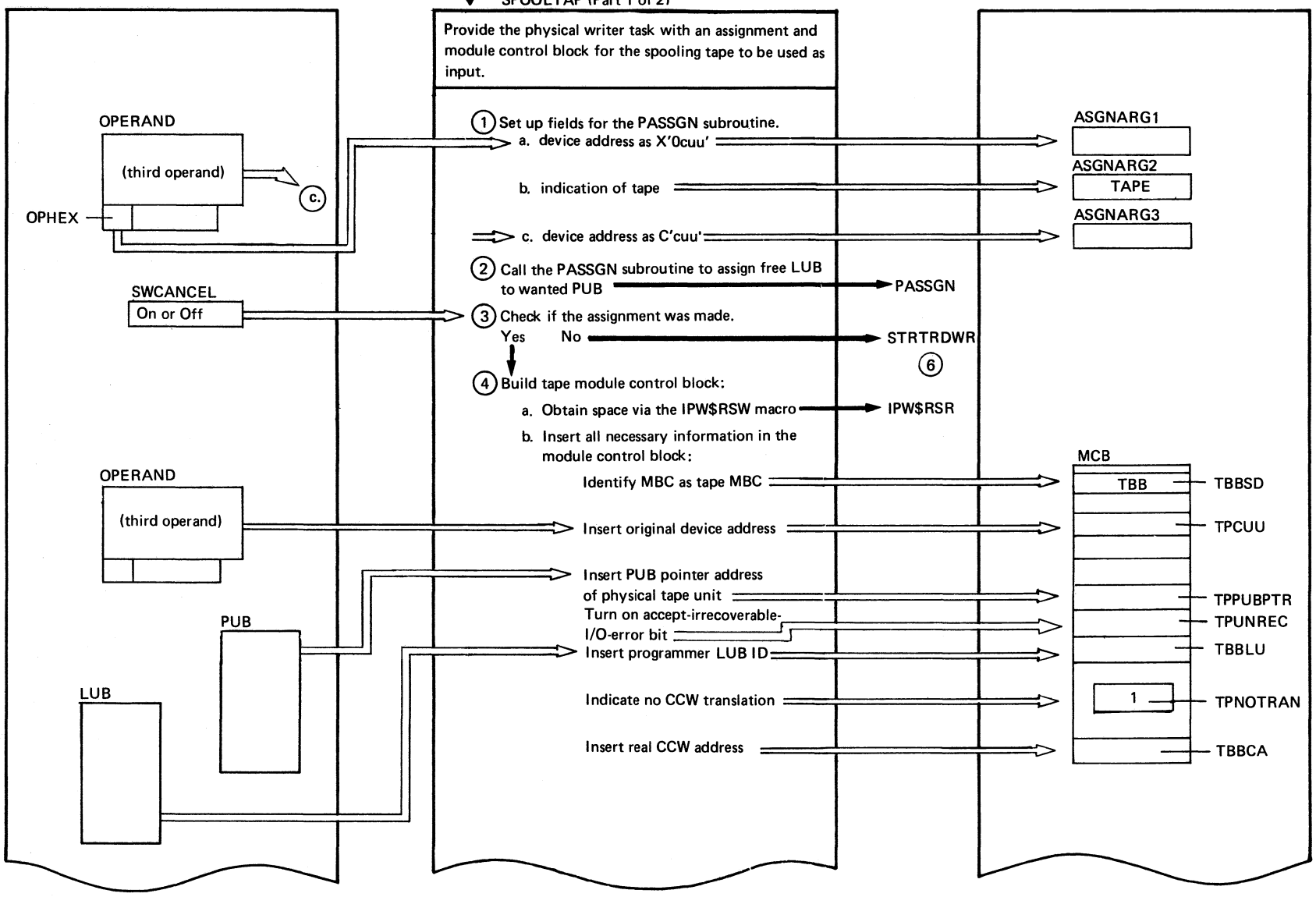
Extended Description	Include Segment	Call Subroutine	Chart
(1) 1R74I INVALID DEVICE SPECIFICATION		MSG	CP75
		PASSGN	CP76
(3)			
(5)	SPOOLTAB		CP18
	CALCLASS		CP19
(6)	DSKTRDR		CP20
(7)	DSKTALT		CP21
(8)	CARDBUFF		CP22
(9)	WRITBUFF		CP22.1
(10)		ATTACH	CP89
(11)		PUNASSGN	CP80

Page of SY33-8577-1, Revised November 24, 1977, By TNL SN33-9241

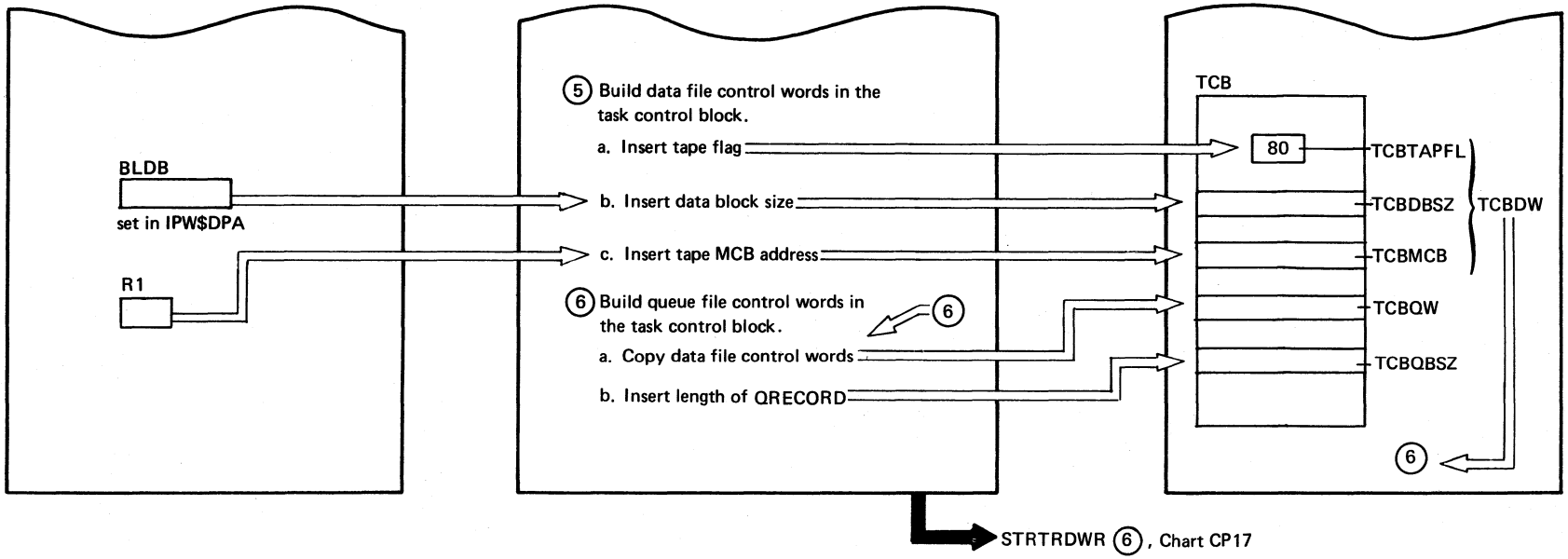
Included by STRTRDWR, Chart CP17 LEVEL 4 Chart CP18

SPOOLTAP (Part 1 of 2)

Provide the physical writer task with an assignment and module control block for the spooling tape to be used as input.

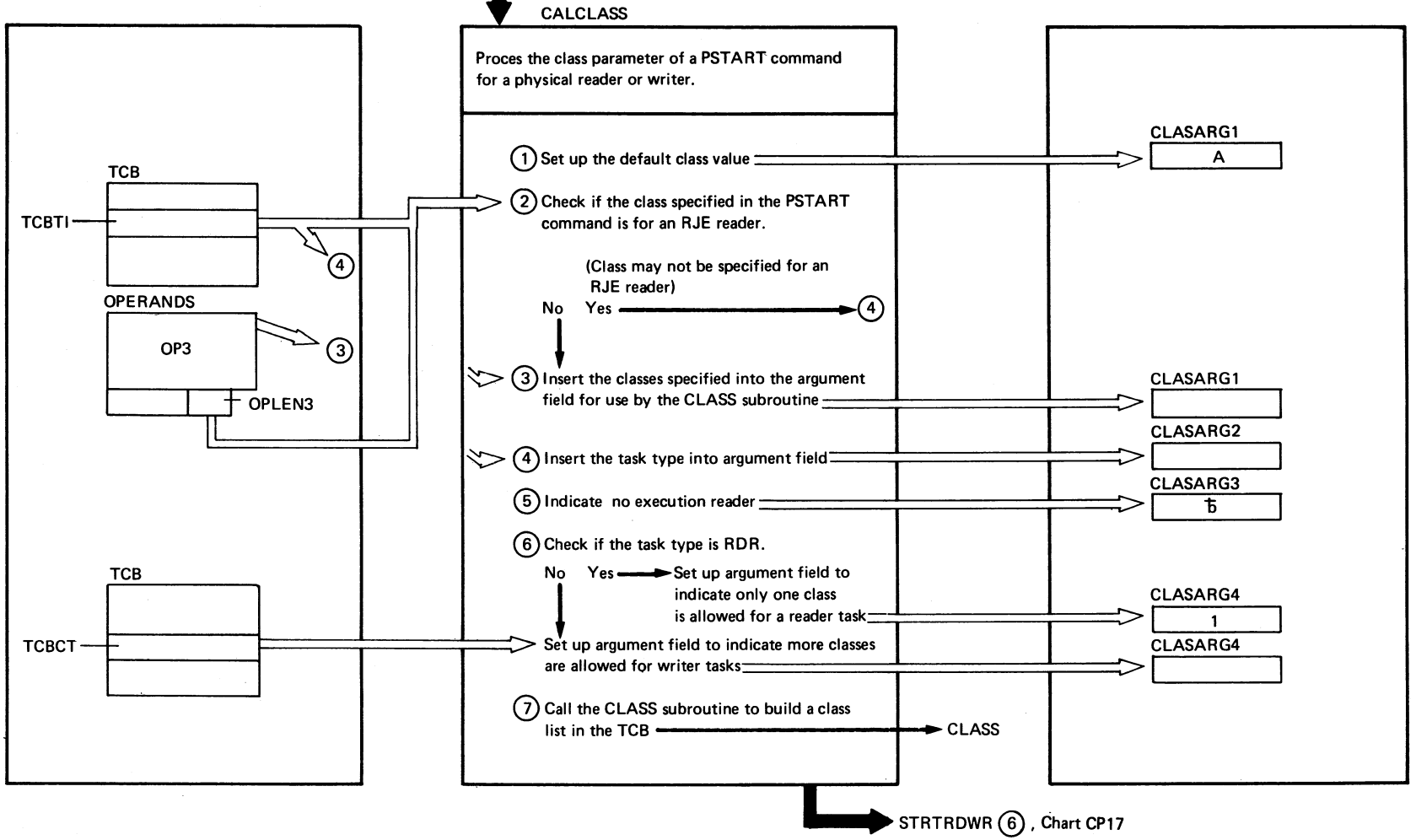


continued on next page



Extended Description	Include Segment	Call Subroutine	Chart
(2)		PASSGN	CP76
(4) a. The IPW$RSW macro uses registers 0, 1, 2, and 3.			

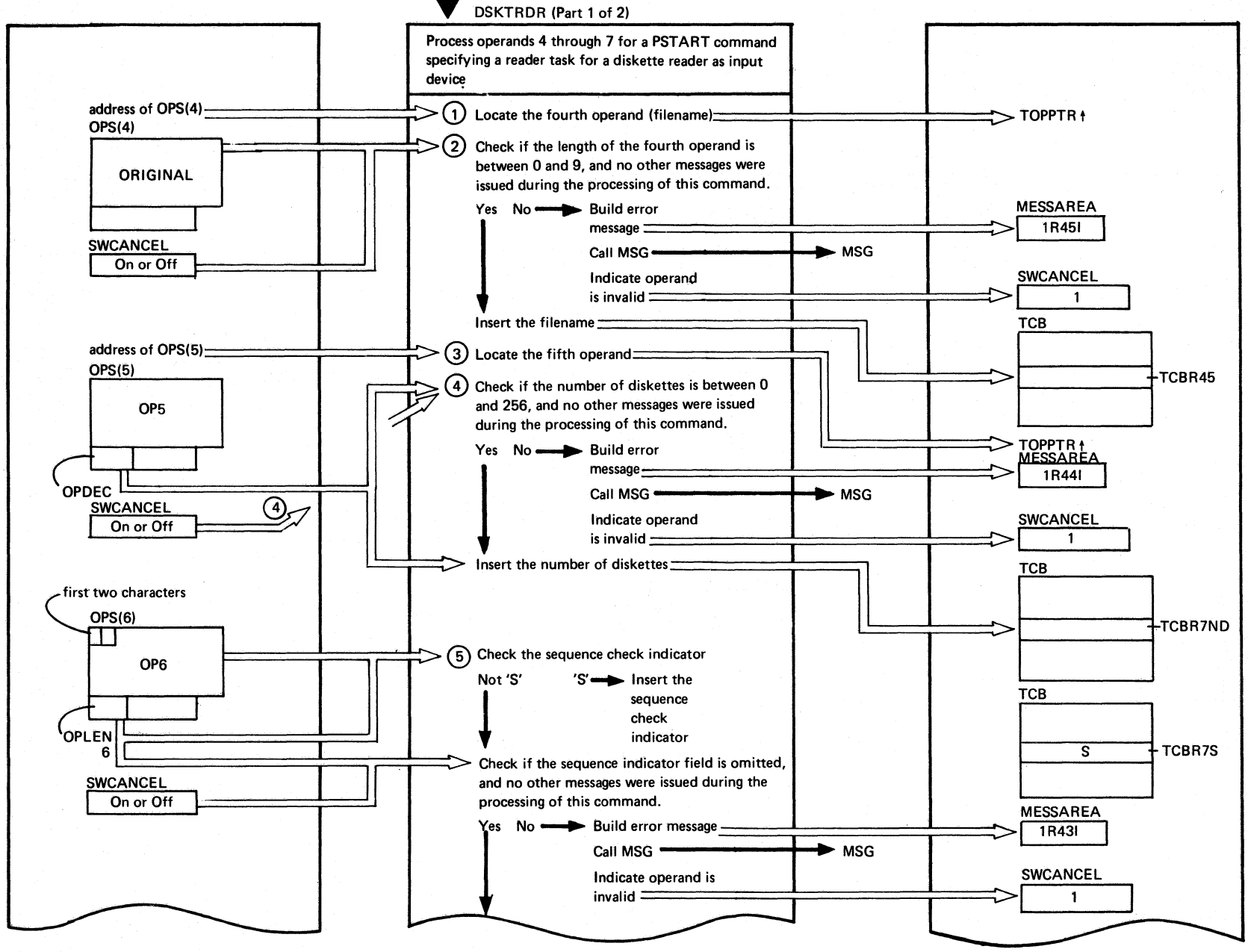
Included by STRTRDWR Chart CP17 LEVEL 4 Chart CP19



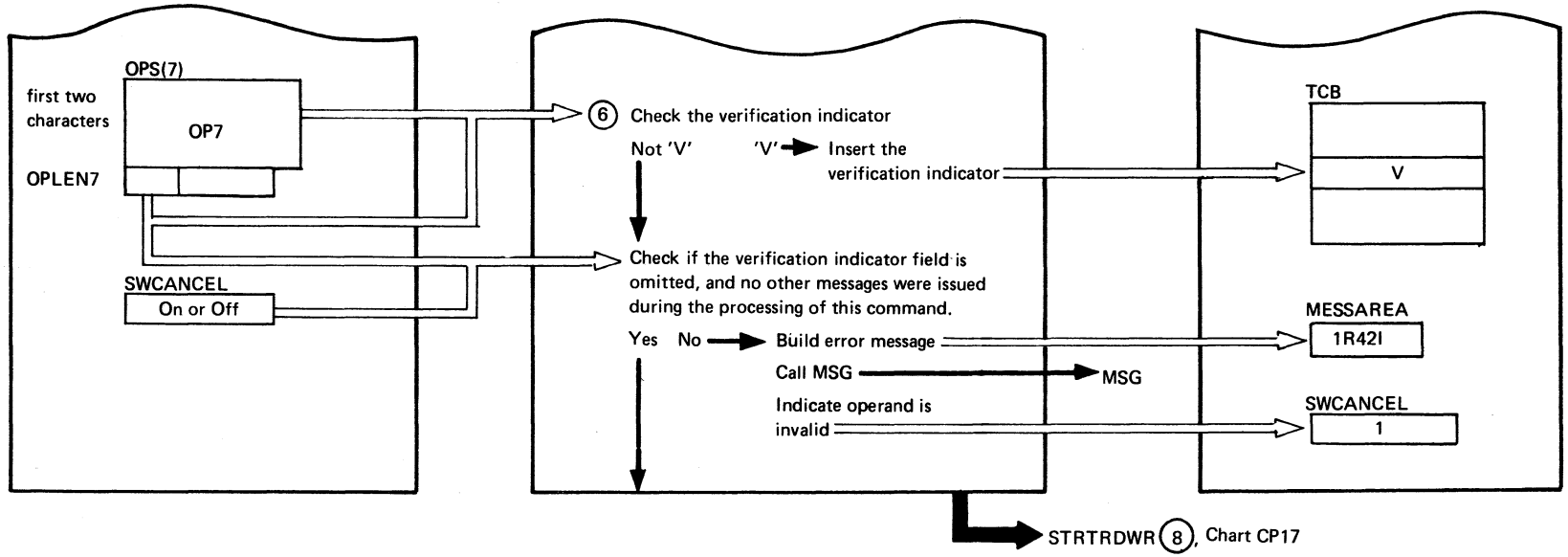
Extended Description	Include Segment	Call Subroutine	Chart
7		CLASS	CP86

Chart CP19

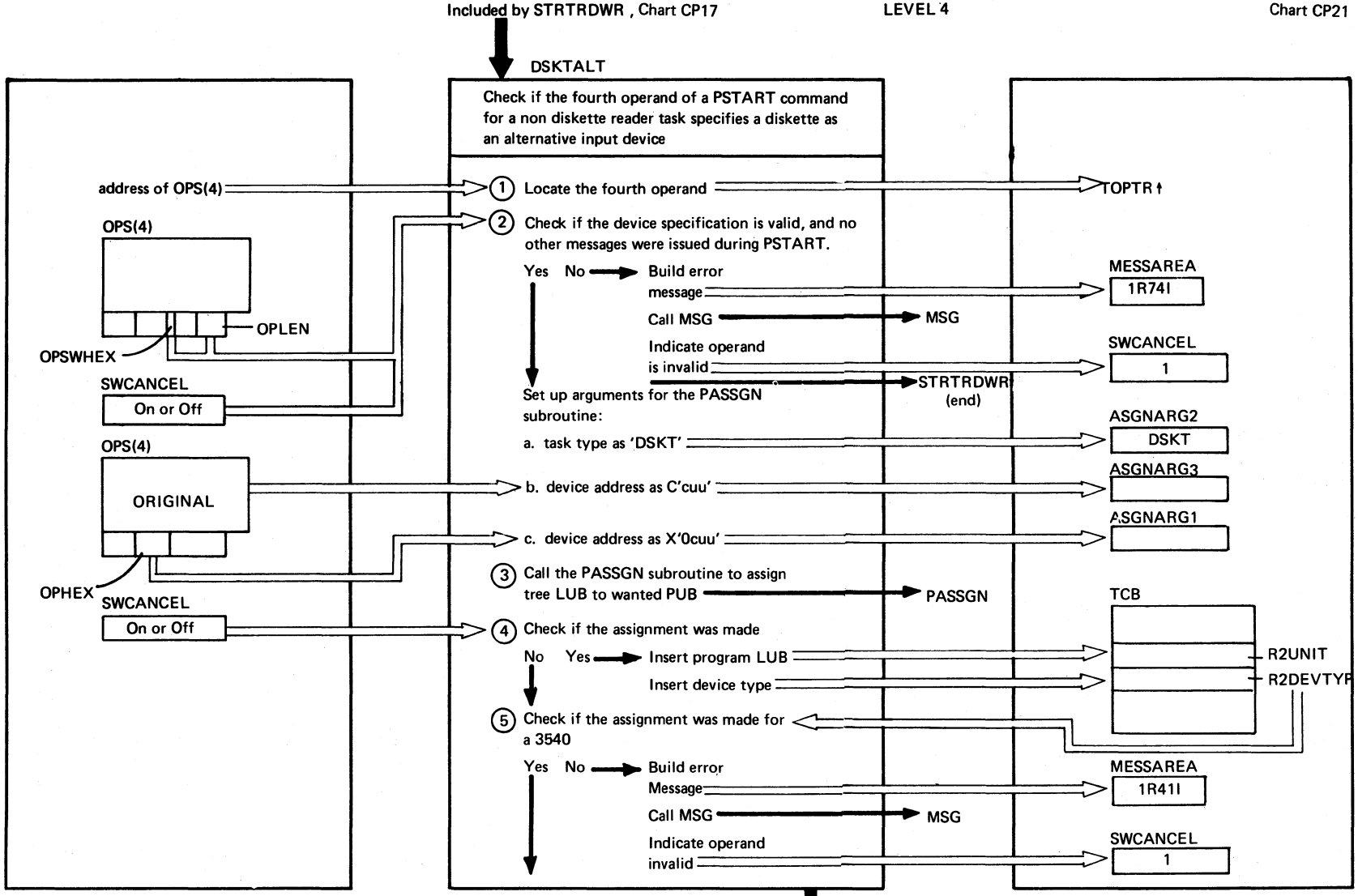
Included by STRTDWR, Chart CP17 LEVEL4 Chart CP20



(continued on next page)

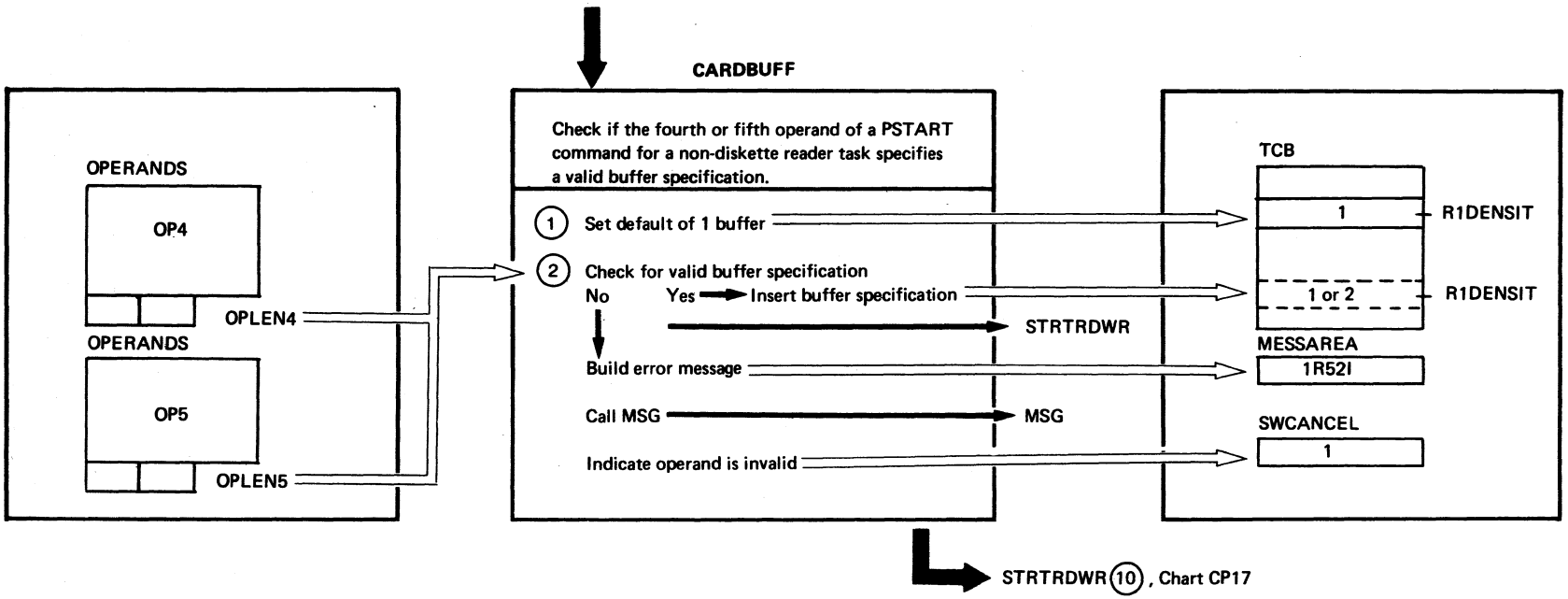


Extended Description		Include Segment	Call Subroutine	Chart
(2)	1R45I OPERAND 4 TOO LONG		MSG	CP75
(4)	1R44I OPERAND 5 INCORRECT		MSG	CP75
(5)	1R43I OPERAND 6 INCORRECT		MSG	CP75
(6)	1R42I OPERAND 7 INCORRECT		MSG	CP75



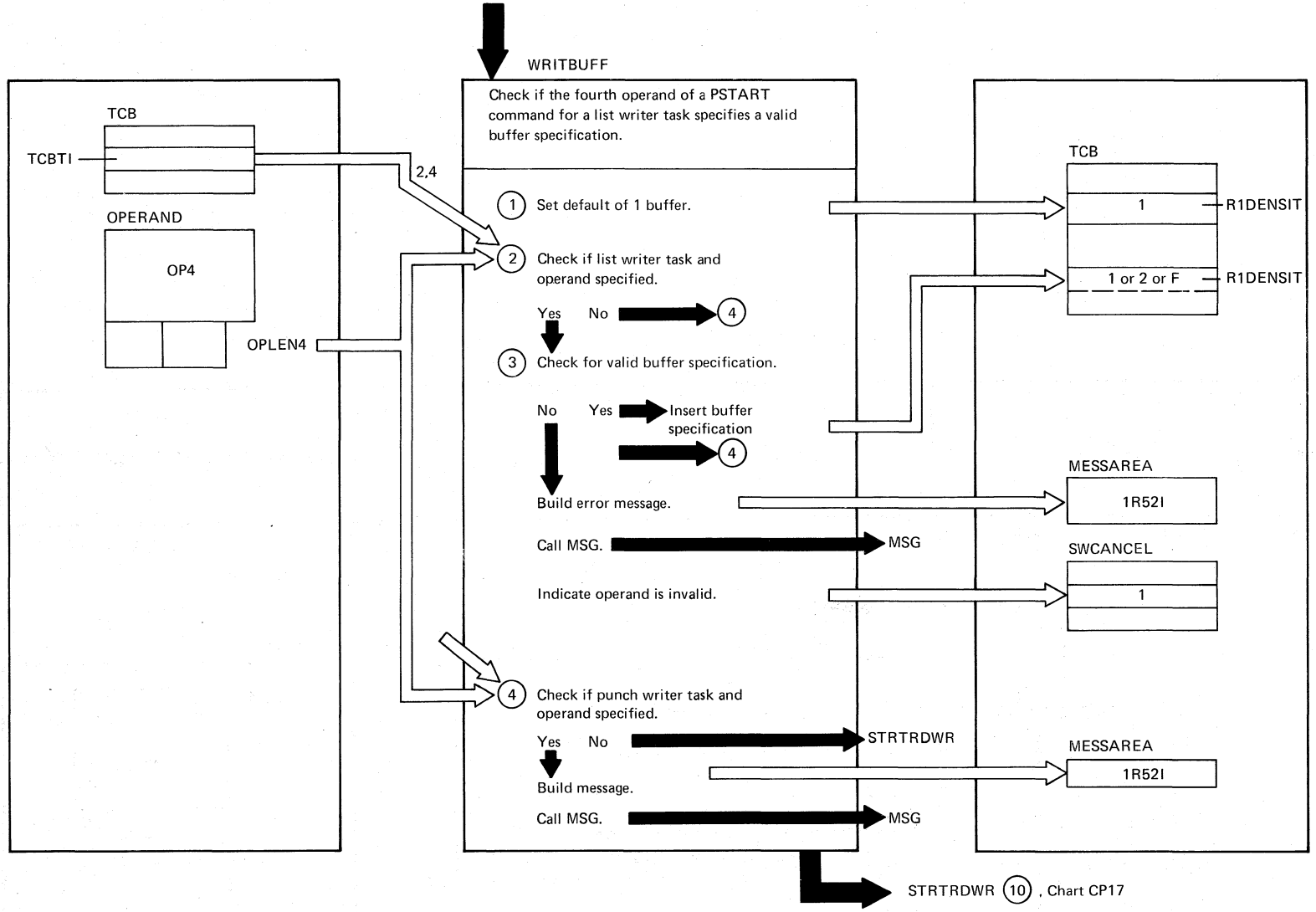
Extended Description		Include Segment	Call Subroutine	Chart
(2)	1R74I INVALID DEVICE SPECIFICATION		MSG	CP75
(3)			PASSGN	CP76
(5)	1R41I INVALID SPECIFICATION FOR DISKETTE		MSG	CP75

Included by STRTRDWR, Chart CP17 LEVEL 4 Chart CP22



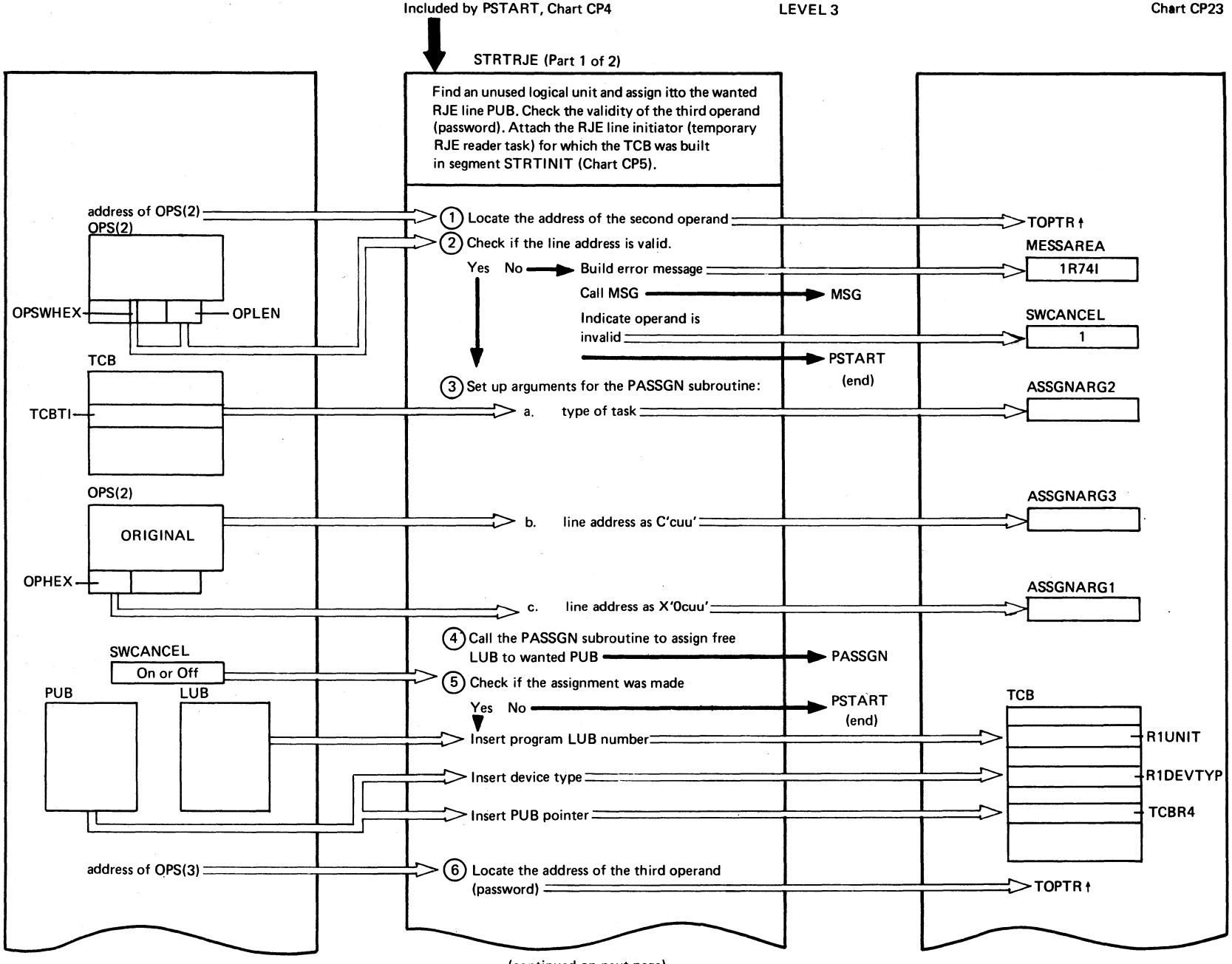
Extended Description	Include Segment	Call Subroutine	Chart
(2) 1R52I BUFFER SPECS NOT 1 or 2		MSG	CP75

Included by STRTRDWR, Chart CP17



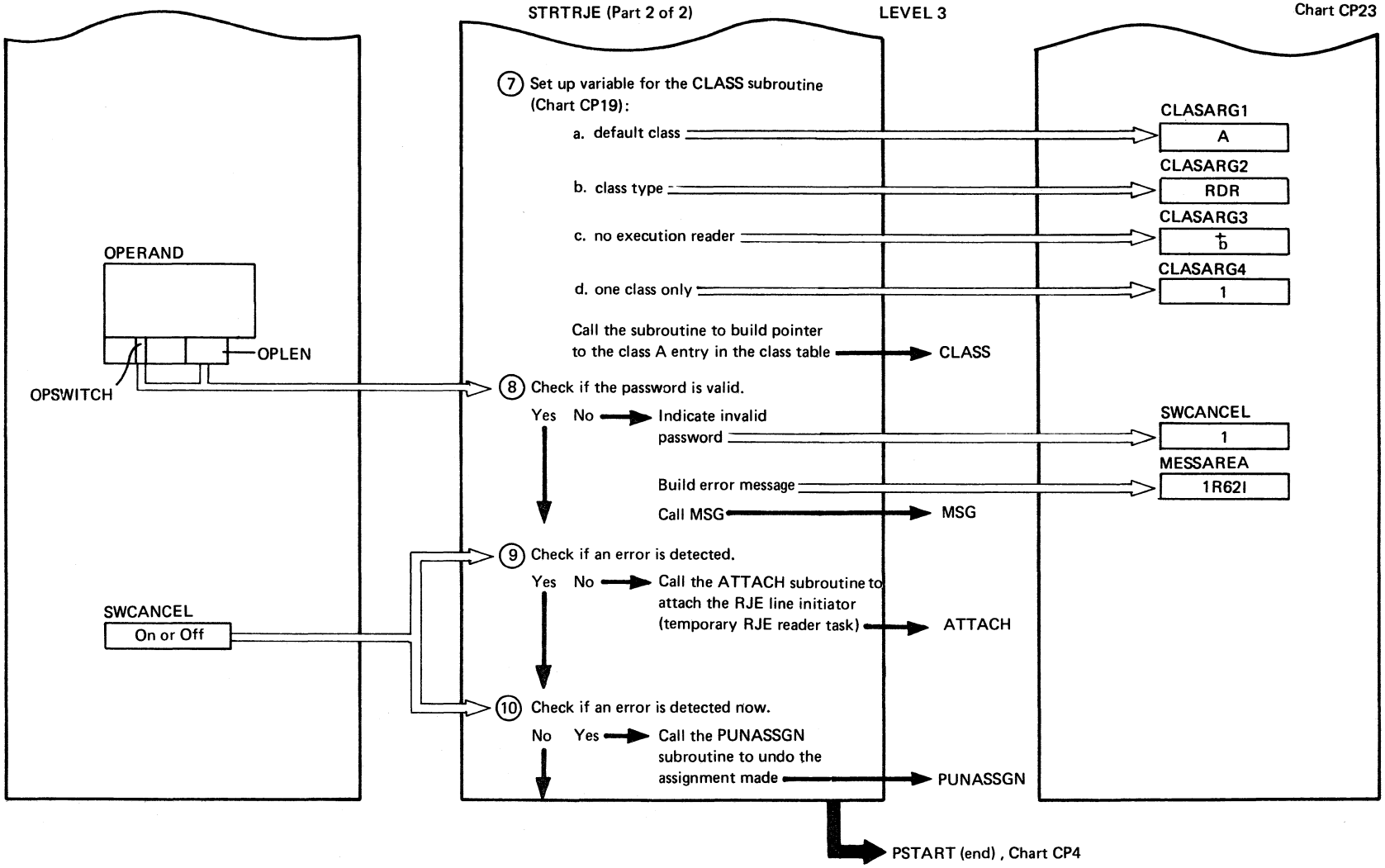
Extended Description	Include Segment	Call Subroutine	Chart
(1)			
(2) If the operand is not specified the length is set to zero.			
(3) Valid specifications are: 1, 2, or D Error message is: 1R52I INVALID BUFFER SPECIFICATION		MSG	CP75
(4) 1R52I OPERAND 4 IGNORED		MSG	CP75

Included by PSTART, Chart CP4 LEVEL 3 Chart CP23

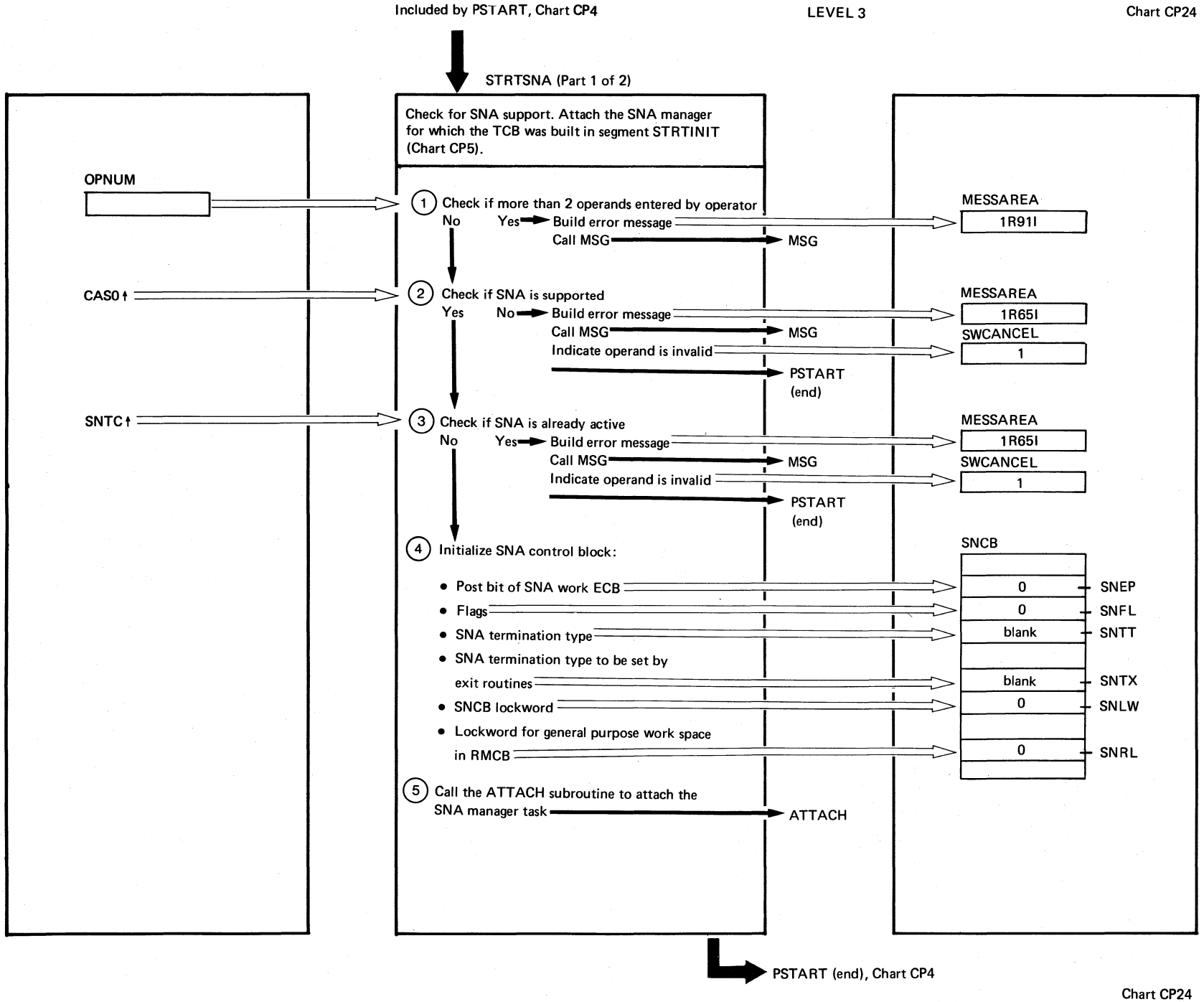


(continued on next page)

Chart CP23



Extended Description	Include Segment	Call Subroutine	Chart
(2) 1R741 INVALID LINE ADDRESS		MSG	CP75
(7)		CLASS	CP86
(8) 1R621 INVALID RJE PASSWORD		MSG	CP75
(9)		ATTACH	CP89
(10)		PUNASSGN	CP80


Included by PSTART, Chart CP4
LEVEL 3
Chart CP24
STRTSNA (Part 1 of 2)
Check for SNA support. Attach the SNA manager for which the TCB was built in segment STRTINIT (Chart CP5).
OPNUM
CASO
SNTC
1 Check if more than 2 operands entered by operator
No
Yes
Build error message
Call MSG
MSG
MESSAREA
1R91I
2 Check if SNA is supported
Yes
No
Build error message
Call MSG
MSG
MESSAREA
1R65I
SWCANCEL
1
Indicate operand is invalid
PSTART
(end)
3 Check if SNA is already active
No
Yes
Build error message
Call MSG
MSG
MESSAREA
1R65I
SWCANCEL
1
Indicate operand is invalid
PSTART
(end)
4 Initialize SNA control block:
SNCB
• Post bit of SNA work ECB
0
SNEP
• Flags
0
SNFL
• SNA termination type
blank
SNTT
• SNA termination type to be set by exit routines
blank
SNTX
• SNCB lockword
0
SNLW
• Lockword for general purpose work space in RMCB
0
SNRL
5 Call the ATTACH subroutine to attach the SNA manager task
ATTACH
PSTART (end), Chart CP4
Chart CP24

Extended Description	Include Segment	Call Subroutine	Chart
(1) 1R91I TOO MANY OPERANDS, FIRST n PROCESSED		MSG	CP75
(2) 1R65I RJE,SNA NOT SUPPORTED		MSG	CP75
(3) 1R65I RJE, SNA ALREADY STARTED		MSG	CP75
(5)		ATTACH	CP89

Included by MAINLINE, Chart CP2

PALTER (Part 1 of 5)

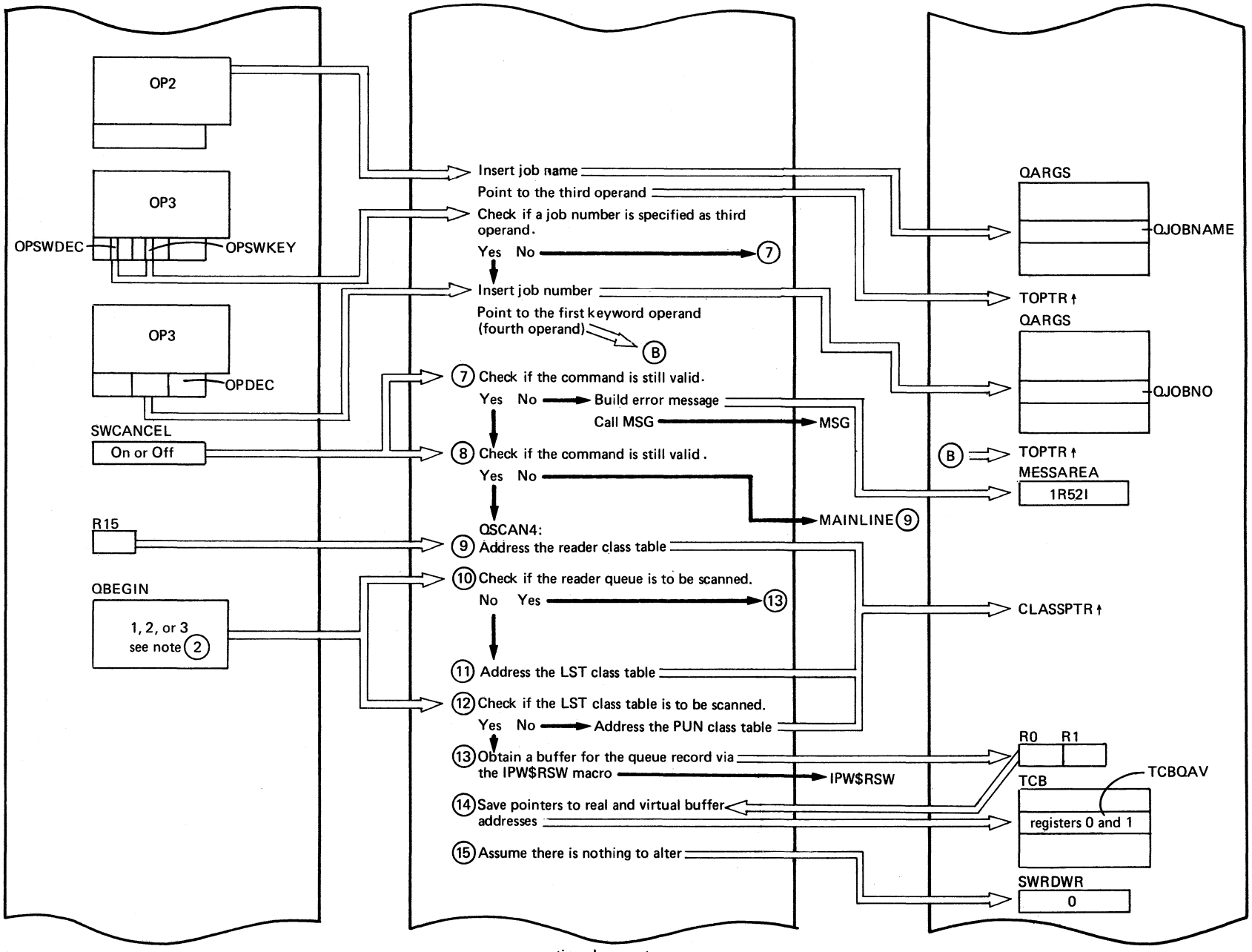
Set up argument fields for the QSCAN4 routine contained within this segment. Check the validity of the second and third operands. Scan the reader, list, or punch queue to find the jobs to be altered.

OPERANDS
OP1
OP2
CPB
CPBID
OP2
OPSWSTAR
OPLEN
SWCANCEL
On or Off
OP2
OPSWSTAR
OPLEN
OPSWALFM
SWCANCEL
On or Off
OP2
OPLEN
OPSWALFM
OP2
OPLEN

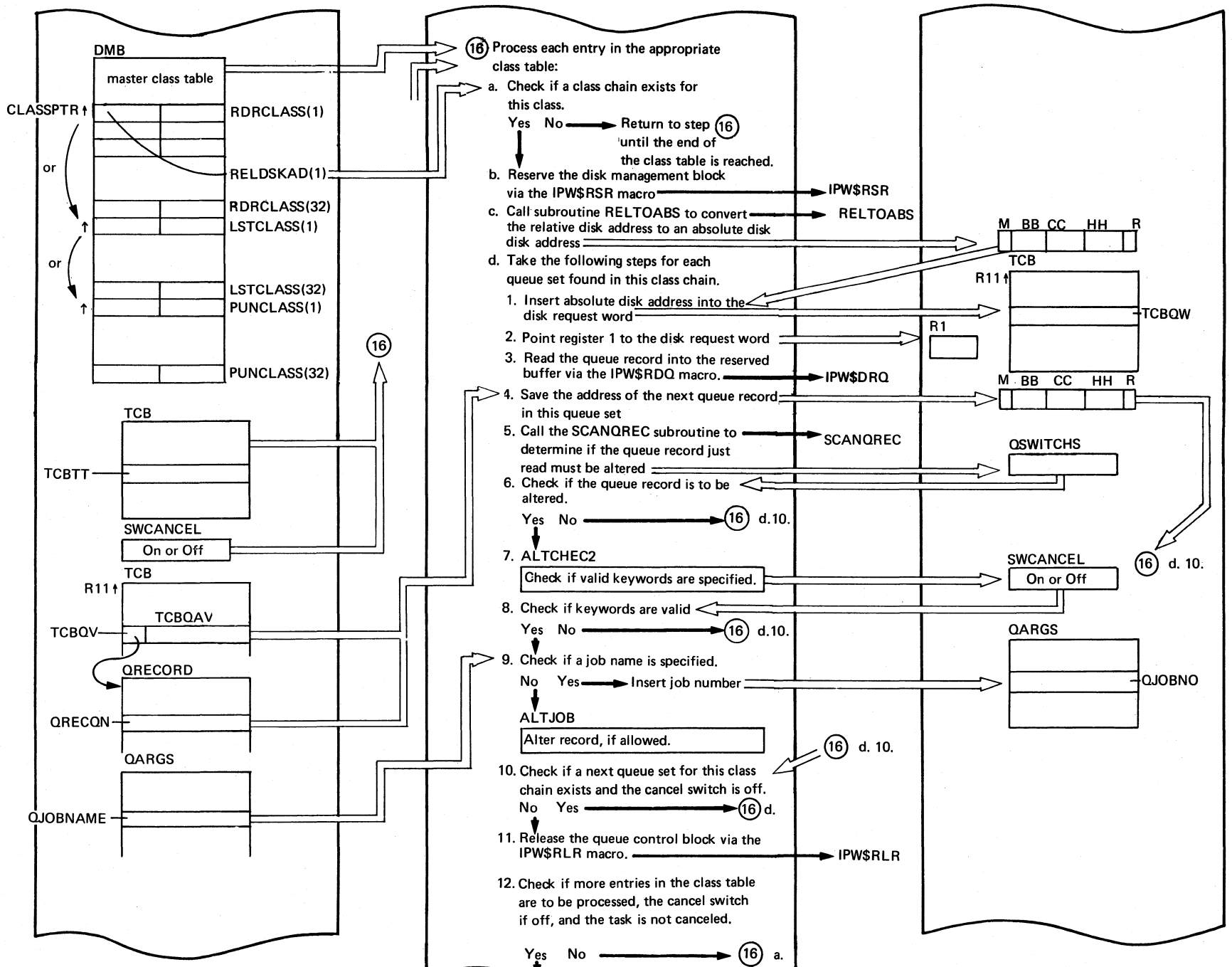
1 Clear arguments for the QSCAN4 (Step 9) routine
2 Check if the first operand specifies a valid queue name and store the result
Yes No → Build error message (B)
Call MSG → MSG
8
3 Store user-ID in arguments for the QSCAN4 (Step 9) routine
4 Check if the second operand is ALL or omitted.
No Yes → Indicate command is valid
Point to the first keyword operand (third operand)
5 Check if the cancel switch is still on and QUEUE,*ABC is specified.
No Yes
a. Indicate command is valid
b. Insert operand length into argument field
c. Insert operand
d. Point to the first keyword operand
6 Check if the cancel switch is still on and QUEUE,jobname or QUEUE,class is specified.
Yes No → 7
Check if QUEUE,class is specified.
No Yes → Indicate command is valid
A → Insert class character
Point to the first keyword operand
7

QARGS
QBEGIN
QIDBIN
B
MESSAREA
1R52I
SWCANCEL
0
TOPTR
SWCANCEL
0
QARGS
QABCLEN
QABCCONT
TOPTR
SWCANCEL
0
QARGS
QCLASS
TOPTR

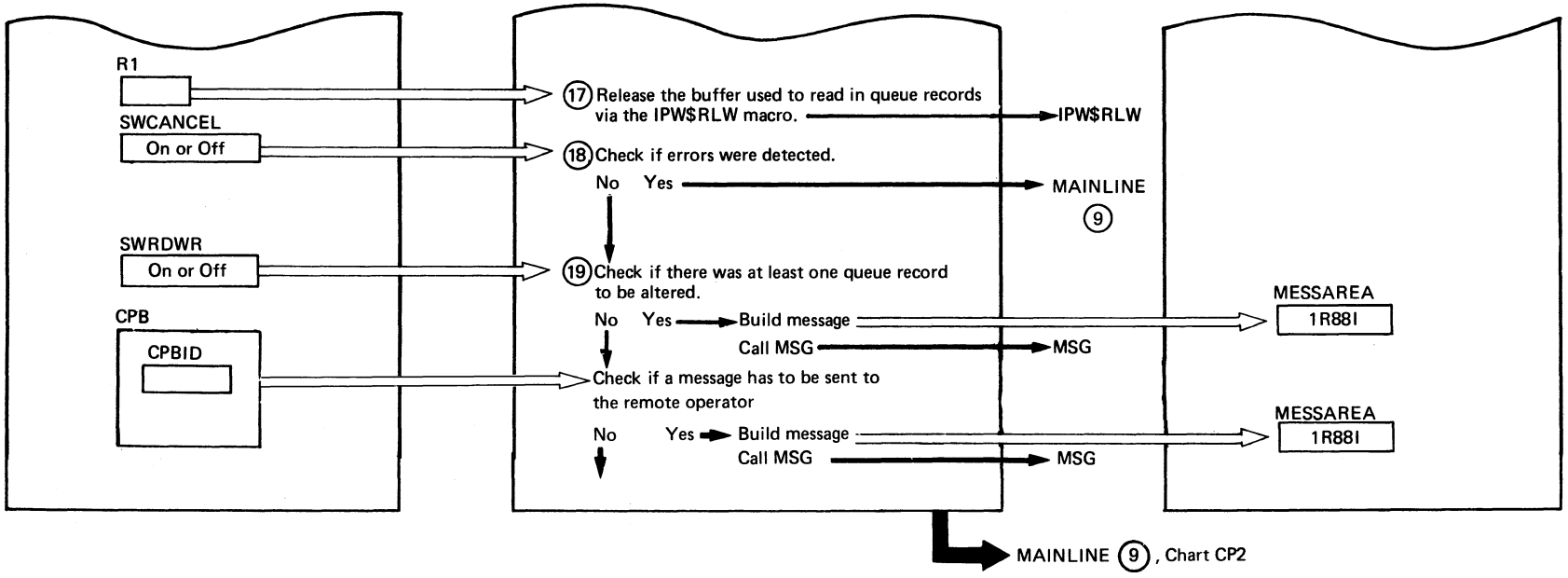
continued on next page



continued on next page

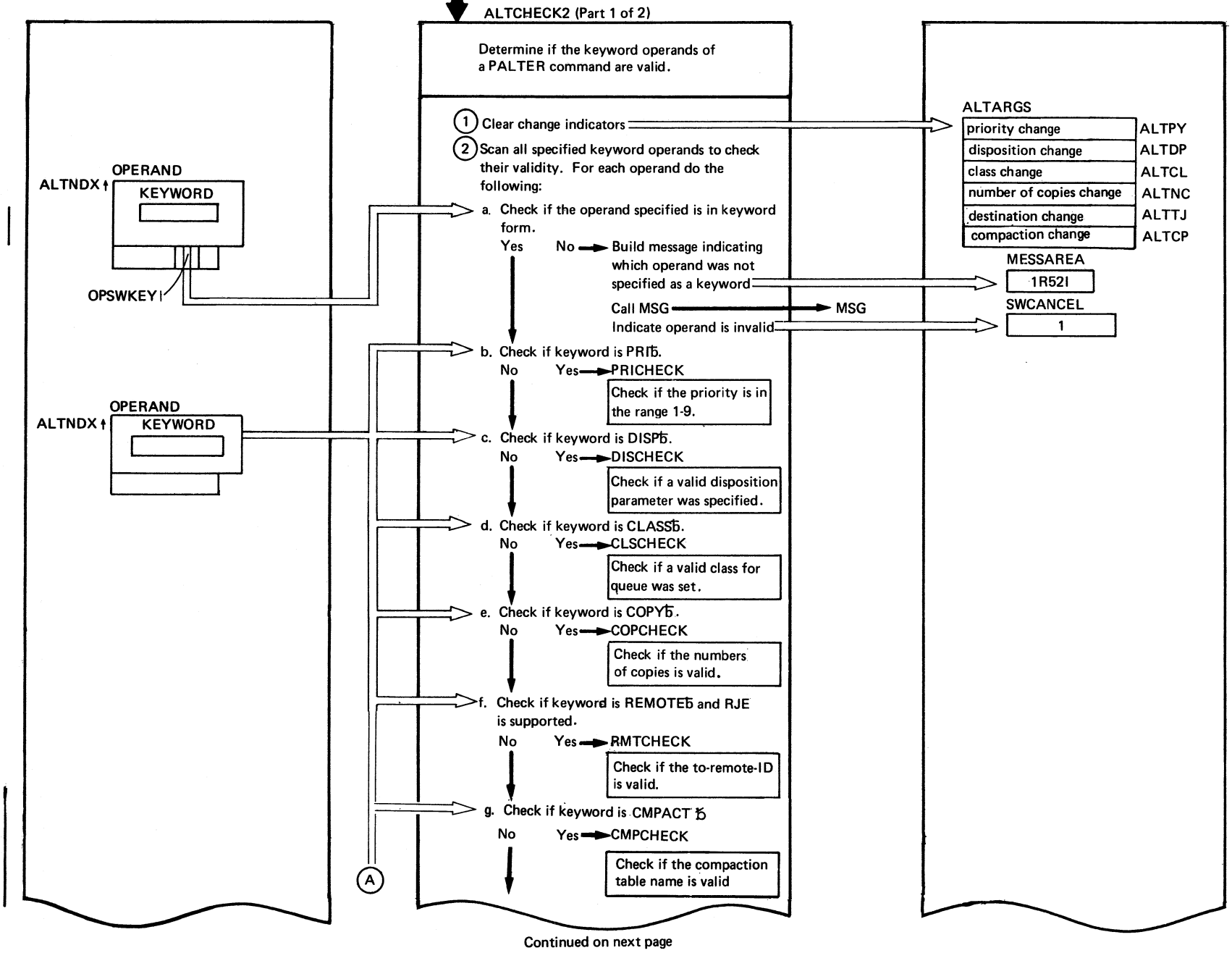


continued next page


R1
SWCANCEL
On or Off
SWRDWR
On or Off
CPB
CPBID
(17) Release the buffer used to read in queue records via the IPW$RLW macro.
IPW$RLW
(18) Check if errors were detected.
No Yes
MAINLINE (9)
(19) Check if there was at least one queue record to be altered.
No Yes
Build message
Call MSG
MSG
MESSAREA
1R88I
Check if a message has to be sent to the remote operator
No Yes
Build message
Call MSG
MSG
MESSAREA
1R88I
MAINLINE (9), Chart CP2

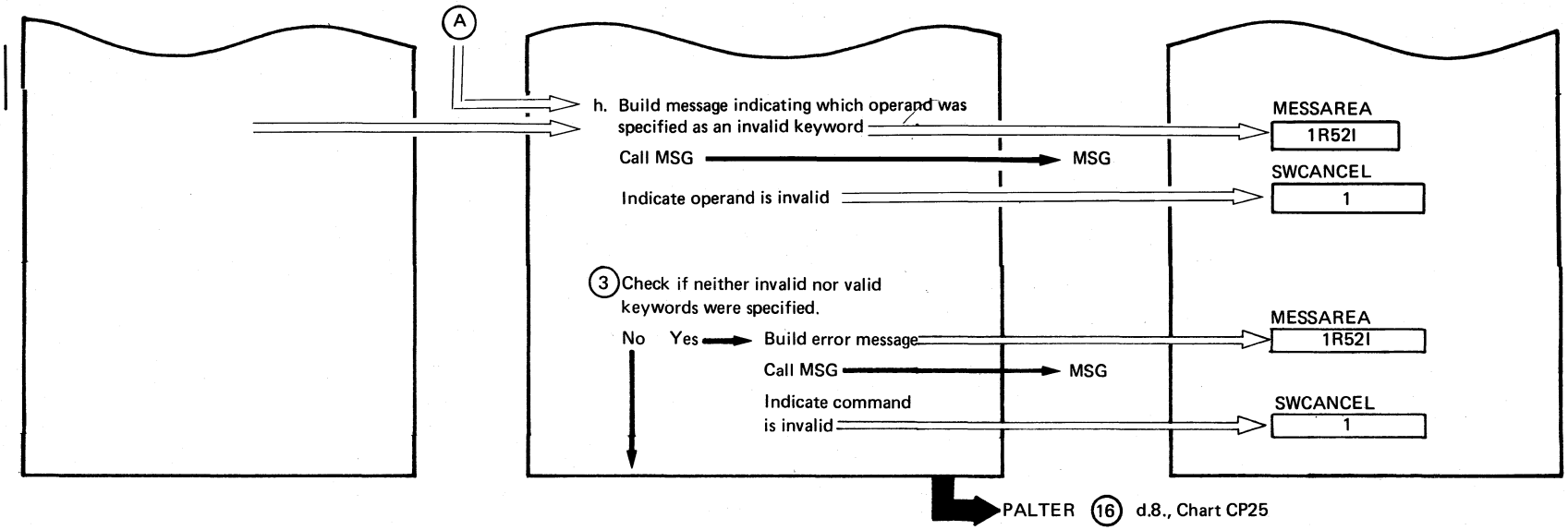
Extended Description	Include Segment	Call Subroutine	Chart
(1) All arguments are set to zero means that all arguments initially meet the selection for process criteria.			
(2) Valid queue names are: RDR, PUN, LST, or PRT for RDR QBEGIN = 1 for PUN QBEGIN = 3 for LST or PRT QBEGIN = 2 } used to indicate which queue is to be scanned.			
1R52I OPERAND 1 NO VALID QUEUE		MSG	CP75
(3) Only the central operator is allowed to alter local jobs.			
(5) The OPSWALFM switch set in subroutine SCANOPND (Chart CP71), if on, indicates a valid alphameric string.			
(7) 1R52I OPERAND 2 INVALID		MSG	CP75
(9) Register 15 is used to address the disk management block. The pointer to the DMB is CAQC. To address the reader class table, add the displacement to the master class table in register 15.			
(13) The IPW$RSW macro uses registers 0, 1, 2, and 3.			
(16) c.		RELTOABS	CP84
d.5.		SCANQREC	CP85
d.7.	ALTCHECK2		CP26
d.9.	ALTJOB		CP32
(19) 1R88I NOTHING TO ALTER		MSG	CP75
1R88I CP READY		MSG	CP75

Included by PALTER, Chart CP25 LEVEL 3 Chart CP26



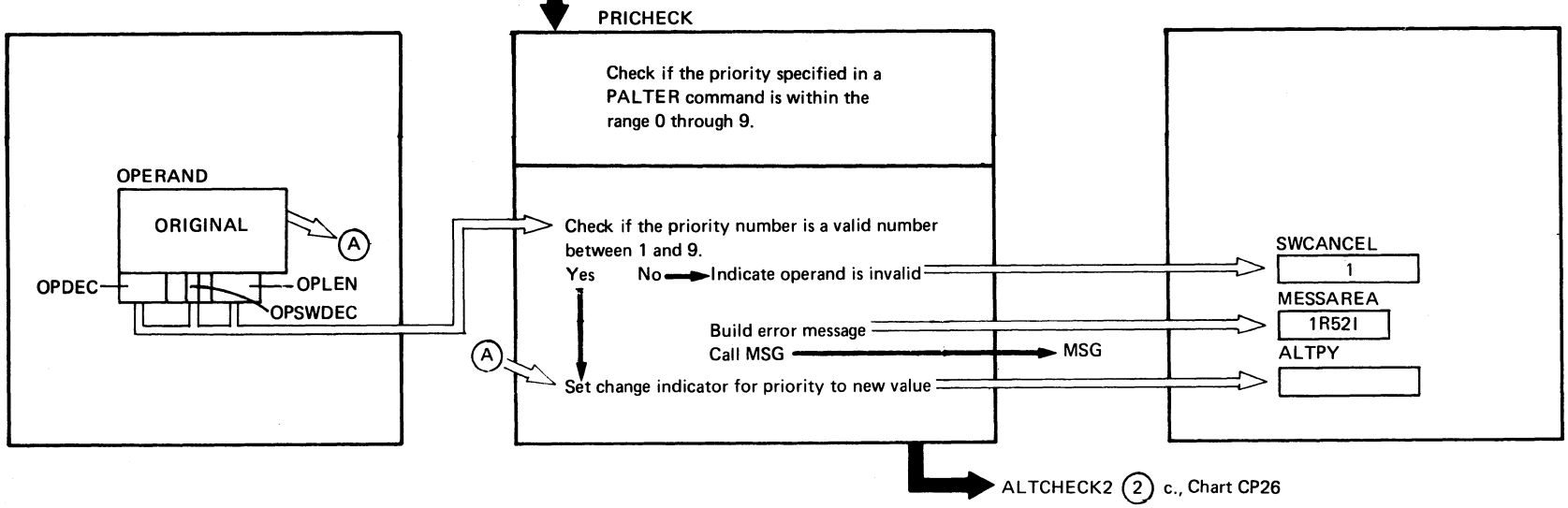
Continued on next page

Chart CP26



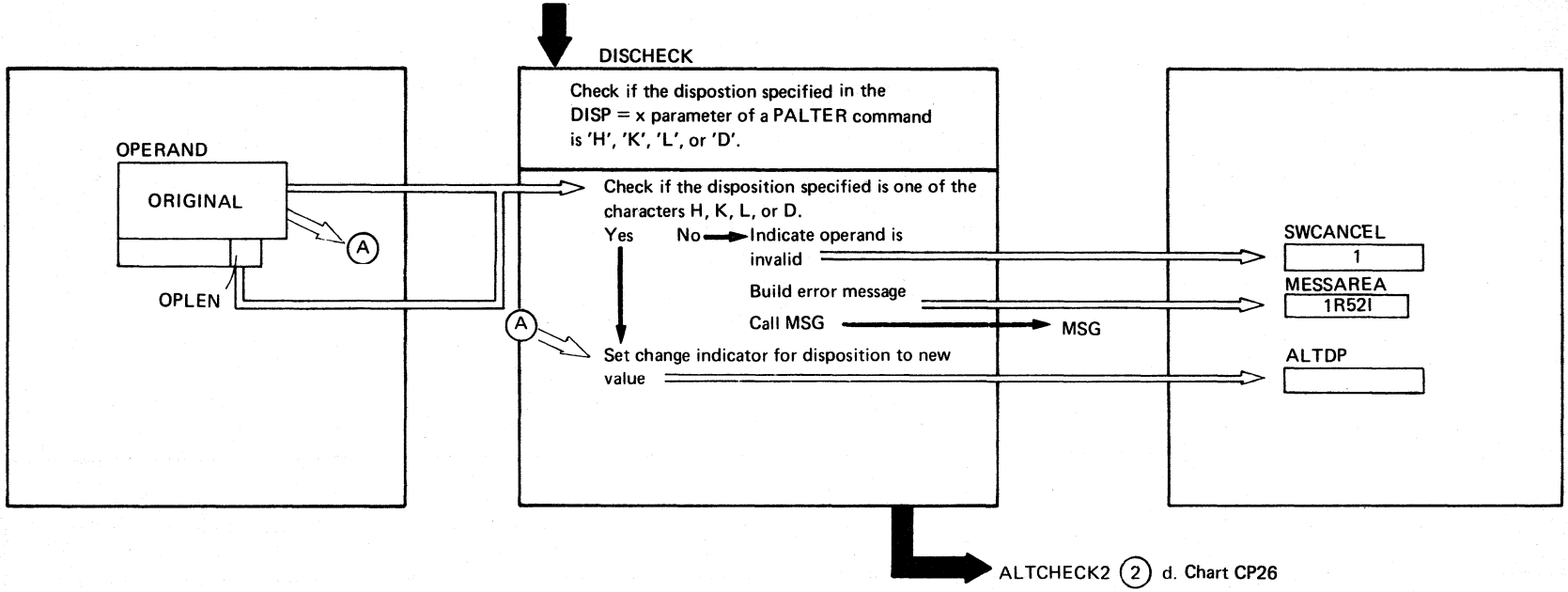
Extended Description	Include Segment	Call Subroutine	Chart
(2) a. 1R52I OPERAND x NOT SPECIFIED AS KEYWORD		MSG	CP
b.	PRICHECK		CP
c.	DISCHECK		CP
d.	CLSCHECK		CP
e.	COPCHECK		CP
f.	RMTCHECK		CP34
g.	CMPCHECK		CP31.1
h. 1R52I OPERAND x INVALID KEYWORD		MSG	CP
(3) 1R52I NO VALID KEYWORD SPECIFIED		MSG	CP75

Included by ALTCHECK2, Chart CP26 LEVEL 4 Chart CP27



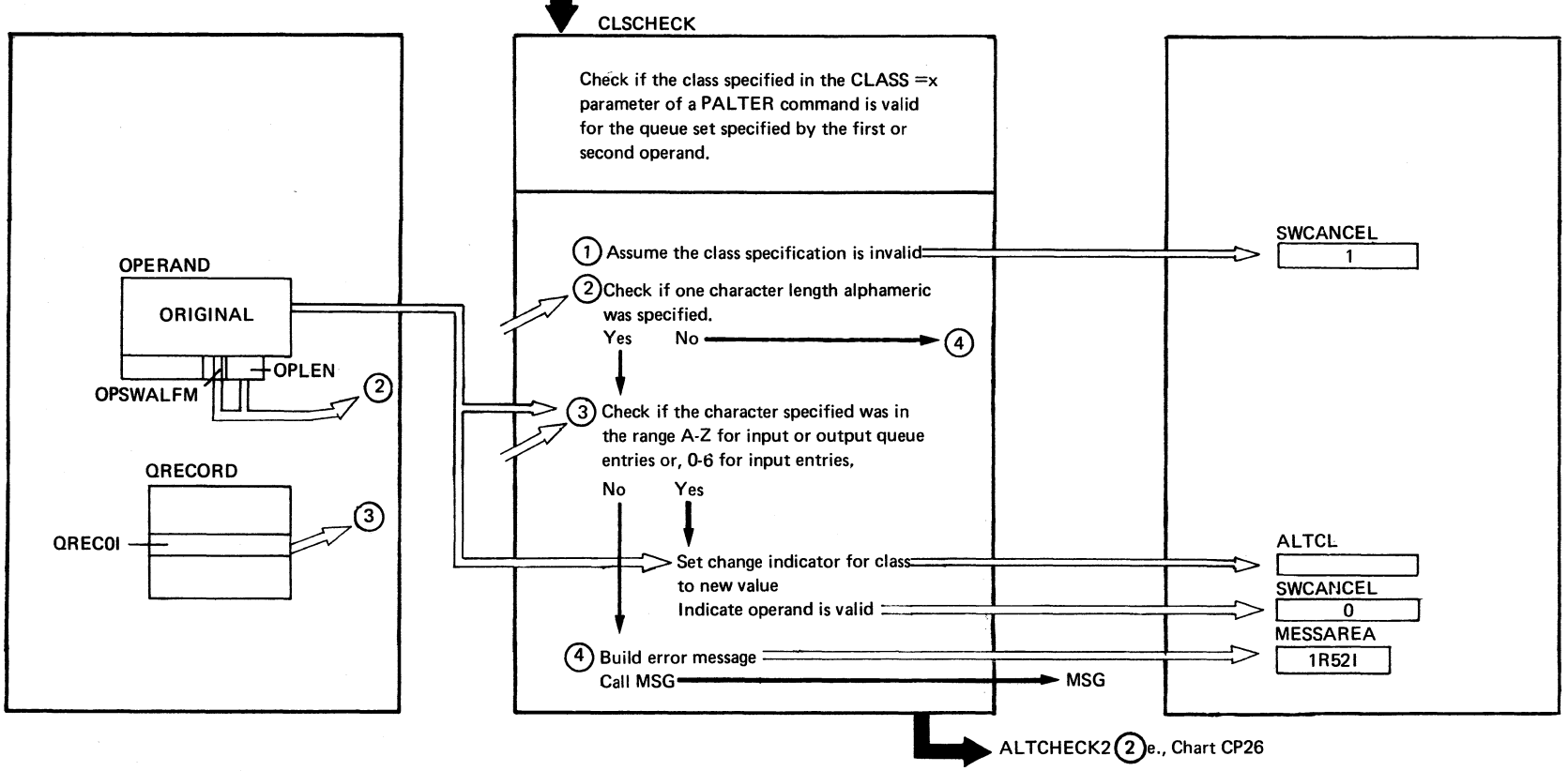
Extended Description	Include Segment	Call Subroutine	Chart
1R521 INVALID PRIORITY		MSG	CP75

Included by ALTCHECK2, Chart CP26 LEVEL 4 Chart CP28

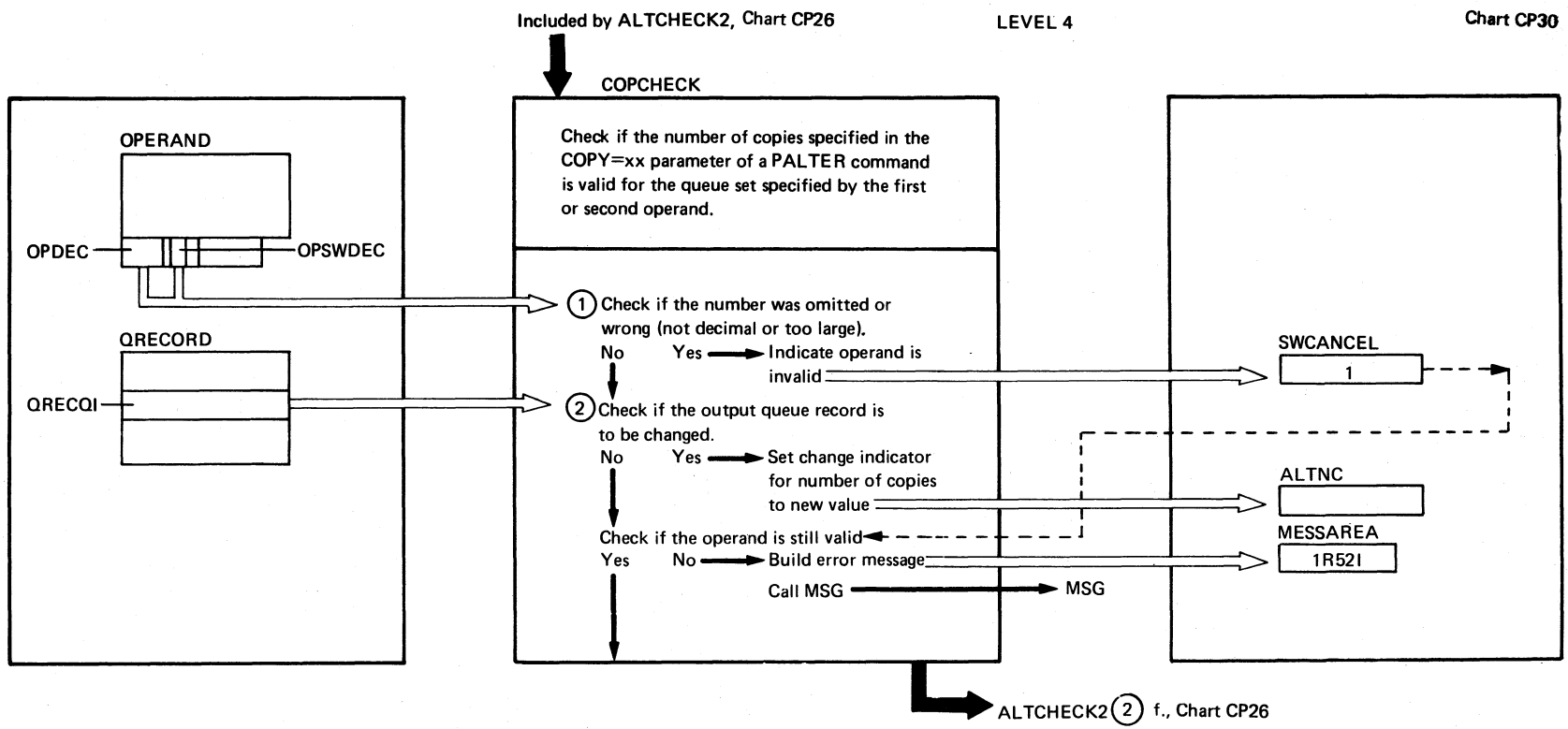


Extended Description	Include Segment	Call Subroutine	Chart
1R52I INVALID DISPOSITION		MSG	CP75

Included by ALTCHECK2, Chart CP26 LEVEL 4 Chart CP29

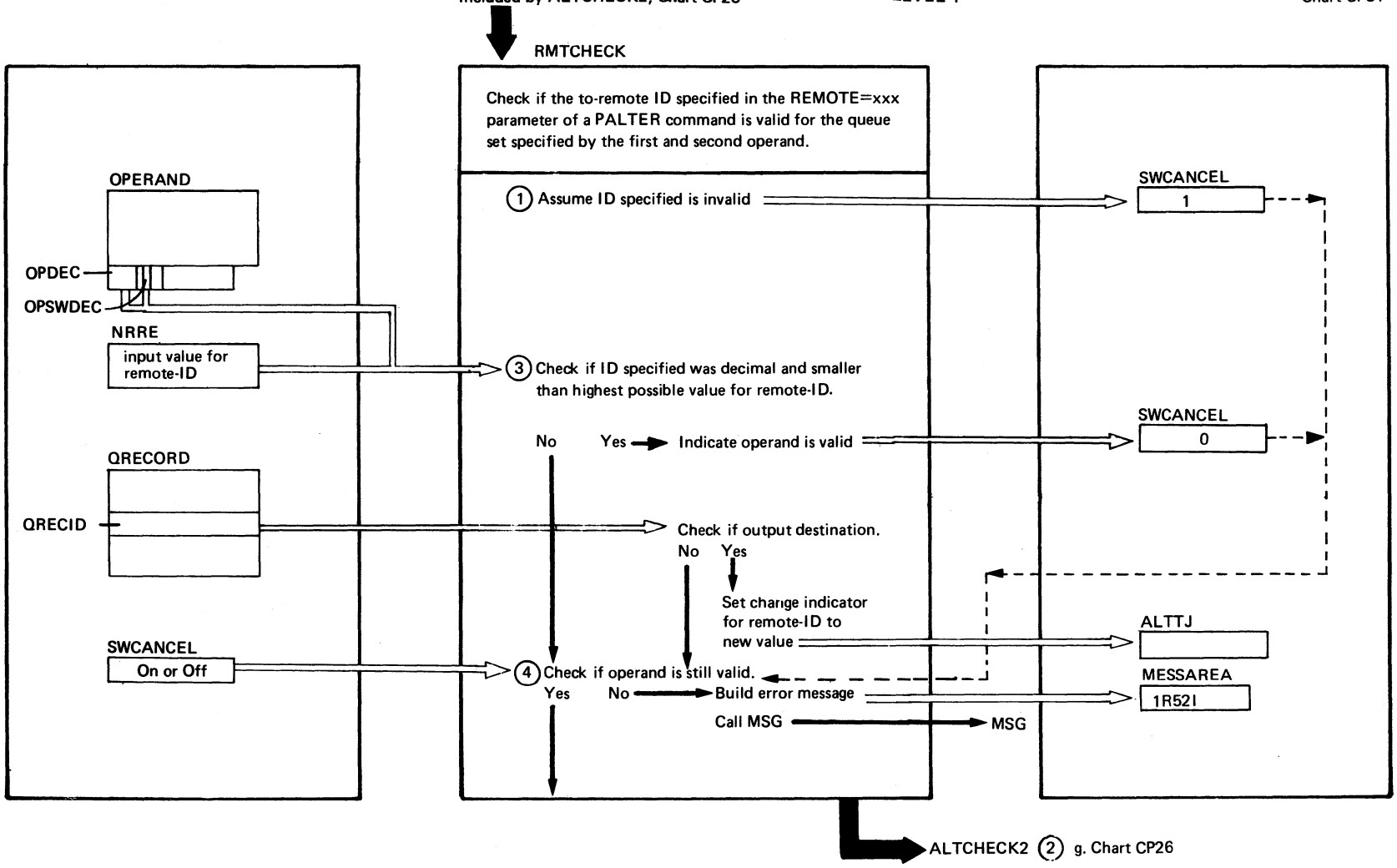


Extended Description	Include Segment	Call Subroutine	Chart
4 1R52I INVALID CLASS		MSG	CP75



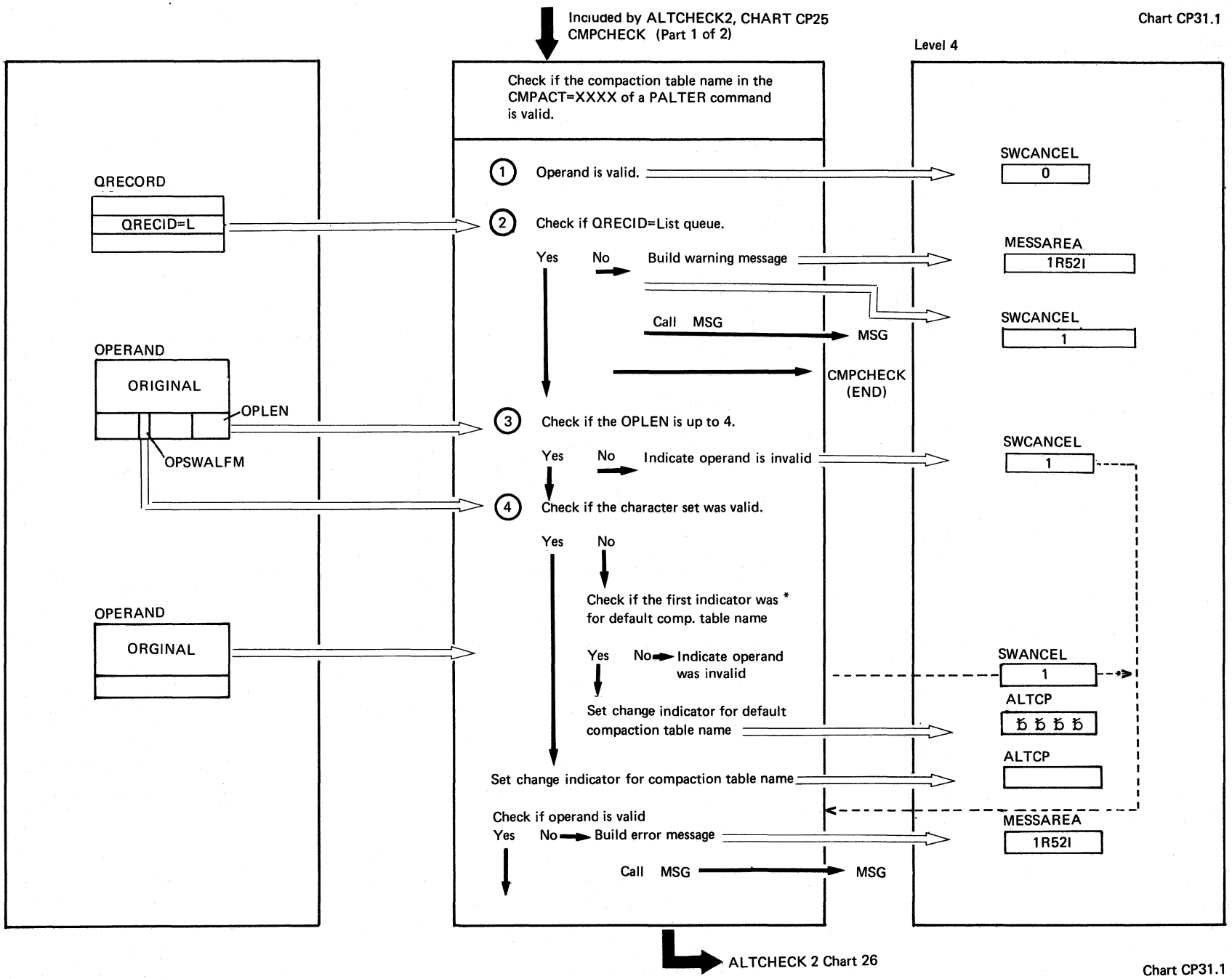
Extended Description	Include Segment	Call Subroutine	Chart
2 1R52I INVALID NUMBER OF COPIES		MSG	CP75

Included by ALTCHECK2, Chart CP26 LEVEL 4 Chart CP31



Extented Description	Include Segment	Call Subroutine	Chart
(4) 1R521 INVALID REMOTE ID		MSG	CP75

Chart CP31


Included by ALTCHECK2, CHART CP25
CMPCHECK (Part 1 of 2)
Chart CP31.1
Level 4
Check if the compaction table name in the CMPACT=XXXX of a PALTER command is valid.
QRECORD
QRECID=L
OPERAND
ORIGINAL
OPLEN
OPSWALFM
OPERAND
ORGINAL
1 Operand is valid.
SWCANCEL
0
2 Check if QRECID=List queue.
Yes
No
Build warning message
MESSAREA
1R52I
Call MSG
MSG
SWCANCEL
1
CMPCHECK (END)
3 Check if the OPLEN is up to 4.
Yes
No
Indicate operand is invalid
SWCANCEL
1
4 Check if the character set was valid.
Yes
No
Check if the first indicator was * for default comp. table name
Yes
No
Indicate operand was invalid
SWANCEL
1
Set change indicator for default compaction table name
ALTCP
ƀ ƀ ƀ ƀ
Set change indicator for compaction table name
ALTCP
Check if operand is valid
Yes
No
Build error message
MESSAREA
1R52I
Call MSG
MSG
ALTCHECK 2 Chart 26
Chart CP31.1

Extended Description	Include Segment	Call Subroutine/Macro	Chart
(1) If QRECID ≠ L, then the queue is not LST and the CMPACT operand is ignored. Call MSG for message: CMPACT IGNORED.			
(2) Check if CMPACT name length is up to 4 characters and a valid alphameric string was specified.			
(3) If the compaction table name is an invalid character string, issue error message, otherwise check if the first character is an * for default compaction table name.			
(4) 1R52I INVALID CMPACT NAME		MSG	CP75
or			
1R52I NO VALID KEYWORD SPECIFIED		MSG	CP75

Included by PALTER, Chart CP25

LEVEL3

ALTJOB (Part 1 of 3)

Check if the queue set may be altered. (It may not if it is being processed, unless only the number of copies is changed.) Remove the queue set from the class chain via an IPW$DQS macro. Make the necessary changes and add the queue set to the same or another class chain via an IPW$AQS macro. If the alter operation results in no changes in the attributes of the queue set, the PALTER command is ignored for this queue set.

ALTARGS
ALTPY
QRECORD
QRECPY

ALTARGS
ALTDP
QRECORD
QRECDP

ALTARGS
ALTCL
QRECORD
QRECCL

ALTARGS
ALTNC
QRECORD
QRECNC
QRECQI
QRECCTJ

ALTTJSW
ALTARGS
ALTTJ

(1) Suppress the message
NOTHING TO ALTER → SWRDWR 1

(2) Assume altering results in no changes → SWALT 0

(3) Check if the priority is to be altered and will result in a different priority.
No
Yes → Indicate altering is necessary

(4) Check if the disposition is to be altered and will result in a different disposition.
No
Yes → Indicate altering is necessary

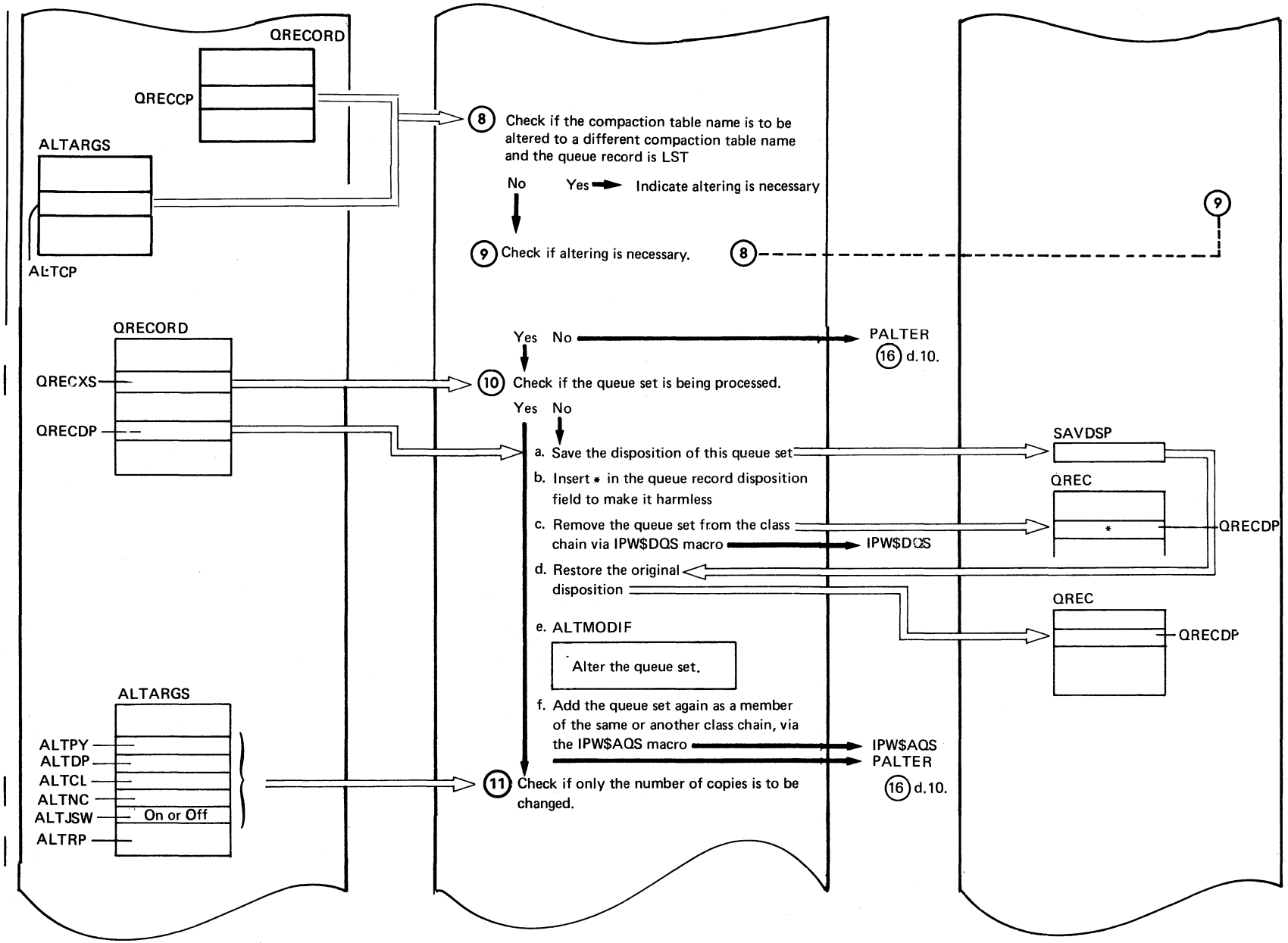
(5) Check if the class is to be altered and will result in a different class.
No
Yes → Indicate altering is necessary → SWALT 1

(6) Check if the number of copies is to be altered and will result in a different number of copies and the queue record does not belong to a reader queue entry.
No
Yes → Indicate altering is necessary

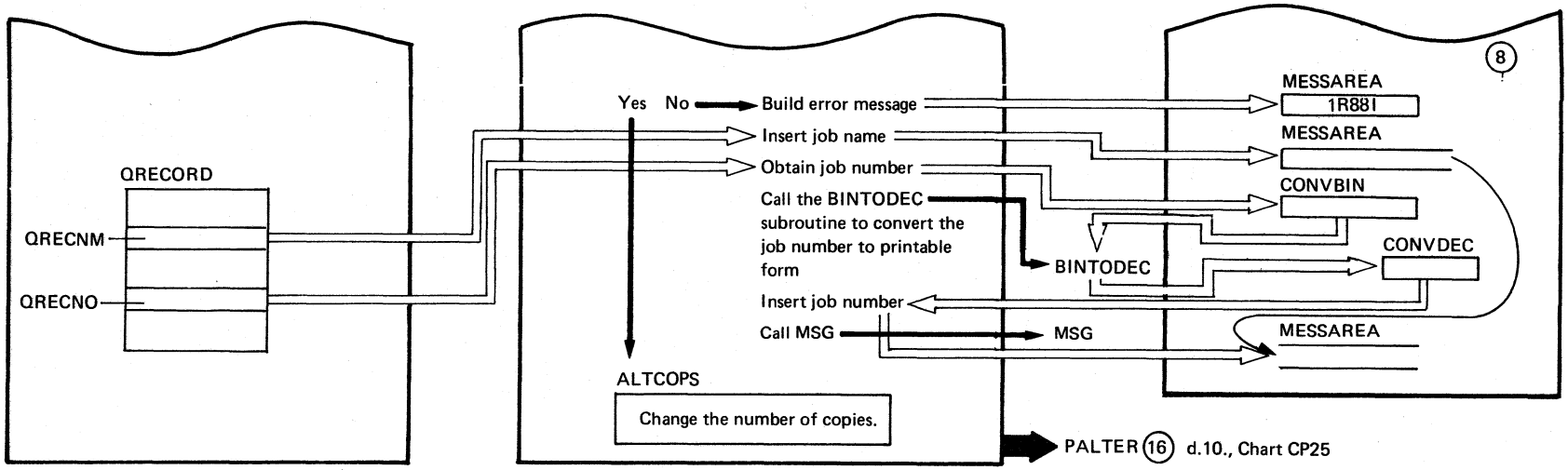
(7) Check if the to-remote-ID is to be altered and will result in a different to-remote-ID, and the queue record does not belong to a reader queue entry.
No
Yes → Indicate altering is necessary

(8)

Continued on next page



Continued on next page



Extended Description	Include Segment	Call Subroutine	Chart
(10) 1R881 JOB xxxxxxxx xxxx CANNOT BE ALTERED job name job number		BINTODEC MSG	CP83 CP75

Included by ALTJOB, Chart CP33 LEVEL 4 Chart CP33

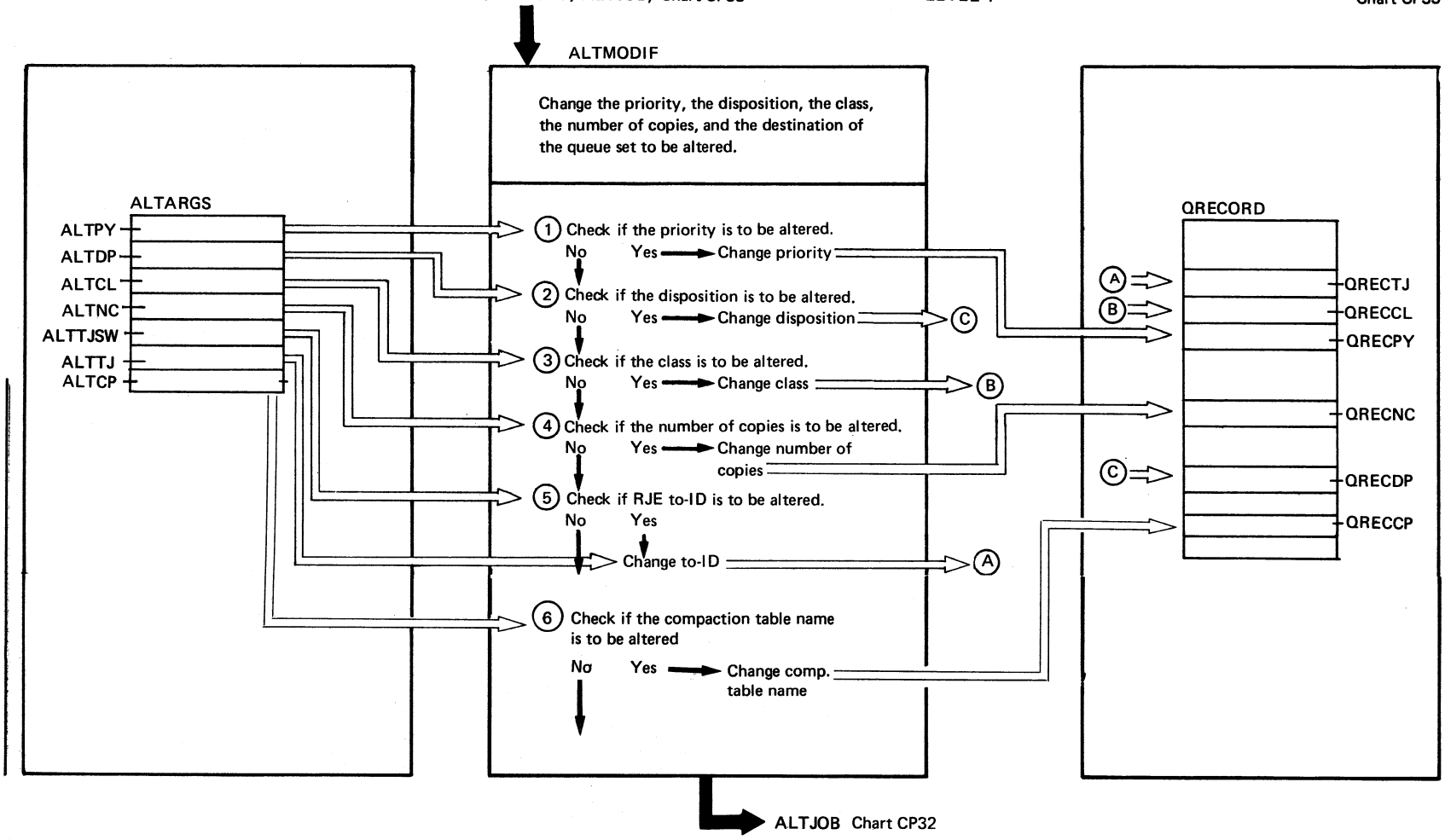
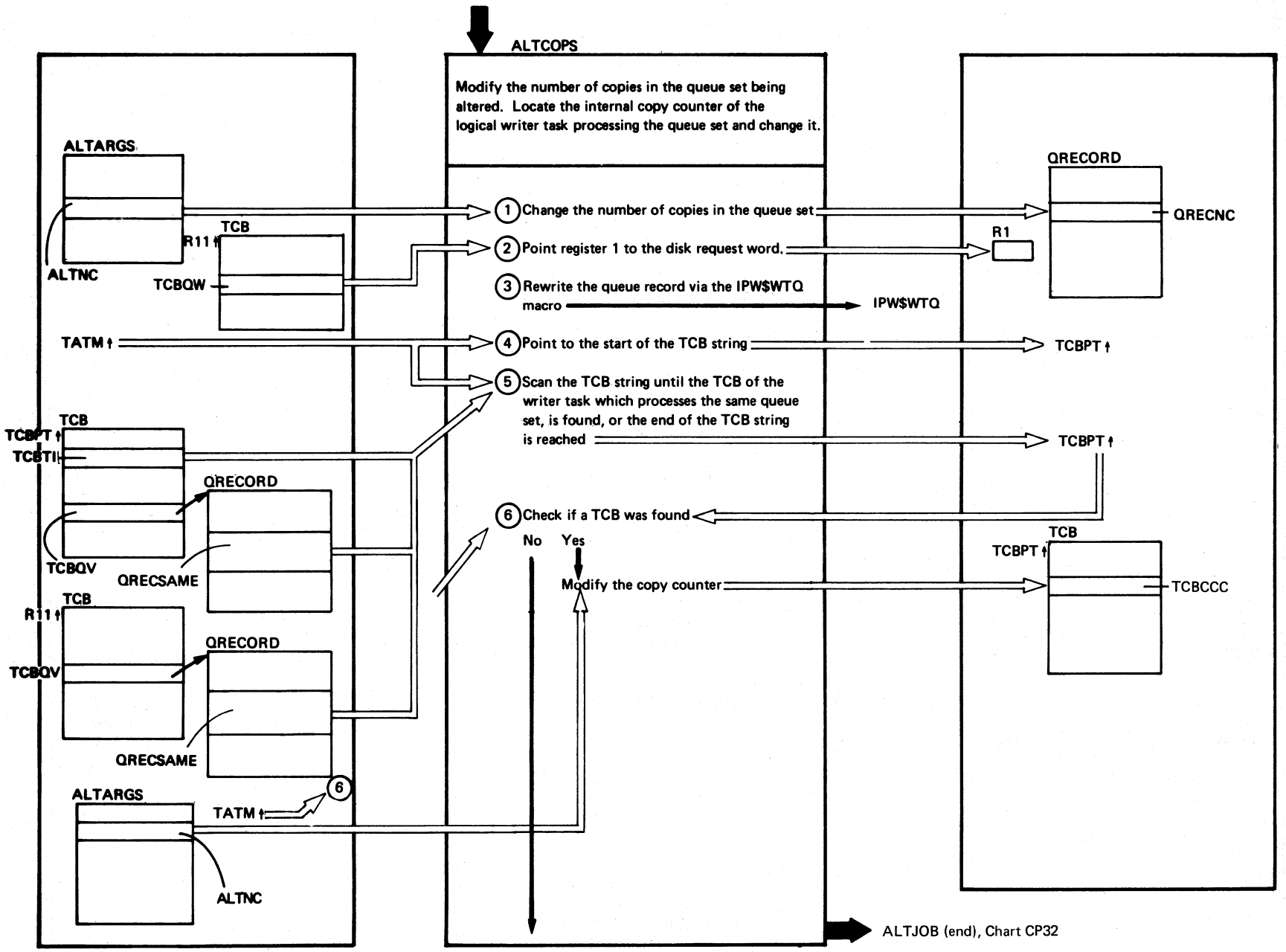
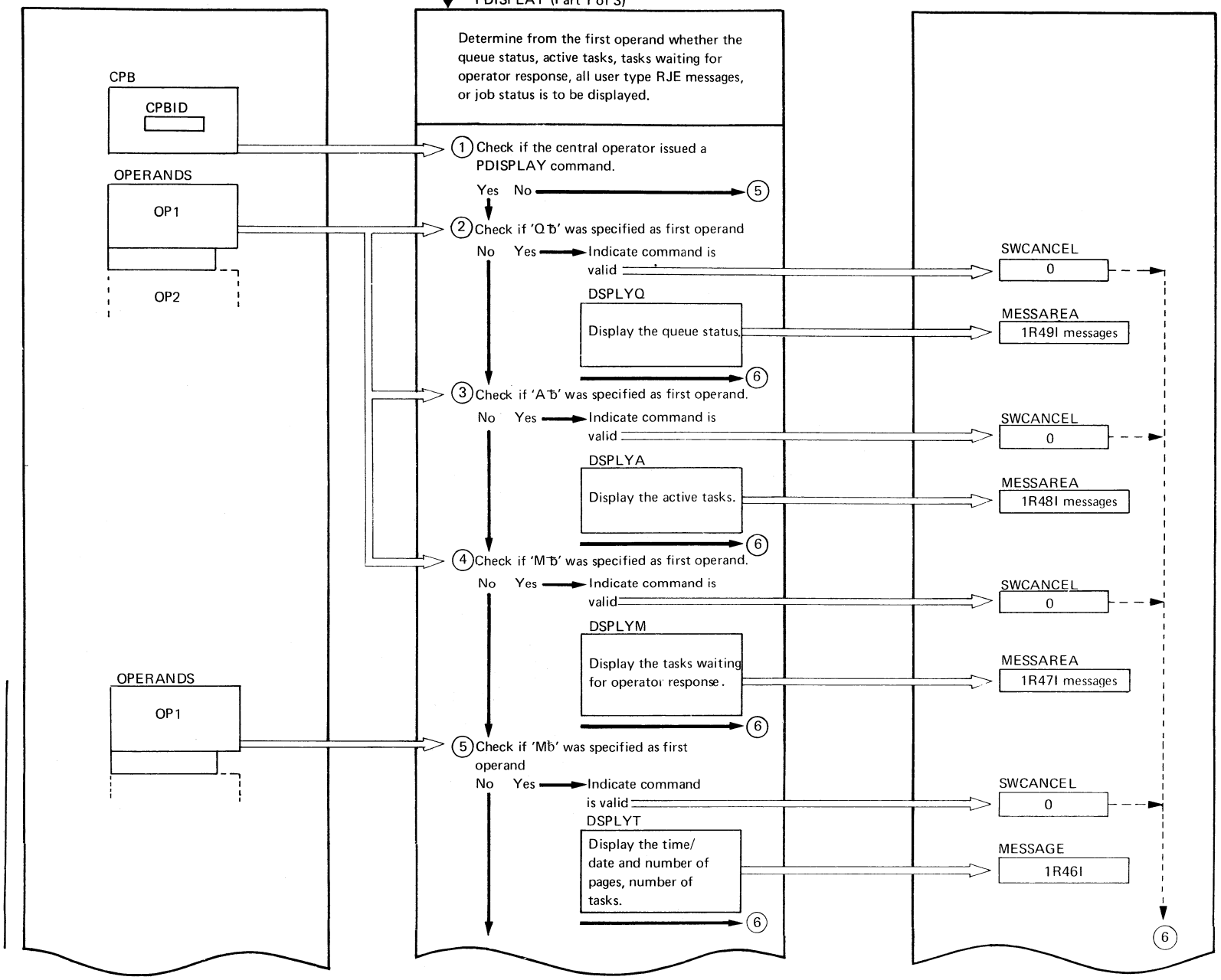


Chart CP33


ALTCOPS
Modify the number of copies in the queue set being altered. Locate the internal copy counter of the logical writer task processing the queue set and change it.
ALTARGS
ALTNC
TCB
R11
TCBQW
TATM
TCBPT
TCBTI
QRECORD
TCBQV
QRECSAME
QRECNC
R1
IPW$WTQ
TCBCCC
1 Change the number of copies in the queue set
2 Point register 1 to the disk request word.
3 Rewrite the queue record via the IPW$WTQ macro
4 Point to the start of the TCB string
5 Scan the TCB string until the TCB of the writer task which processes the same queue set, is found, or the end of the TCB string is reached
6 Check if a TCB was found
No
Yes
Modify the copy counter
ALTJOB (end), Chart CP32

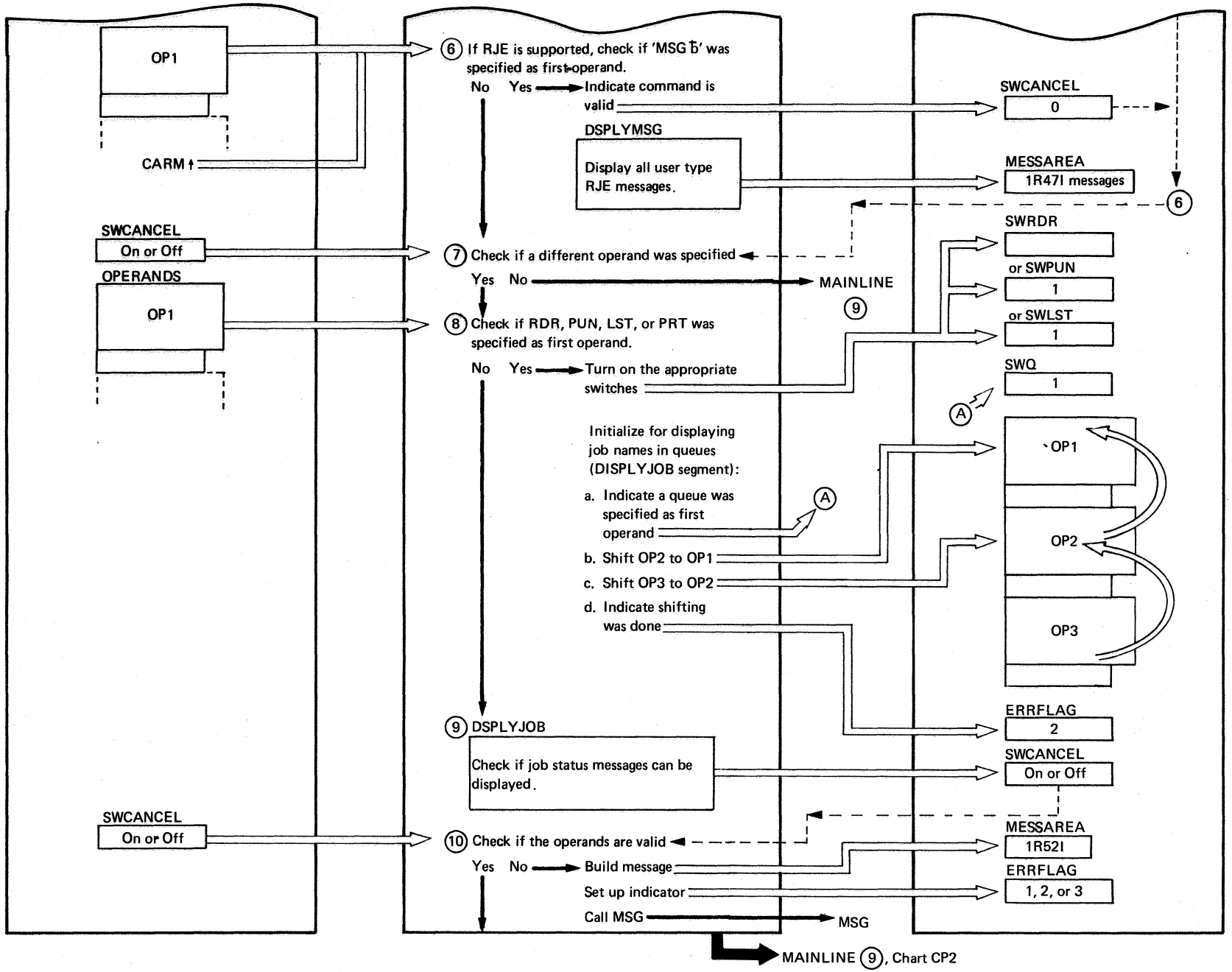
Included by MAINLINE, Chart CP2 LEVEL 2 Chart CP35

PDISPLAY (Part 1 of 3)



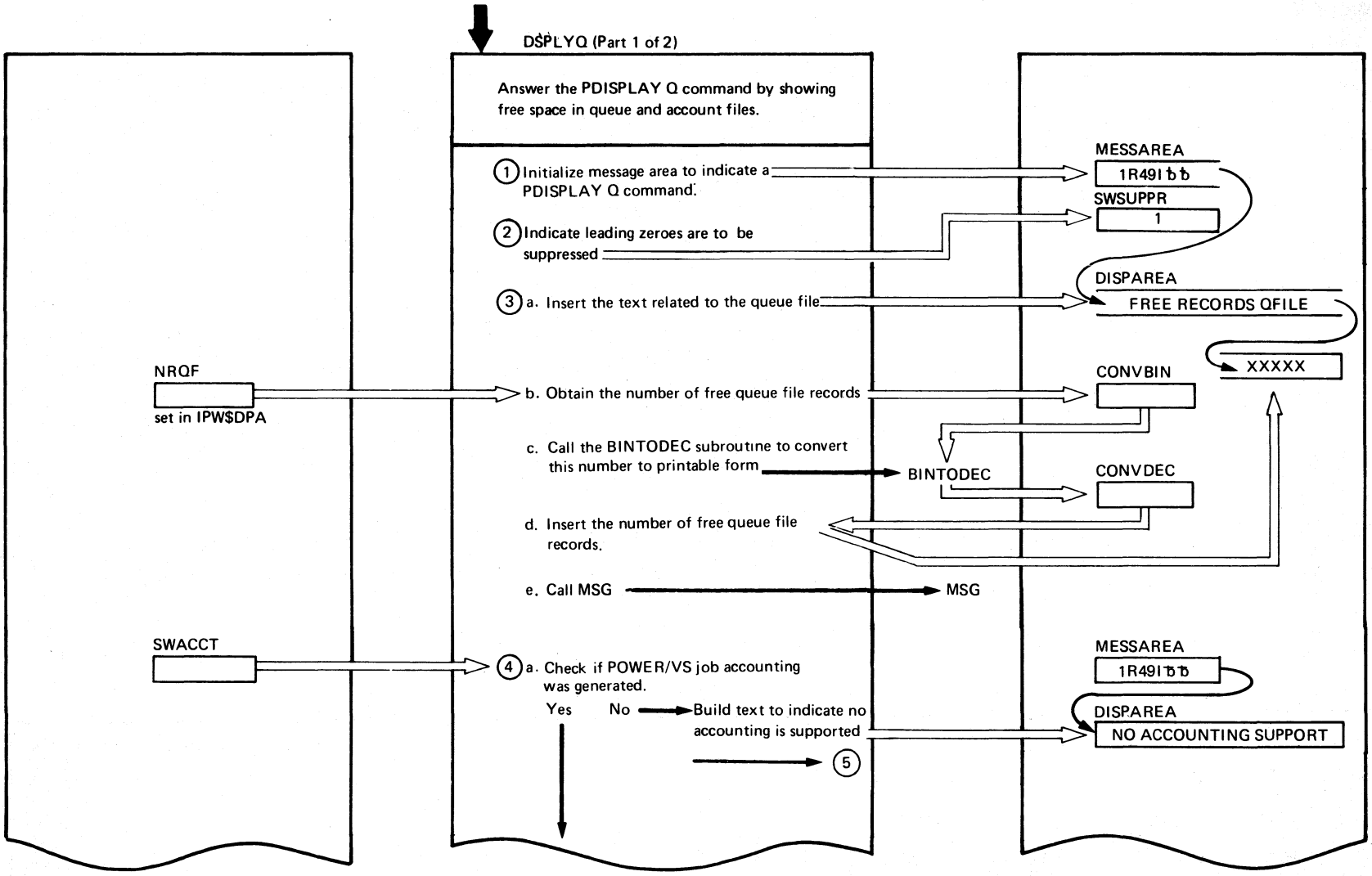
continued on next page

Chart CP35

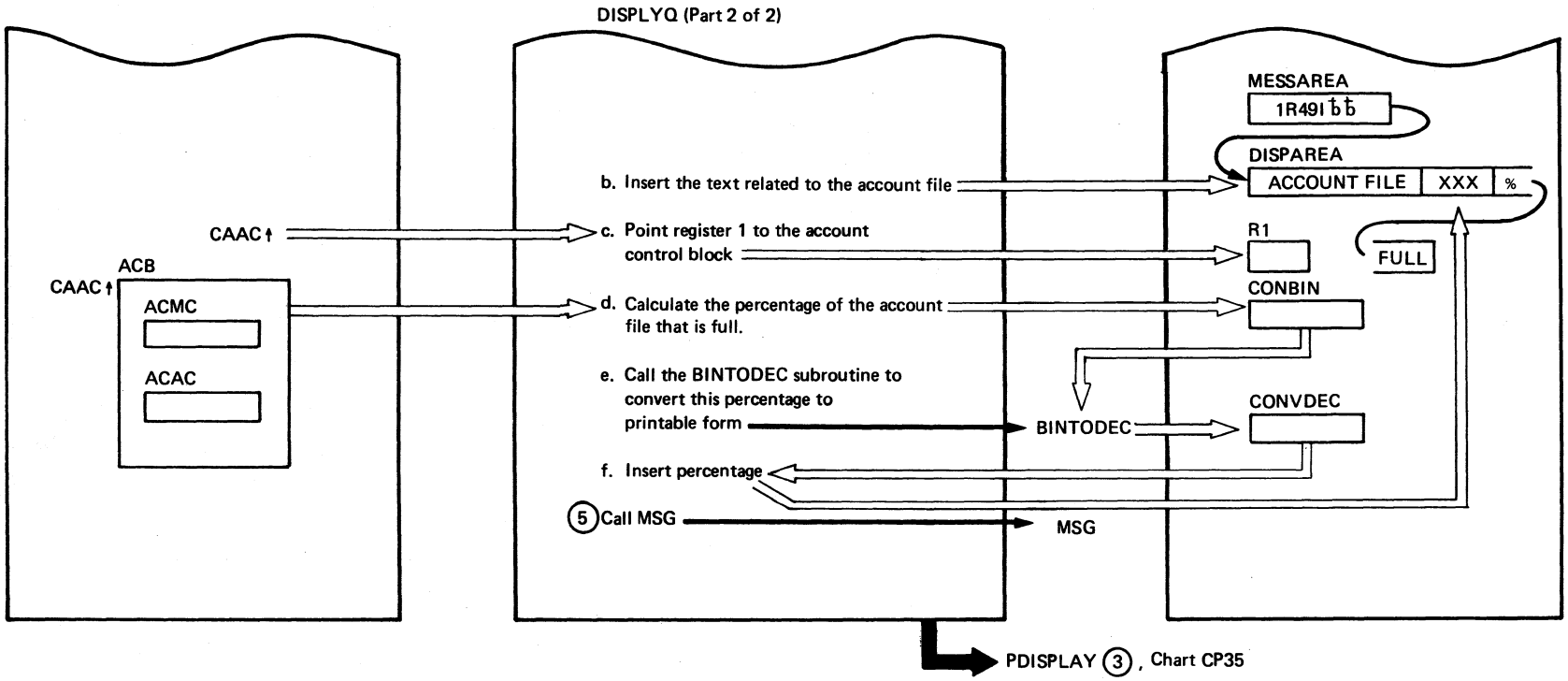

OP1
CARM
SWCANCEL
On or Off
OPERANDS
OP1
SWCANCEL
On or Off
6 If RJE is supported, check if 'MSG B' was specified as first operand.
No
Yes
Indicate command is valid
DSPLYMSG
Display all user type RJE messages.
7 Check if a different operand was specified
Yes
No
MAINLINE 9
8 Check if RDR, PUN, LST, or PRT was specified as first operand.
No
Yes
Turn on the appropriate switches
Initialize for displaying job names in queues (DISPLYJOB segment):
a. Indicate a queue was specified as first operand
b. Shift OP2 to OP1
c. Shift OP3 to OP2
d. Indicate shifting was done
9 DSPLYJOB
Check if job status messages can be displayed.
10 Check if the operands are valid
Yes
No
Build message
Set up indicator
Call MSG
MSG
MAINLINE 9, Chart CP2
SWCANCEL
0
MESSAREA
1R47I messages
6
SWRDR
or SWPUN
1
or SWLST
1
SWQ
1
A
OP1
OP2
OP3
ERRFLAG
2
SWCANCEL
On or Off
MESSAREA
1R52I
ERRFLAG
1, 2, or 3

Extended Description	Include Segment	Call Subroutine	Chart
(2)	DSPLYQ		CP36
(3)	DSPLYA		CP37
(4)	DSPLYM		CP38
(5)	DSPLYT		CP95
(6)	DSPLYMSG		CP39
(9)	DSPLYJOB		CP40
(10) 1R52I OPERAND 1 INVALID		MSG	CP75

Included by PDISPLAY, Chart CP35 LEVEL 3 Chart CP36



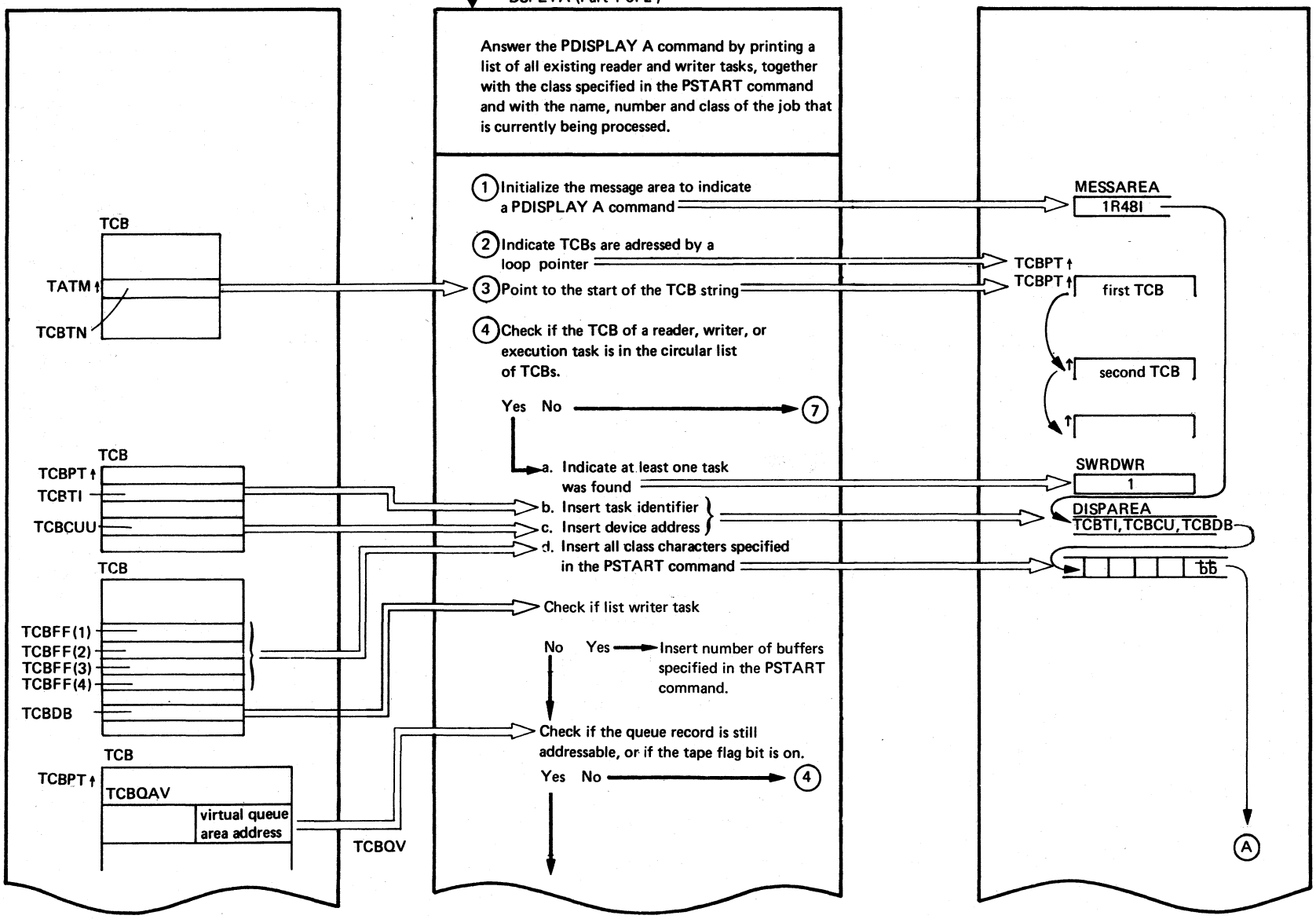
Continued on next page



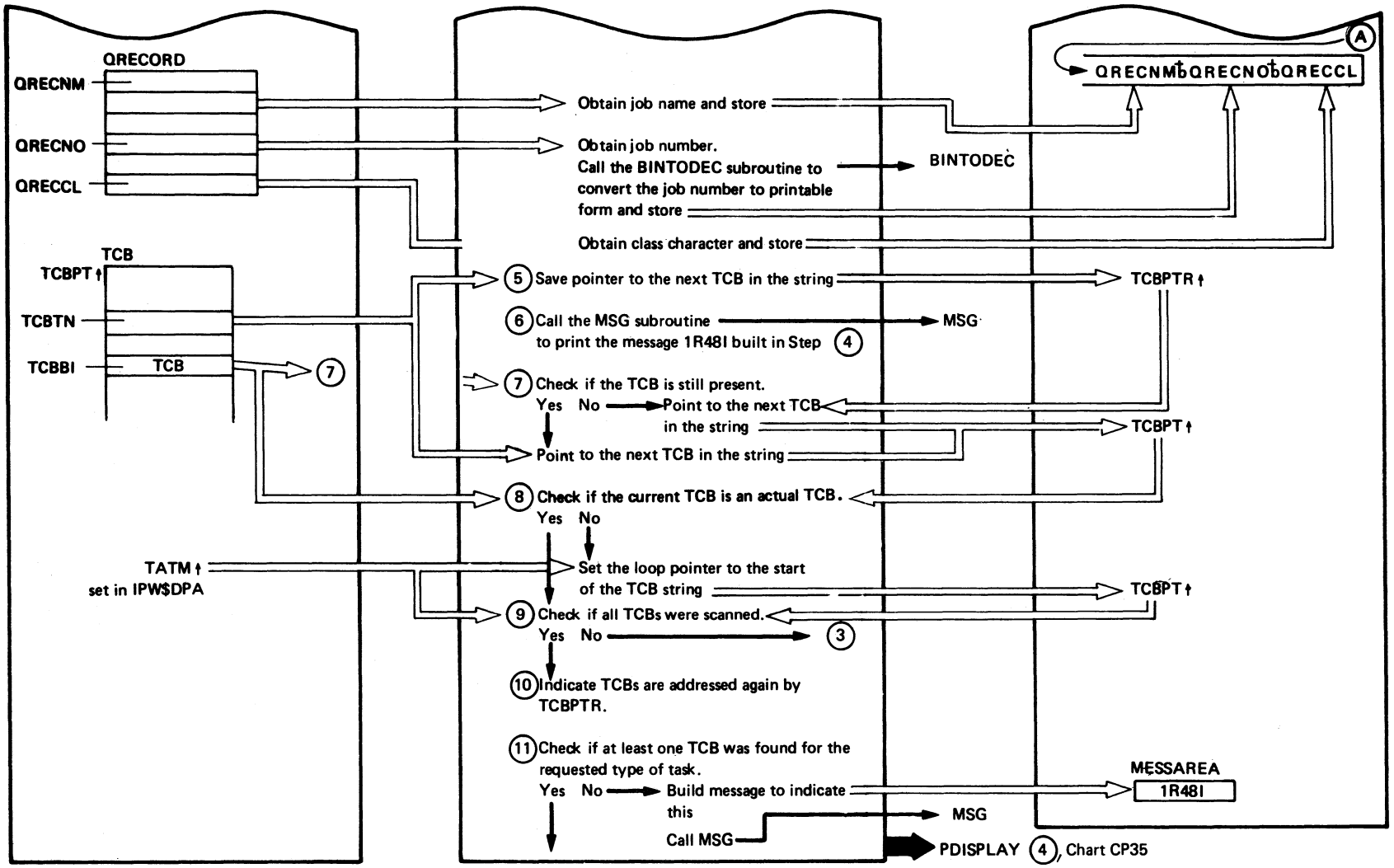
Extended Description	Include Segment	Call Subroutine	Chart
3 c.		BINTODEC	CP83
e.		MSG	CP75
4 d. ACMC in input gives the maximum capacity of the account file. ACAC in input gives the space available in the account file.			
e.		BINTODEC	CP83
5		MSG	CP75

DSPLYA (Part 1 of 2)

Answer the PDISPLAY A command by printing a list of all existing reader and writer tasks, together with the class specified in the PSTART command and with the name, number and class of the job that is currently being processed.

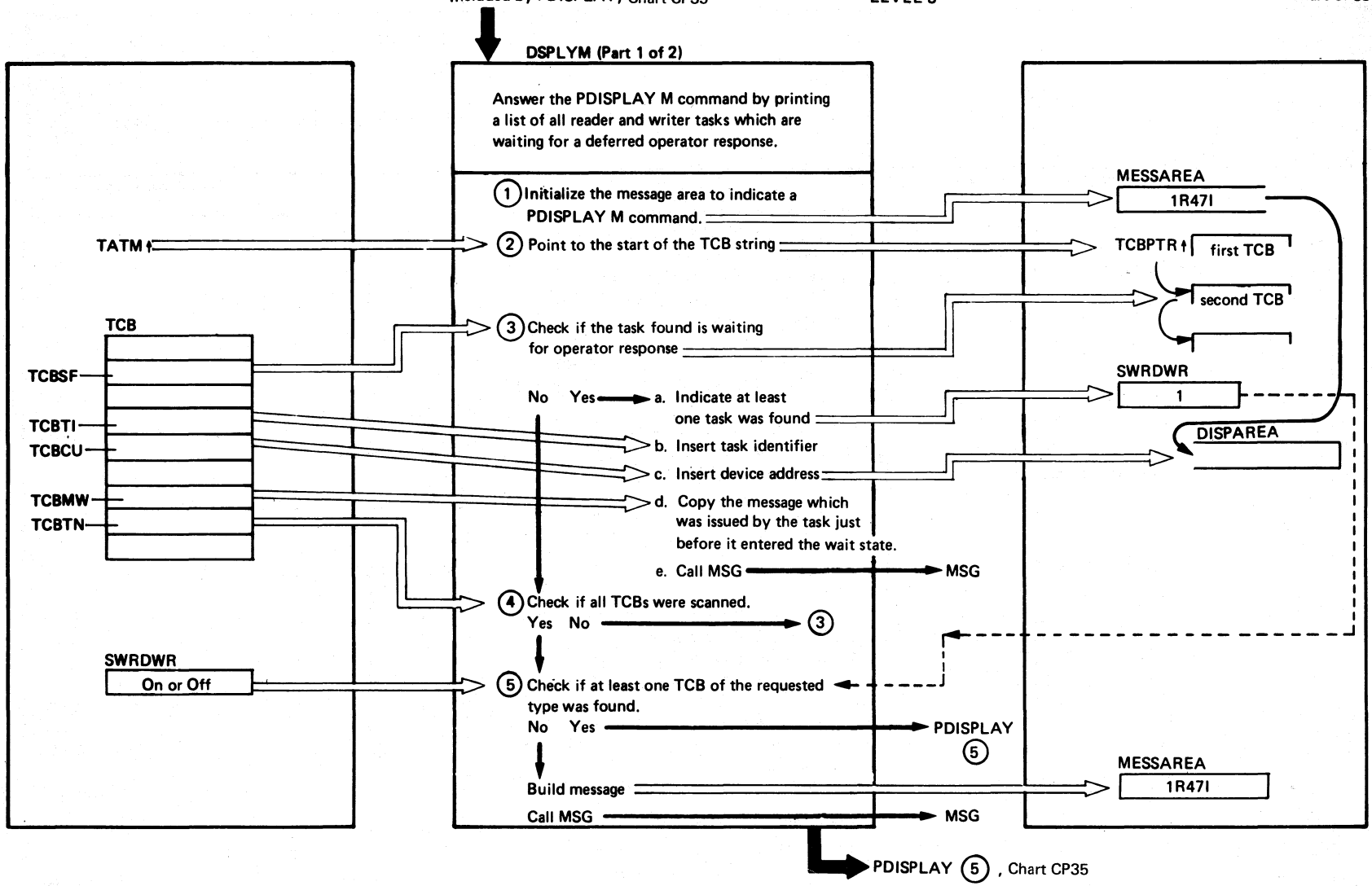


continued on next page



Extended Description	Include Segment	Call Subroutine	Chart
(5) The pointer to the next TCB is saved since the TCB may be lost while message 1R47I is written.			
(9)		MSG	CP75
(11) 1R48I NO READER OR WRITER TASK CURRENTLY ACTIVE		MSG	CP75

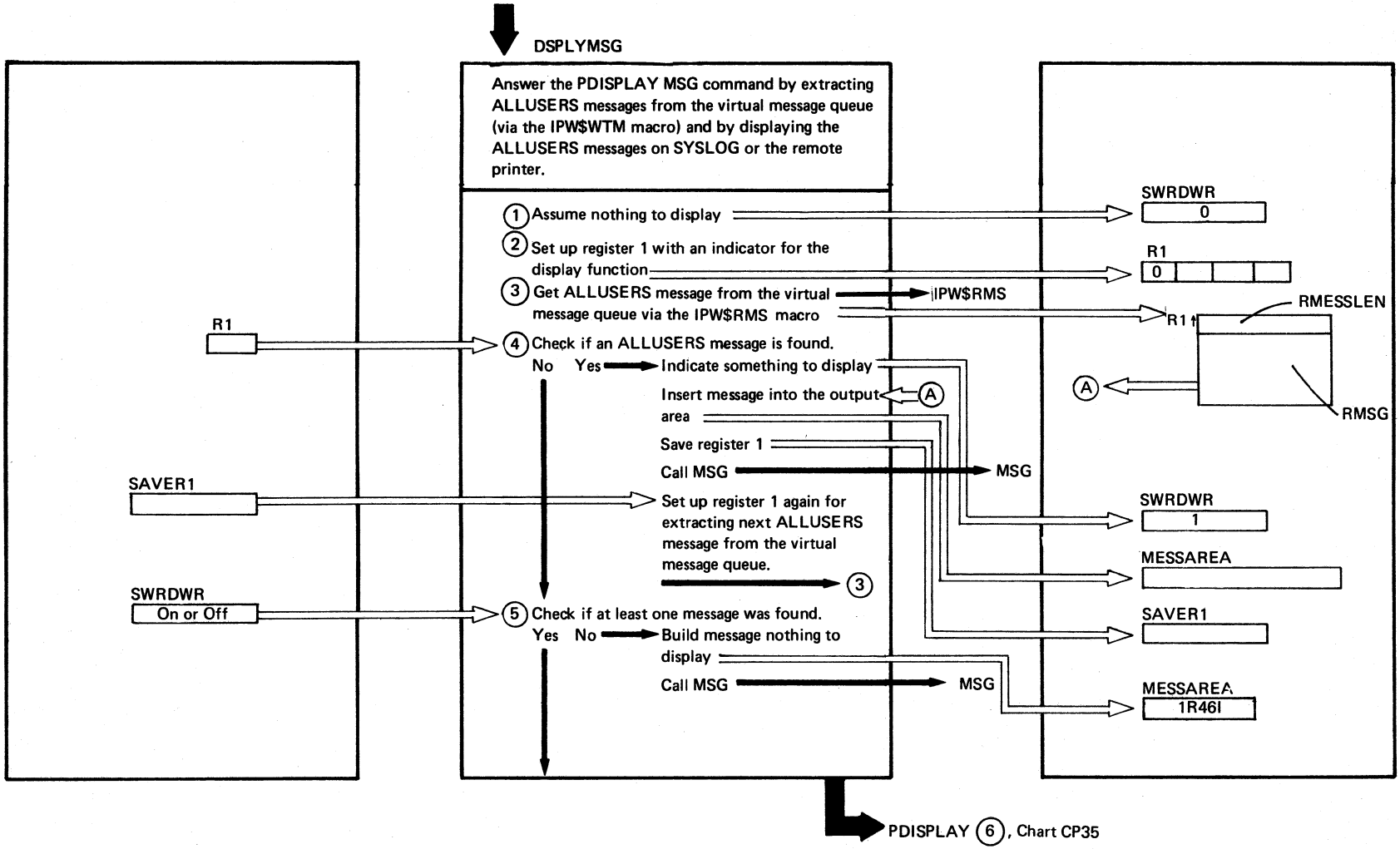
Included by PDISPLAY, Chart CP35 LEVEL 3 Chart CP38



DSPLYM (Part 2 of 2)

Extended Description	Include Segment	Call Subroutine	Chart
(2) **TCBPTR** in output is used as a scan pointer.			
(3) If necessary, a pending message is adapted to the length available in MESSAREA. **TCBMW** in input is a message request word in the TCB, used to address the message still outstanding.		MSG	CP75
(3) and (4) The scan of the TCB string is accomplished through the use of a DO UNTIL statement in the PLS code.			
(4) All TCBs have been scanned when the first TCB is reached again.			
(5) 1R47I NO MESSAGES PENDING		MSG	CP75

Included by PDISPLAY, Chart CP35 LEVEL 3 Chart CP39



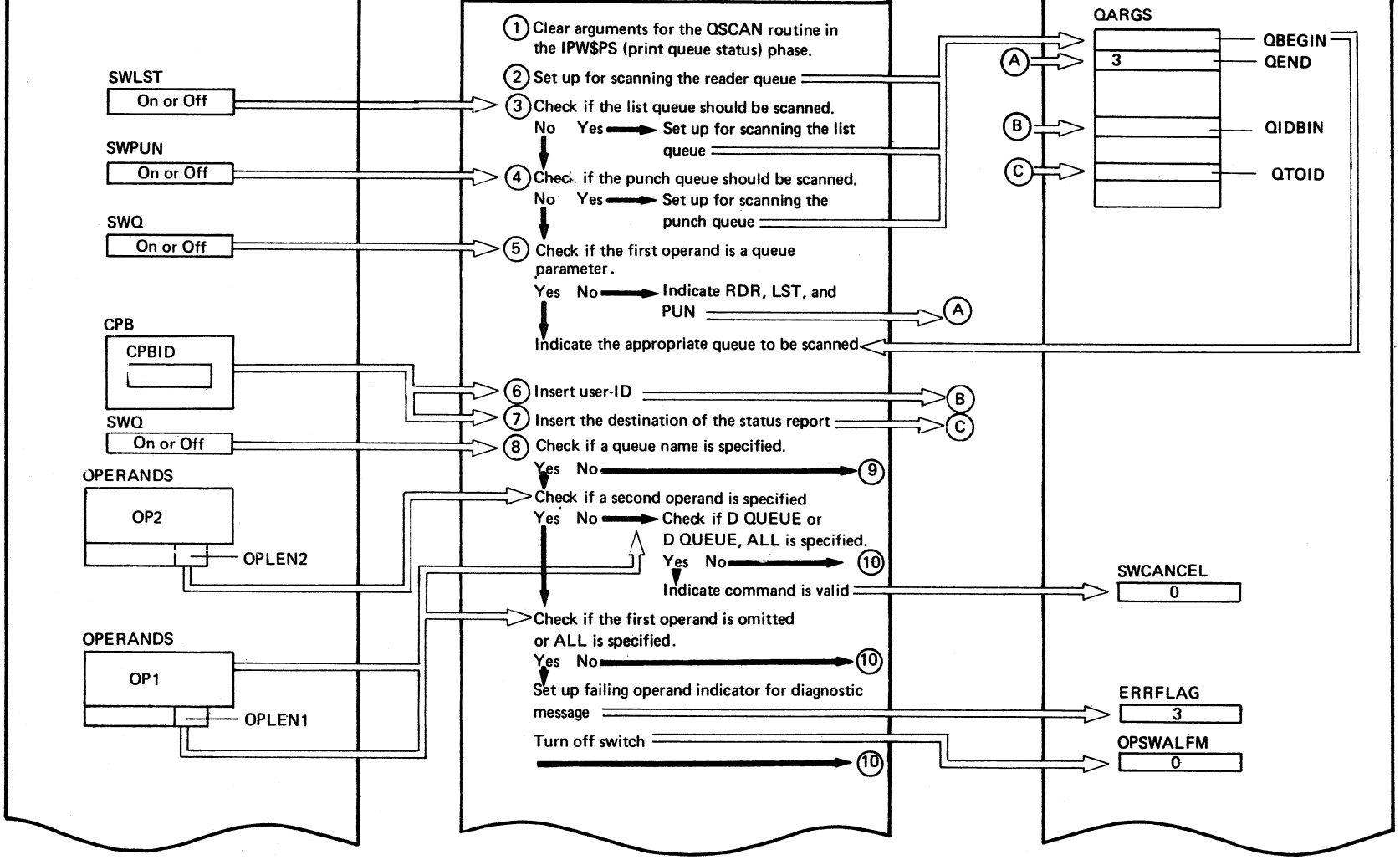
Extended Description	Include Segment	Call Subroutine	Chart
(2) To set up register 1 for the next ALLUSERS message, character D is placed in the high-order byte of register 1. Register 1 is then ORed with SAVER1, which contains the address in the ALLUSERS message queue where the scan must be continued.			
(3) The IPW$RMS macro uses registers 0, 1, 2, and 3. They are restricted in this segment.			
(5) 1R46I NOTHING TO DISPLAY		MSG	CP75

Included by PDISPLAY, Chart CP35 LEVEL 3 Chart CP40

DSPLYJOB (Part 1 of 4)

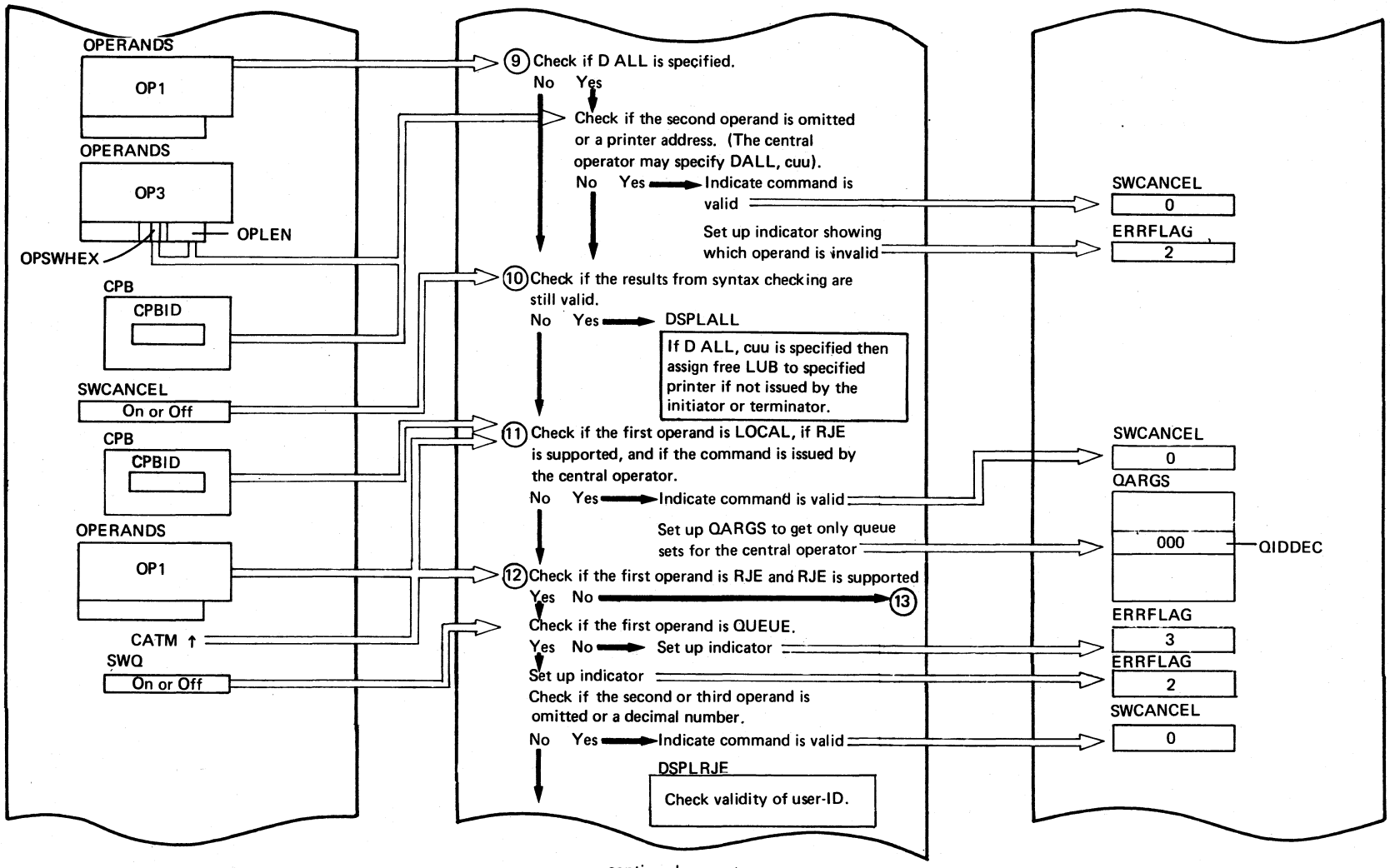
Process the PDISPLAY commands with a first operand that is not 'Q', 'A', 'M', or MSG.

Note: If the first operand was RDR, LST, or PUN, the SWQ switch was turned on, the second operand was shifted to OP1, and the third operand was shifted to OP2.

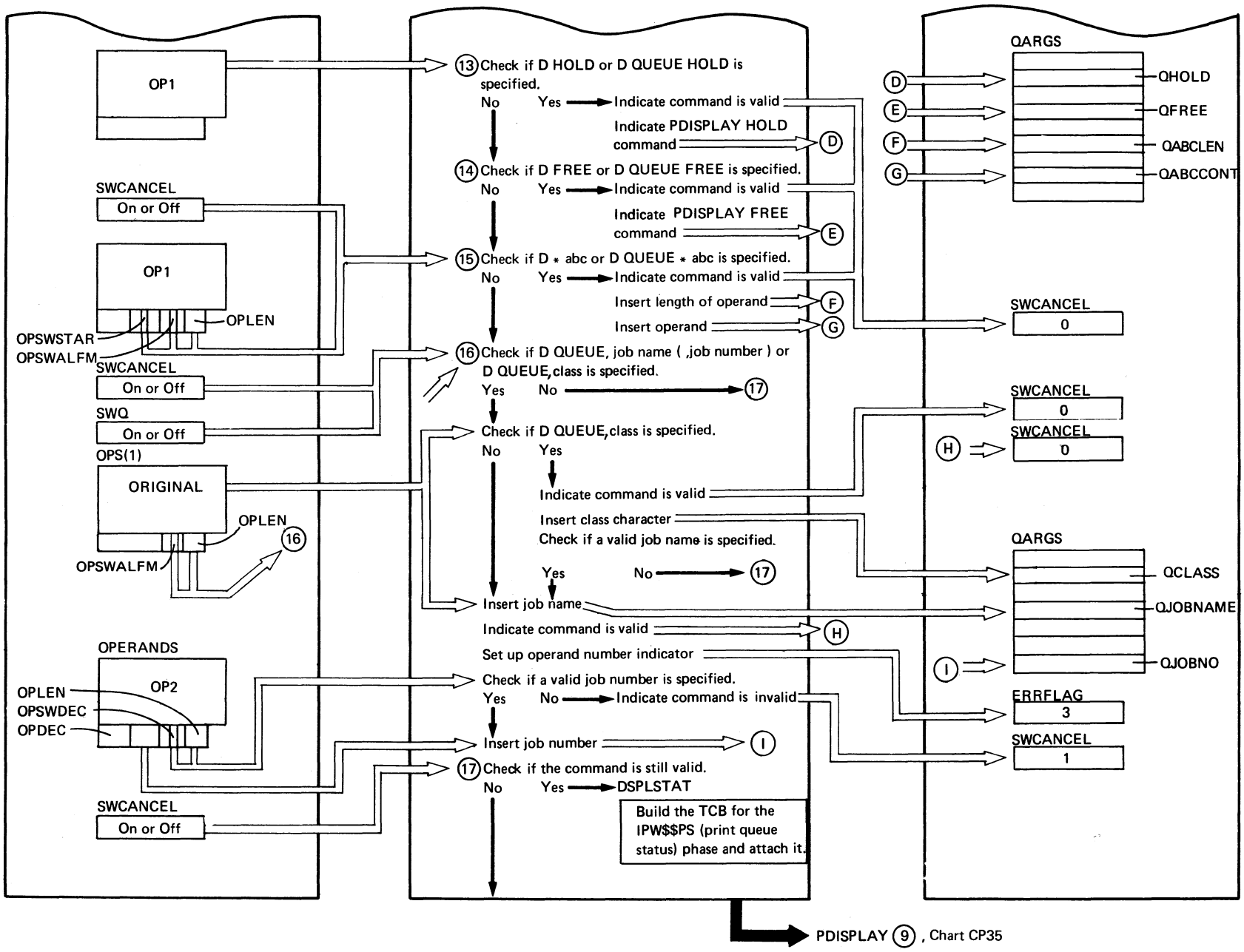


continued on next page

Chart CP40



continued on next page


OP1
SWCANCEL
On or Off
OP1
OPLEN
OPSWSTAR
OPSWALFM
SWCANCEL
On or Off
SWQ
On or Off
OPS(1)
ORIGINAL
OPLEN
16
OPSWALFM
OPERANDS
OP2
OPLEN
OPSWDEC
OPDEC
SWCANCEL
On or Off
13 Check if D HOLD or D QUEUE HOLD is specified.
No
Yes
Indicate command is valid
Indicate PDISPLAY HOLD command
D
14 Check if D FREE or D QUEUE FREE is specified.
No
Yes
Indicate command is valid
Indicate PDISPLAY FREE command
E
15 Check if D * abc or D QUEUE * abc is specified.
No
Yes
Indicate command is valid
Insert length of operand
F
Insert operand
G
16 Check if D QUEUE, job name (,job number) or D QUEUE,class is specified.
Yes
No
17
Check if D QUEUE,class is specified.
No
Yes
Indicate command is valid
Insert class character
Check if a valid job name is specified.
Yes
No
17
Insert job name
Indicate command is valid
H
Set up operand number indicator
I
Check if a valid job number is specified.
Yes
No
Indicate command is invalid
Insert job number
I
17 Check if the command is still valid.
No
Yes
DSPLSTAT
Build the TCB for the IPW$$PS (print queue status) phase and attach it.
QARGS
D
QHOLD
E
QFREE
F
QABCLEN
G
QABCCONT
SWCANCEL
0
SWCANCEL
0
H
SWCANCEL
0
QARGS
QCLASS
QJOBNAME
I
QJOBNO
ERRFLAG
3
SWCANCEL
1
PDISPLAY 9 , Chart CP35

Extended Description	Include Segment	Call Subroutine	Chart
(10)	DSPLYALL		CP41
(12)	DSPLRJE		CP42
(17)	DSPLSTAT		CP43

Included by DSPLYJOB, Chart CP40

LEVEL 4

Chart CP41

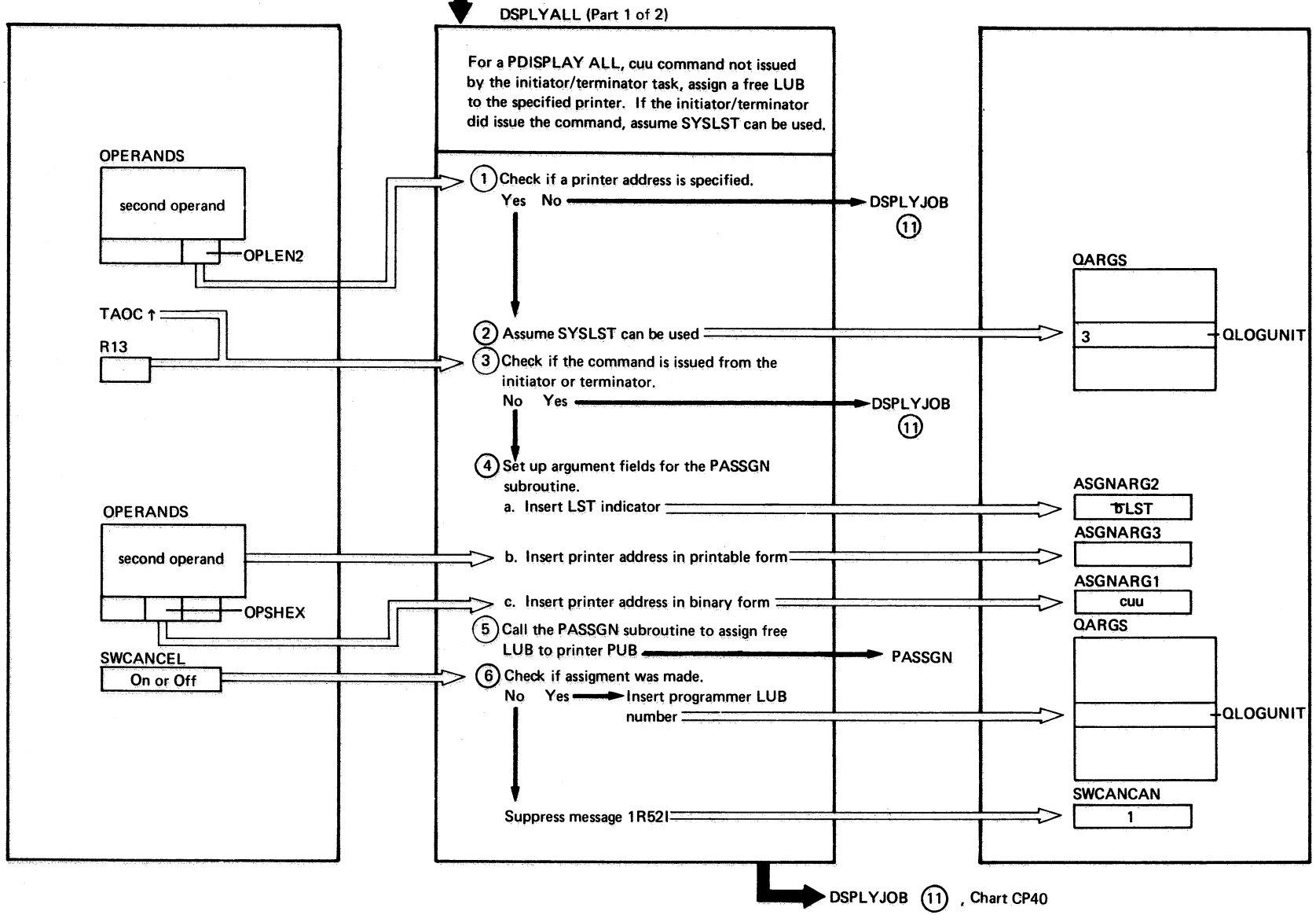
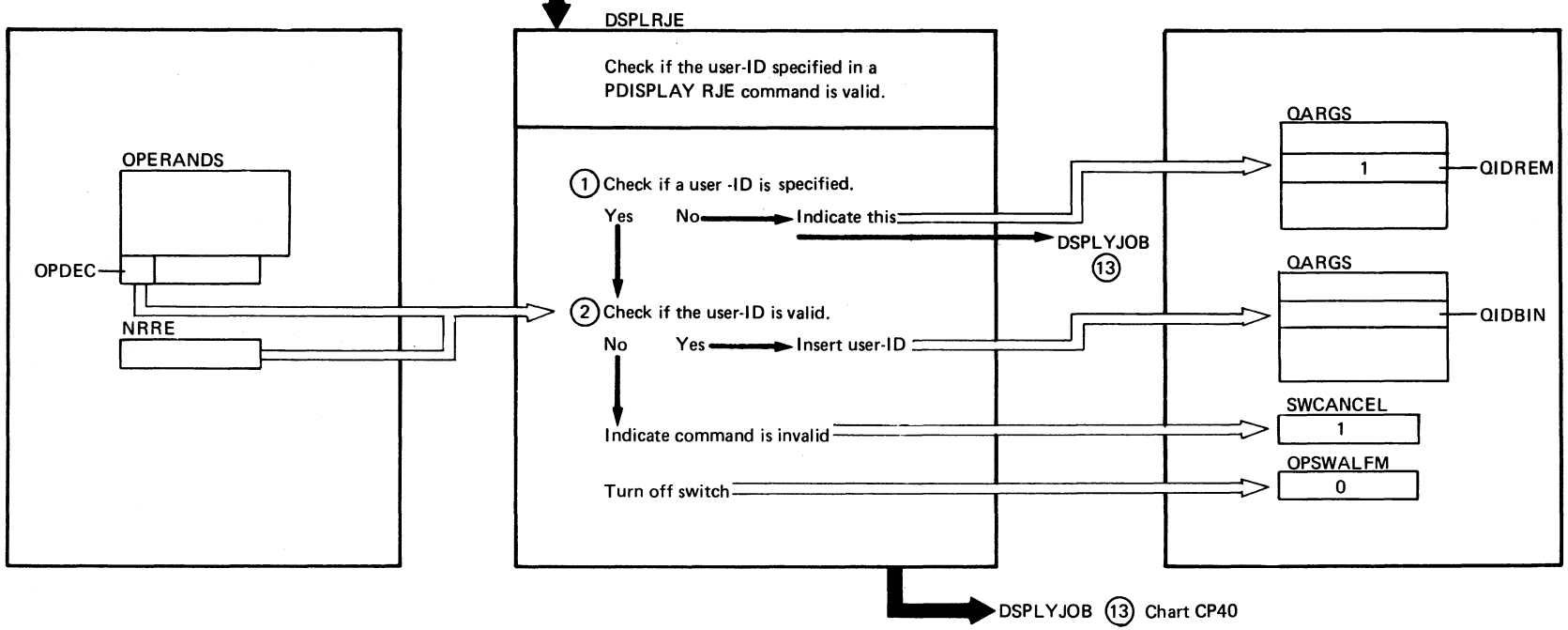


Chart CP41

DSPLYALL (Part 2 of 2)

Extended Description	Include Segment	Call Subroutine	Chart
(3) TAOC points to the permanent command processor TCB. (5) (6) Message 1R52I is suppressed since the error was diagnosed already by the PASSGN subroutine.		PASSGN	CP76

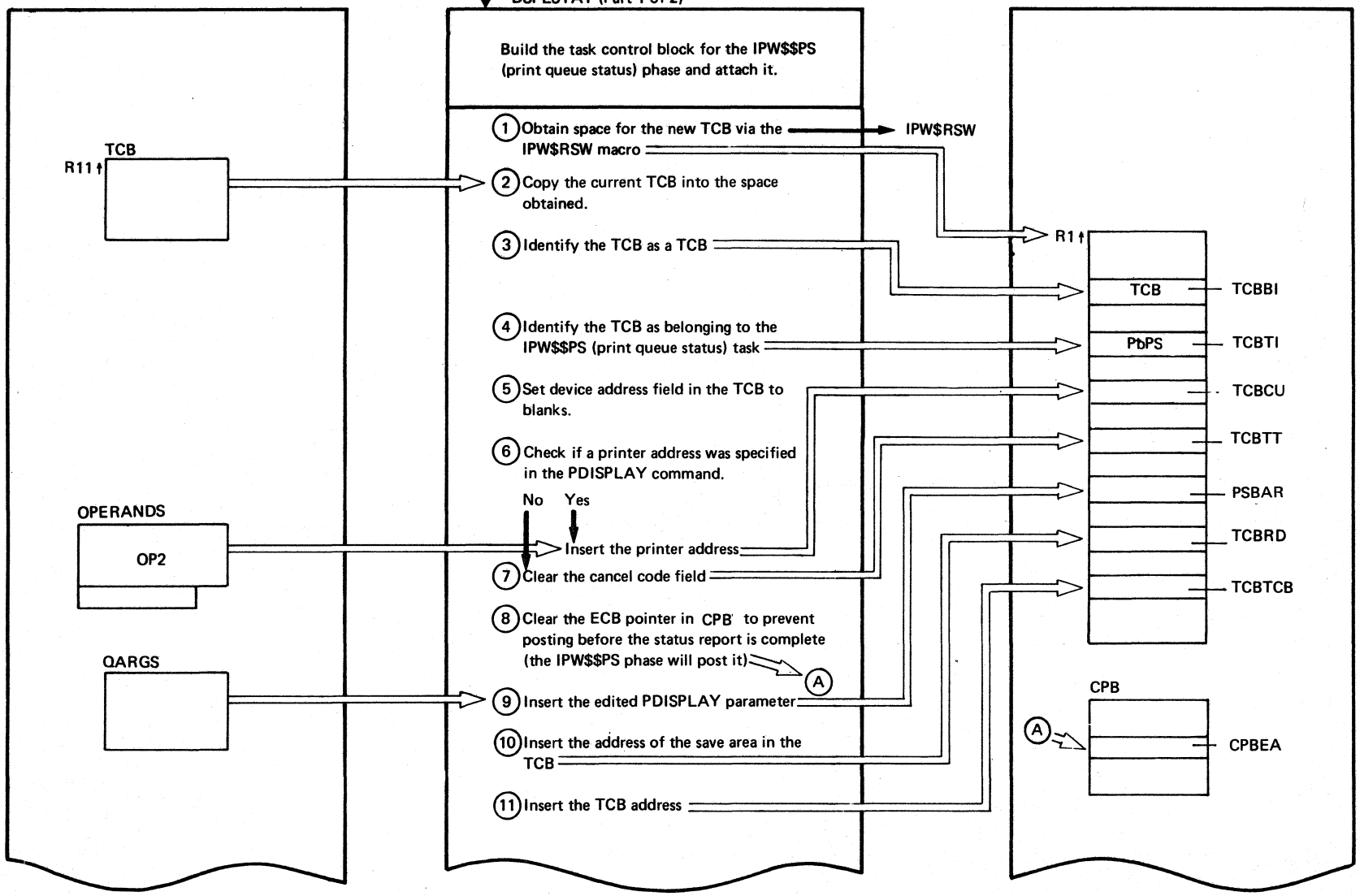
Included by DSPLYJOB, Chart CP40 LEVEL 4 Chart CP 42



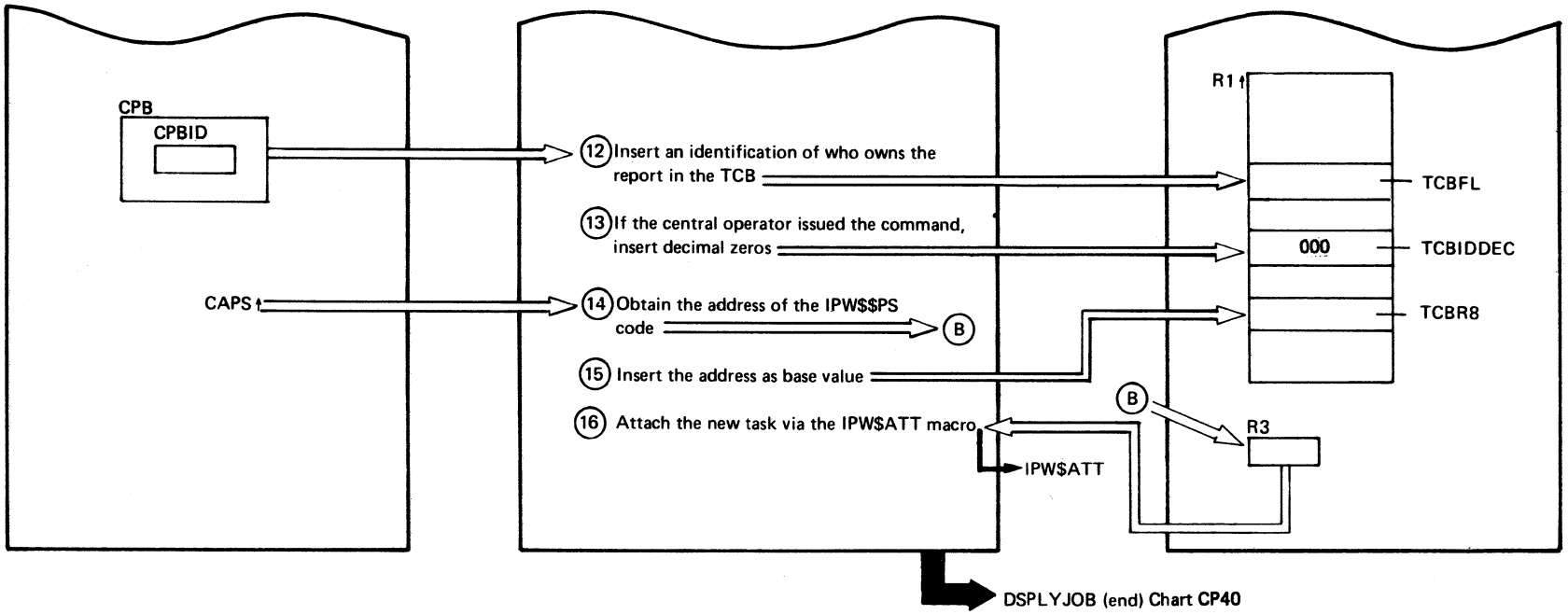
Included by DSPLYJOB, Chart CP40

LEVEL 4

DSPLSTAT (Part 1 of 2)



continued on next page



Extended Description	Include Segment	Call Subroutine	Chart
This segment uses registers 0, 1, 2, and 3.			

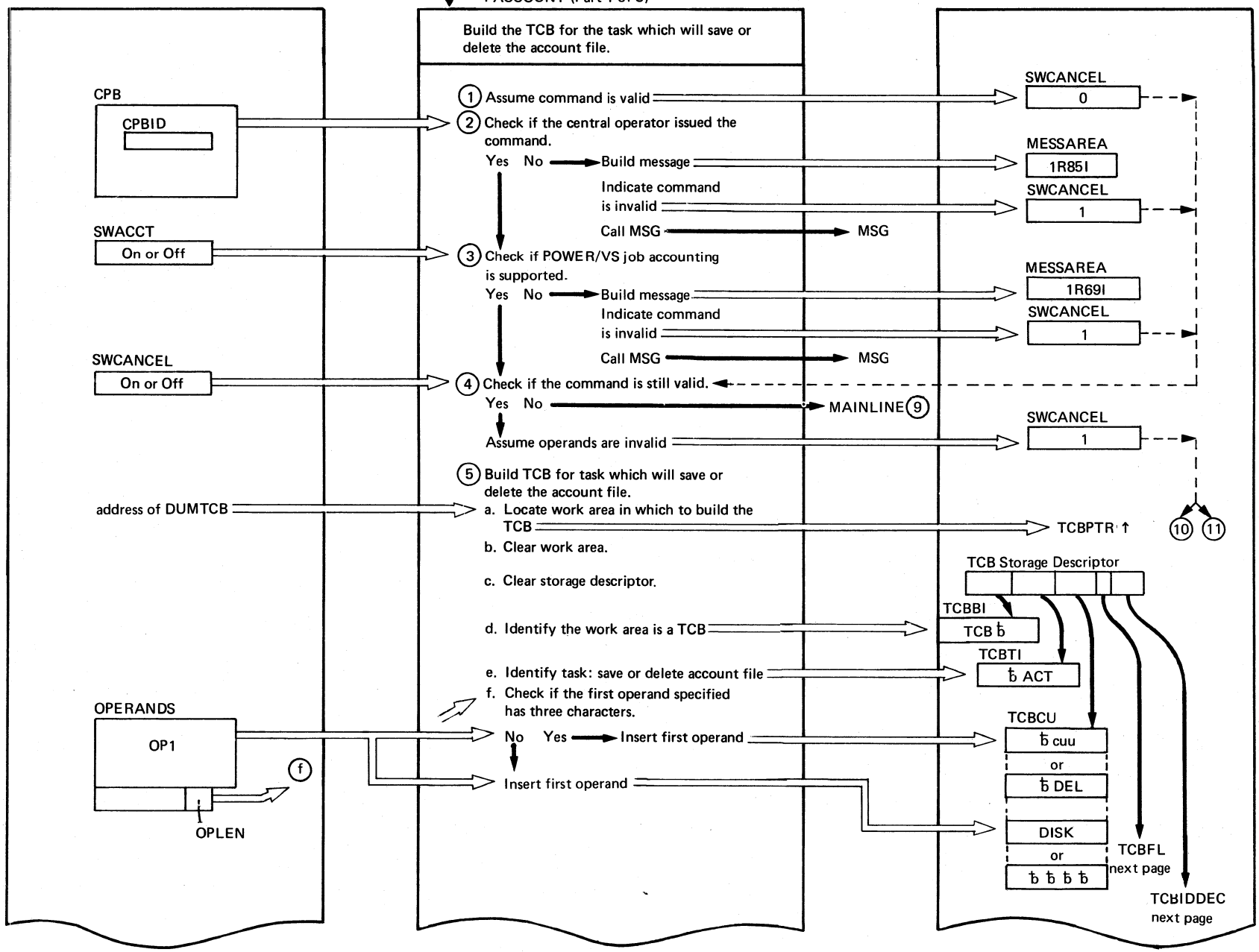
Included by MAINLINE, Chart CP2

LEVEL 2

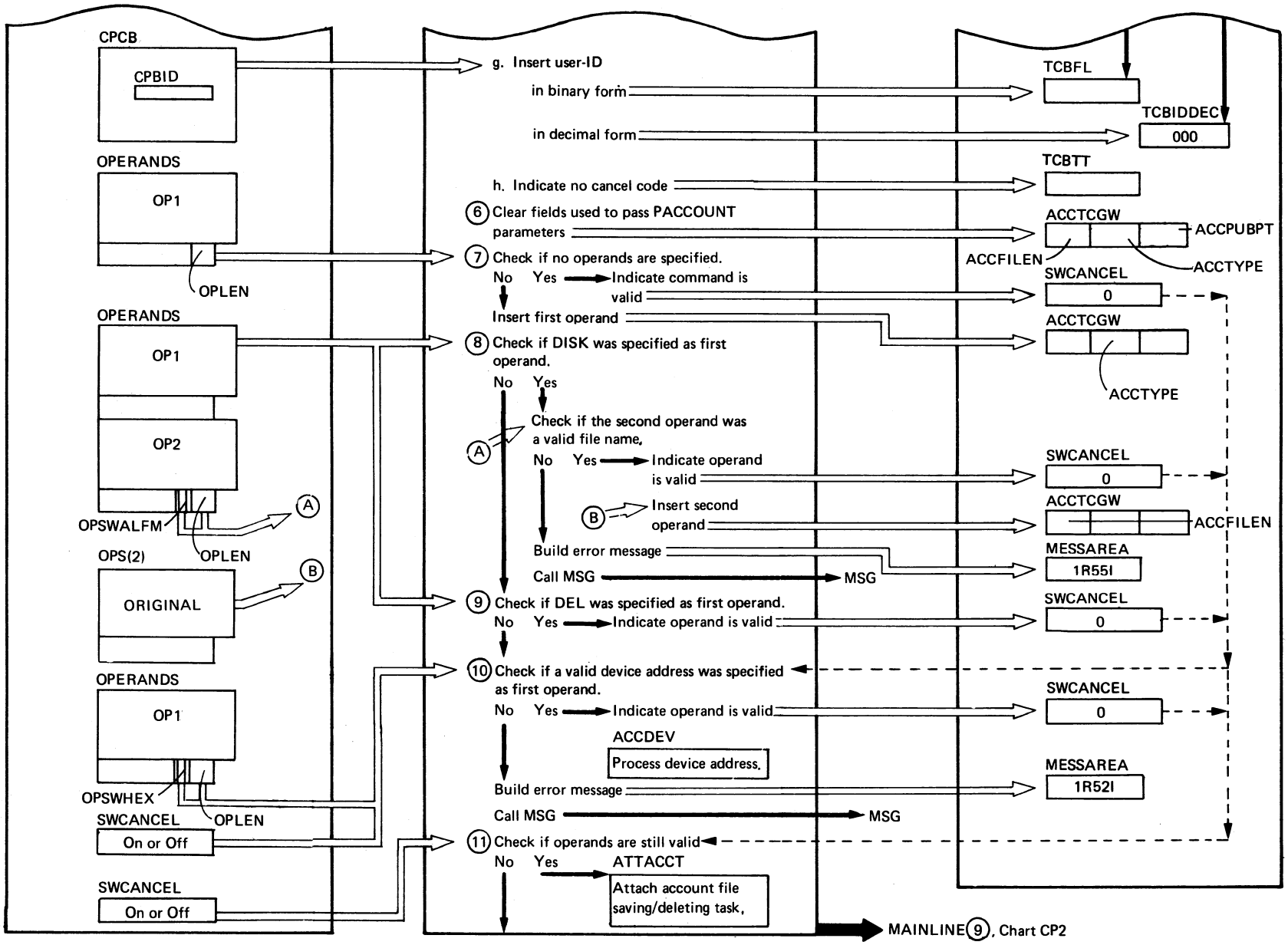
Chart CP44

PACCOUNT (Part 1 of 3)

Build the TCB for the task which will save or delete the account file.



Continued on next page


CPCB
CPBID
OPERANDS
OP1
OPLEN
OPERANDS
OP1
OP2
OPSWALFM
OPLEN
OPS(2)
ORIGINAL
OPERANDS
OP1
OPSWHEX
SWCANCEL
OPLEN
On or Off
SWCANCEL
On or Off
g. Insert user-ID
in binary form
in decimal form
h. Indicate no cancel code
6 Clear fields used to pass PACCOUNT parameters
7 Check if no operands are specified.
No Yes Indicate command is valid
Insert first operand
8 Check if DISK was specified as first operand.
No Yes
A
Check if the second operand was a valid file name.
No Yes Indicate operand is valid
B
Insert second operand
Build error message
Call MSG
MSG
9 Check if DEL was specified as first operand.
No Yes Indicate operand is valid
10 Check if a valid device address was specified as first operand.
No Yes Indicate operand is valid
ACCDEV
Process device address.
Build error message
Call MSG
MSG
11 Check if operands are still valid
No Yes ATTACCT
Attach account file saving/deleting task.
MAINLINE 9, Chart CP2
TCBFL
TCBIDDEC
000
TCBTT
ACCTCGW
ACCPUBPT
ACCFILEN
ACCTYPE
SWCANCEL
0
ACCTCGW
ACCTYPE
SWCANCEL
0
ACCTCGW
ACCFILEN
MESSAREA
1R551
SWCANCEL
0
SWCANCEL
0
MESSAREA
1R521
SWCANCEL
0

Extended Description	Include Segment	Call Subroutine	Chart
(2) 1R85I COMMAND INVALID FOR REMOTE OPERATOR		MSG	CP75
(3) 1R69I NO ACCOUNTING SUPPORT		MSG	CP75
(8) 1R55I INVALID FILENAME		MSG	CP75
(10) 1R52I OPERAND 1 INVALID		MSG	CP75
	ACCDEV		CP45
(11)	ATTACCT		CP47

Included by PACCOUNT, Chart CP44 LEVEL 3 Chart CP45

ACCDEV (Part 1 of 2)

Check if the device address specified in a PACCOUNT command corresponds to a tape drive. Assign free programmer LUB to specified PUB.

OPERANDS

OPHEX

OPERAND

ORIGINAL

PUBDEVTP

SWCANCEL

On or Off

(1) Scan the PUB table for an entry corresponding to the device address specified

PUB pointer pointing to entry found or to the end of the PUB table.

(2) Check if a PUB entry is found.

Yes No → Build error message

Call MSG → MSG

MESSAREA 1R58I

Indicate operand is invalid → (6)

SWCANCEL 1

(3) Save PUB pointer

ACCPUBPT ↑

Set up fields for the PASSGN subroutine:

a. hexadecimal device address

ASGNARG1

b. printable device address

ASGNARG3

(4) Check if the device is a tape.

Yes No → Build error message

Call MSG → MSG

MESSAREA 1R74I

Indicate operand is invalid

SWCANCEL 1

a. ACCOP23

Process operands 2 and 3 specifying density and file name.

ASGNARG2 TAPE

b. Set up field for the PASSGN subroutine

ACCTYPE TAPE

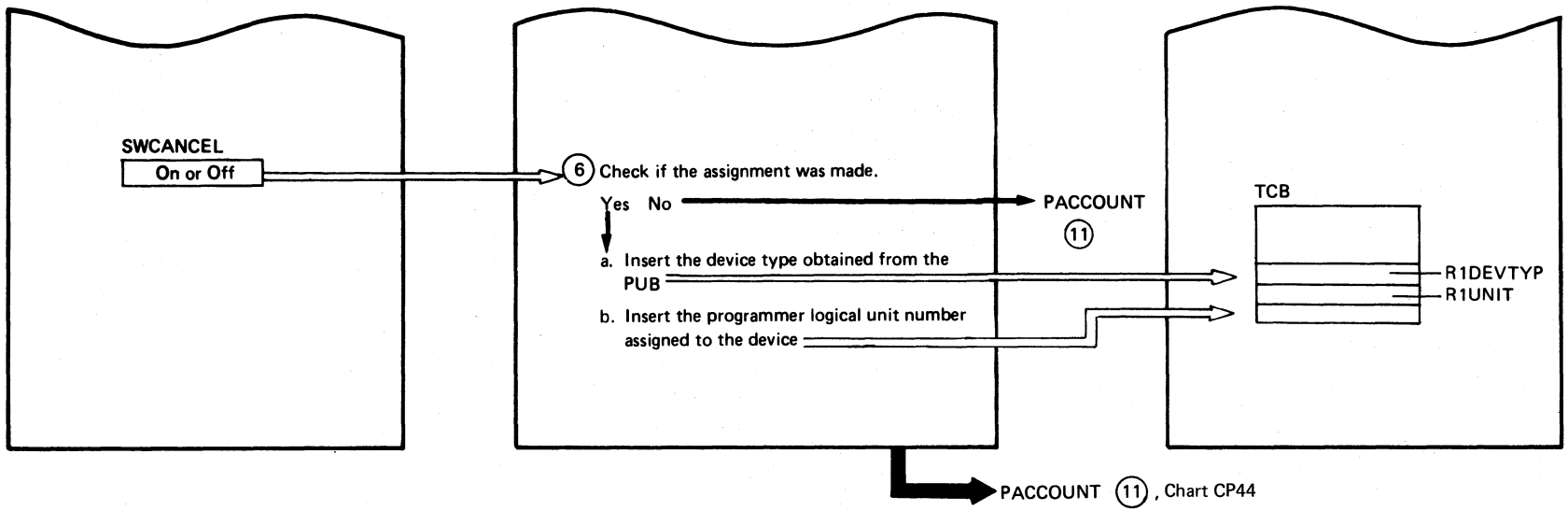
Set up field for task which will save/delete account file

(5) Check if the command is still valid.

Yes No → PACCOUNT (11)

Call the PASSGN subroutine to assign free programmer LUB to wanted device → PASSGN

continued on next page



Extended Description	Include Segment	Call Subroutine	Chart
(2) 1R58I DEVICE xxx NOT KNOWN device address	ACCOP23	MSG	CP75
(4) 1R74I INVALID TAPE SPECIFICATION The contents of the PUBDEVTP field is in the range >X'4F' and <X'60' and ≠ X'51'		MSG	CP75 CP46
(5)		PASSGN	CP76

Included by ACCDEV, Chart CP45

ACCOP23

LEVEL 4

Chart CP46

Check the validity of the second and third operands, if present, of a PACCOUNT command specifying a tape drive to save the account file.

PUB

PUB7T

DENSIT7

DENSIT9

OPERANDS

OP2

OPSWHEX

OPLEN

OPERANDS

OP3

OPLEN

OPSWALFM

(1) Initialize the density field in the TCB → R1DENSIT: 0

(2) Check if the device is a 7-track tape.

No

Yes

Scan the table of valid densities for 7-track tapes to find a match with the density specified.

If a match is found insert density → TCB: R1DENSIT

(3) Scan the table of valid densities for 9-track tapes to find a match with density specified

If a match is found insert density → TCB: R1DENSIT

(4) Check if the density is valid or omitted.

Yes No → Build error message → MESSAREA: 1R53I

Call MSG → MSG

Indicate operand is invalid → SWCANCEL: 1

(5) Check if the third operand specifies a valid file name.

No Yes → Insert the file name into the parameter field for the save or delete task → ACCTCGW: ACCFILEN

ACCDEV (4) b.

Check if the third operand was omitted.

Yes No → Build error message → MESSAREA: 1R55I

Call MSG → MSG

Indicate command is invalid → SWCANCEL: 1

ACCDEV, (4) b., Chart CP45

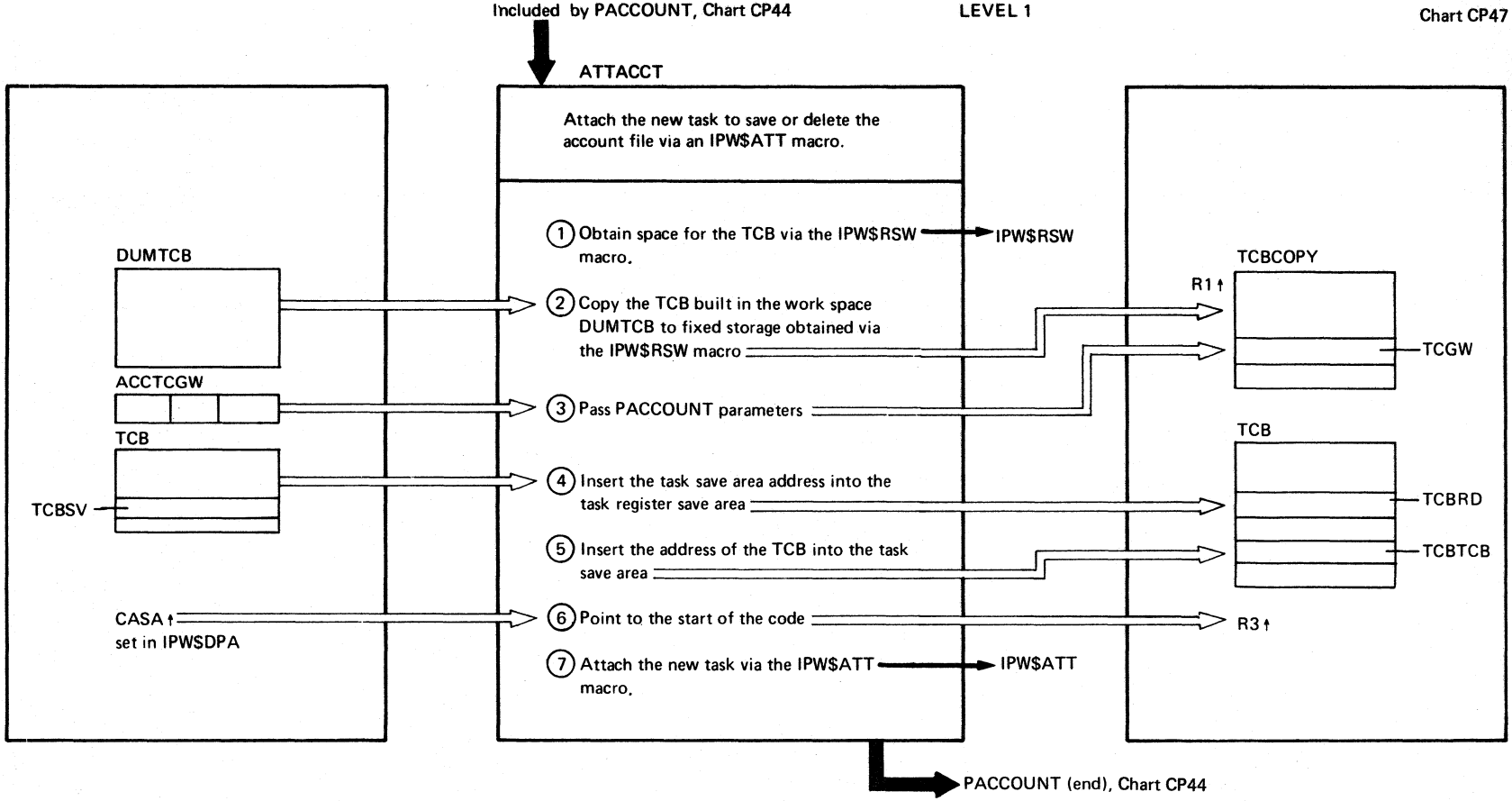
Extended Description	Include Segment	Call Subroutine	Chart
(4) 1R53I INVALID DENSITY		MSG	CP75
(5) 1R55I INVALID FILENAME		MSG	CP75

Chart CP46

Included by PACCOUNT, Chart CP44

LEVEL 1

Chart CP47



Extended Description	Include Segment	Call Subroutine	Chart
(1) and (7) The IPW$RSW and IPW$ATT macros use Registers 0, 1, 2, and 3			

From MAINLINE, Chart CP2

LEVEL 2

Chart CP48

PEND (Part 1 of 3)

Insert a cancel code in all task control blocks. Assign SYSLST LUB to the specified printer address. Issue the SETPEA and DUMP macro if an immediate stop is required. Change initiator TCB and activate the terminator task.

CPB
CPBCM

CPB
CPBID

TCB
TAIT↑
TCBSF

OPERAND
OPLEN
OPSWHEX

(1) Assume command is valid → SWCANCEL 0

(2) Check if the command was entered in short format.
No
Yes → Indicate command is invalid → SWCANCEL 1
Build message → MESSAREA 1R78I
Call MSG → MSG

(3) Check if a remote user issued the command.
No
Yes → Indicate command is invalid → SWCANCEL 1
Build message → MESSAREA 1R85I
Call MSG → MSG

(4) Check if POWER/VS intialization was completed.
Yes
No → Indicate command is invalid → SWCANCEL 1
Build message → MESSAREA 1R89I
Call MSG → MSG

(5) Check if the command is still valid
Yes
No → (10)

PENDOP1

(6) Check the first operand for validity.

(7) Check if a printer address was specified in the first or second operand.
No
Yes

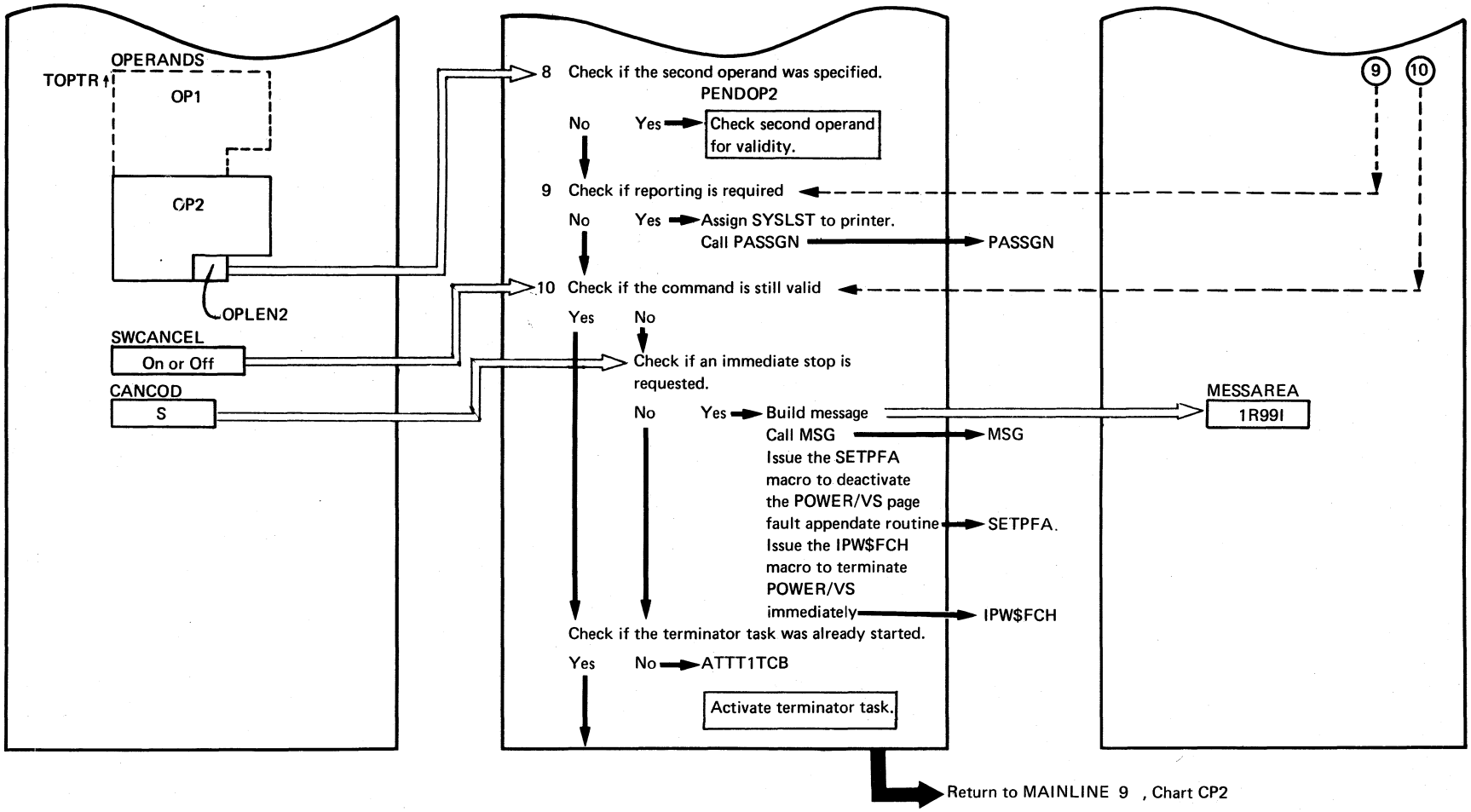
Check if the operand is valid.
No
Yes → Indicate reporting is required → SWREPORT 1
Indicate command is invalid → SWCANCEL 1
Build message → MESSAREA 1R74I
Call MSG → MSG

(9) (10)

continued on next page

Chart CP48

PEND (Part 2 of 3) LEVEL 2

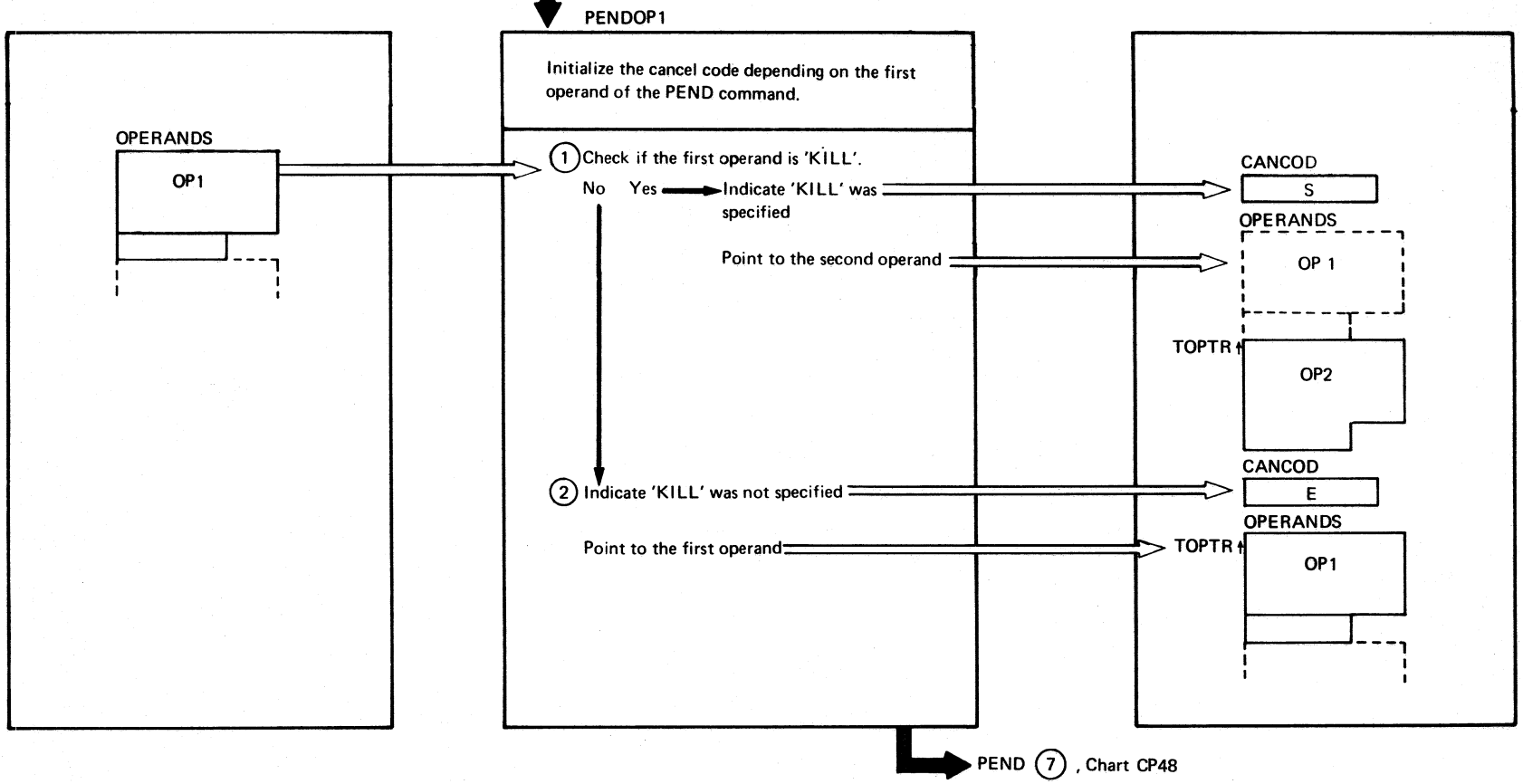


Extended Description	Include Segment	Call Subroutine	Chart
(2) 1R78I SHORT 'PEND' NO LONGER SUPPORTED		MSG	CP75
(3) 1R85I COMMAND INVALID FOR REMOTE OPERATOR		MSG	CP75
(4) 1R89I POWER/VS INITIALIZATION NOT COMPLETE		MSG	CP75
(6)	PENDOP1		CP49
(7) 1R74I OPERAND x INVALID PRINTER SPECIFICATION x: 1 or 2		MSG	CP75
(8)	PENDOP2		CP50
(9) The following arguments are built as input to subroutine PASSGN : ASGNARG1 = OPHEX ASGNARG2 = SLST ASGNARG3 = ORIGINAL		PASSGN	CP76
(10) 1R99I POWER/VS HAS BEEN TERMINATED	ATTT1TCB	MSG	CP51 CP75

Included by PEND, Chart CP48

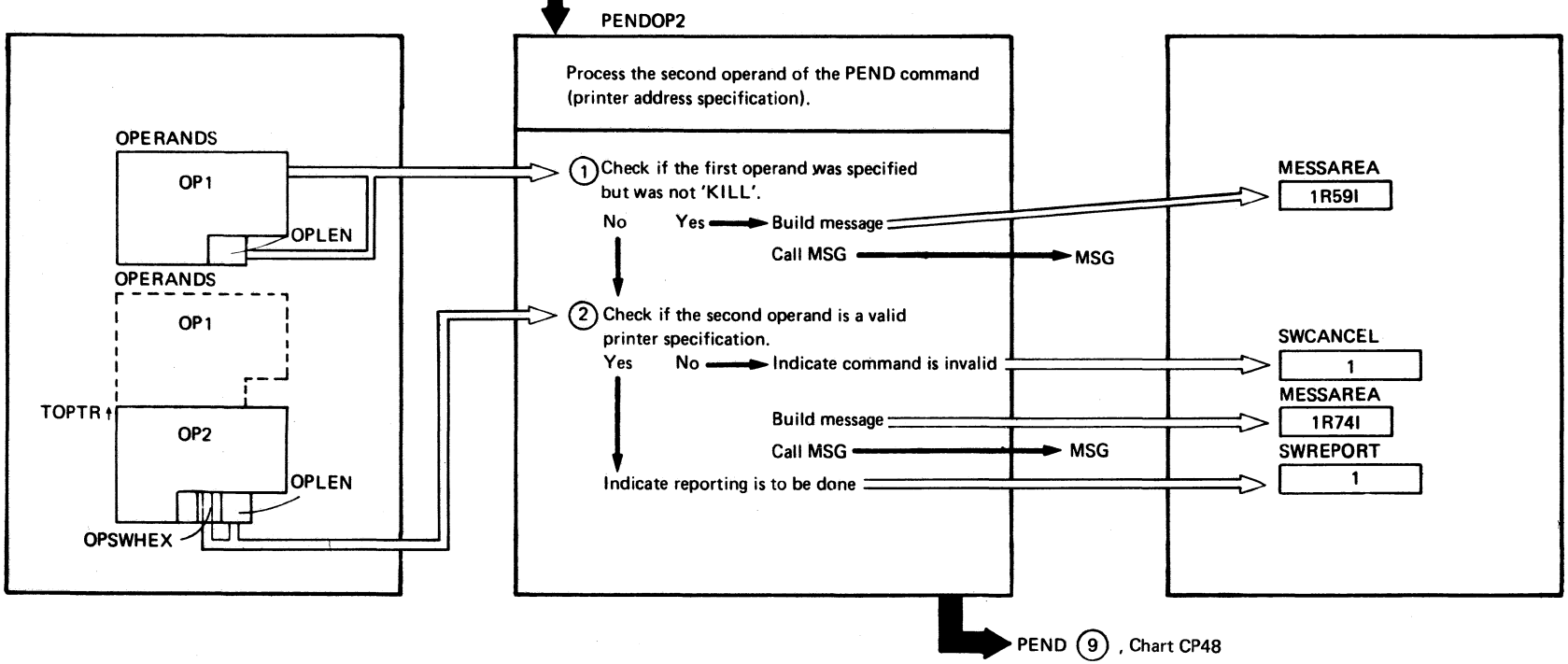
LEVEL 3

Chart CP49



Extended Description	Include Segment	Call Subroutine	Chart
(1) The CANCOD field is used to insert the cancel code 'E' into all TCBs used when PEND is issued. (2) If PEND 'KILL' was specified, the second operand, if specified, should be the device address of of the printer.			

Included by PEND, Chart CP48 LEVEL 3 Chart CP50



Extended Description	Include Segment	Call Subroutine	Chart
(1) 1R59I OPERAND 2 IGNORED		MSG	CP75
(2) 1R74I OPERAND x INVALID PRINTER SPECIFICATION		MSG	CP75

x = 2

Included by PEND, Chart CP48

LEVEL 3

ATTT1TCB (Part 1 of 2)

Simulate the PSTOP command for all non-SNA tasks. Turn on the stop bit in all line control blocks. Notify the central operator and all signed-on remote users that POWER/VS is going down. Change initiator TCB and activate the terminator task.

TCB
TCBSTATE
TCBSF
CANCOD
LCBSIGN
SNTC↑
CAS0↑
SNDS
SNTT

1. Build shutdown message
2. Insert termination code in all TCBs except for SNA tasks and post tasks dispatchable if required
3. Turn on stop bit in all line control blocks.
4. Check if an RJE user is signed on.
   - No
   - Yes → Indicate destination of shutdown message
   - Call MSG → MSG
5. Check if SNA is active
   - No
   - Yes
   - Check if a higher termination type has already been set
     - Yes
     - No
       - Set termination type
       - Indicate SNA is to be stopped
       - Post SNA work ECB
6. Terminate the POWER/VS session.
   - a. Identify terminator task
   - b. Post terminator TCB dispatchable
   - c. Notify central operator
   - d. Call MSG → MSG

MESSAREA
1R991

TCB
TCBTT
CANCOD
TCBSF
TMCD

CPB
CPBID
0

SNDS
E — SNTT
I — SNSS
I — SNEP

TCB
TCBTI
TCBTT
TCBSF
TMCD

CPB
CPBID
0

PEND (end), Chart CP 48

ATTT1TCB (Part 2 of 2)

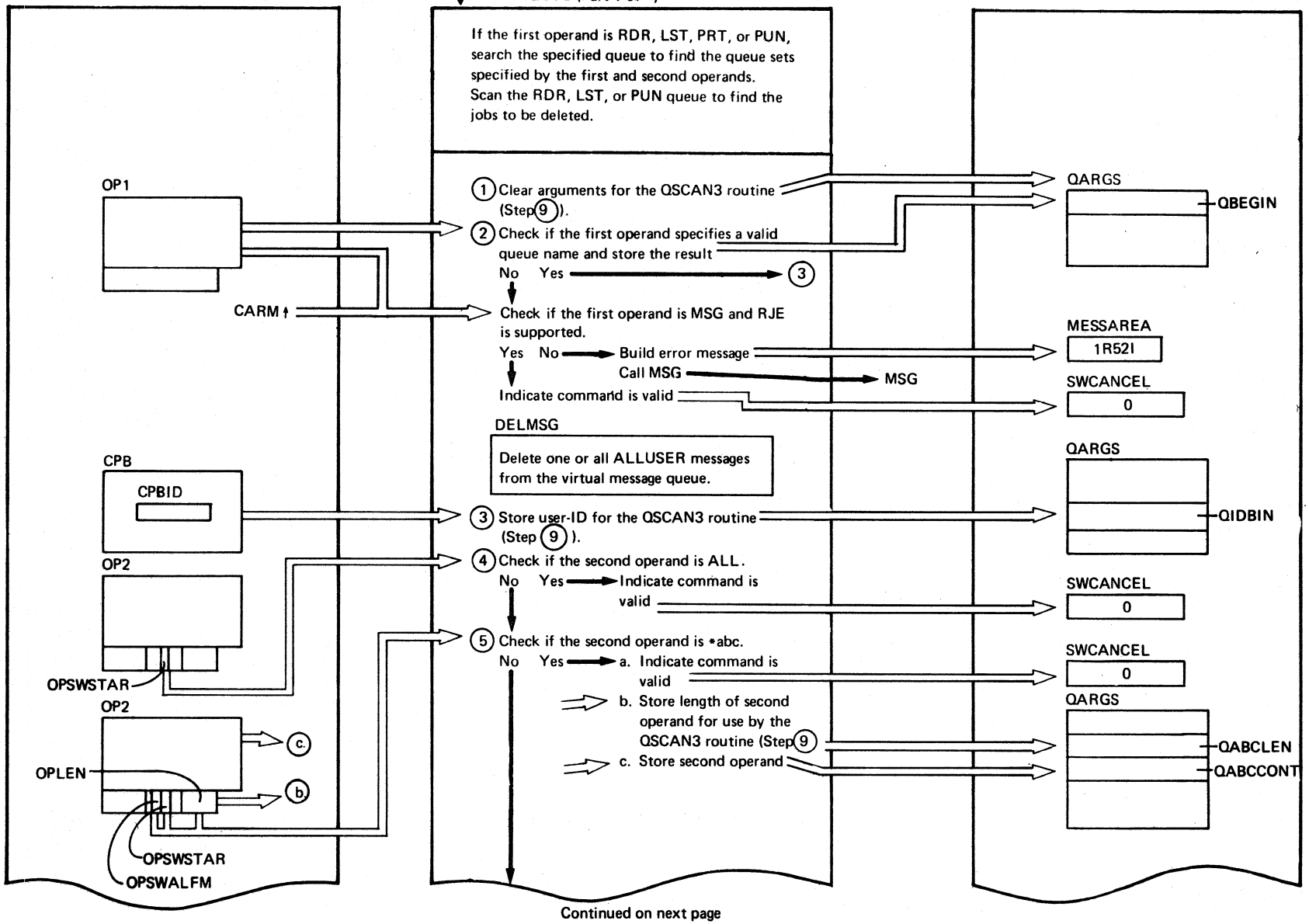
Extended Description	Include Segment	Call Subroutine	Chart
(1) 1R99I POWER/VS IS IN THE SHUTDOWN PERIOD			
(2) Termination code will be inserted in all TCBs Except: . SNA task TCBs . Execution writer TCBs . Execution reader TCBs from writer-only partitions. All TCBs will be set dispatchable Except: . Command processor TCB . Status report task TCB . Terminator task TCB			
(4)		MSG	CP75
(6)		MSG	CP75

Included by MAINLINE, Chart CP2

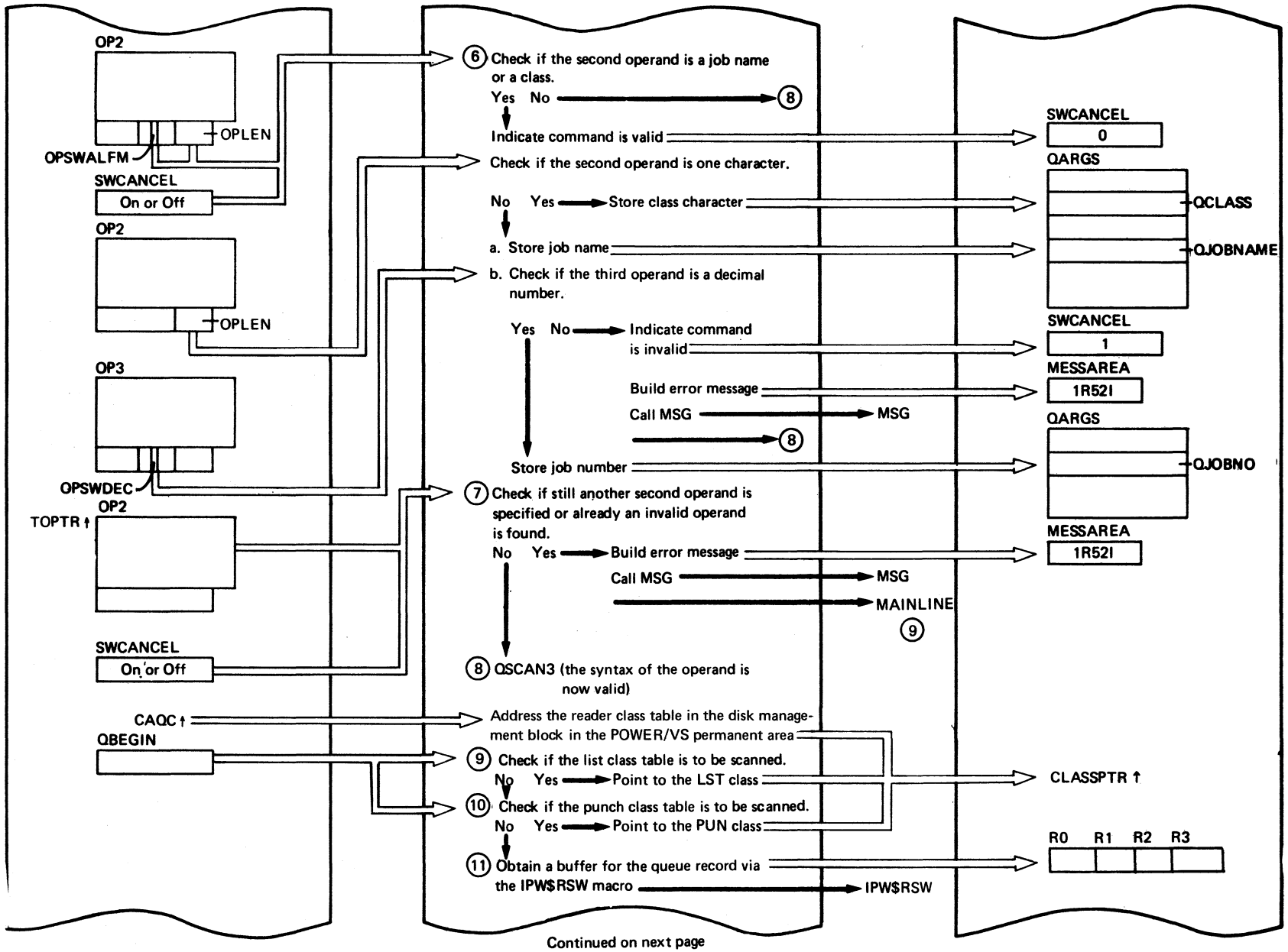
LEVEL 2

PDELETE (Part 1 of 4)

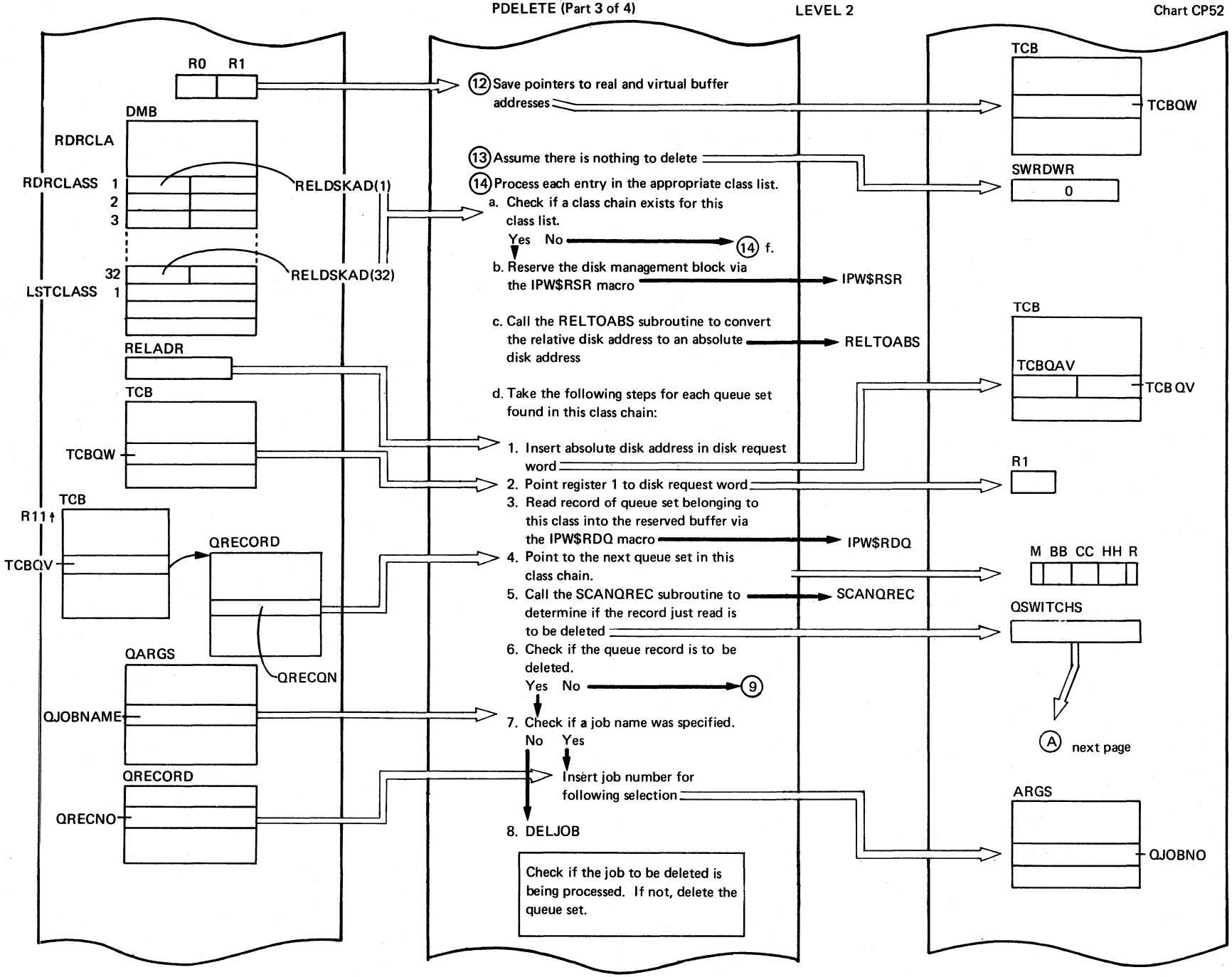
If the first operand is RDR, LST, PRT, or PUN, search the specified queue to find the queue sets specified by the first and second operands. Scan the RDR, LST, or PUN queue to find the jobs to be deleted.



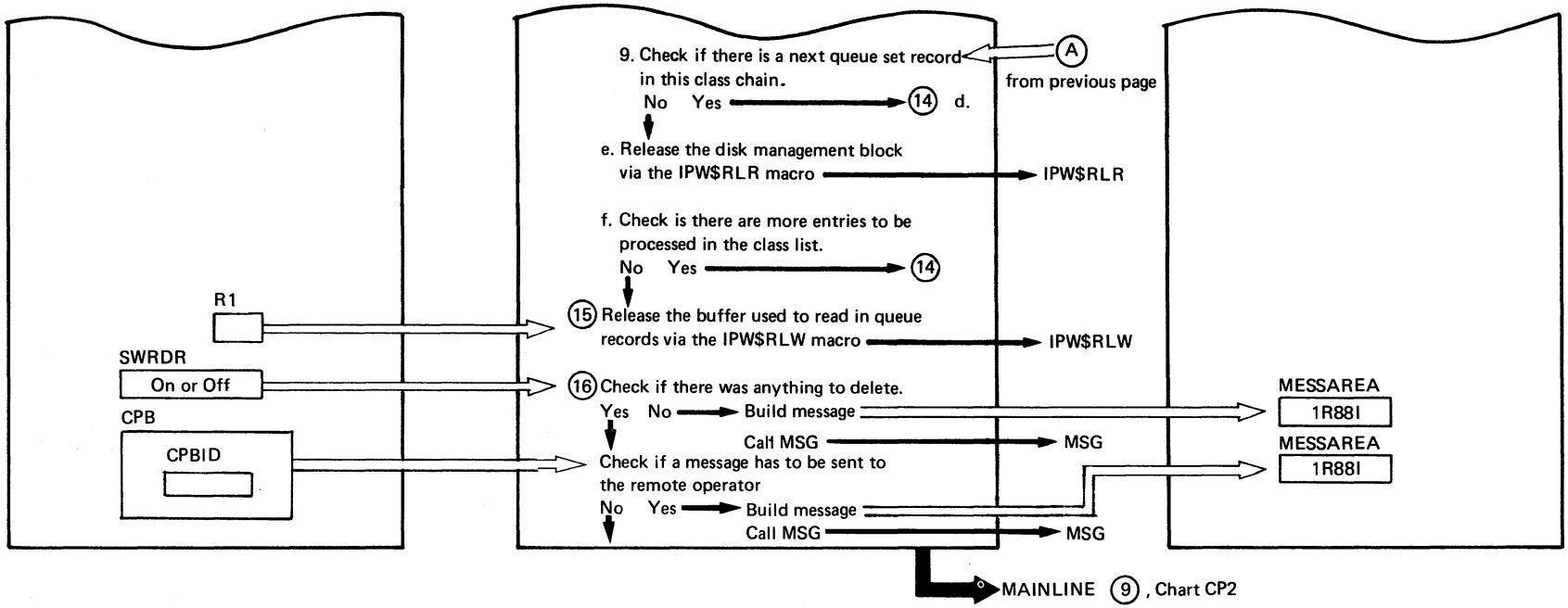
Continued on next page



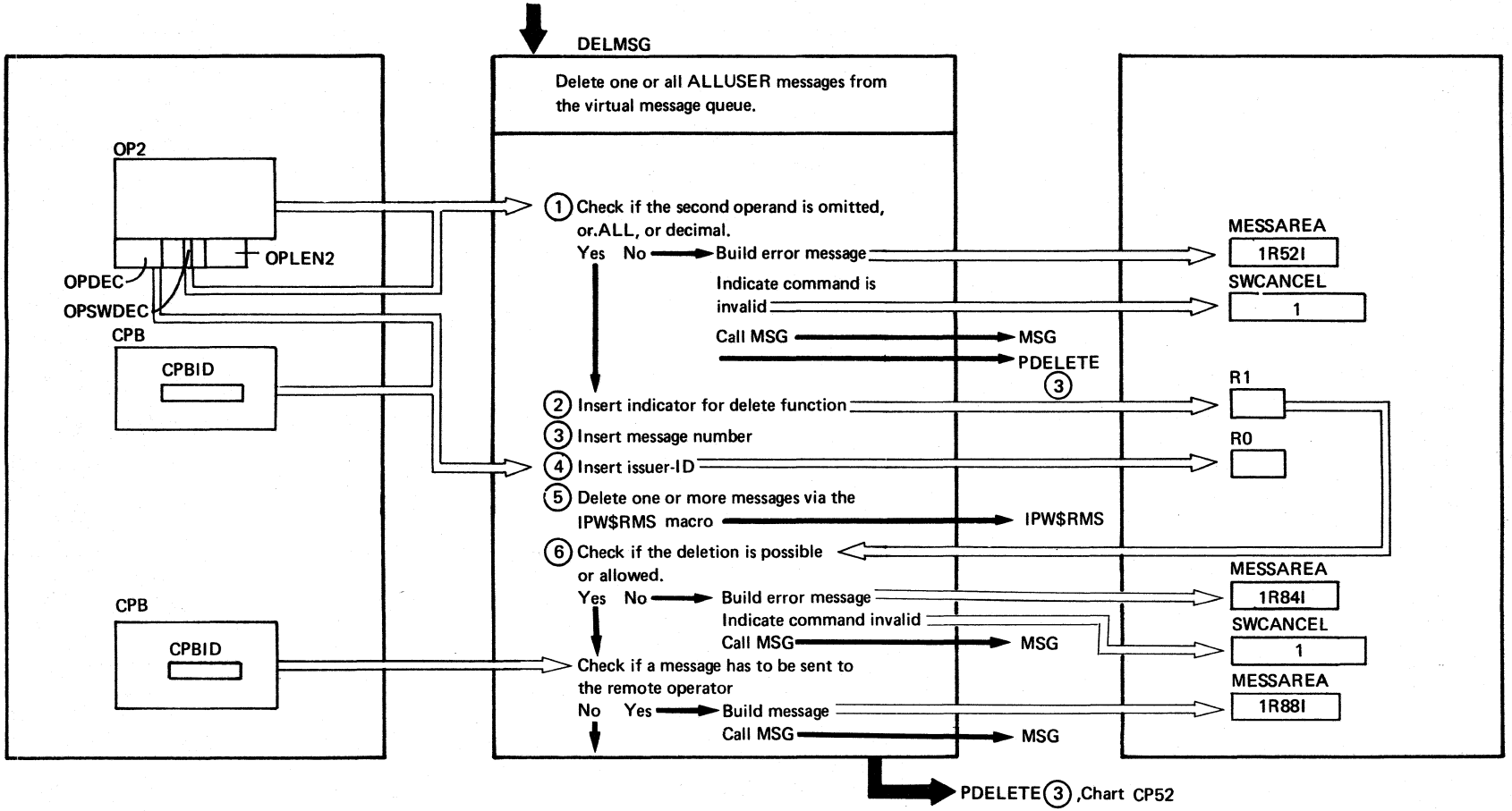
Continued on next page



Continued on next page

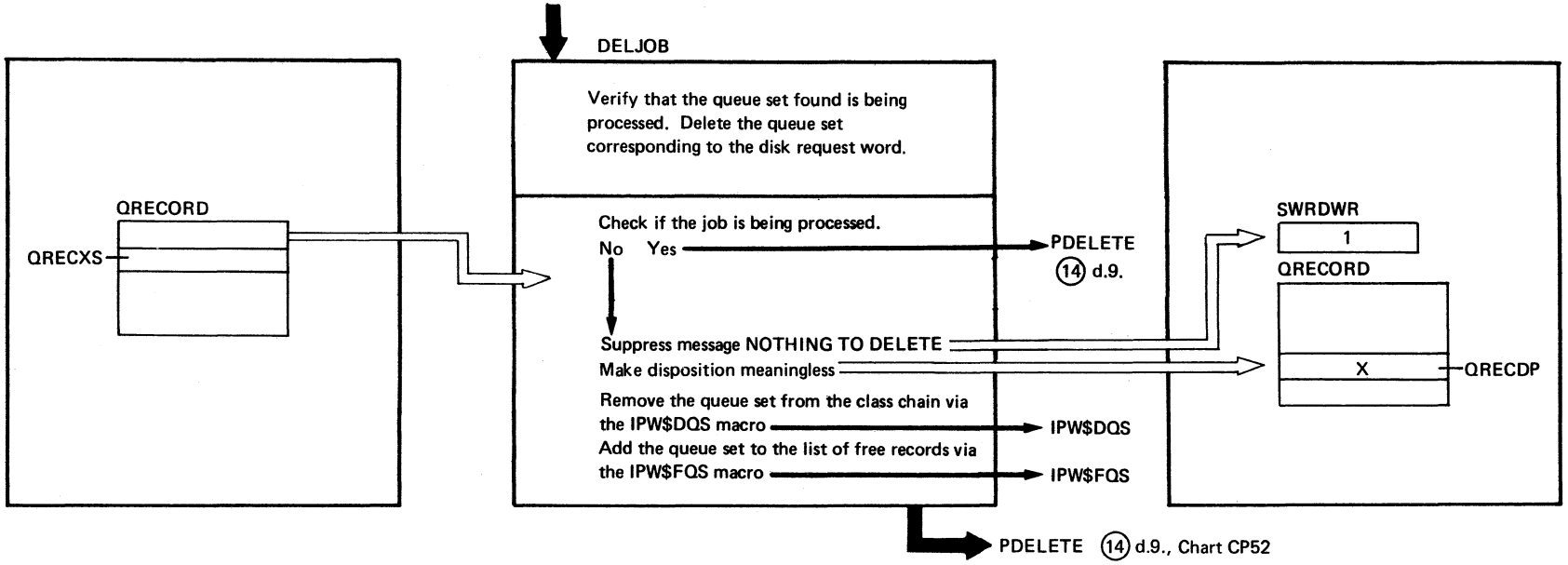


Extended Description			Include Segment	Call Subroutine	Chart
(2)	1R52I	OPERAND 1 NO VALID QUEUE		MSG	CP75
			DELMSG		CP53
(6)	b. 1R52I	OPERAND 3 NOT DECIMAL		MSG	CP75
(7)	1R52I	OPERAND 2 INVALID		MSG	CP75
(8)	Register 15 is used to calculate this address.				
(14)	c.			RELTOABS	CP84
	d.5.			SCANQREC	CP85
	d.8.		DELJOB		CP54
(16)	1R88I	NOTHING TO DELETE		MSG	CP75
	1R88I	CP READY		MSG	CP75



Extended Description	Include Segment	Call Subroutine	Chart
1 1R52I OPERAND 2 INVALID		MSG	CP75
5 The IPW$RMS macro uses registers 0, 1, 2, and 3			
6 1R84I DELETION NOT ALLOWED OR IMPOSSIBLE		MSG	CP75
1R88I CP READY		MSG	CP75

Included by PDELETE, Chart CP52 LEVEL 3 Chart CP54



Included by MAINLINE, Chart CP2

LEVEL 2

Chart CP55

PSTOP Part (1 of 3)

Provide the task control block specified by the first operand with a cancel code.
Note that '* . . STOP' RJE commands are not supposed to be processed by this program.

OPERAND
ORIGINAL
OPSWHEX
OPLEN

(1) Initialize the default cancel code and clear all argument fields for the TCBSCAN subroutine. → CANCOD: S

(2) Check if a valid partition identifier was specified.
No / Yes →
a. Assume command is valid → SWCANCEL: 0
b. Insert cancel code for execution reader → CANCOD: E
c. Set up fields for the TCBSCAN subroutine → ARGTI: EbBG or EbFn

(3) Check if a device address was specified.
No / Yes →
a. Assume command is valid → SWCANCEL: 0
b. Set up fields for the TCBSCAN subroutine → ARGCU: bcuu

SWANCEL
On or Off

(4) Check if one of the operands mentioned above was specified.

OPERAND
ORIGINAL

Yes / No

Check if stop SNA was requested.

No / Yes

Assume command is valid → SWCANCEL: 0

PSTOPSNA
Stop SNA

→ MAINLINE (9)

Build error message → MESSAREA: 1R52I

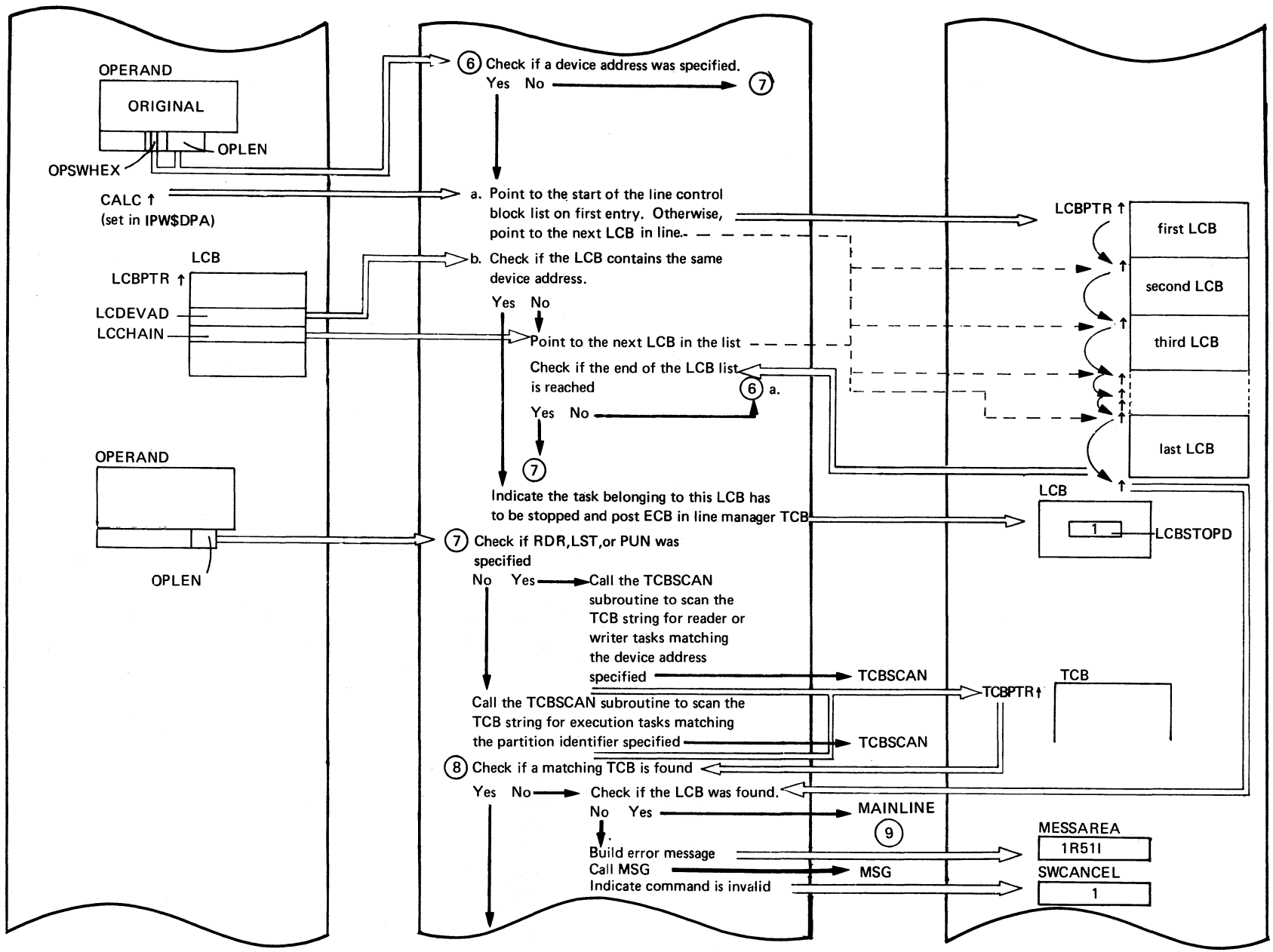
Call MSG → MSG

→ MAINLINE (9)

(5) Initialize the line control block pointer → LCBPTR↑ = 0

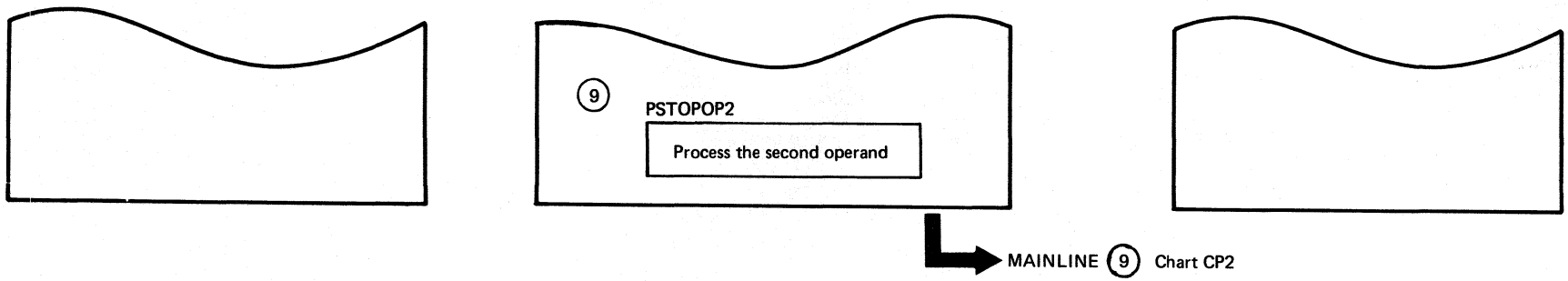
continued on next page

PSTOP (Part 2 of 3) LEVEL 2



(Continued on next page)

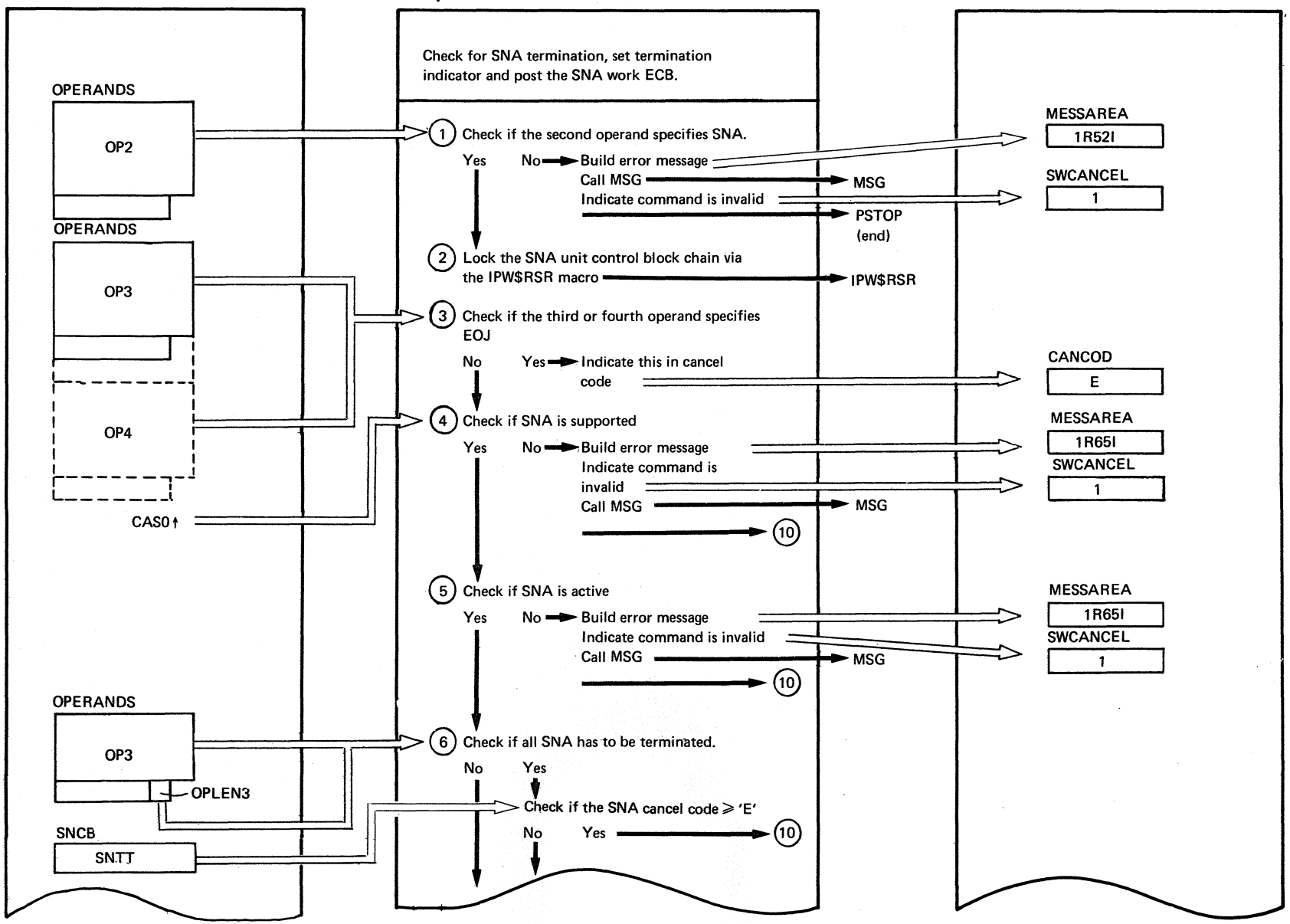
PSTOP (Part 3 of 3)



Extended Description	Include Segment	Call Subroutine	Chart
	PSTOPSNA		CP56
(4) 1R52I OPERAND 1 MISSING OR INVALID		MSG	CP75
(7)		TCBSCAN	CP82
(8) 1R51I OPERAND 1 DESIGNATES NON-EXISTING TASK		MSG	CP75
(9)	PSTOPOP2		CP57

Included by PSTOP, Chart CP55 LEVEL 3 Chart CP56

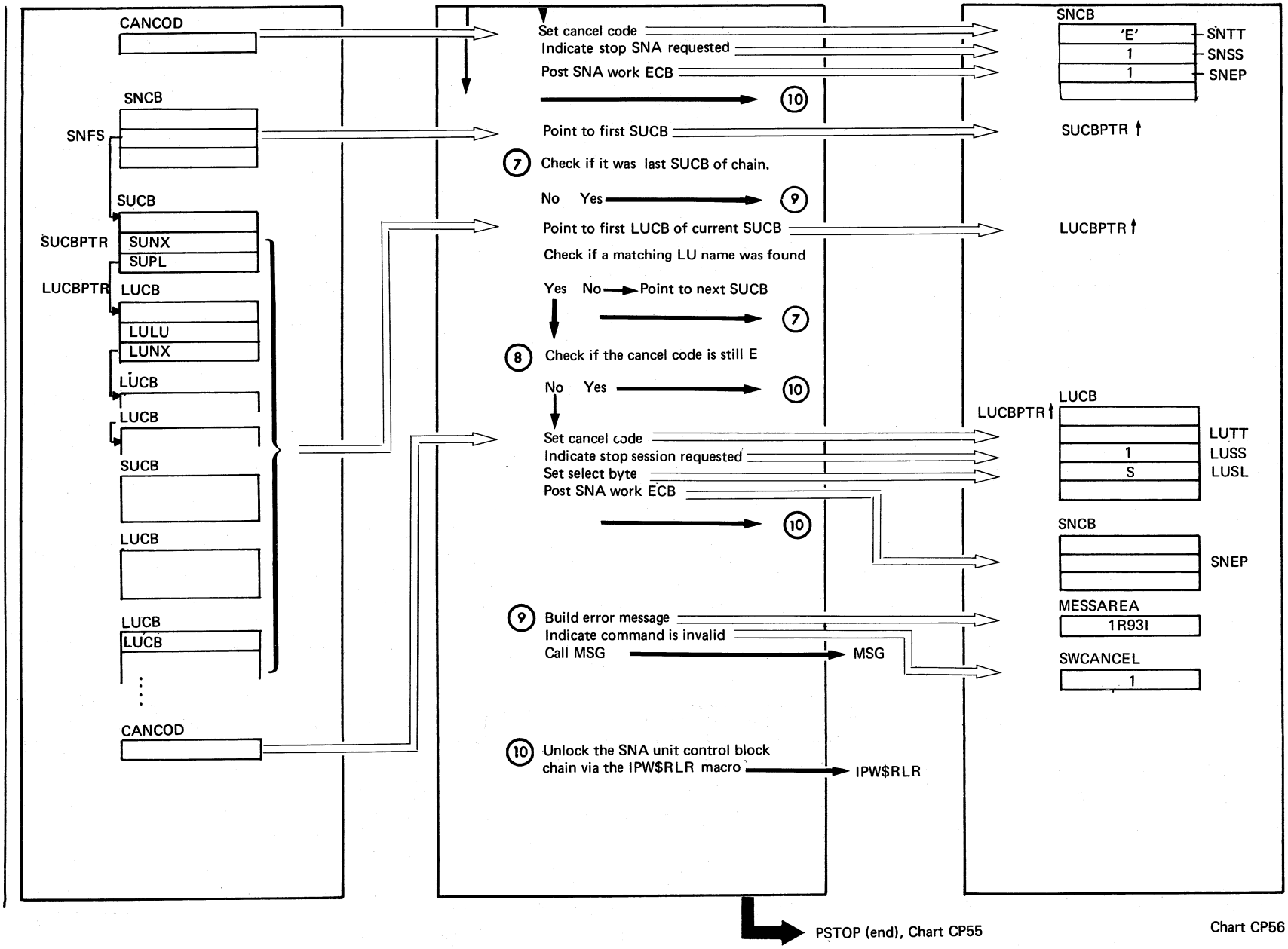
PSTOPSNA (Part 1 of 3)



Continued on next page

Chart CP56

PSTOPSNA (Part 2 of 3)



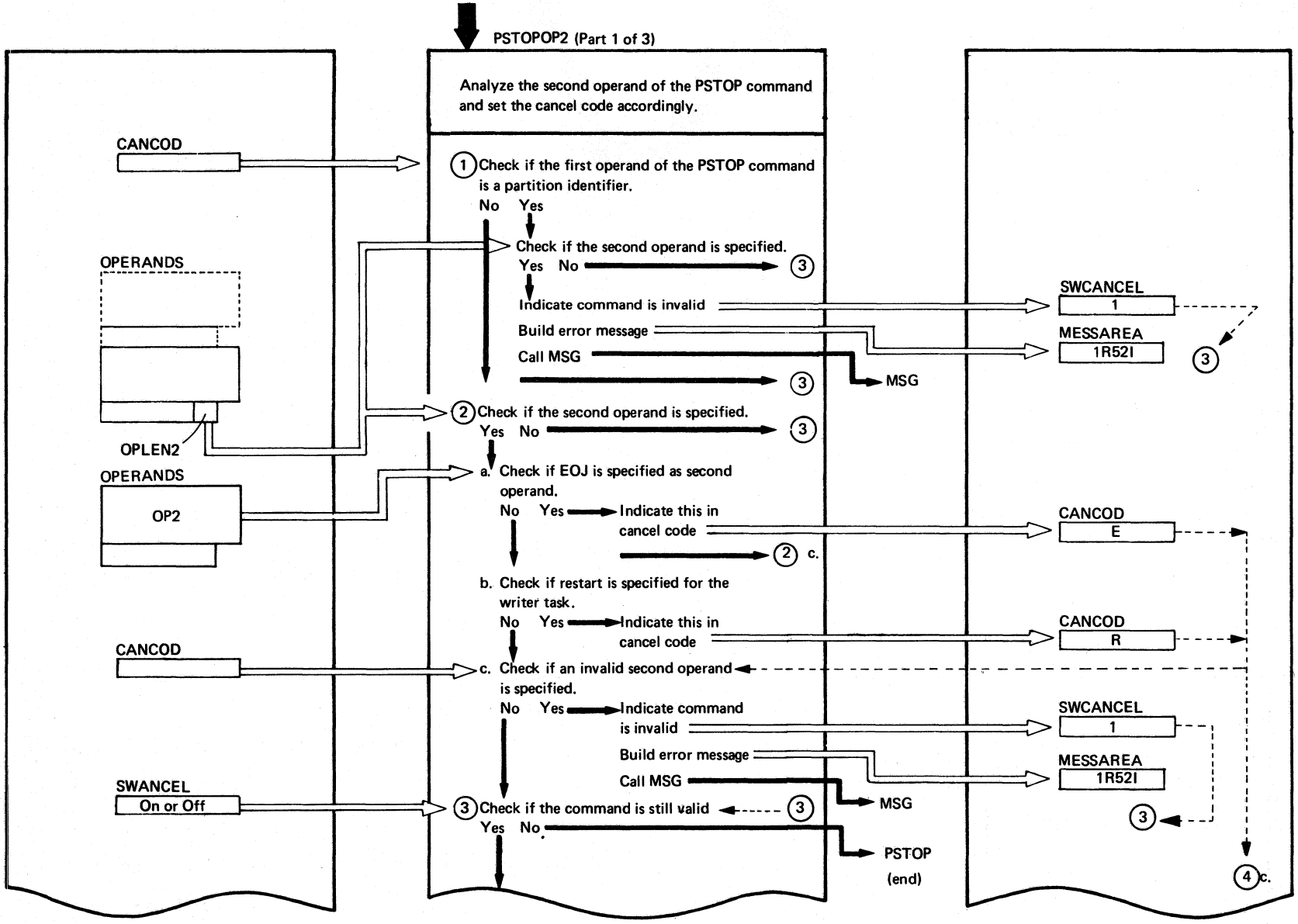
PSTOPSNA (Part 3 of 3)

Extended Description	Include Segment	Call Subroutine	Chart
(1) 1R52I OPERAND 2 INVALID		MSG	CP75
(3) The IPW$RSR macro uses registers 0, 1, 2, and 3 which are restricted in this segment.			
(4) 1R65I RJE, SNA NOT SUPPORTED		MSG	CP75
(5) 1R65I RJE, SNA NOT ACTIVE		MSG	CP75
(8) 1R93I NO SESSION ESTABLISHED FOR xxxxxxxx ↑ logical unit name specified in PSTOP command		MSG	CP75

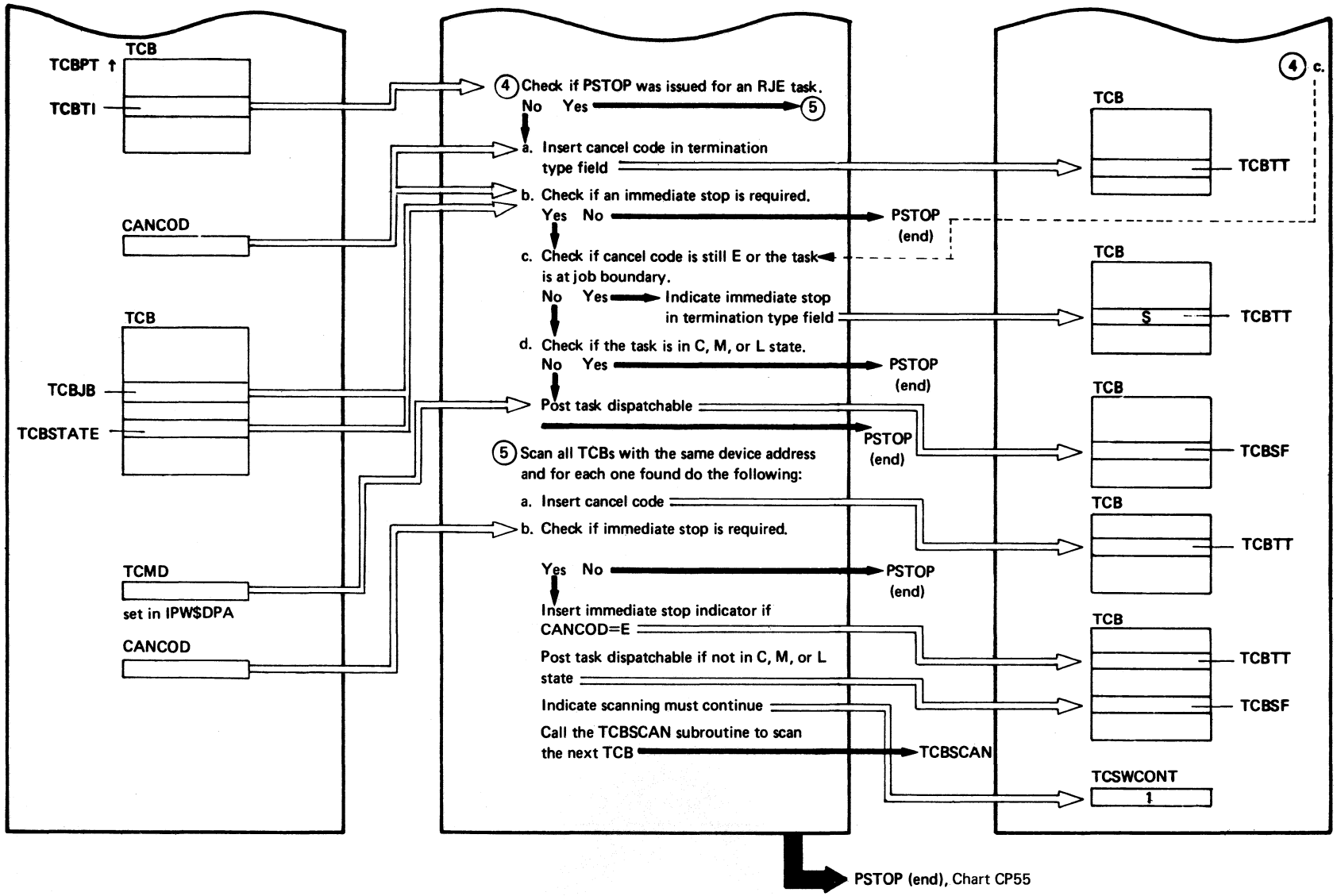
Included by PSTOP, Chart CP55

LEVEL 3

Chart CP57



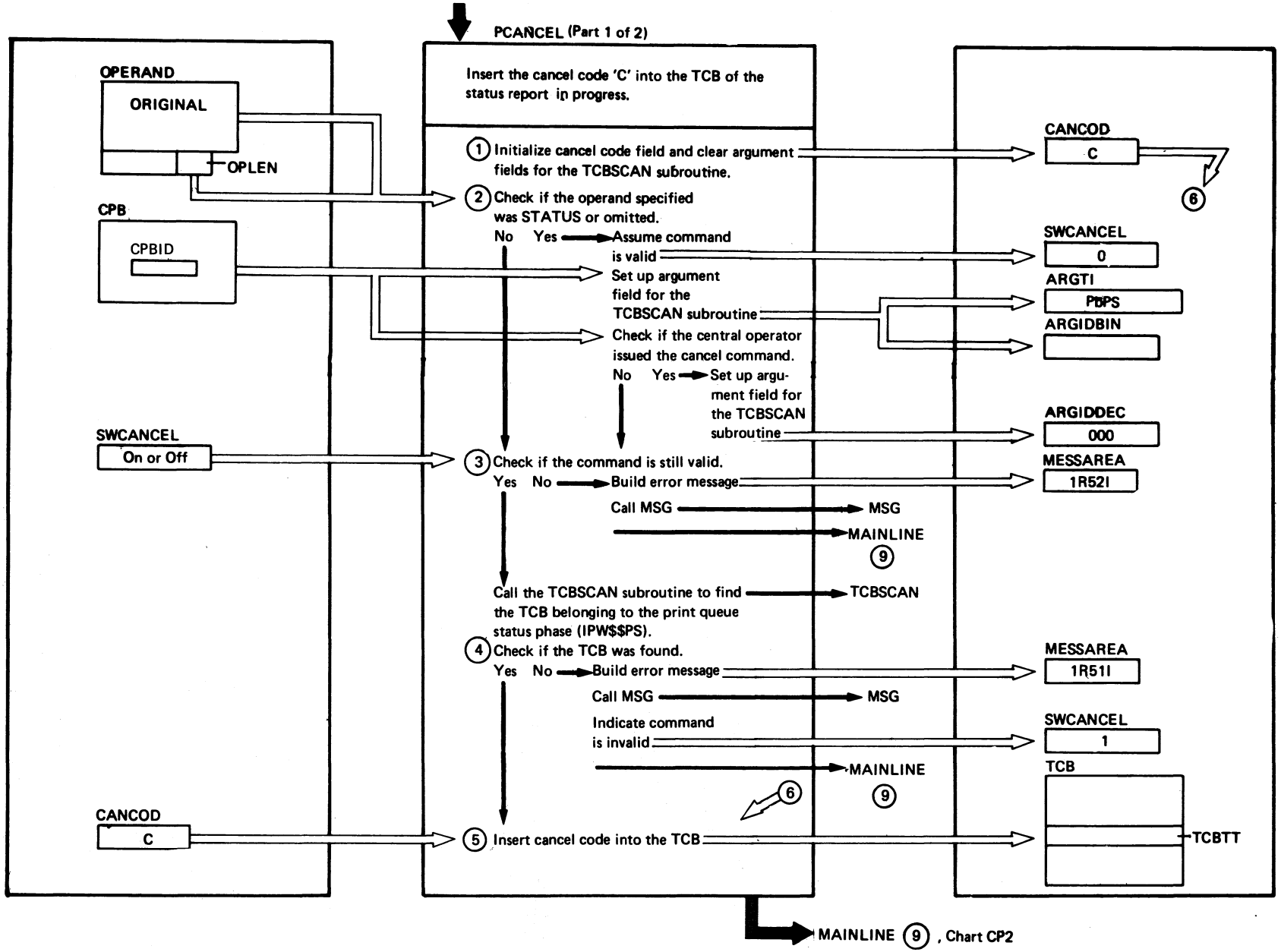
continued on next page


TCB
TCBPT
TCBTI
CANCOD
TCB
TCBJB
TCBSTATE
TCMD
set in IPW$DPA
CANCOD
4 Check if PSTOP was issued for an RJE task.
No Yes 5
a. Insert cancel code in termination type field
b. Check if an immediate stop is required.
Yes No PSTOP (end)
c. Check if cancel code is still E or the task is at job boundary.
No Yes Indicate immediate stop in termination type field
d. Check if the task is in C, M, or L state.
No Yes PSTOP (end)
Post task dispatchable
PSTOP (end)
5 Scan all TCBs with the same device address and for each one found do the following:
a. Insert cancel code
b. Check if immediate stop is required.
Yes No PSTOP (end)
Insert immediate stop indicator if CANCOD=E
Post task dispatchable if not in C, M, or L state
Indicate scanning must continue
Call the TCBSCAN subroutine to scan the next TCB
TCBSCAN
4 c.
TCB
TCBTT
TCB
S
TCBTT
TCB
TCBSF
TCB
TCBTT
TCB
TCBTT
TCBSF
TCSWCONT
1
PSTOP (end), Chart CP55

Extended Description	Include Segment	Call Subroutine	Chart
(1) 1R52I NO SECOND OPERAND ALLOWED FOR PSTOP PARTITION		MSG	CP75
(2) c. 1R52I SECOND OPERAND INVALID		MSG	CP75
(5) b.		TCBSCAN	CP82

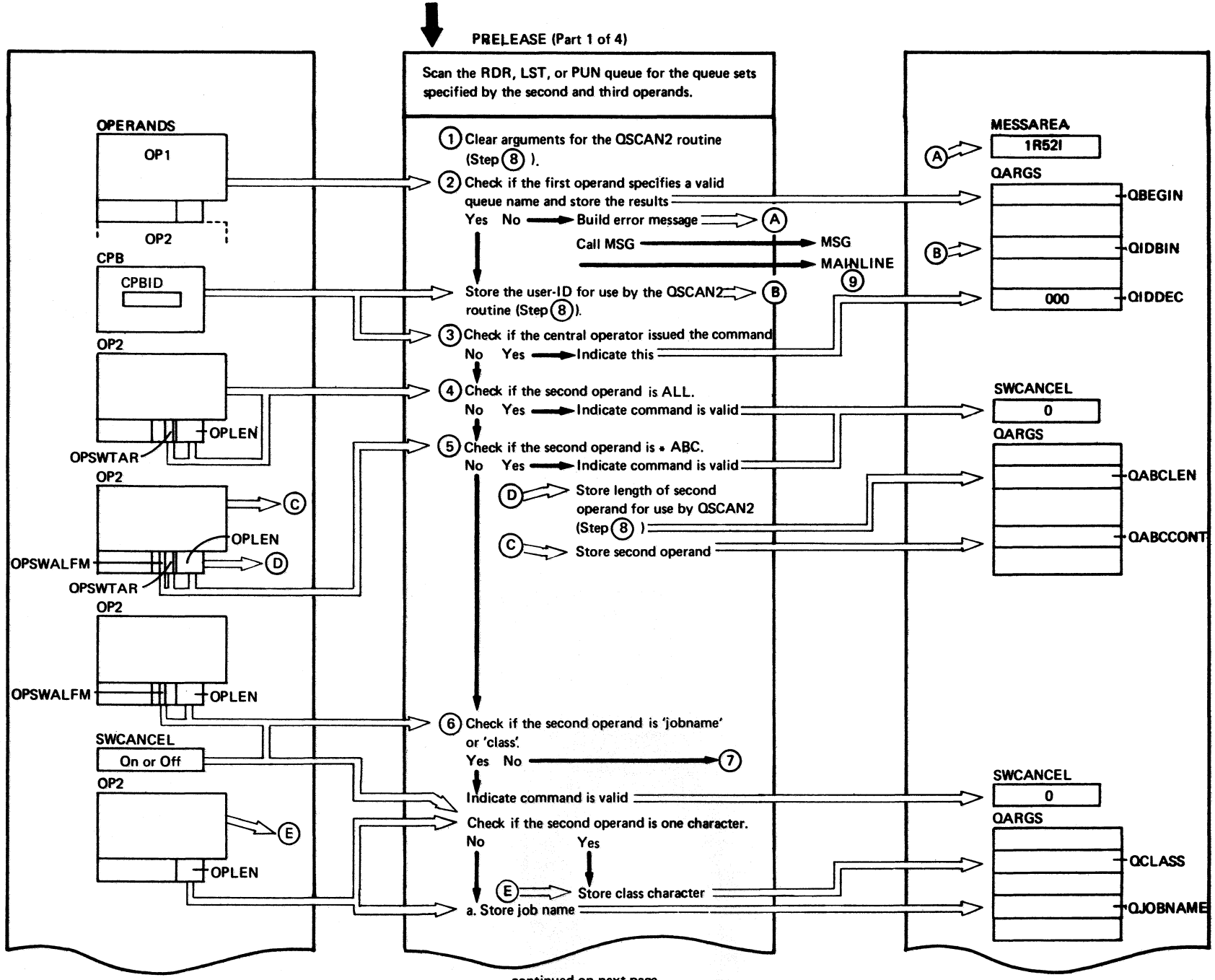
Included by MAINLINE, Chart CP2

LEVEL 2



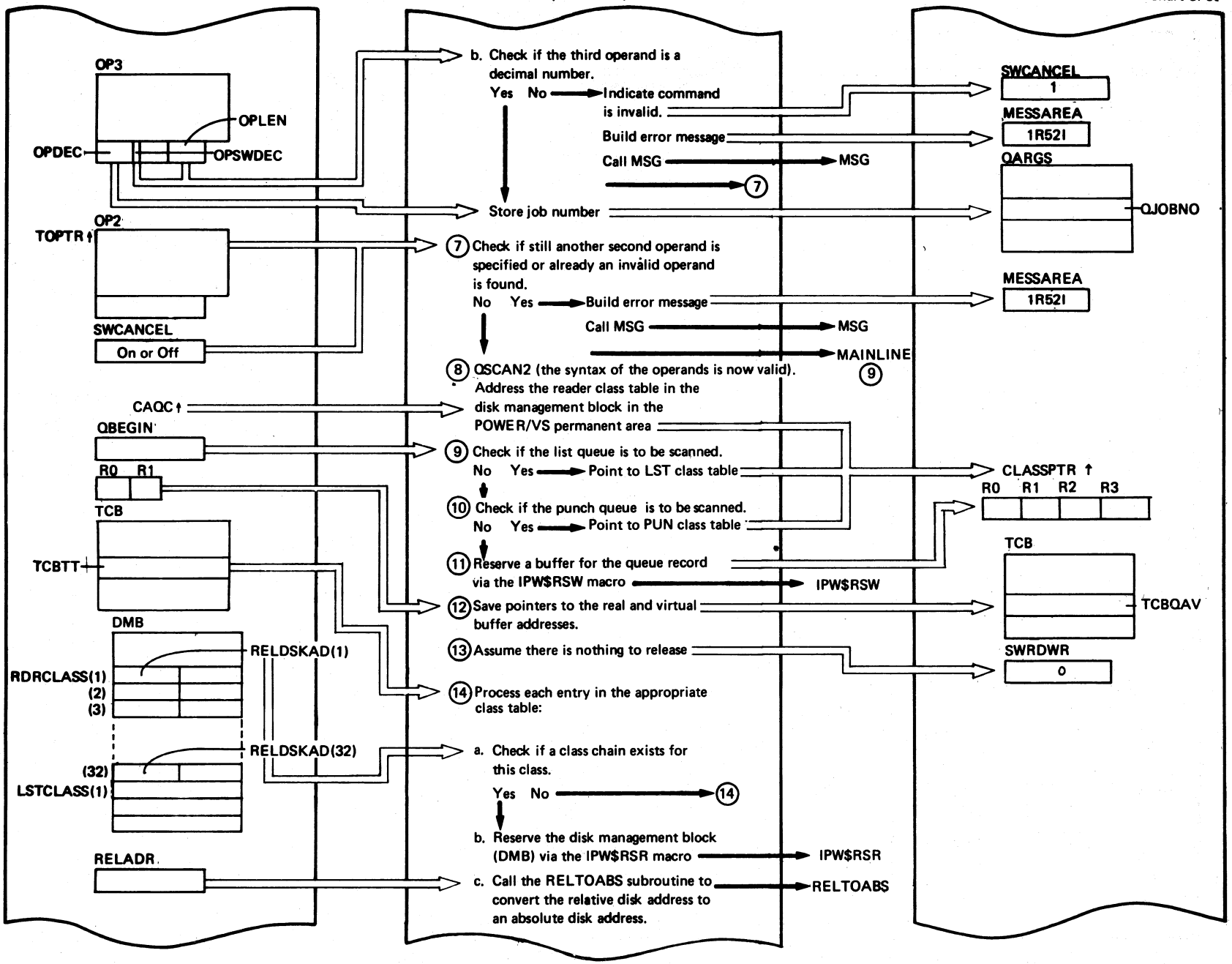
Extended Description	Include Segment	Call Subrouitne	Chart
(2) ARGIDBIN is used to indicate that only the remote operator can cancel a remote report. ARGIDDEC is used to indicate that only the local operator can cancel a local report.			
(3) 1R52I OPERAND 1 NOT 'STATUS'		MSG	CP75
(4) 1R51I NO STATUS REPORT IN PROGRESS		MSG	CP75

Included by MAINLINE, Chart CP2 LEVEL 2 Chart CP59

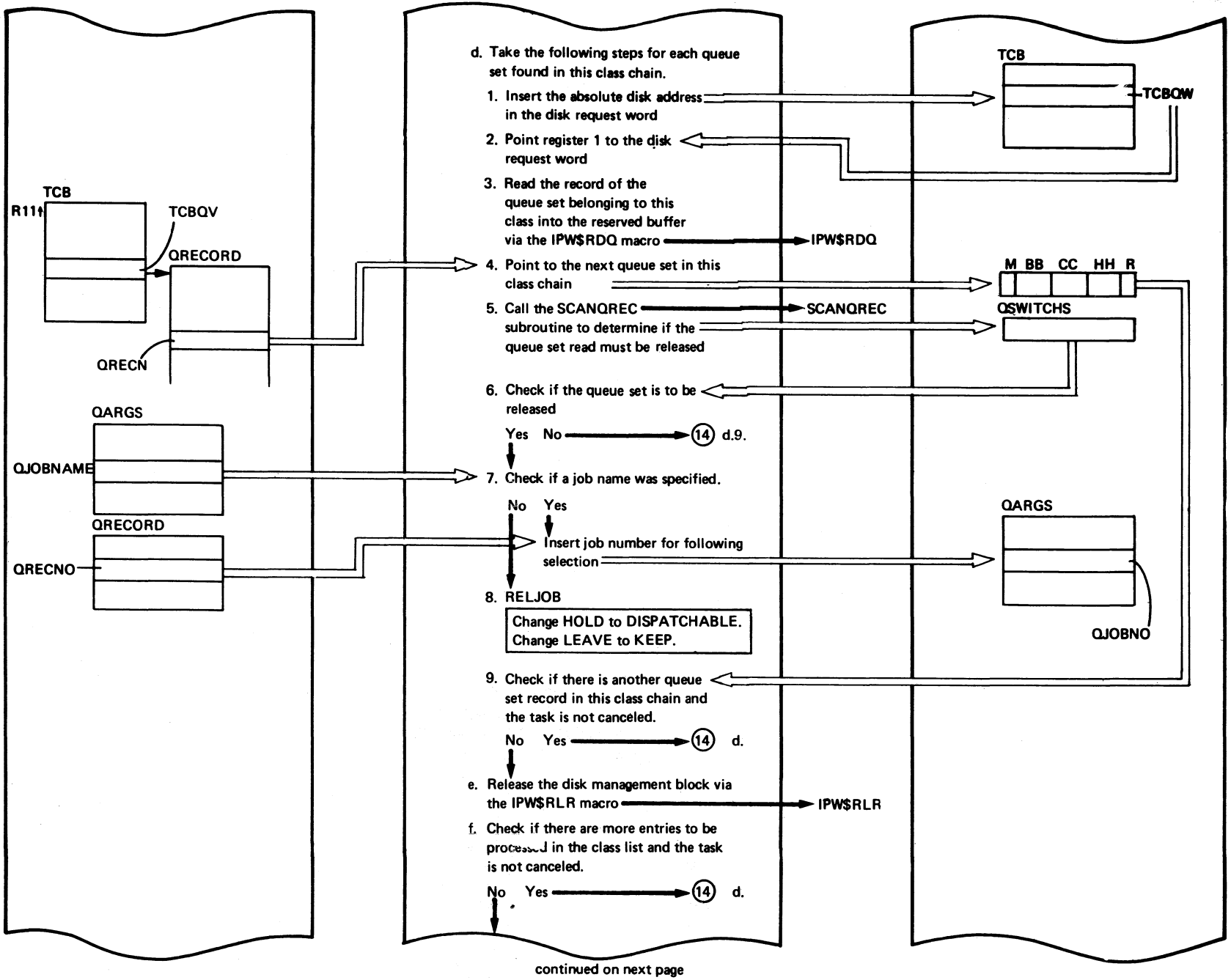


continued on next page

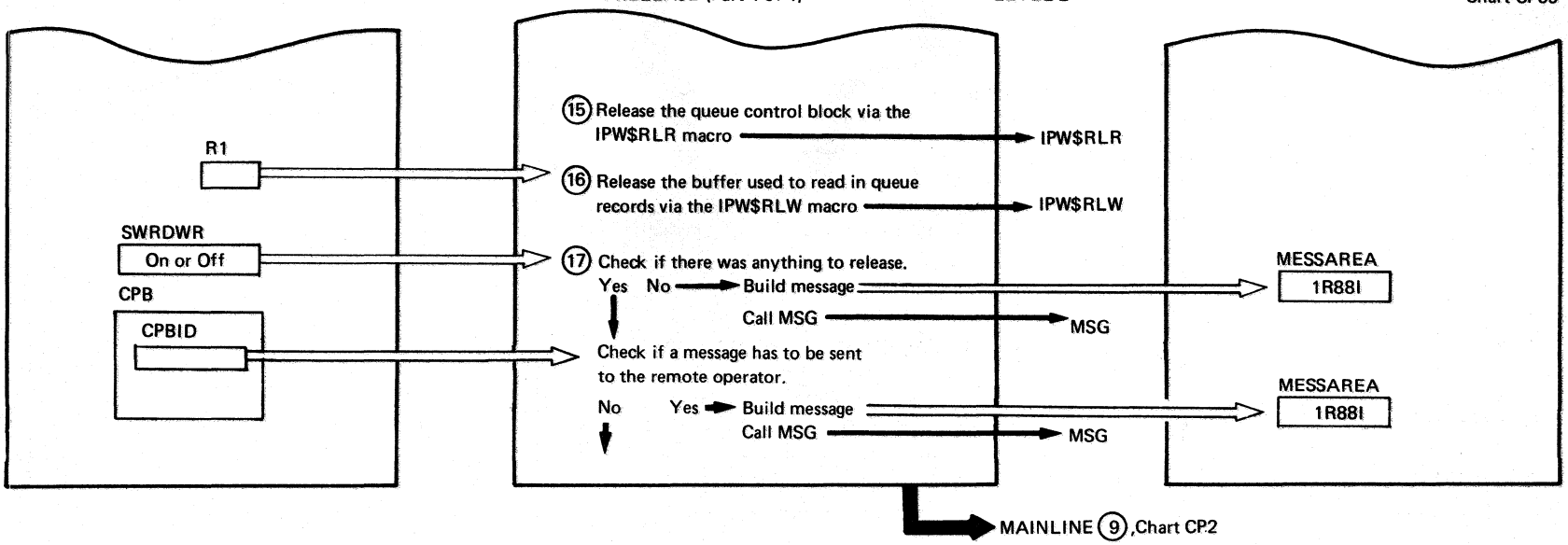
Chart CP59



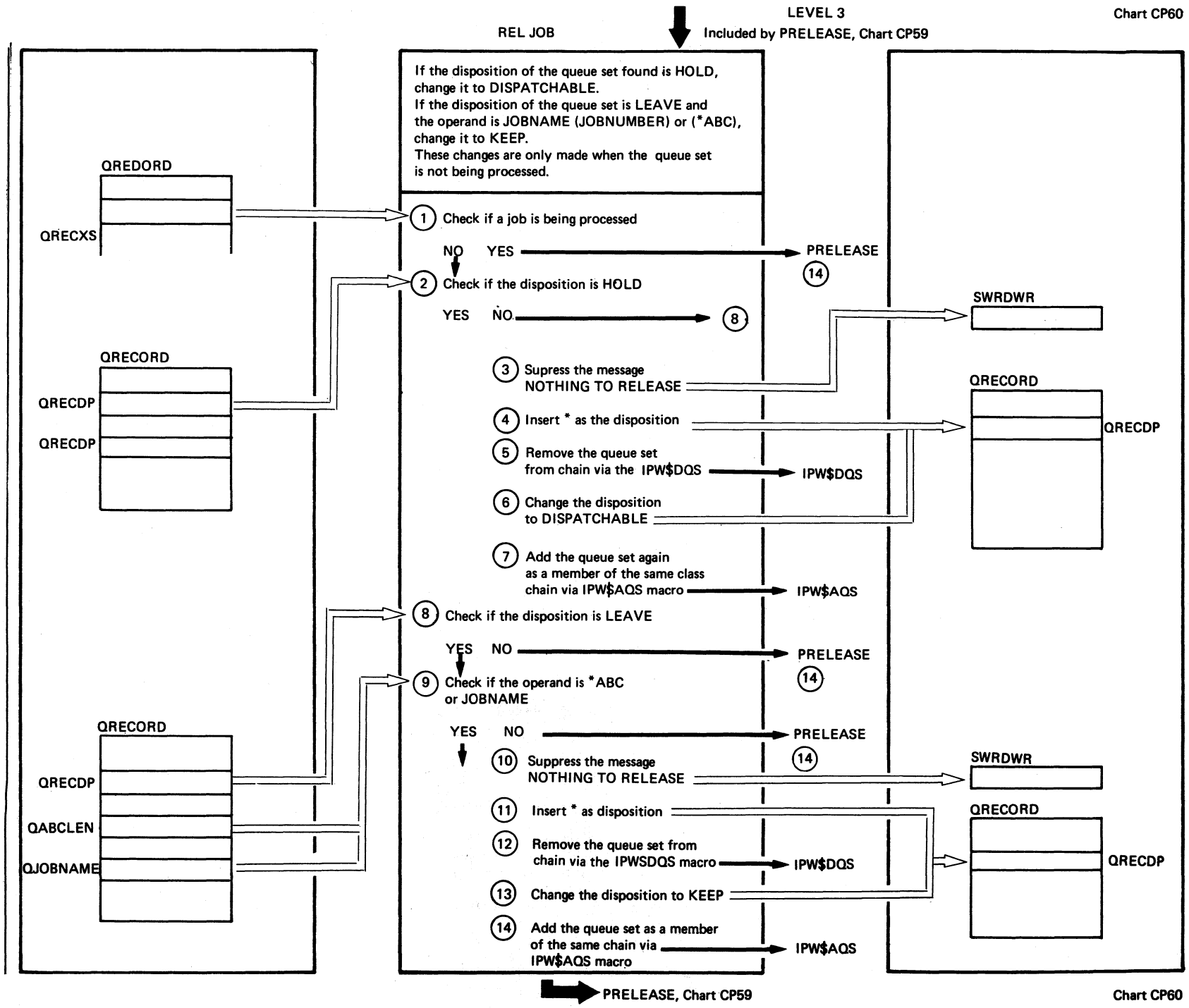
continued on next page



continued on next page



Extended Description	Include Segment	Call Subroutine	Chart
(2) 1R52I OPERAND 1 NO VALID QUEUE		MSG	CP75
(6) b. 1R52I OPERAND 3 NOT DECIMAL		MSG	CP75
(7) 1R52I OPERAND 2 INVALID		MSG	CP75
(14) b.		RELTOABS	CP84
d.		SCANQREC	CP85
d. 8	RELJOB		CP60
(17) 1R88I NOTHING TO RELEASE		MSG	CP75
1R88I CP READY		MSG	CP75


REL JOB
LEVEL 3
Included by PRELEASE, Chart CP59
Chart CP60
If the disposition of the queue set found is HOLD, change it to DISPATCHABLE.
If the disposition of the queue set is LEAVE and the operand is JOBNAME (JOBNUMBER) or (*ABC), change it to KEEP.
These changes are only made when the queue set is not being processed.
QREDORD
QRECXS
QRECORD
QRECDP
QRECDP
QRECORD
QRECDP
QABCLEN
QJOBNAME
1 Check if a job is being processed
NO
YES
PRELEASE
14
2 Check if the disposition is HOLD
YES
NO
8
SWRDWR
3 Supress the message NOTHING TO RELEASE
QRECORD
QRECDP
4 Insert * as the disposition
5 Remove the queue set from chain via the IPW$DQS
IPW$DQS
6 Change the disposition to DISPATCHABLE
7 Add the queue set again as a member of the same class chain via IPW$AQS macro
IPW$AQS
8 Check if the disposition is LEAVE
YES
NO
PRELEASE
14
9 Check if the operand is *ABC or JOBNAME
YES
NO
PRELEASE
14
10 Suppress the message NOTHING TO RELEASE
SWRDWR
11 Insert * as disposition
QRECORD
QRECDP
12 Remove the queue set from chain via the IPWSDQS macro
IPW$DQS
13 Change the disposition to KEEP
14 Add the queue set as a member of the same chain via IPW$AQS macro
IPW$AQS
PRELEASE, Chart CP59
Chart CP60

Included by MAINLINE ,Chart CP2

LEVEL 2

Chart CP61

PRESTART (Part 1 of 4)

If the second operand is a valid signed or unsigned decimal term, or not specified, insert the second operand in the task control block at label TCRS.

(1) Clear argument fields for the TCBSCAN subroutine.

CPB

CPBID

(2) Check if the command was issued by the central operator.

RESTOCL

No Yes → Find the TCB of the task to be restarted by the central operator.

RESTREMT

Find the TCB of the writer task to be restarted by a remote operator.

TCB

TCBPT

TCBJB

(3) Check if the command is valid and the task is at job boundary.

No Yes → Indicate this → SWANCEL 1

Build message → MESSAREA 1R57I

Call MSG → MSG

SWCANCEL

On or Off

TCB

TCBPT

TCBRSTYP

(4) Check if the command is valid and a PSETUP or PRESTART command was issued.

No Yes → Indicate command is invalid → SWCANCEL 1

Build message → MESSAREA 1R82I

Call MSG → MSG

SWCANCEL

On or Off

(5) Check if the command is valid

Yes No → (12)

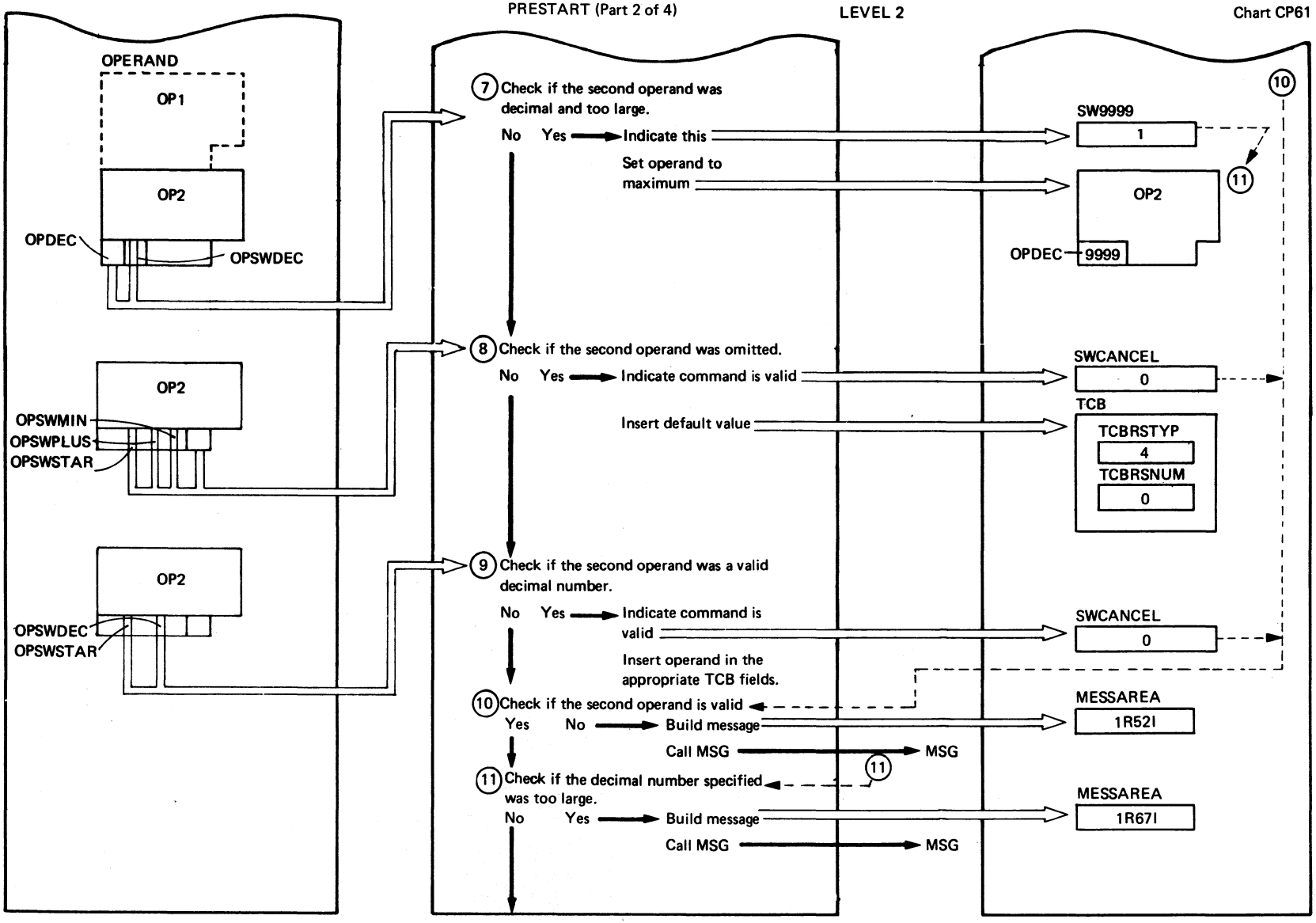
SWCANCEL

On or Off

Process the second operand

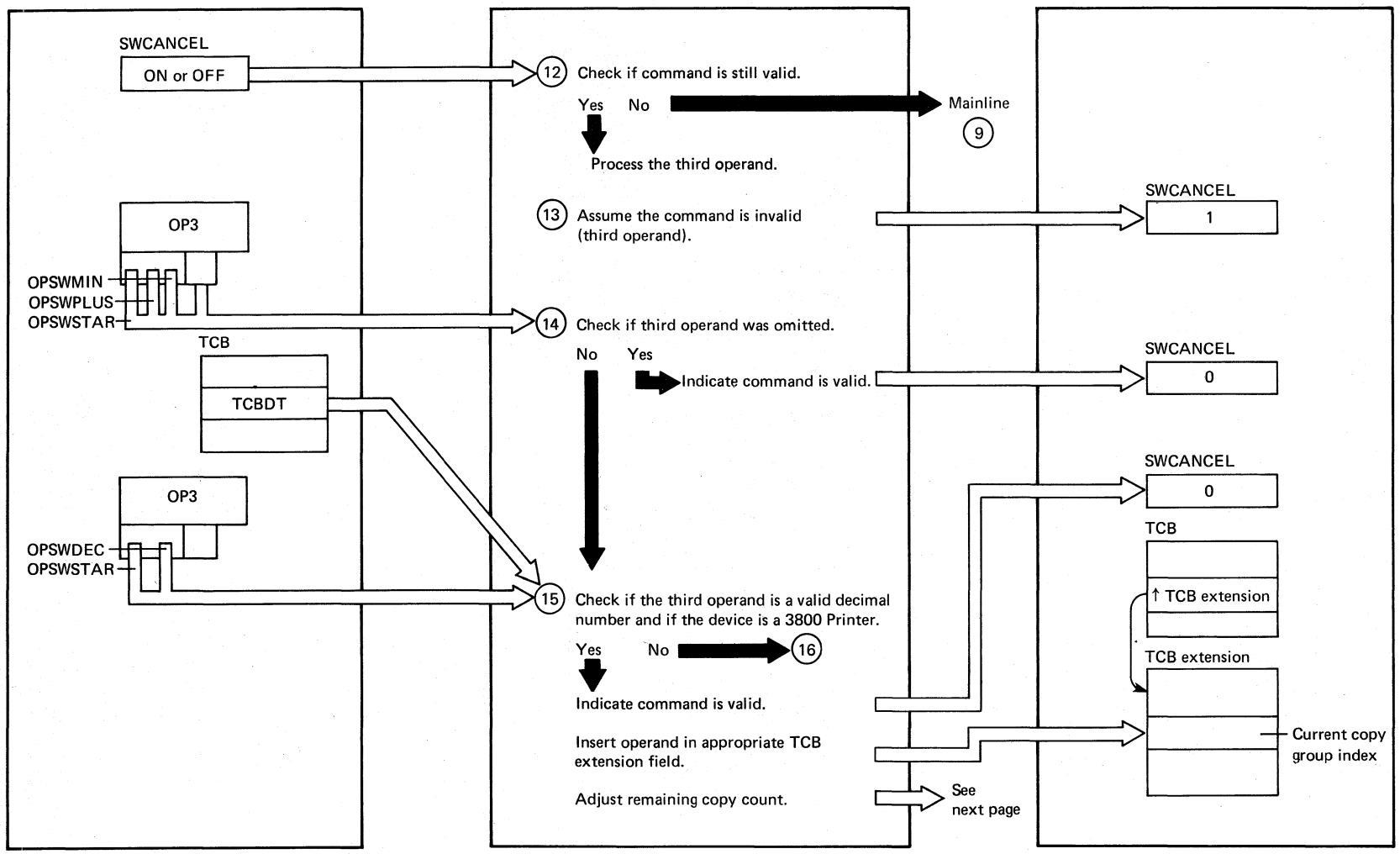
(6) Assume the command is invalid (second operand) → SWCANCEL 1 → (10)

continued on next page



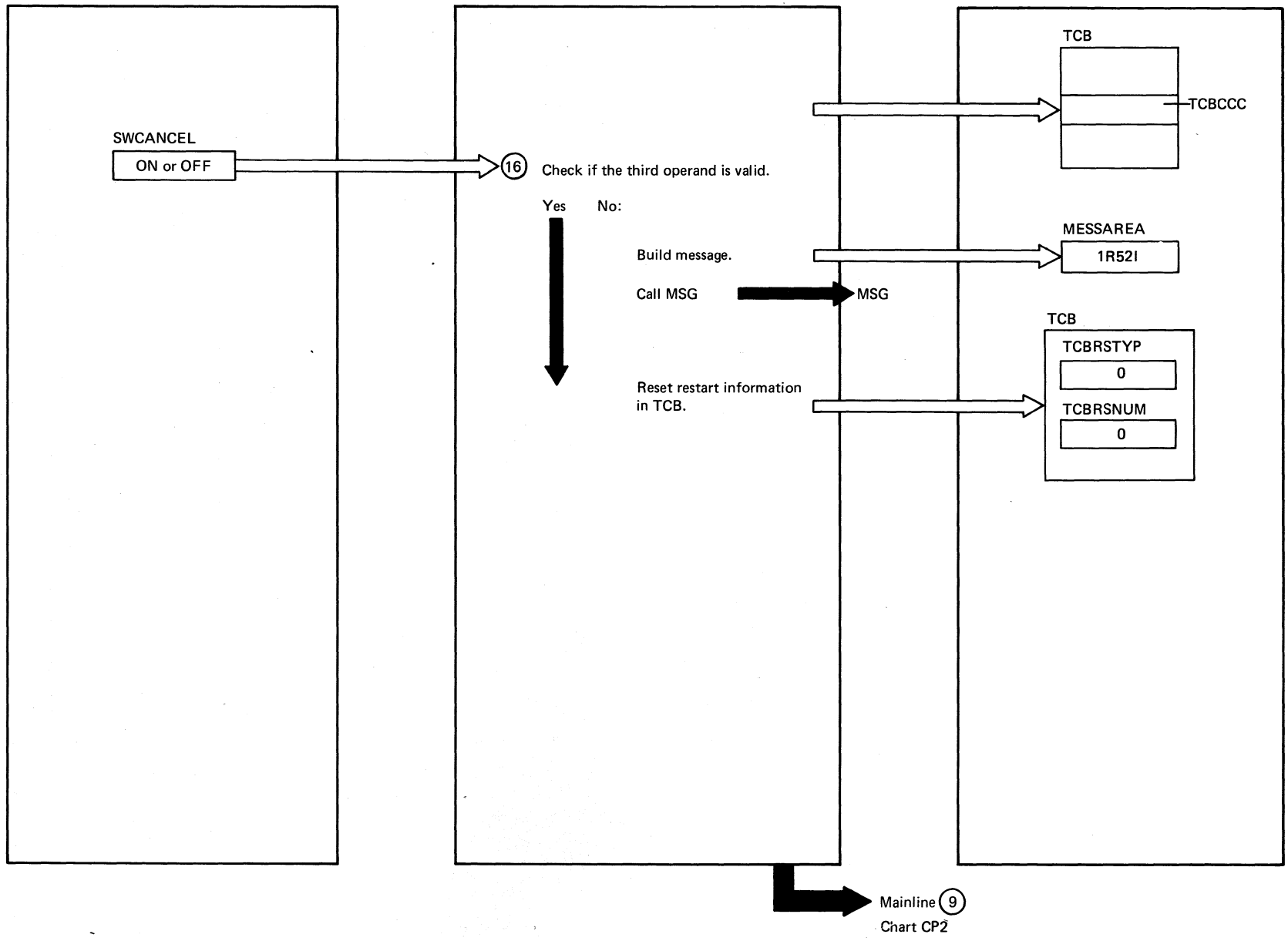
Continued on next page

Page of SY33-8577-1, Revised November 24, 1977, By TNL SN33-9241



Extended Description	Include Segment	Call Subroutine	Chart
(12)			
(13)			
(14)			
(15)			
(16) 1R52I OPERAND 3 invalid		MSG	CP75

Chart CP61

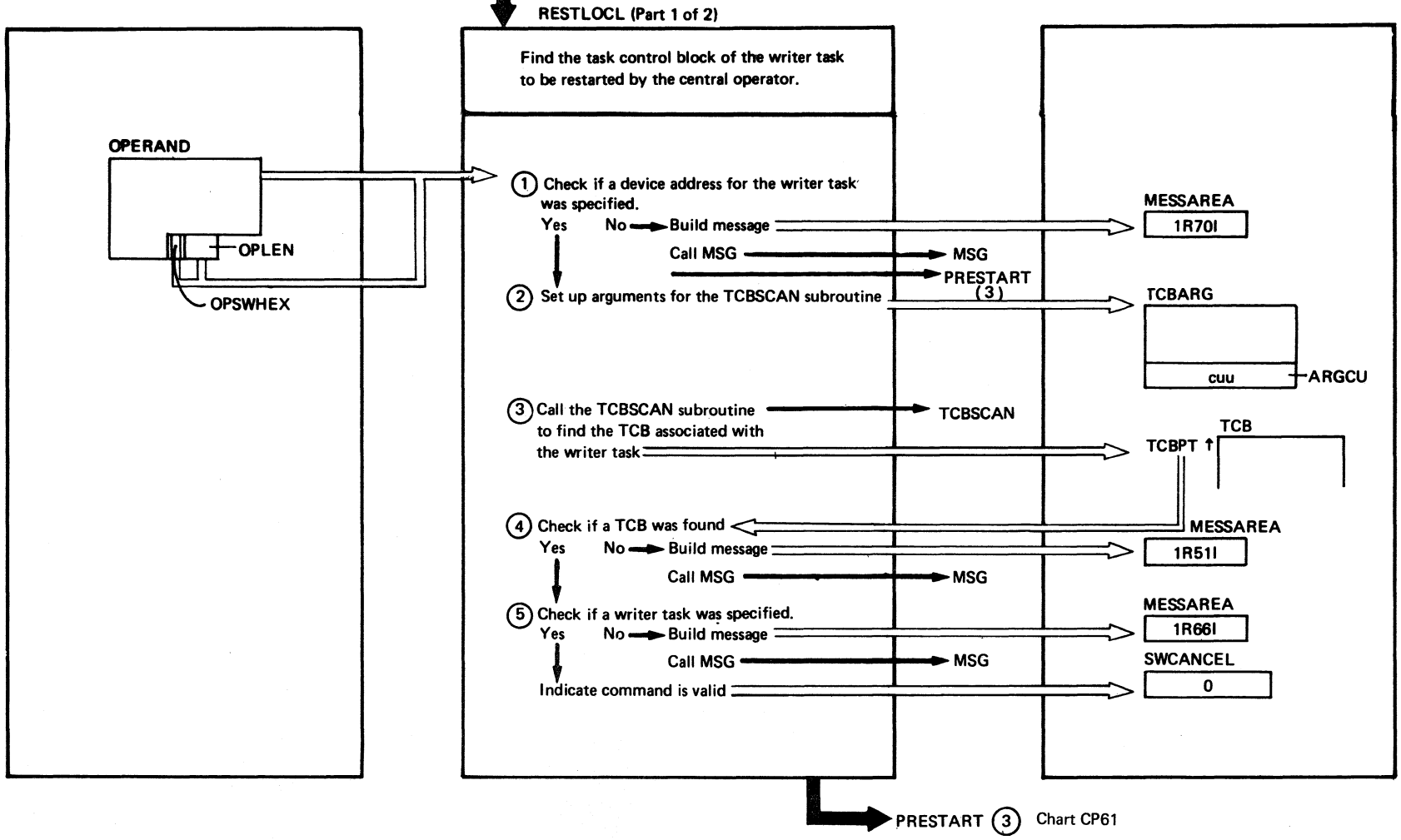


Extended Description	Include Segment	Call Subroutine	Chart
(2)	RESTLOCL RESTREMT		CP62 CP63
(3) 1R57I COMMAND IGNORED, TASK IS AT JOB BOUNDARY		MSG	CP75
(4) 1R82I 'PSETUP' OR 'PRESTART' IN PROGRESS		MSG	CP75
(8) The output field TCBRSNUM contains the number of pages or cards. The output field TCBRSTYP contains: bit 4 : no sign 8 : plus sign 12 : minus sign			
(10) 1R52I OPERAND 2 NEITHER DECIMAL NOR OMITTED		MSG	CP75
(11) 1R67I OPERAND 2 REDUCED TO 9999		MSG	CP75

Page of SY33-8571-1, Added November 24, 1977, By TNL SN33-9241

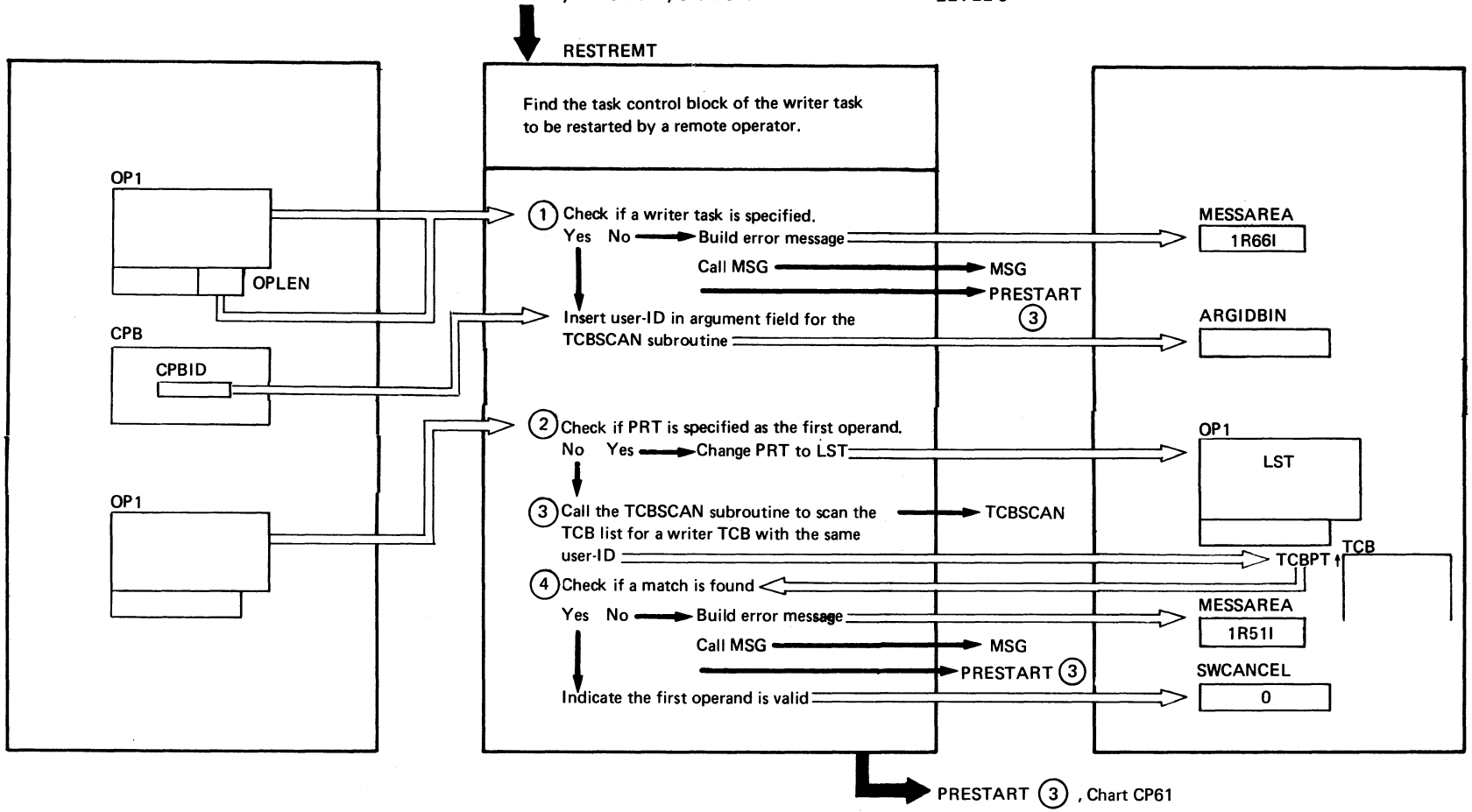
Included by PRESTART, Chart CP61

LEVEL 3

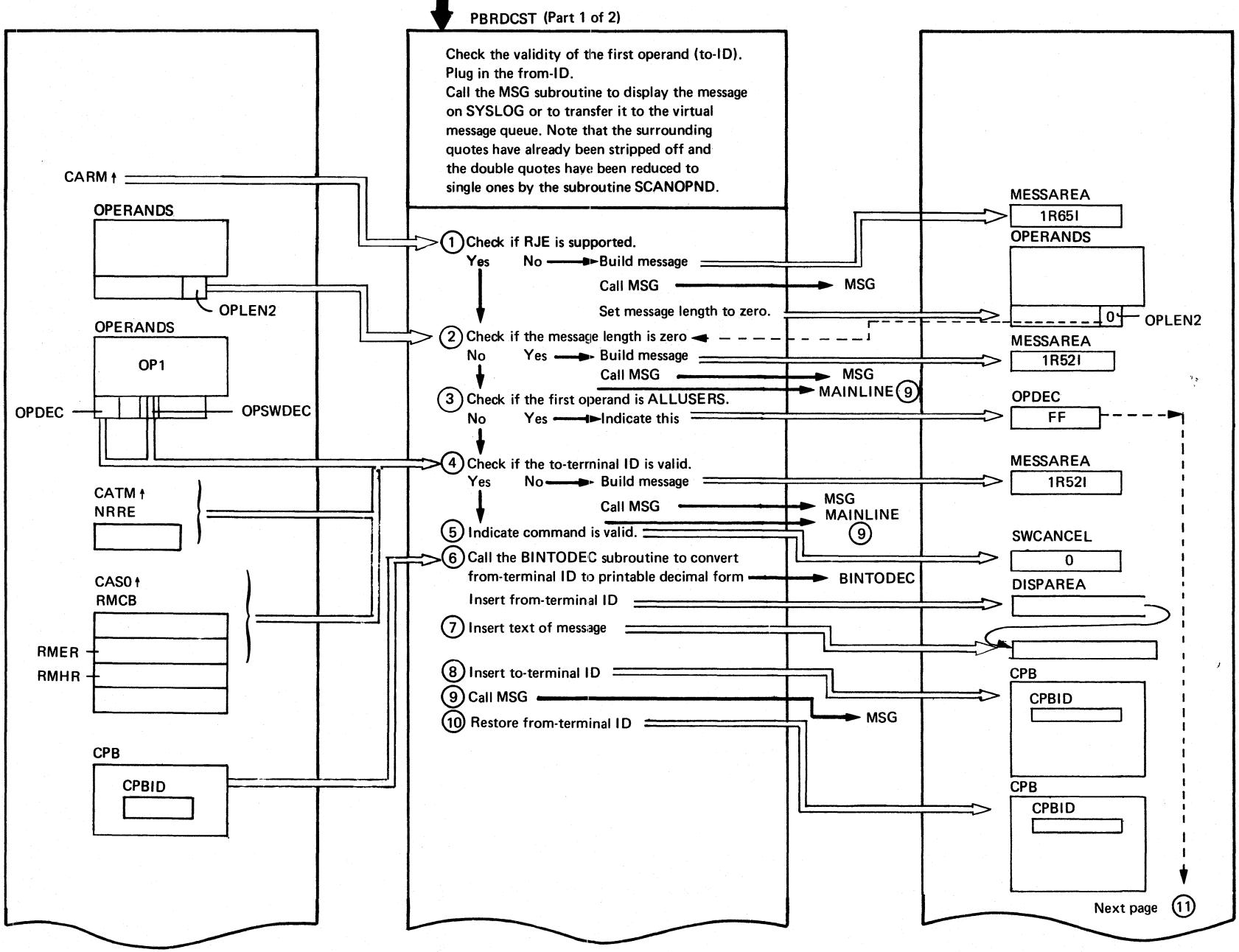


Extended Description		Call Subroutine	Chart
(1) 1R70I NO DEVICE ADDRESS SPECIFIED		MSG	CP75
(3)		TCBSCAN	CP82
(4) 1R51I OPERAND 1 DESIGNATES NON-EXISTING TASK		MSG	CP75
(5) 1R66I NO WRITER TASK SPECIFIED		MSG	CP75

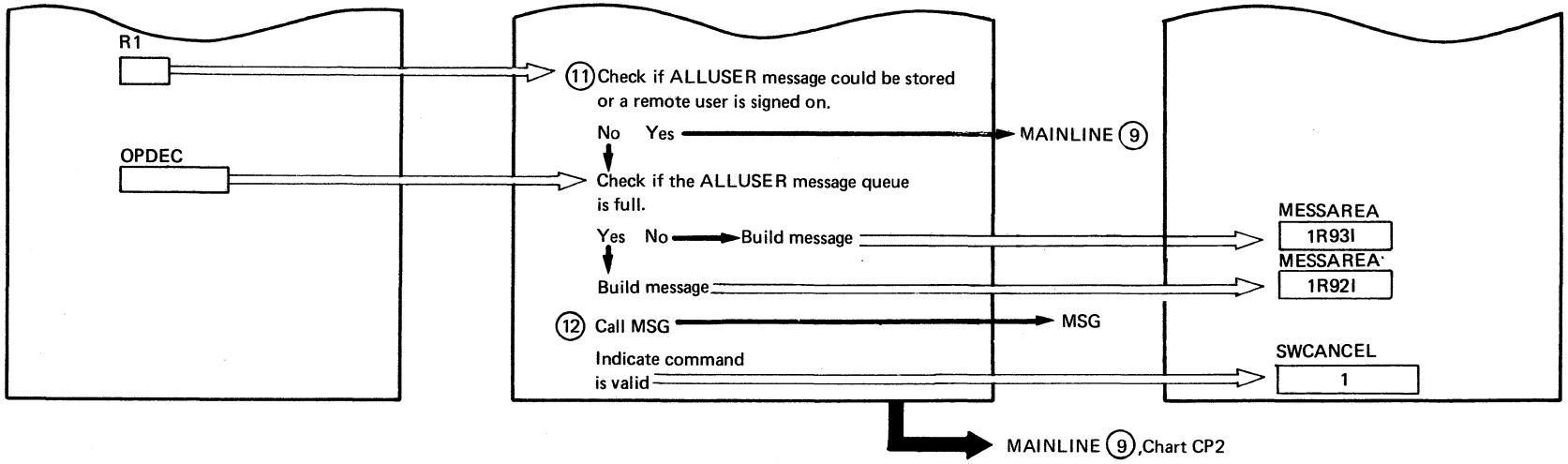
Included by PRESTART, Chart CP61 LEVEL 3 Chart CP63



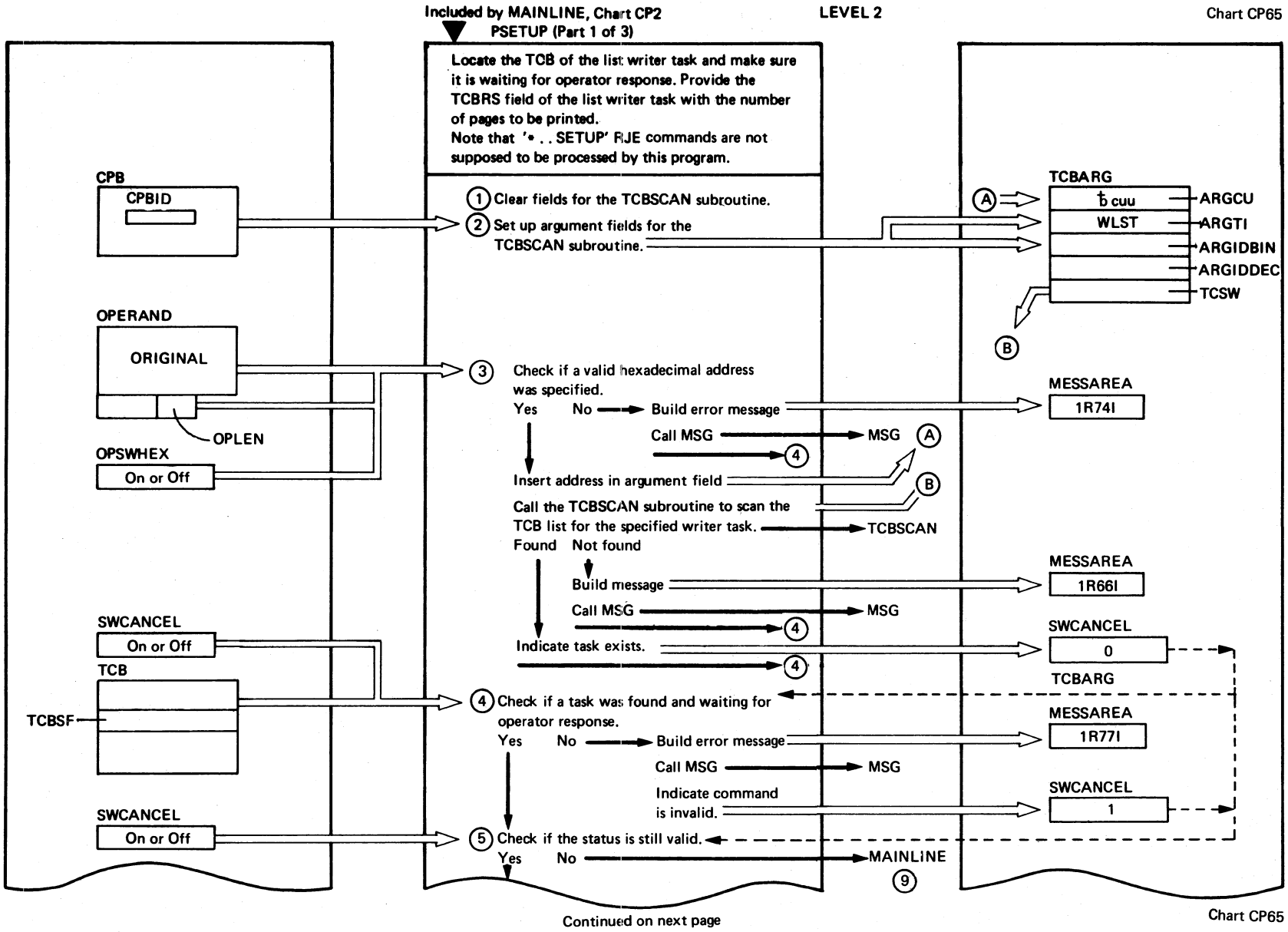
Extended Description	Include Segment	Call Subroutine	Chart
1 1R66I NO WRITER TASK SPECIFIED		MSG	CP75
3		TCBSCAN	CP82
4 1R51I OPERAND 1 DESIGNATES NON-EXISTING TASK		MSG	CP75

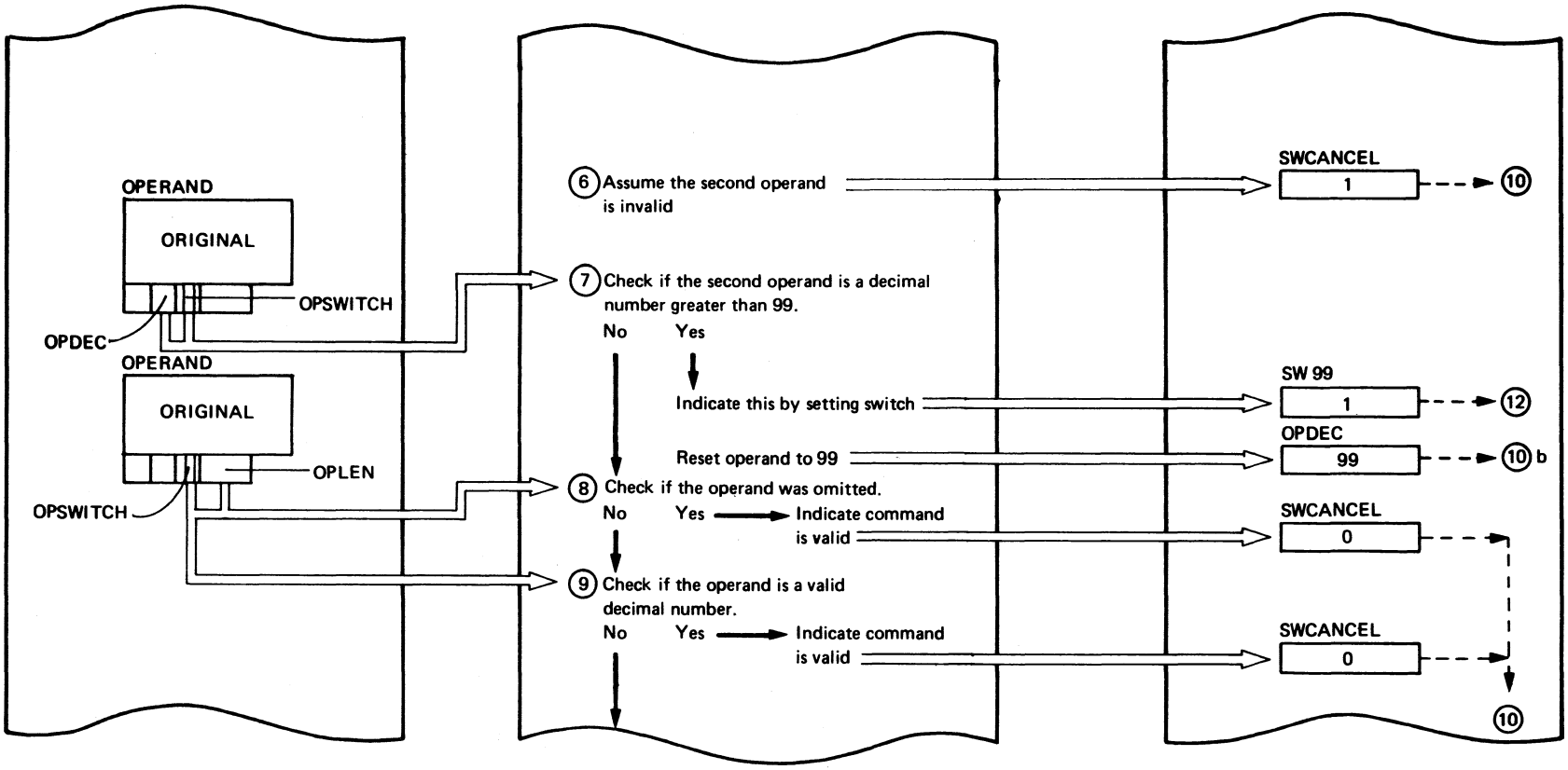


Continued on next page

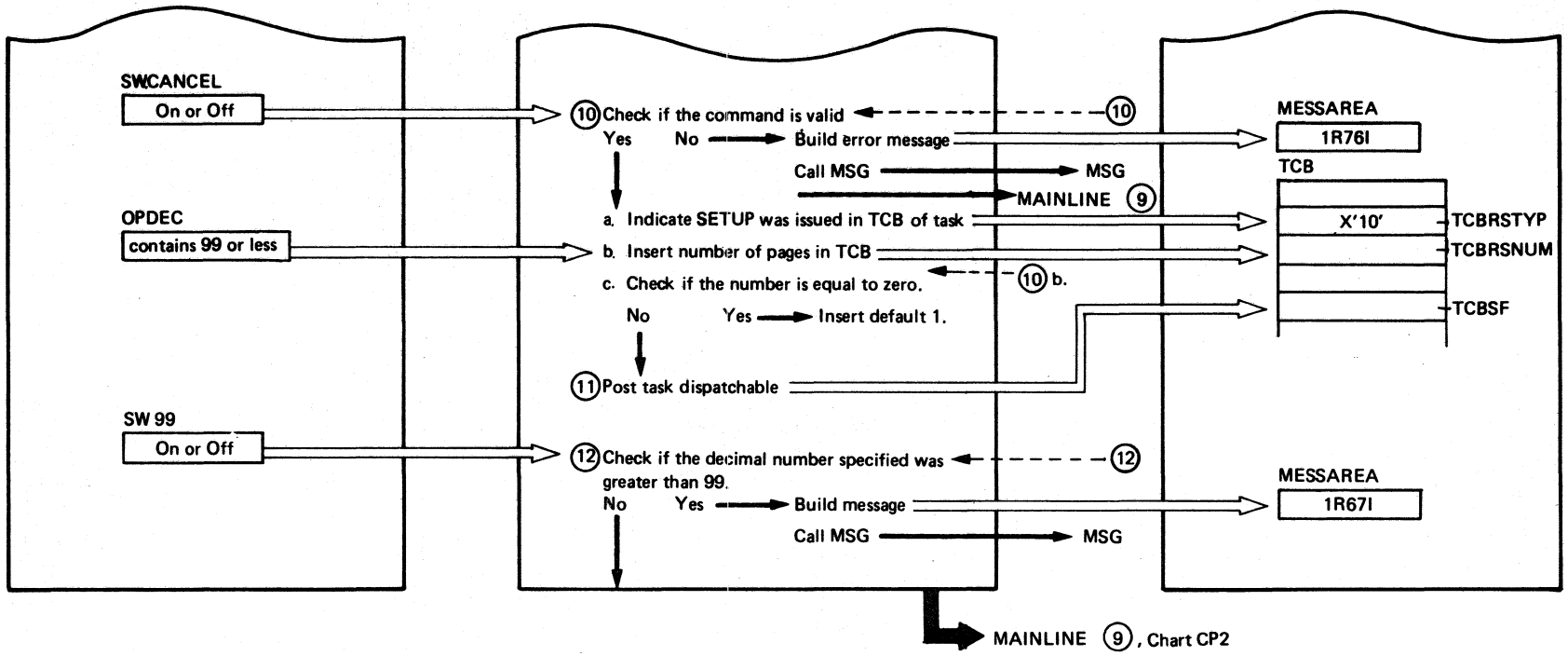


Extended Description			Include Segment	Call Subroutine	Chart
(1)	1R65I	RJE-BSC NOT SUPPORTED		MSG	CP75
(2)	1R52I	OPERAND 1 OR 2 INVALID		MSG	CP75
(4)	The to-terminal ID is valid when a decimal number within the range or when ALLUSERS is specified.				
	1R52I	OPERAND 1 INVALID DESTINATION		MSG	CP75
(6)				BINTODEC	CP83
(9)				MSG	CP75
(11)	This is determined by the IPW$WTM macro that was issued in the MSG subroutine.				
	1R93I	REMOTE xxx CURRENTLY NOT SIGNED ON (xxx: remote-ID)			
	1R92I	ALLUSER MESSAGE QUEUE IS FULL		MSG	CP75
(12)					


Included by MAINLINE, Chart CP2
PSETUP (Part 1 of 3)
LEVEL 2
Chart CP65
Locate the TCB of the list writer task and make sure it is waiting for operator response. Provide the TCBRS field of the list writer task with the number of pages to be printed.
Note that '* . . SETUP' RJE commands are not supposed to be processed by this program.
CPB
CPBID
OPERAND
ORIGINAL
OPLEN
OPSWHEX
On or Off
SWCANCEL
On or Off
TCB
TCBSF
SWCANCEL
On or Off
1 Clear fields for the TCBSCAN subroutine.
2 Set up argument fields for the TCBSCAN subroutine.
3 Check if a valid hexadecimal address was specified.
Yes
No
Build error message
Call MSG
MSG
4
Insert address in argument field
Call the TCBSCAN subroutine to scan the TCB list for the specified writer task.
TCBSCAN
Found
Not found
Build message
Call MSG
MSG
4
Indicate task exists.
4
4 Check if a task was found and waiting for operator response.
Yes
No
Build error message
Call MSG
MSG
Indicate command is invalid.
5 Check if the status is still valid.
Yes
No
MAINLINE
9
TCBARG
A
B
bcuu
WLST
ARGCU
ARGTI
ARGIDBIN
ARGIDDEC
TCSW
MESSAREA
1R74I
MESSAREA
1R66I
SWCANCEL
0
TCBARG
MESSAREA
1R77I
SWCANCEL
1
Continued on next page
Chart CP65



Continued on next page



Extended Description		Include Segment	Call Subroutine	Chart
(3)	1R74I NO PRINTER ADDRESS SPECIFIED		MSG	CP75
			TCBSCAN	CP82
	1R661 xxx LIST WRITER TASK DOES NOT EXIST		MSG	CP75
	cuu		MSG	CP75
(4)	1R77I TASK NOT WAITING FOR OPERATOR		MSG	CP75
(10)	1R76I NUMBER OF PAGES NOT DECIMAL		MSG	CP75
(12)	1R67I NUMBER OF PAGES REDUCED TO 99		MSG	CP75

Included by MAINLINE, Chart CP2 LEVEL 2 Chart CP66

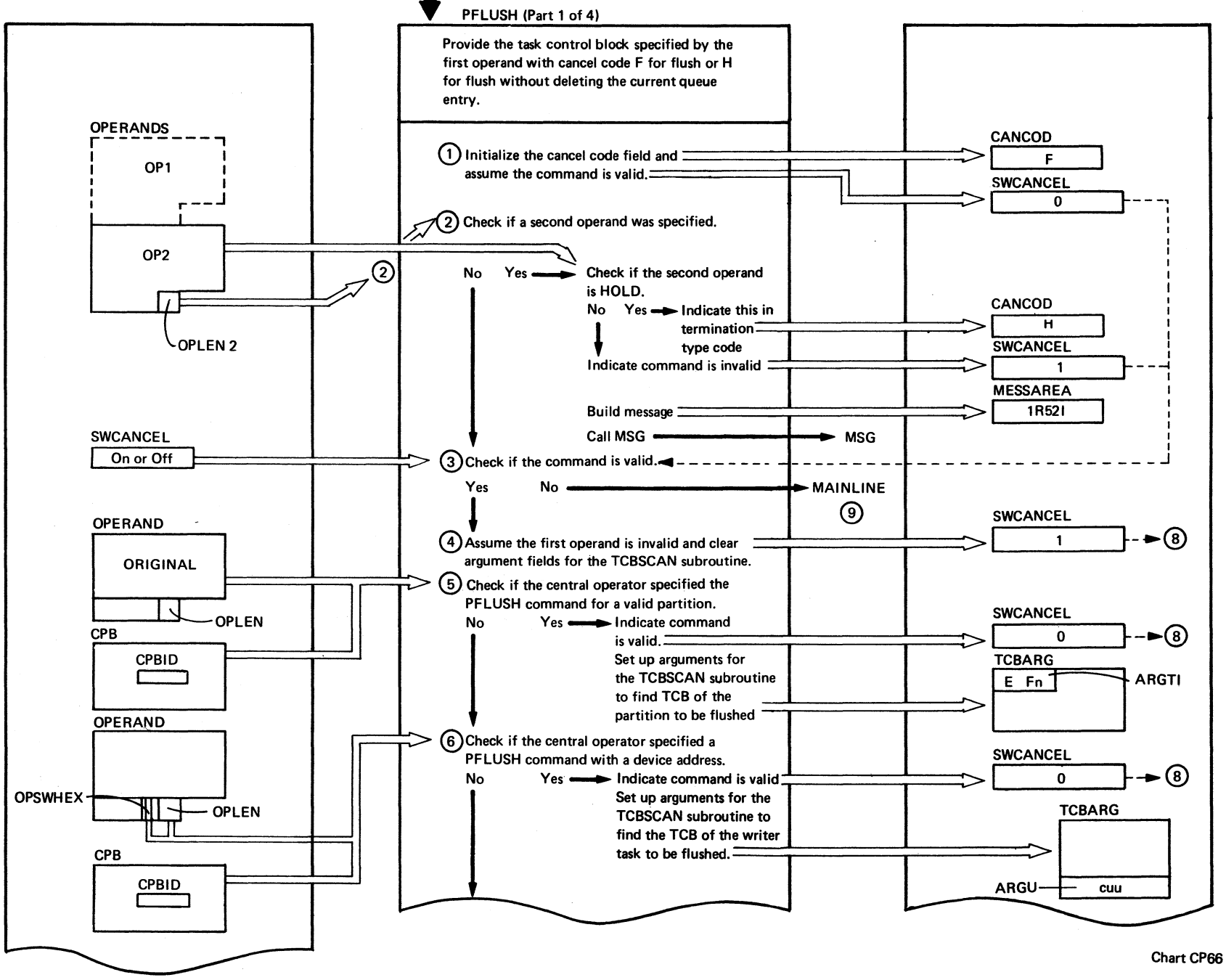
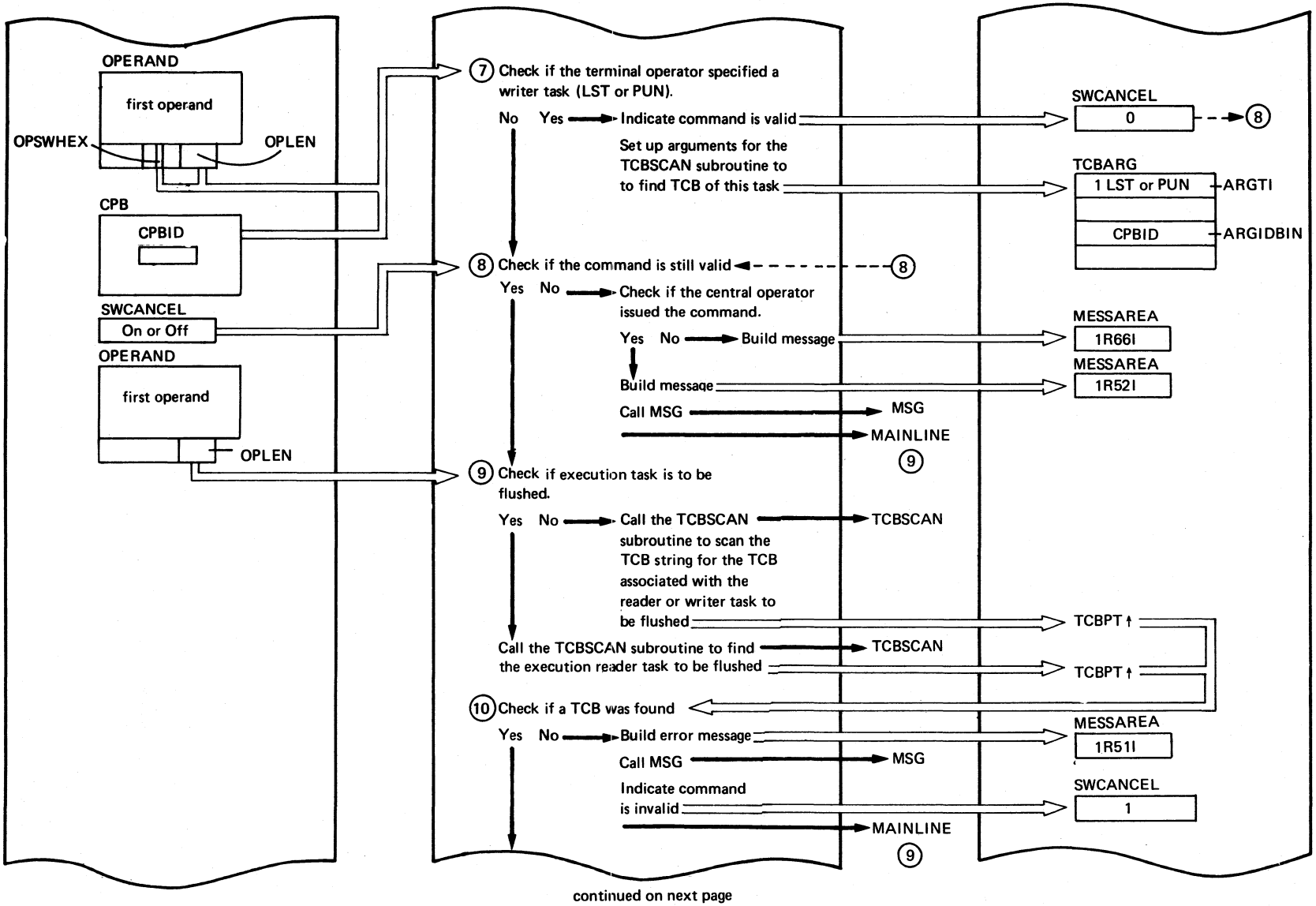
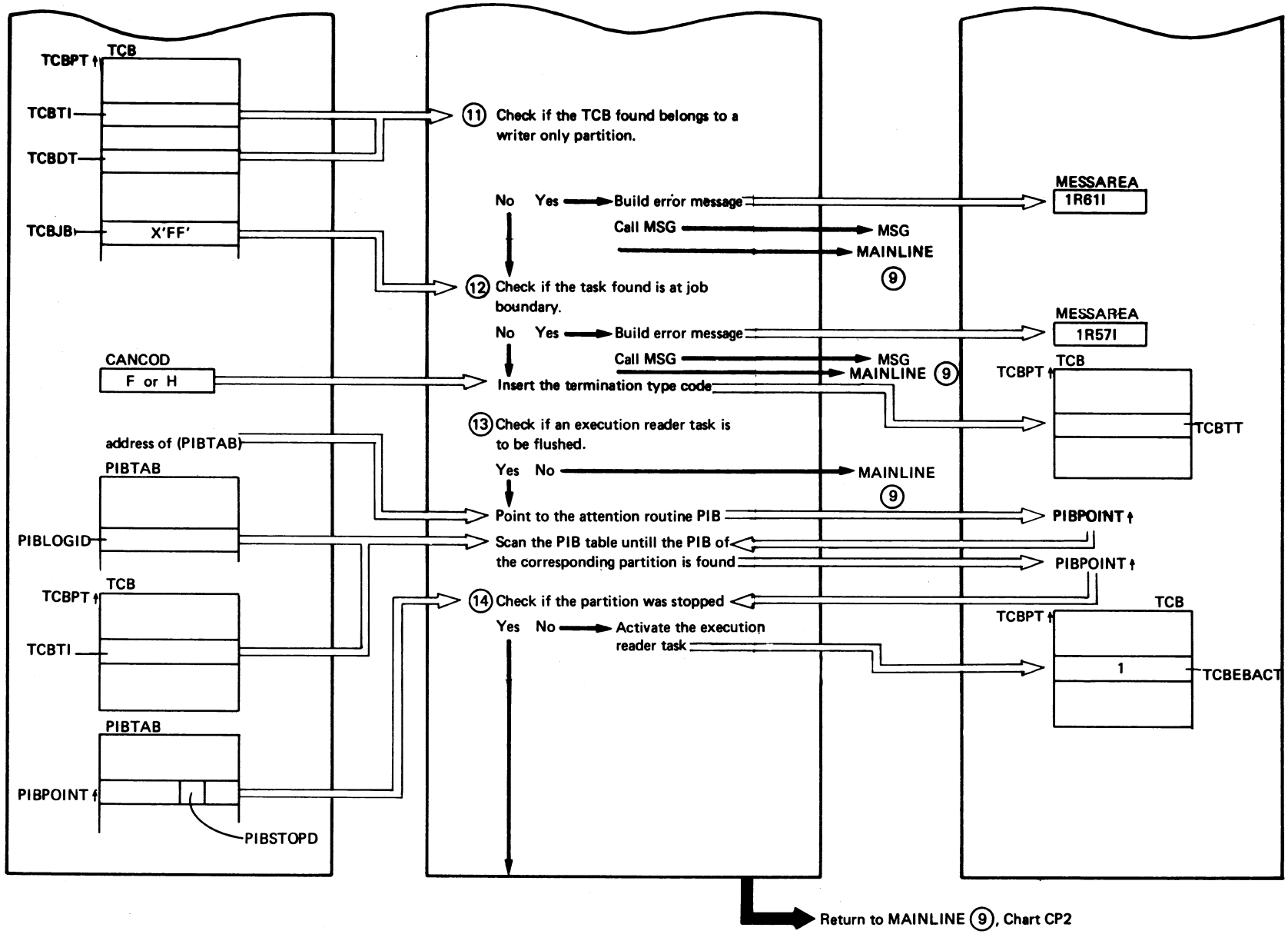


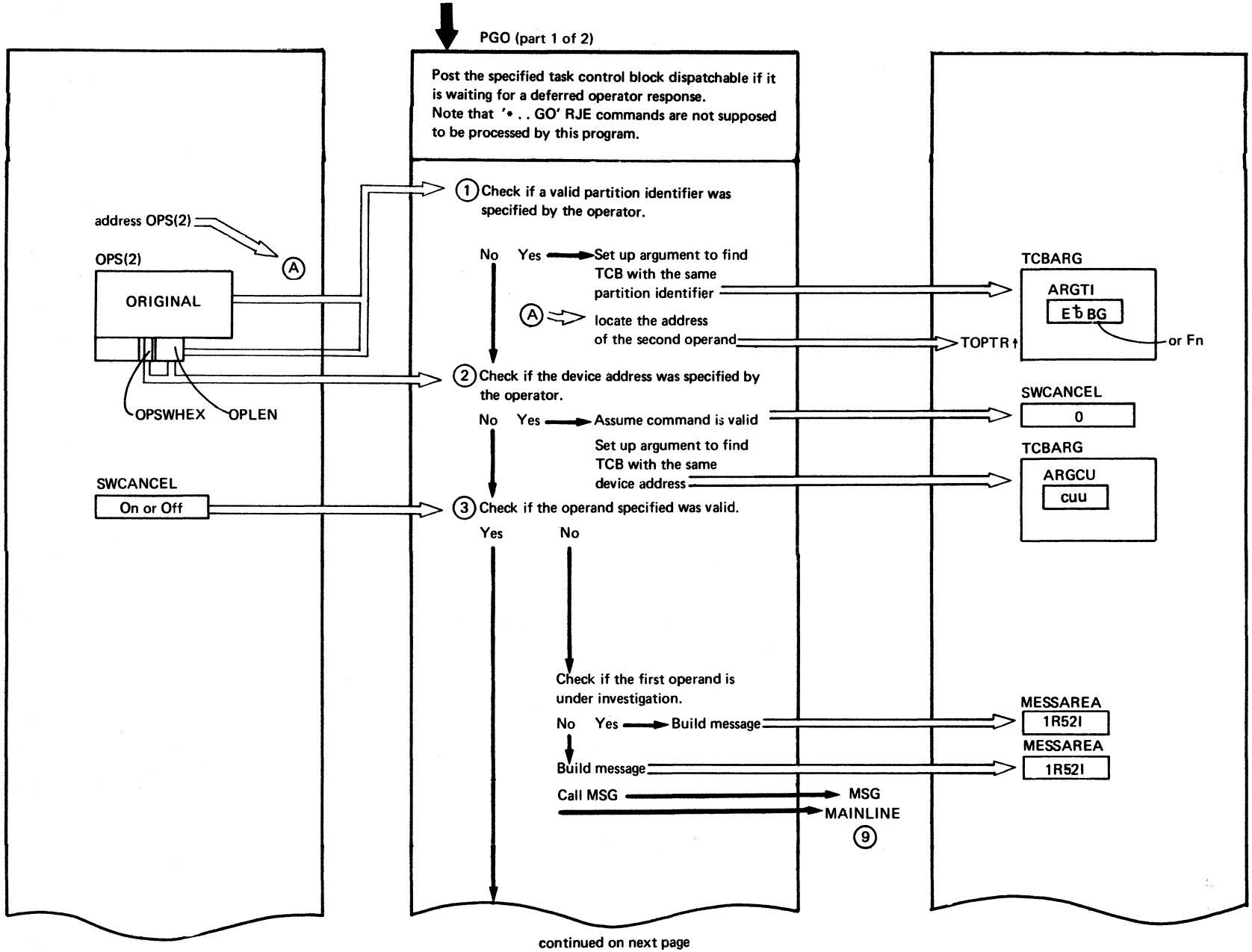
Chart CP66

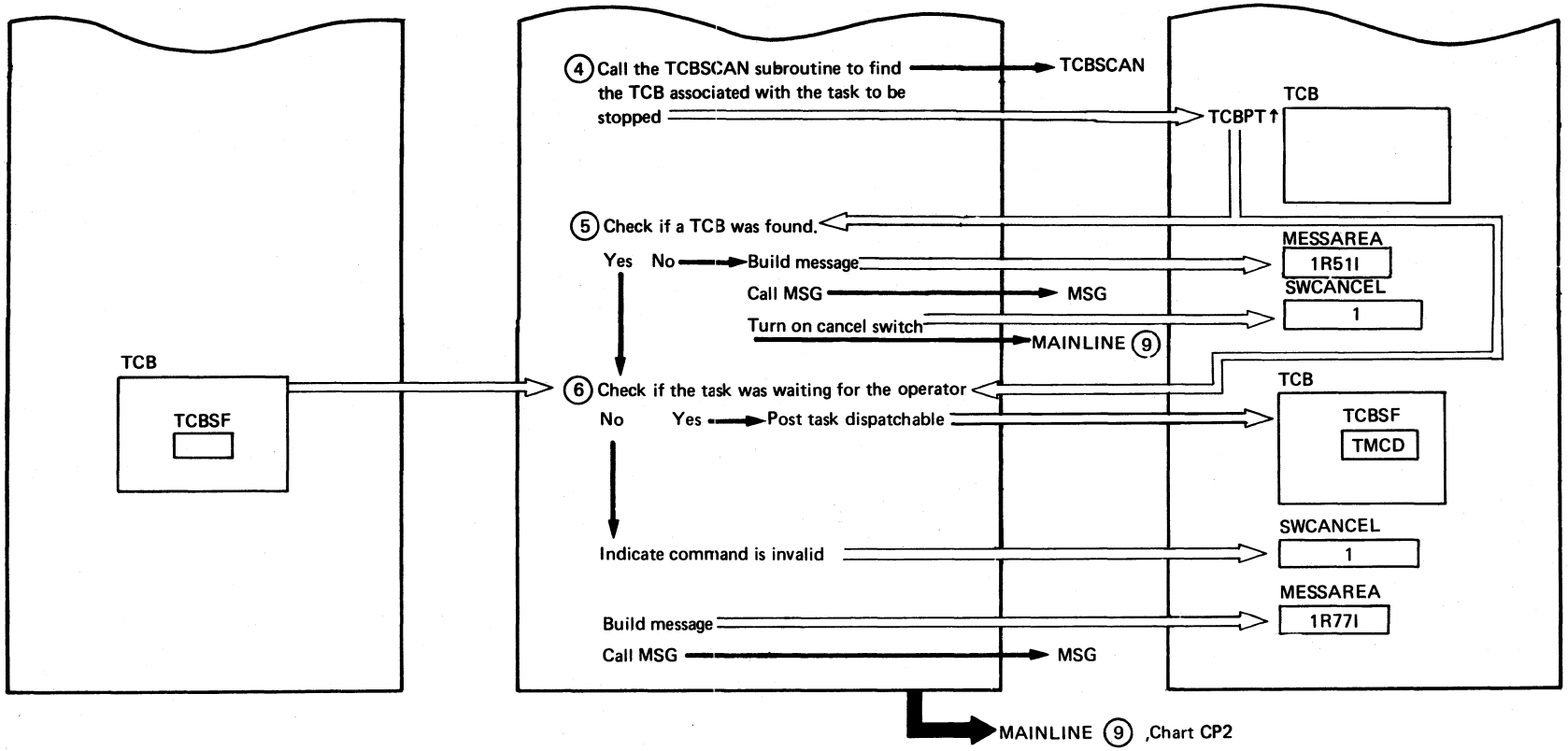


continued on next page


TCB
TCBPT
TCBTI
TCBDT
TCBJB
X'FF'
(11) Check if the TCB found belongs to a writer only partition.
No Yes Build error message
MESSAREA
1R61I
Call MSG
MSG
MAINLINE
(9)
(12) Check if the task found is at job boundary.
No Yes Build error message
MESSAREA
1R57I
CANCOD
F or H
Call MSG
MSG
MAINLINE (9)
Insert the termination type code
TCB
TCBPT
TCBTT
(13) Check if an execution reader task is to be flushed.
address of (PIBTAB)
PIBTAB
Yes No
MAINLINE
(9)
Point to the attention routine PIB
PIBPOINT
PIBLOGID
Scan the PIB table untill the PIB of the corresponding partition is found
PIBPOINT
TCB
TCBPT
TCBTI
(14) Check if the partition was stopped
Yes No
Activate the execution reader task
TCB
TCBPT
1
TCBEBACT
PIBTAB
PIBPOINT
PIBSTOPD
Return to MAINLINE (9), Chart CP2

Extended Description	Include Segment	Call Subroutine	Chart
(2) 1R52I NEITHER 'HOLD' NOR OMITTED		MSG	CP75
(8) 1R66I NO WRITER TASK SPECIFIED		MSG	CP75
1R52I OPERAND 1 MISSING OR INVALID			
(9)		TCBSCAN	CP82
(10) 1R51I OPERAND 1 DESIGNATES NON-EXISTING TASK		MSG	CP75
(11) The device to be spooled is a 1052 or CRT console.		MSG	CP75
1R61I INVALID FOR WRITER-ONLY PARTITION		MSG	CP75
(12) 1R57I COMMAND INGNORED, TASK IS AT JOB BOUNDARY			


PGO (part 1 of 2)
Post the specified task control block dispatchable if it is waiting for a deferred operator response. Note that '* . . GO' RJE commands are not supposed to be processed by this program.
address OPS(2)
OPS(2)
ORIGINAL
OPSWHEX
OPLEN
SWCANCEL
On or Off
1 Check if a valid partition identifier was specified by the operator.
No
Yes
Set up argument to find TCB with the same partition identifier
A
locate the address of the second operand
TOPTR
TCBARG
ARGTI
EbBG
or Fn
2 Check if the device address was specified by the operator.
No
Yes
Assume command is valid
SWCANCEL
0
Set up argument to find TCB with the same device address
TCBARG
ARGCU
cuu
3 Check if the operand specified was valid.
Yes
No
Check if the first operand is under investigation.
No
Yes
Build message
MESSAREA
1R52I
Build message
MESSAREA
1R52I
Call MSG
MSG
MAINLINE
9
continued on next page

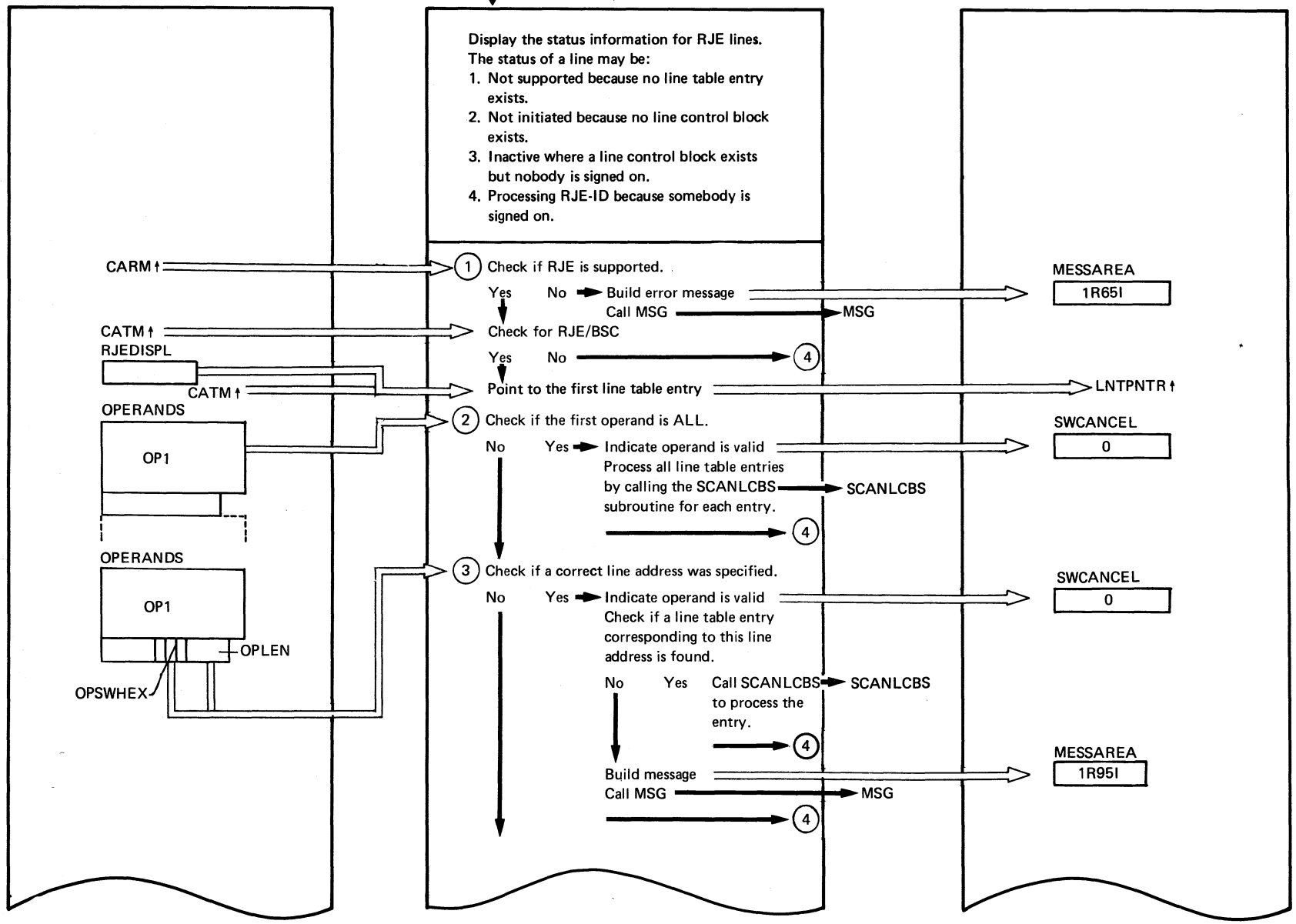


Extended Description	Include Segment	Call Subroutine	Chart
3 1R52I OPERAND 1 MISSING OR INVALID 1R52I OPERAND 2 NO DEVICE ADDRESS		MSG	CP75
4		TCBSCAN	CP82
5 1R51I NON-EXISTING TASK DESIGNATED		MSG	CP75
6 1R77I TASK NOT WAITING FOR OPERATOR		MSG	CP75

Included by MAINLINE, Chart CP2

LEVEL 2

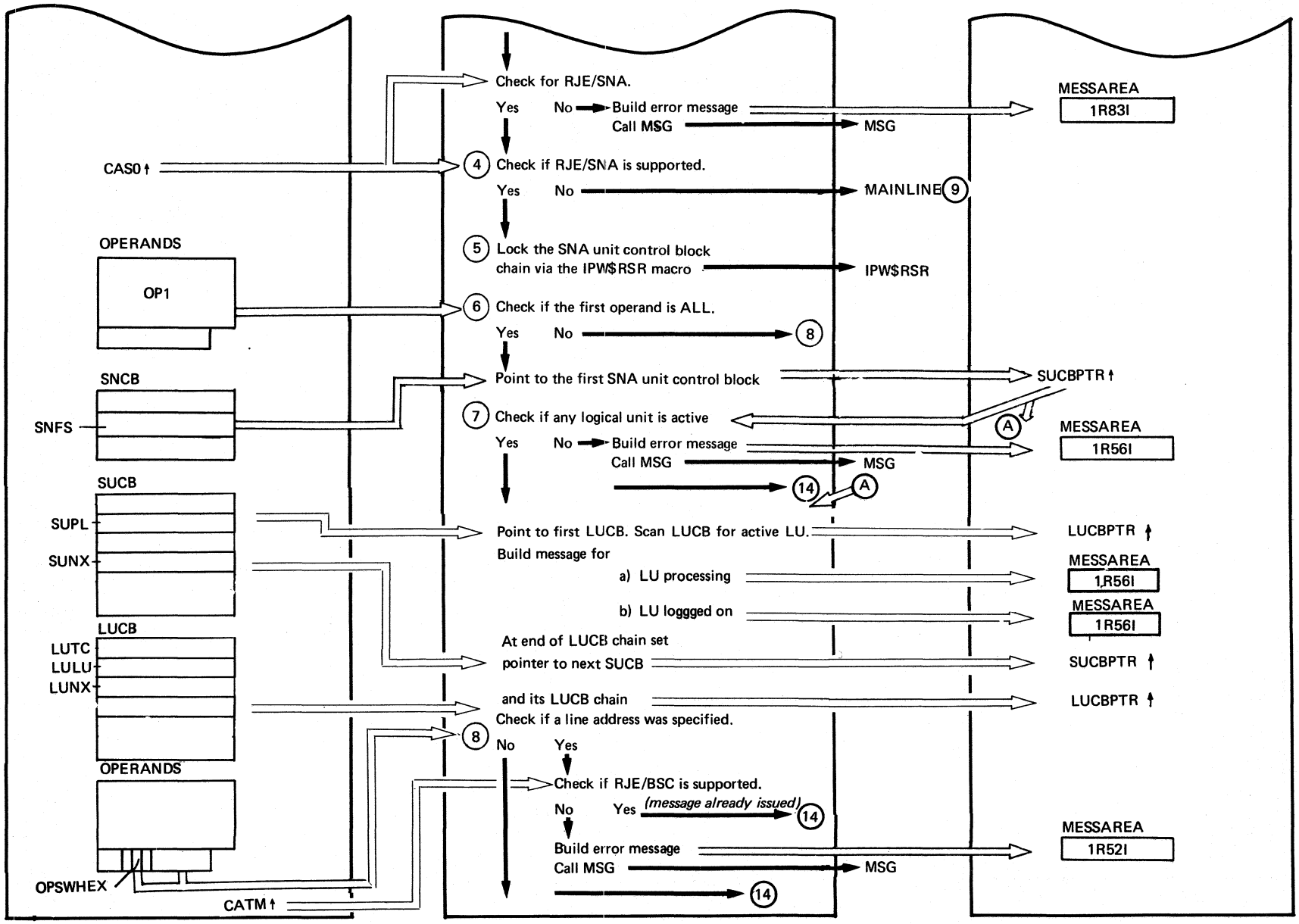
PINQUIRE (Part 1 of 4)



(Continued on next page)

LEVEL 2

PINQUIRE (Part 2 of 4)



PINQUIRE (Part 3 of 4) LEVEL 2

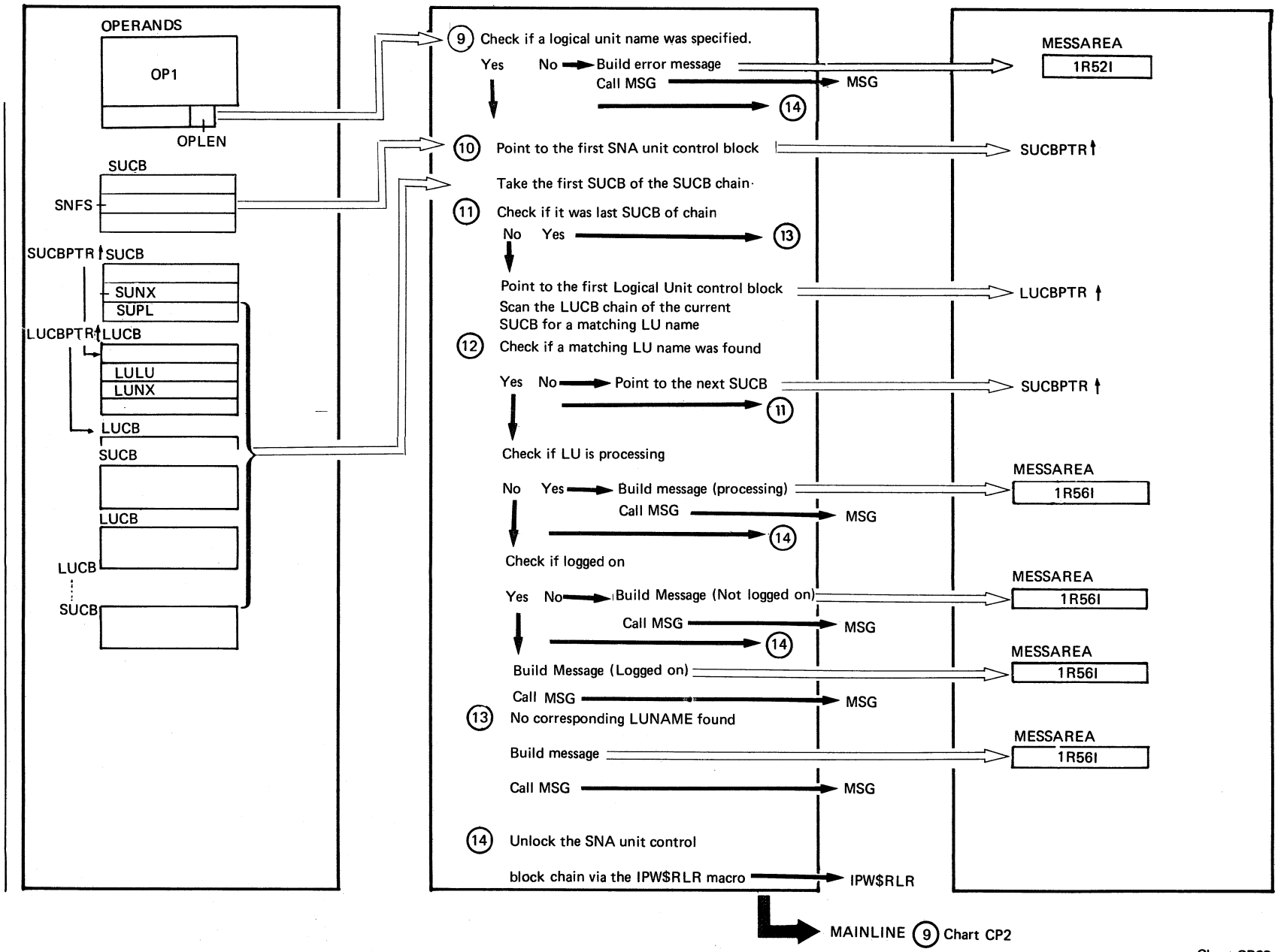


Chart CP68

Extended Description	Include Segment	Call Subroutine	Chart
(1) 1R65I RJE-BSC NOT SUPPORTED		MSG	CP75
(2)	SCANLCBS		CP69
(3)	SCANLCBS		CP69
1R95I LINE xxx NOT SUPPORTED ↑ RJE line address		MSG	CP75
1R83I OPERAND NEITHER 'ALL' NOR LINE ADDRESS		MSG	CP75
(5) The IPW$RSR macro uses registers 0, 1, 2, and 3 which are restricted in this segment.			
(7) 1R56I NO LOGICAL UNIT LOGGED ON		MSG	CP75
1R56I xxxxxxxx PROCESSING xxx ↑ logical unit name ↑ remote ID		MSG	CP75
1R56I xxxxxxxx LOGGED ON ↑ logical unit name		MSG	CP75
(8) 1R52I INVALID OPERAND		MSG	CP75
(9) 1R52I INVALID OPERAND		MSG	CP75
(12) 1R56I xxxxxxxx PROCESSING xxx ↑ logical unit name ↑ remote ID		MSG	CP75
1R56I xxxxxxxx NOT LOGGED ON ↑ logical unit name			

Called by PINQUIRE, Chart CP68 LEVEL 2 Chart CP69

SCANLCBS (Subroutine included in PINQUIRE)

Provide status information for the line table entry being processed.

LCBPRT ↑
LCB
LCBSIGN

1. Convert line address from binary to printable hexadecimal → BTHOUT
2. Scan LCBs to find requested LCB containing the device address.
3. Check if an LCB was found.
   Yes No → Build message line not initiated → MESSAREA 1R56I
4. Check if the terminal is active.
   Yes No → Build message indicating terminal is inactive → MESSAREA 1R56I
   Build message indicating who logged on → MESSAREA 1R56I
5. Call MSG → MSG

Return to calling segment
PINQUIRE, Chart CP68

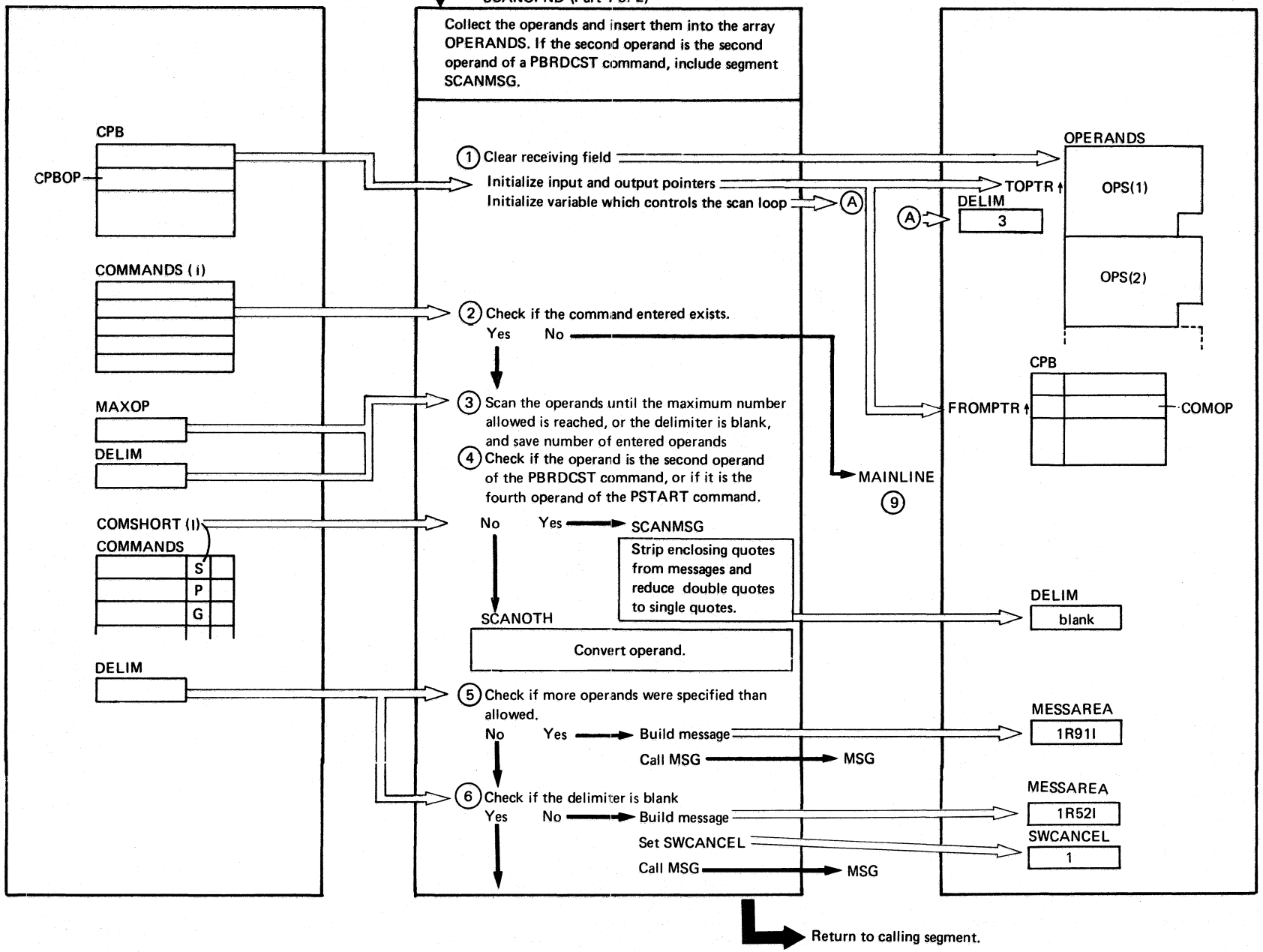
Extended Description	Include Segment	Call Subroutine	Chart
(3) 1R56I xxx NOT INITIATED RJE line address		MSG	CP75
(4) 1R56I xxx INACTIVE RJE line address			
1R56I xxx PROCESSING yyy RJE line address remote-ID			
(5)			

Chart CP69

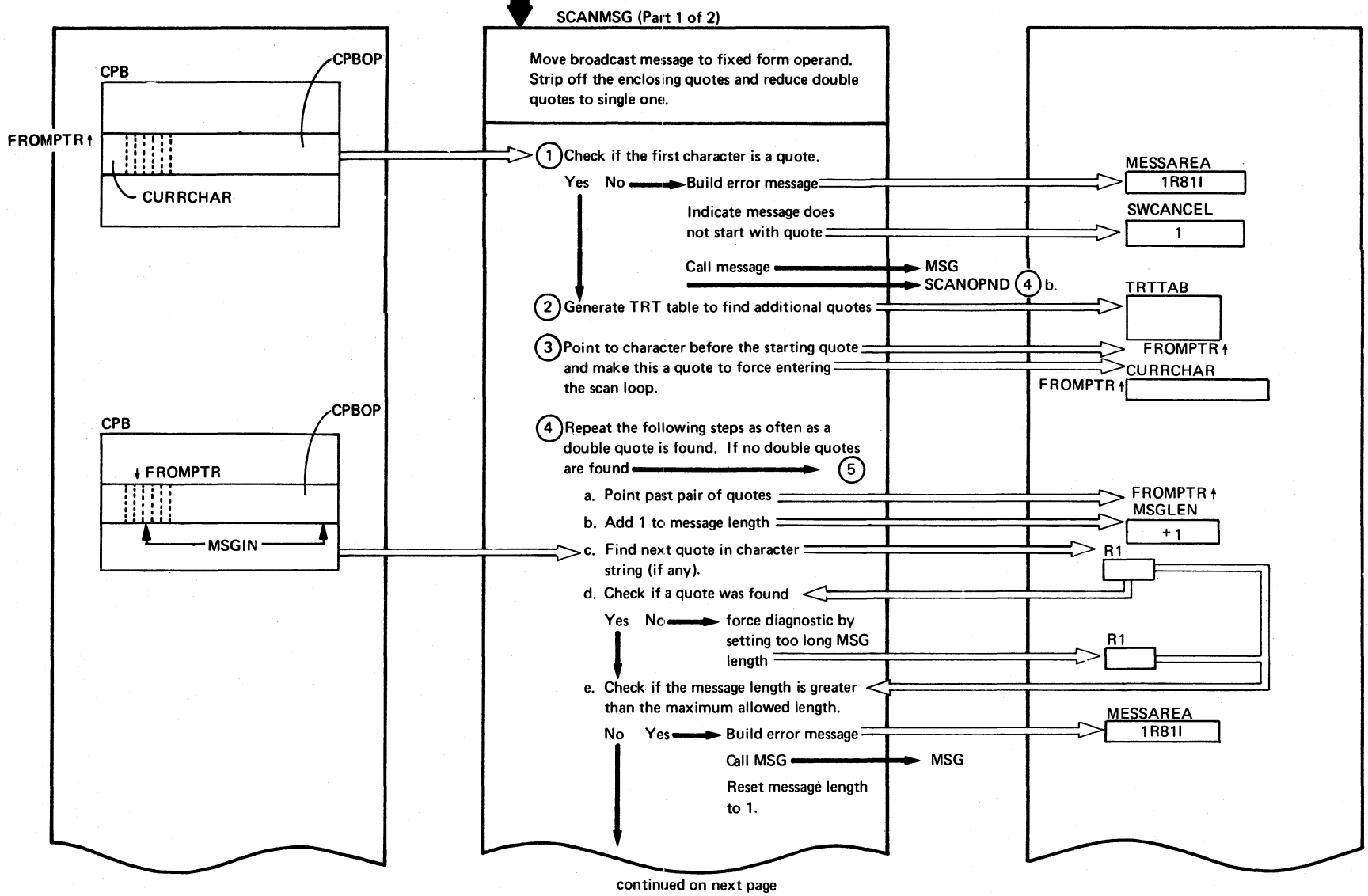
From calling segment

LEVEL 2

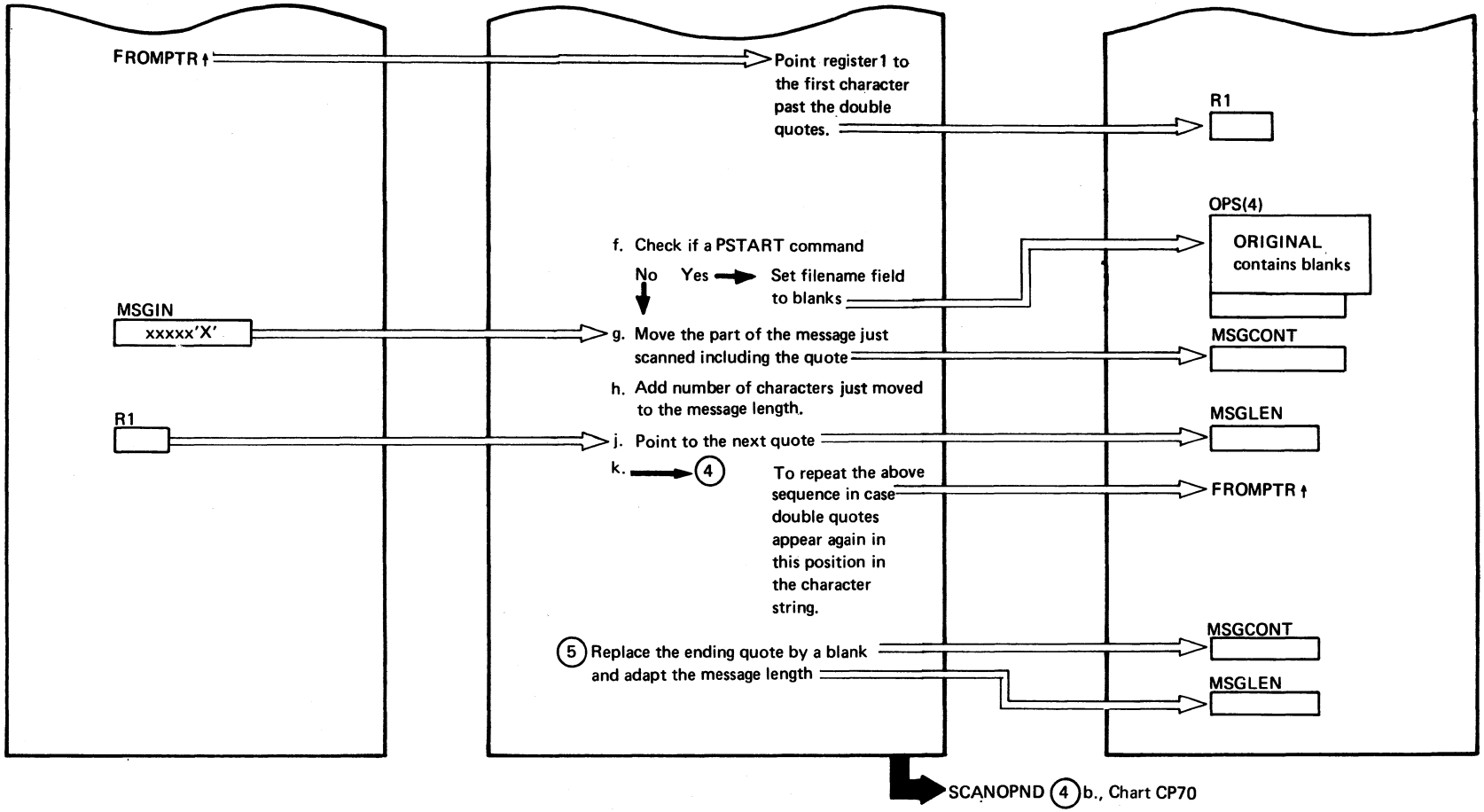
Chart CP70



Extended Description		Include Subroutine	Call Subroutine	Chart
(1)	Register 14 is saved as return register. Input field CBPOP in CPB is built by either the IPW$ICP macro or the attention interface. DELIM is a field to control the scan loop. When it contains a comma the scan continues. When it is blank the scan stops.			
(2)	The index number of the command entered is compared with the dimension of the array COMMANDS.			
(3)	The CPOP field is scanned as long as a comma delimits each operand and the maximum number of operands is not reached. Registers 1 and 2 are used in translate and test instructions. The TOPTR is updated to point to the location where to store the next operand.			
(4)		SCANMSG SCANOTH		CP71 CP72
(5)	1R91I TOO MANY OPERANDS, FIRST n PROCESSED		MSG	CP75
(6)	1R52I DELIMITER NOT BLANK OR COMMA		MSG	CP75



continued on next page



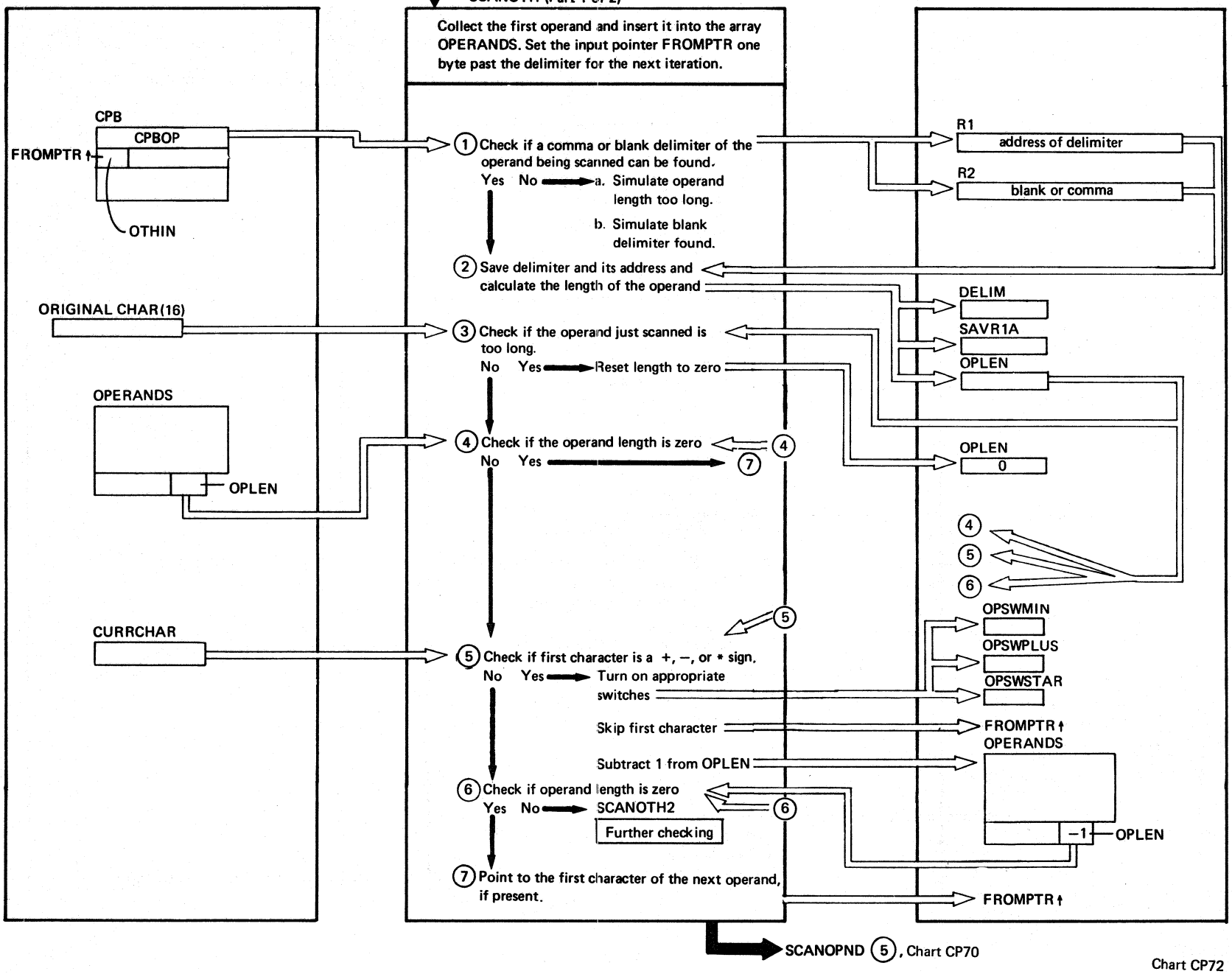
Extended Description	Include Segment	Call Subroutine	Chart
(1) 1R81I MESSAGE DOES NOT START WITH QUOTE		MSG	CP75
(4)e. 1R81I MESSAGE TOO LONG OR NO CLOSING QUOTE		MSG	CP75

Included by SCANOPND, Chart CP70

LEVEL 3

SCANOTH (Part 1 of 2)

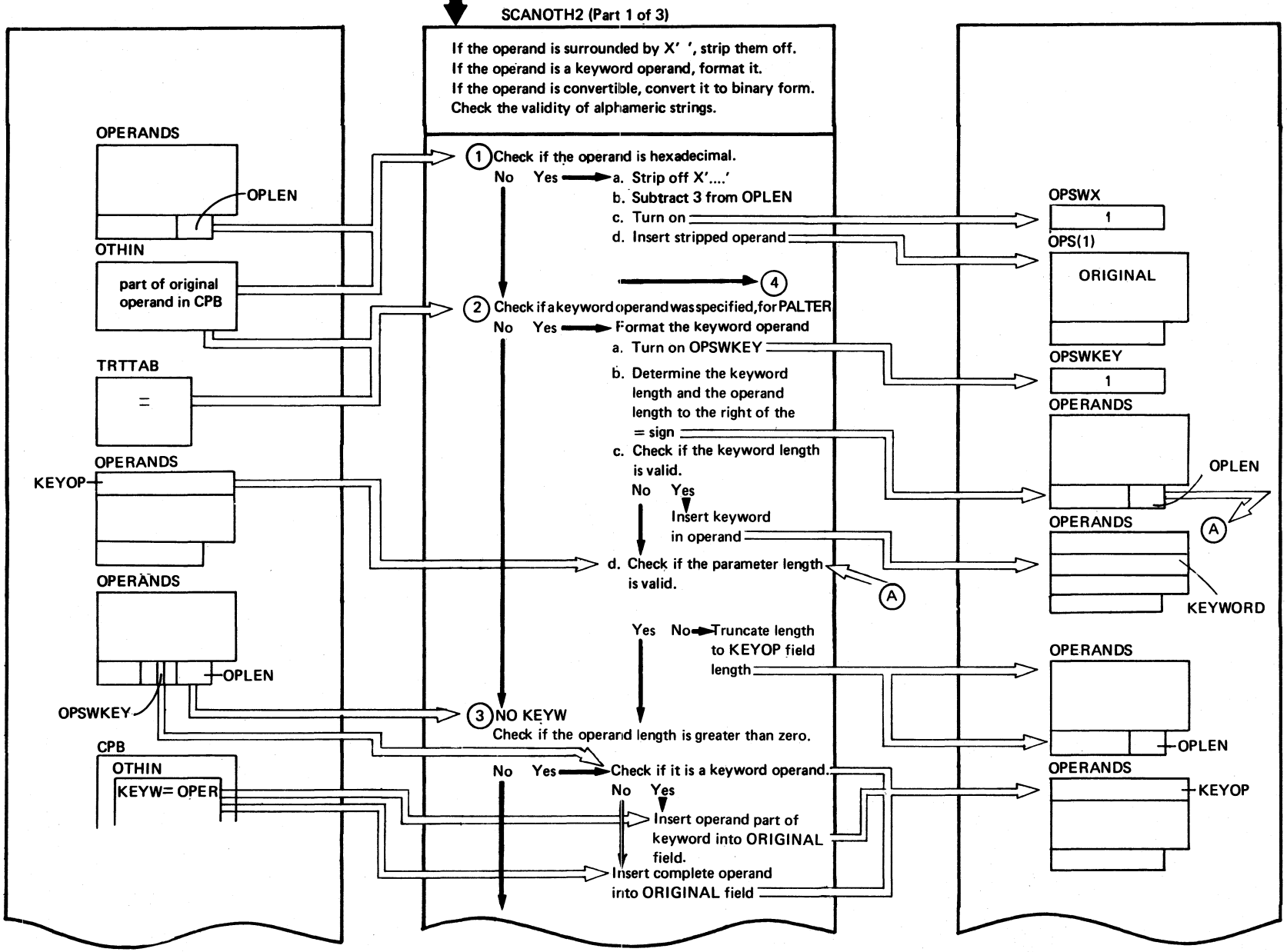
Collect the first operand and insert it into the array OPERANDS. Set the input pointer FROMPTR one byte past the delimiter for the next iteration.



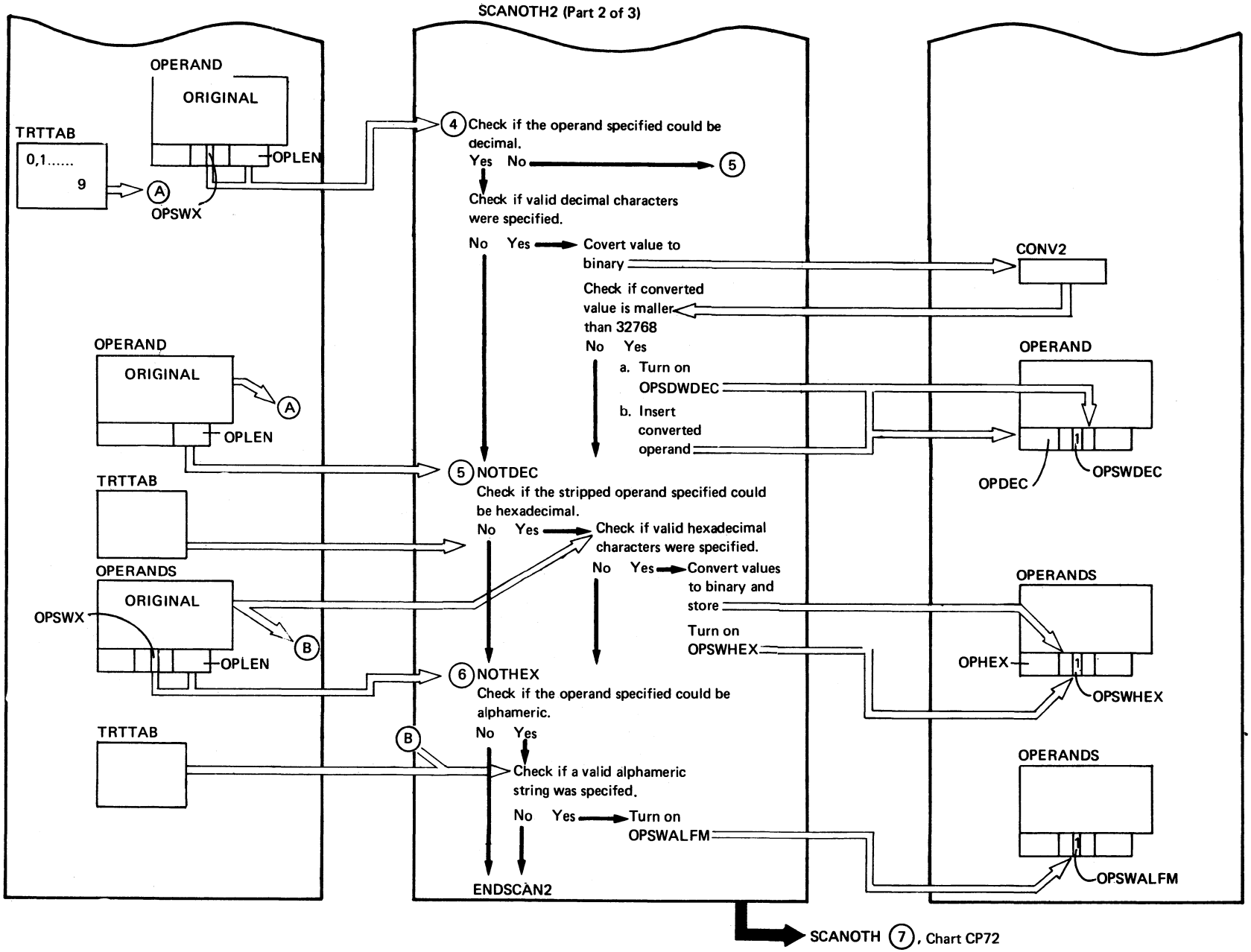
SCANOPND (5), Chart CP70

Extended Description	Include Segment	Call Subroutine	Chart
(1) OTHIN in input is part of the operands field in which the operand being scanned resides. OTHIN is always addressed by the FROMPTR.			
(6)	SCANOTH2		CP73

Included by SCANOTH, Chart CP72 LEVEL 4 Chart CP73



Continued on next page

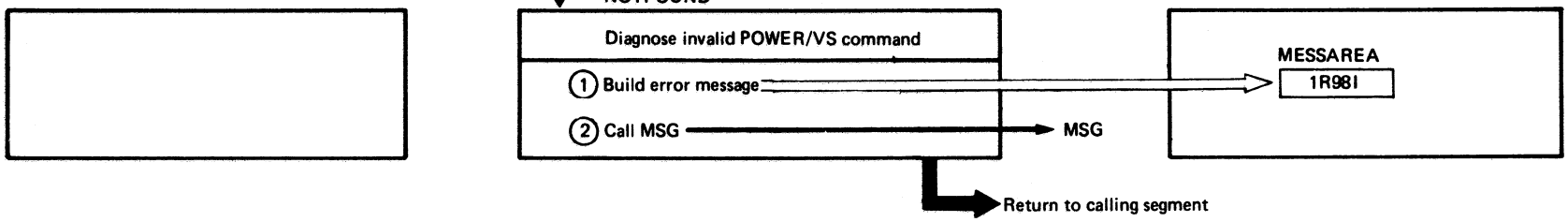

SCANOTH2 (Part 2 of 3)
OPERAND
ORIGINAL
TRTTAB
0,1.....
9
OPLEN
A
OPSWX
4 Check if the operand specified could be decimal.
Yes No
5
Check if valid decimal characters were specified.
No Yes
Covert value to binary
CONV2
Check if converted value is maller than 32768
No Yes
a. Turn on OPSDWDEC
b. Insert converted operand
OPERAND
OPDEC
1
OPSWDEC
OPERAND
ORIGINAL
A
OPLEN
5 NOTDEC
Check if the stripped operand specified could be hexadecimal.
TRTTAB
No Yes
Check if valid hexadecimal characters were specified.
No Yes
Convert values to binary and store
OPERANDS
ORIGINAL
OPSWX
B
OPLEN
Turn on OPSWHEX
OPERANDS
OPHEX
1
OPSWHEX
6 NOTHEX
Check if the operand specified could be alphameric.
TRTTAB
B
No Yes
Check if a valid alphameric string was specifed.
No Yes
Turn on OPSWALFM
OPERANDS
1
OPSWALFM
ENDSCAN2
SCANOTH 7, Chart CP72

Extended Description	Include Segment	Call Subroutine	Chart
(1) If the operand is surrounded by X'....' and therefore hexadecimal, the operand length (OPLEN) is 5 or 6 bytes. c. OPSWX is turned on to indicate that no conversion to decimal is necessary. (2) A translate and test (TRT) table for locating an equal sign (=) is generated. Using a TRT instruction, the operand is then scanned for an equal sign. If an equal sign is found, the operand is split into the keyword part and the operand part and inserted into a fixed formatted array. If the operand part specified is longer than 8 bytes, only the first 8 bytes are stored. (4) As under (2), for detecting a hexadecimal, decimal, or alphameric string, an appropriate TRT table is generated and a TRT instruction is executed. Registers 1 and 2 are therefore restricted.			

From calling segment

LEVEL 2

Chart CP74

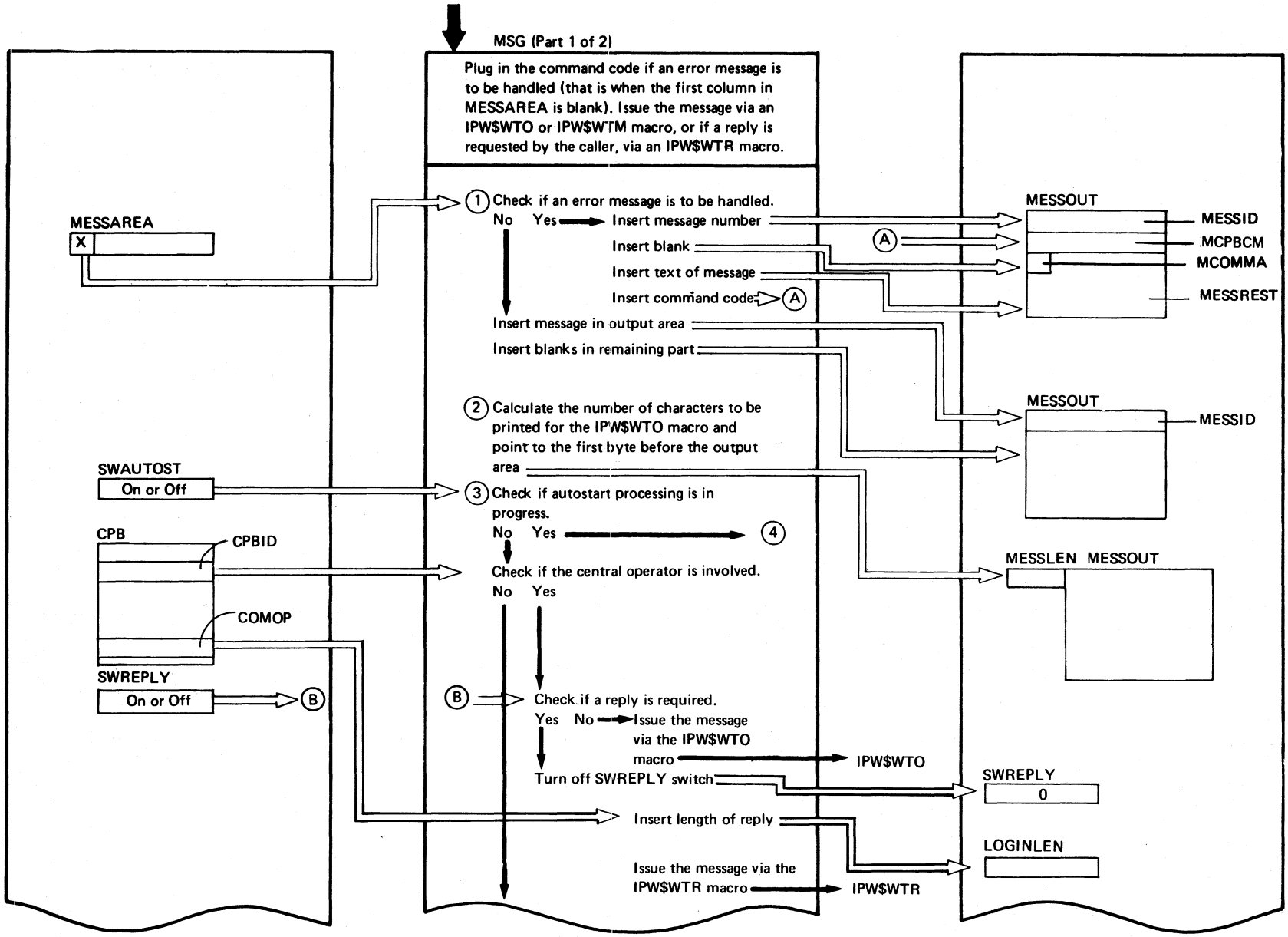


Extended Description	Include Segment	Call Subroutine	Chart
1 1R981 INVALID POWER/VS COMMAND CODE			
2		MSG	CP75

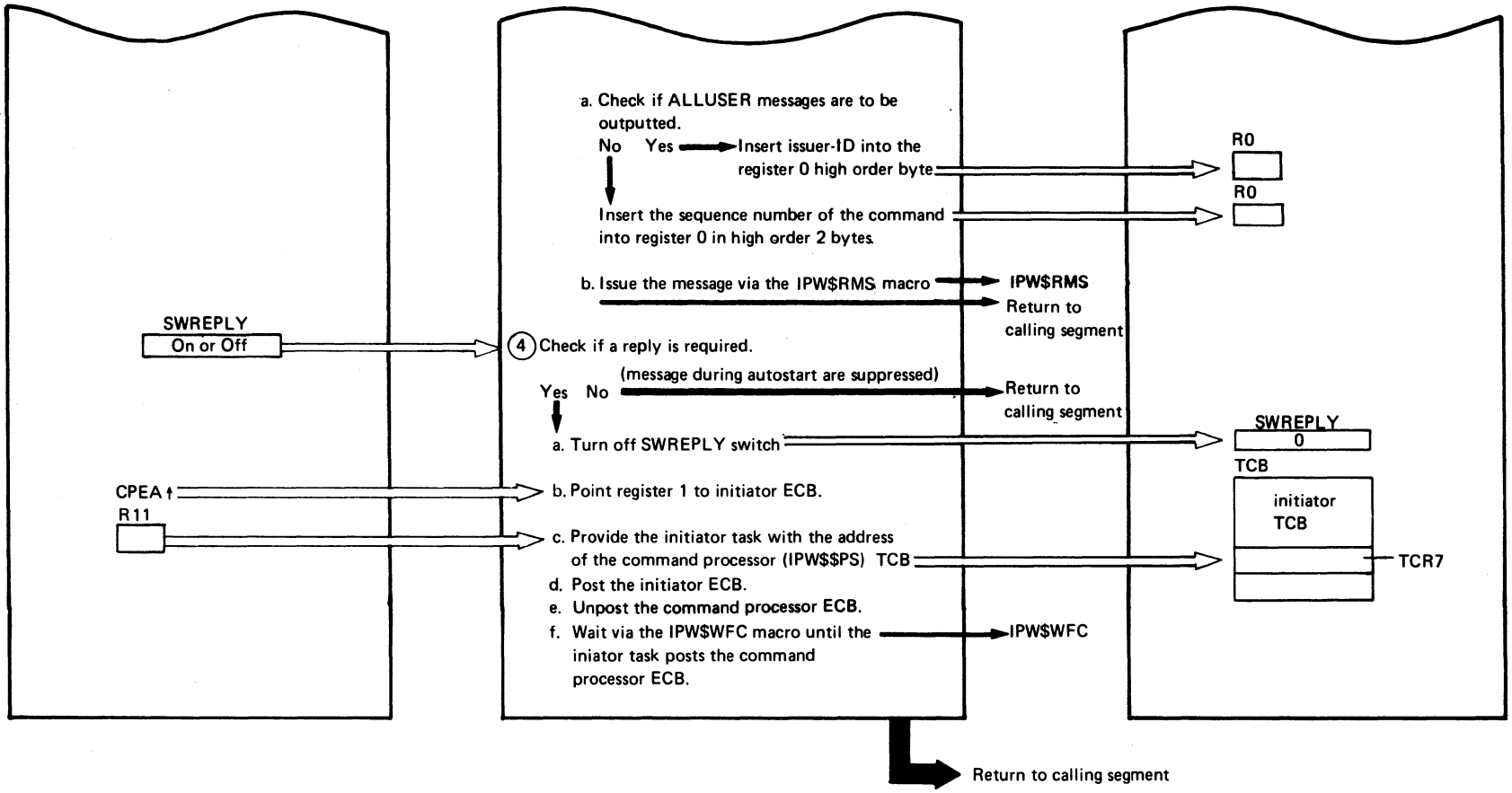
From calling segment

LEVEL 2

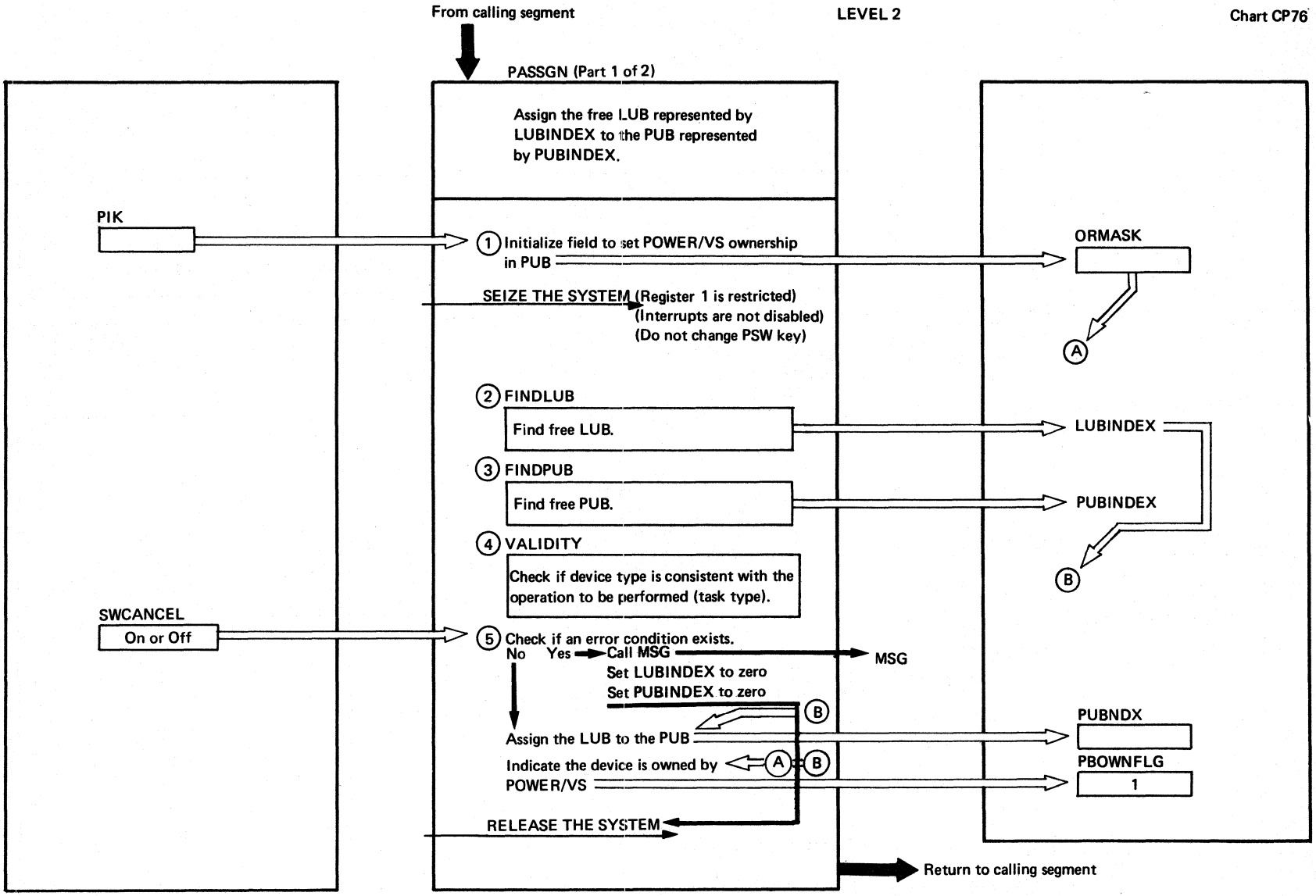
Chart CP75



Continued on next page

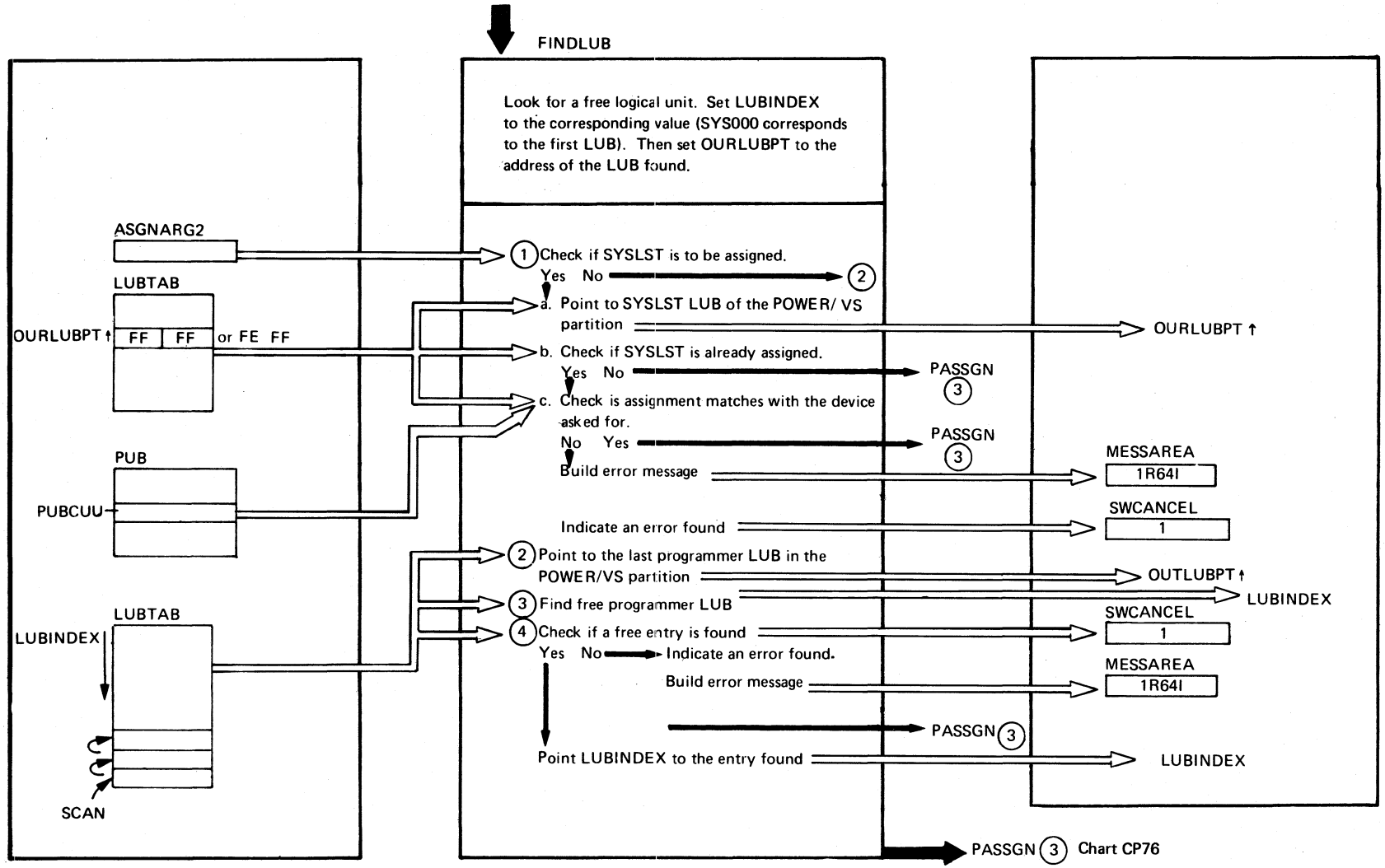


Extended Description	Include Segment	Call Subroutine	Chart
The IPWWTO, IPWWTR, and IPW$RMS macros use registers 0, 1, 2, and 3.			


From calling segment
LEVEL 2
Chart CP76
PASSGN (Part 1 of 2)
Assign the free LUB represented by LUBINDEX to the PUB represented by PUBINDEX.
PIK
1 Initialize field to set POWER/VS ownership in PUB
ORMASK
SEIZE THE SYSTEM (Register 1 is restricted)
(Interrupts are not disabled)
(Do not change PSW key)
A
2 FINDLUB
Find free LUB.
LUBINDEX
3 FINDPUB
Find free PUB.
PUBINDEX
4 VALIDITY
Check if device type is consistent with the operation to be performed (task type).
B
SWCANCEL
On or Off
5 Check if an error condition exists.
No
Yes
Call MSG
MSG
Set LUBINDEX to zero
Set PUBINDEX to zero
B
PUBNDX
Assign the LUB to the PUB
Indicate the device is owned by POWER/VS
A
B
PBOWNFLG
1
RELEASE THE SYSTEM
Return to calling segment

Extended Description	Include Segment	Call Subroutine	Chart
(1) PIK = '10'X ORMASK = '01'X '20'X '02'X '30'X '04'X '40'X '08'X '50'X '10'X '60'X '20'X '70'X '40'X			
(2)	FINDLUB		CP77
(3)	FINDPUB		CP78
(4)	VALIDITY		CP79
(5)		MSG	CP75

Included by PA$SGN, Chart CP76 LEVEL 3 Chart CP77

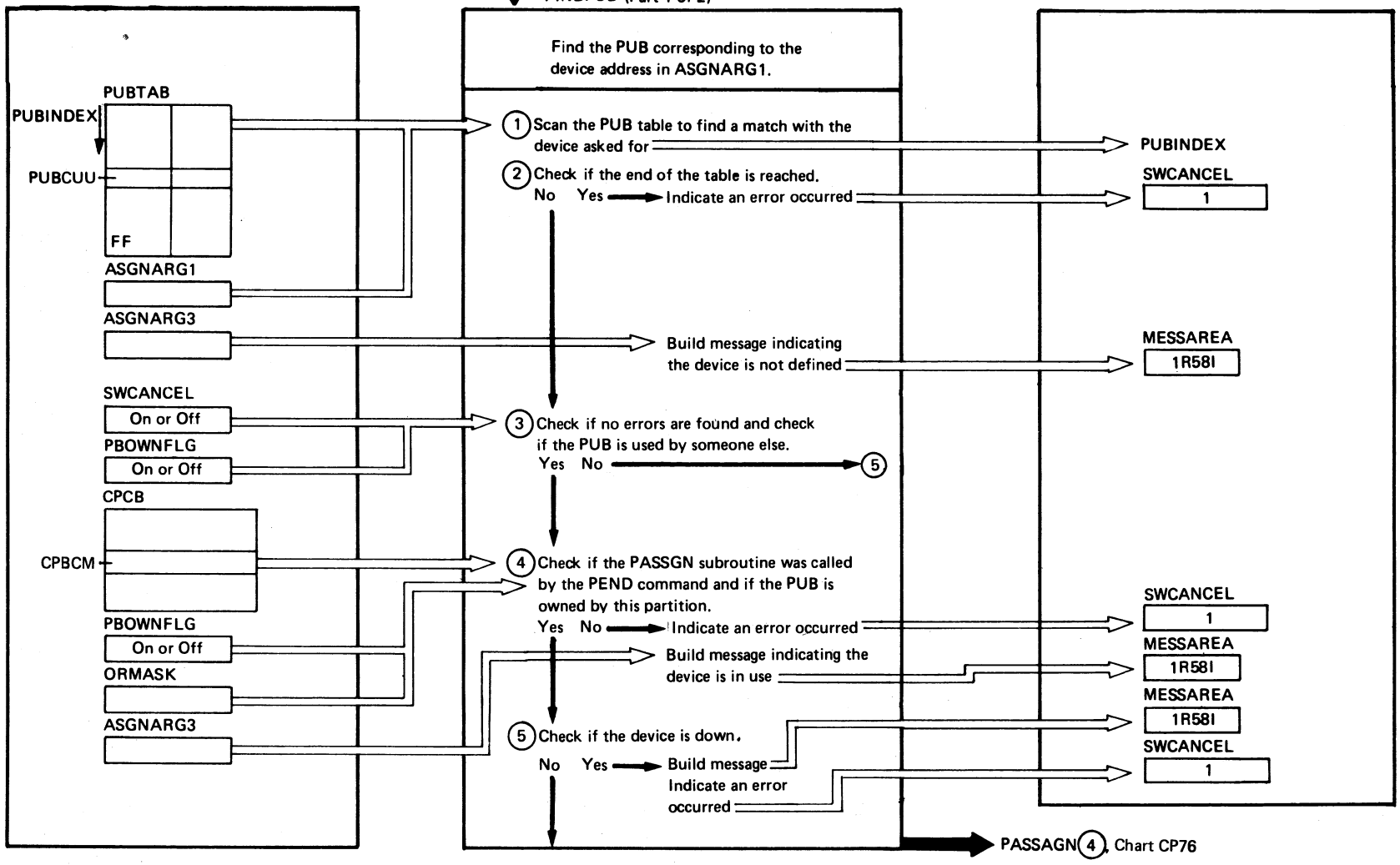


Extended Description	Include Segment	Call Subroutine	Chart
(1) c. 1R64I SYSLST LUB NOT AVAILABLE			
(4) 1R64I NO FREE LUB AVAILABLE			

Included by PASSGN, Chart CP76

LEVEL 3

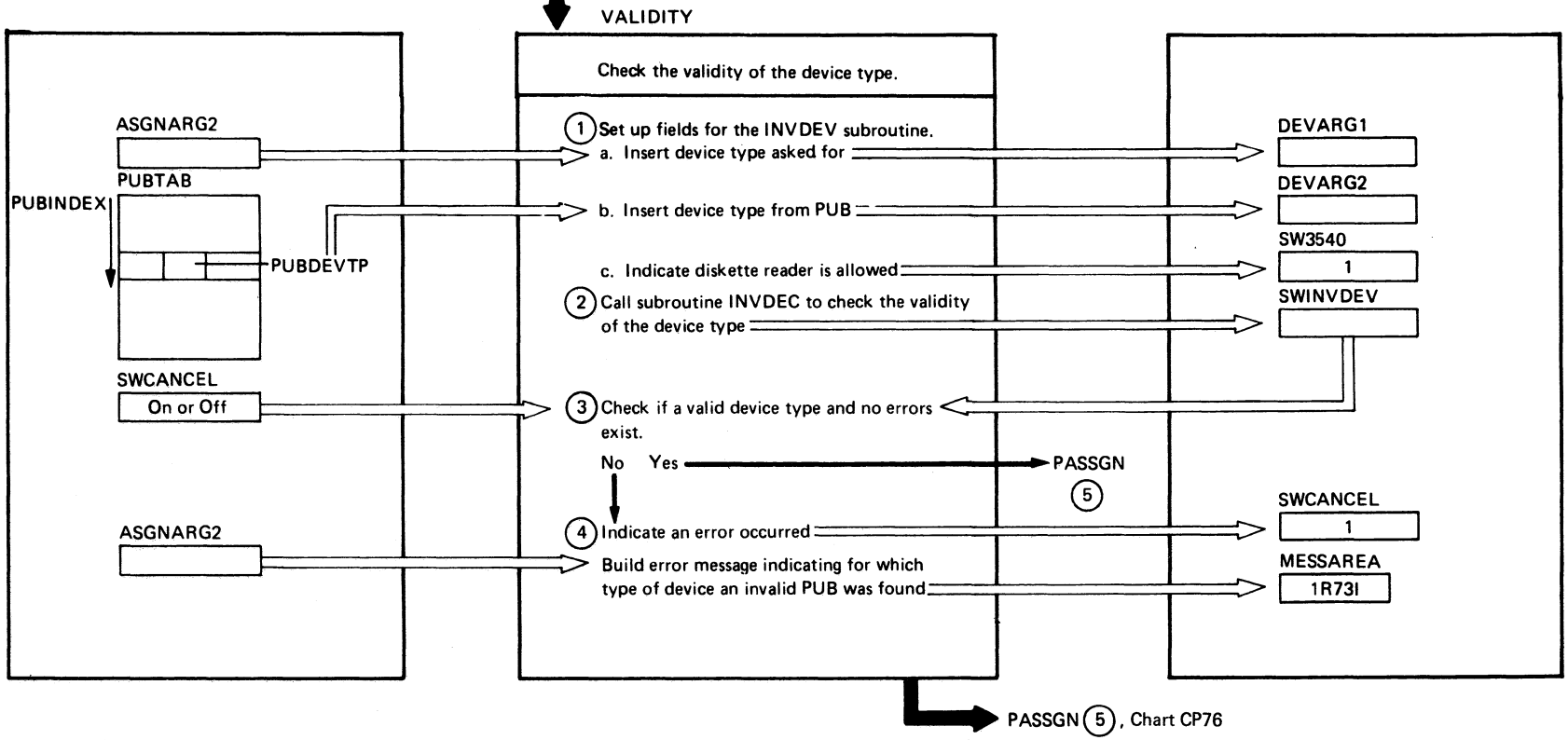
Chart CP78



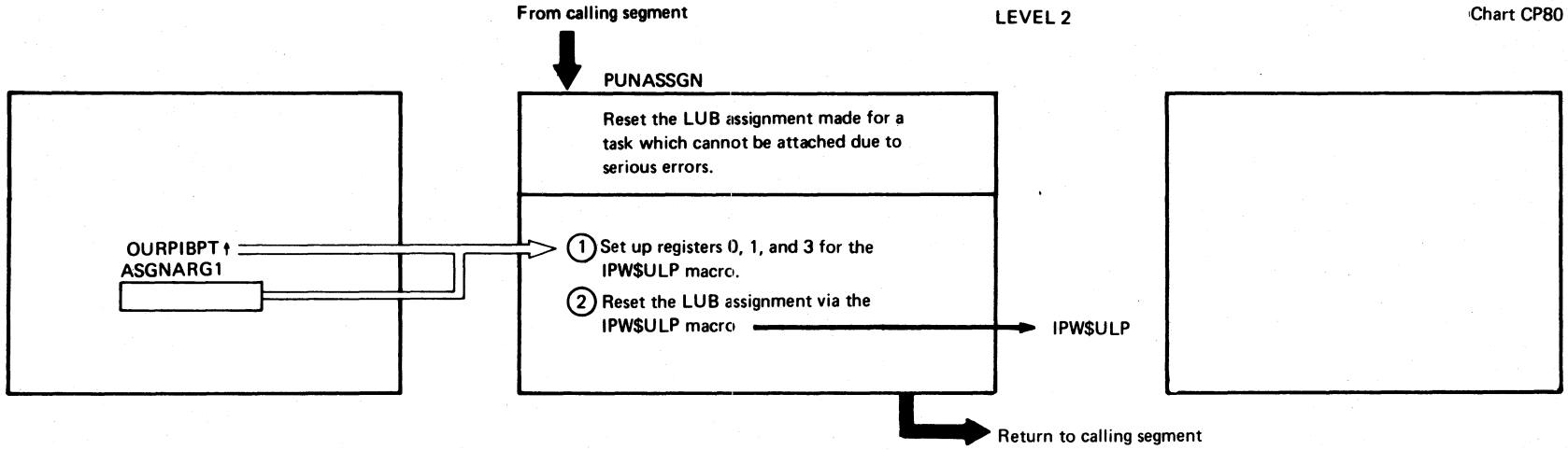
FINDPUB (Part 2 of 2)

Extended Description	Include Segment	Call Subroutine	Chart
(2) 1R58I DEVICE xxx NOT KNOWN ↑ device address (4) 1R58I DEVICE xxx IN USE ↑ device address (5) 1R58I DEVICE xxx IS DOWN ↑ device address			

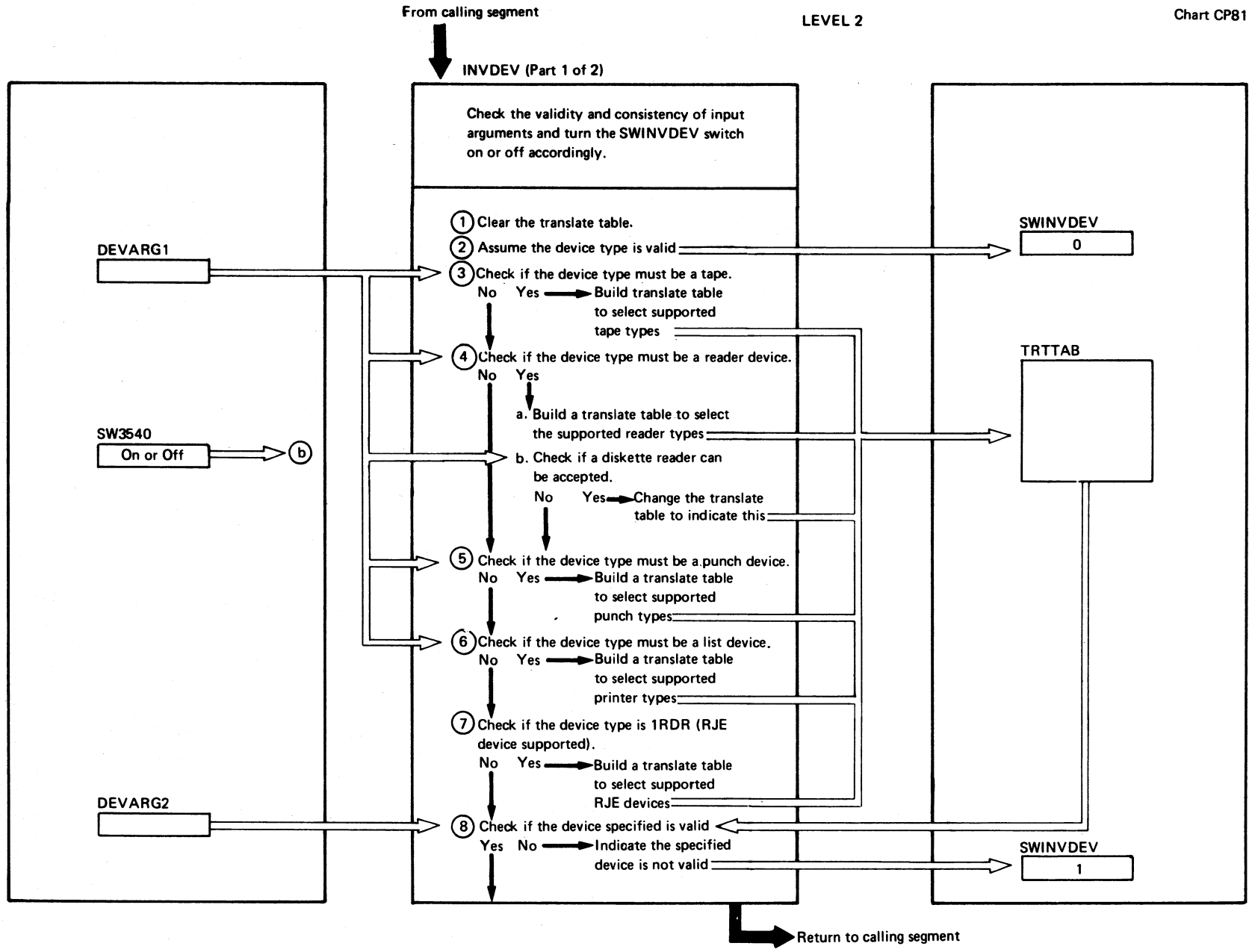
Included by PASSGN, Chart CP76 LEVEL 3 Chart CP79



Extended Description	Include Segment	Call Subroutine	Chart
(1) ASGNARG2 contains TAPE, RDR, 1RDR, PUN, LST, or SLST			
(4) 1R731 INVALID DEVICE TYPE FOR xxxx ↑ RDR (for RJE line) PRINTER, PUN, or TAPE.			

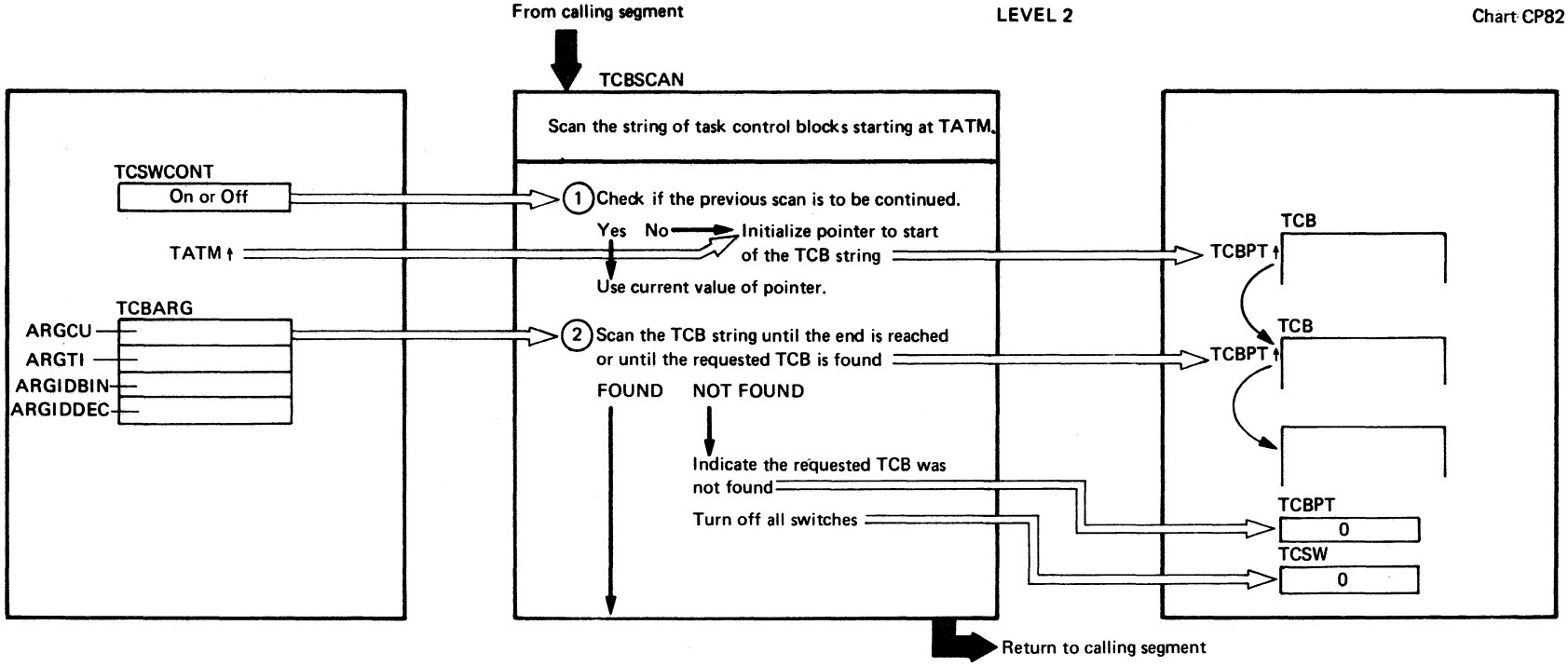


Extended Description	Include Segment	Call Subroutine	Chart
(1) Register 0 = 0 means: unassign the LUB identified by the CCB pointed to by register 3. Register 1 points to the PIB of the POWER/VS partition. Register 3 points to six bytes before the LUB identifier. (2) The IPW$ULP macro uses registers 0, 1, 2, and 3.			

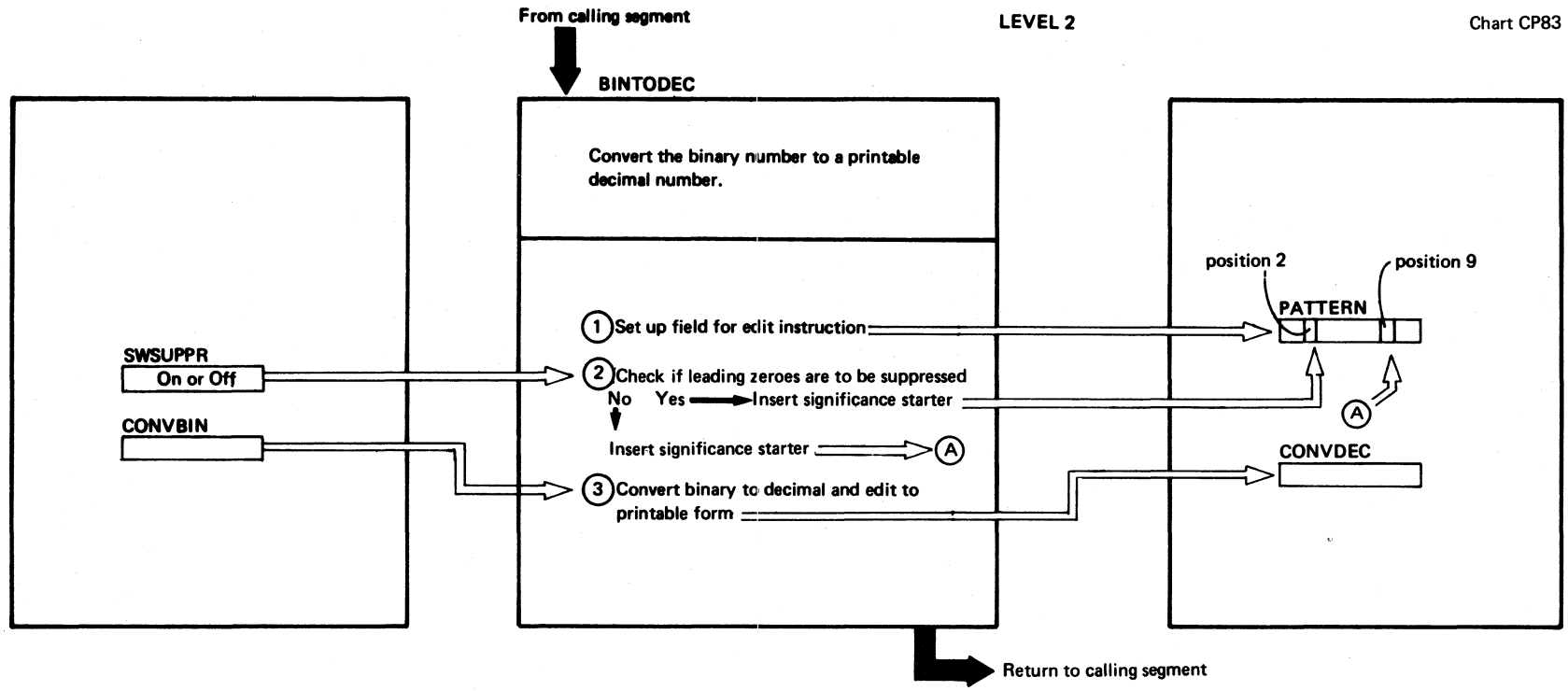

From calling segment
LEVEL 2
Chart CP81
INVDEV (Part 1 of 2)
Check the validity and consistency of input arguments and turn the SWINVDEV switch on or off accordingly.
DEVARG1
SW3540
On or Off
DEVARG2
1 Clear the translate table.
2 Assume the device type is valid
3 Check if the device type must be a tape.
No Yes Build translate table to select supported tape types
4 Check if the device type must be a reader device.
No Yes
a. Build a translate table to select the supported reader types
b. Check if a diskette reader can be accepted.
No Yes Change the translate table to indicate this
5 Check if the device type must be a punch device.
No Yes Build a translate table to select supported punch types
6 Check if the device type must be a list device.
No Yes Build a translate table to select supported printer types
7 Check if the device type is 1RDR (RJE device supported).
No Yes Build a translate table to select supported RJE devices
8 Check if the device specified is valid
Yes No Indicate the specified device is not valid
SWINVDEV
0
TRTTAB
SWINVDEV
1
Return to calling segment
Chart CP81

Extended Description	Include Segment	Call Subroutine	Chart
(3) The input field DEVARG1 can contain: TAPE b RDR b PUN b LST 1 RDR SLST The following is a list of supported devices: tape devices : 2400, 3420, 3410 reader devices : 2501, 2540R, 3504, 3505, 1442N1, 2520B1, 3525RP, 2560, 5425, 3540 punch devices : 2520B2, 2540P, 1442N2, 3525P, 1442N1, 2520B1, 3525RP, 5425 printer devices : 1403, 1443, 1403U, 3211, 3203, 3800, 5203, 5203U RJE devices : 2701, 2703 (8) The input field DEVARG2 contains the device type code obtained from the PUB.			

Page of SY33-8577-1, Revised November 24, 1977, By TNL SN33-9241



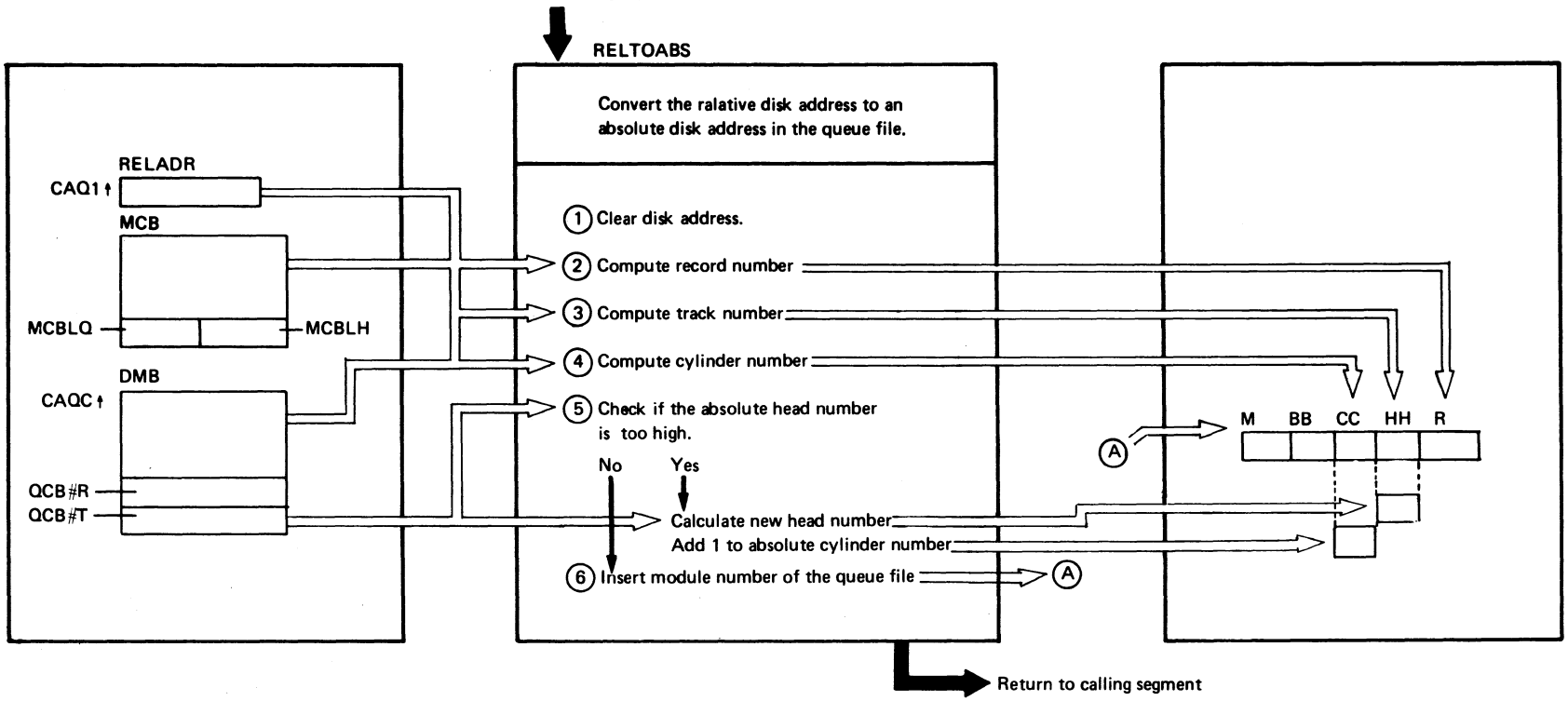
Extended Description	Include Segment	Call Subroutine	Chart
(1) In this subroutine the pointer TCBPT and not TCBPTR is used to qualify TCBs. (2) The end of the TCB string is reached when TCBPT=TATM. One to four criteria set up by the calling routine in the TCBARG field have to be met. TCBARG can contain: 1 ARGCU: device address or X'00' in last byte 2 ARGTI: task identifier or X'00' in last byte 3 ARGINBIN: binary RJE user-ID or 0 4 ARGIDDEC : decimal RJE user-ID or X'00' in last byte. The four TCBARG fields are compared with the following fields in the TCB: 1 TCBCU 2 TCBTI 3 TCBFL 4 TCBIDDEC A criterion is met if the argument field matches the corresponding field in the TCB or if the argument field contains zero.			


From calling segment
LEVEL 2
Chart CP83
BINTODEC
Convert the binary number to a printable decimal number.
position 2
position 9
PATTERN
1 Set up field for edit instruction
SWSUPPR
On or Off
2 Check if leading zeroes are to be suppressed
No
Yes
Insert significance starter
A
CONVBIN
Insert significance starter
A
CONVDEC
3 Convert binary to decimal and edit to printable form
Return to calling segment

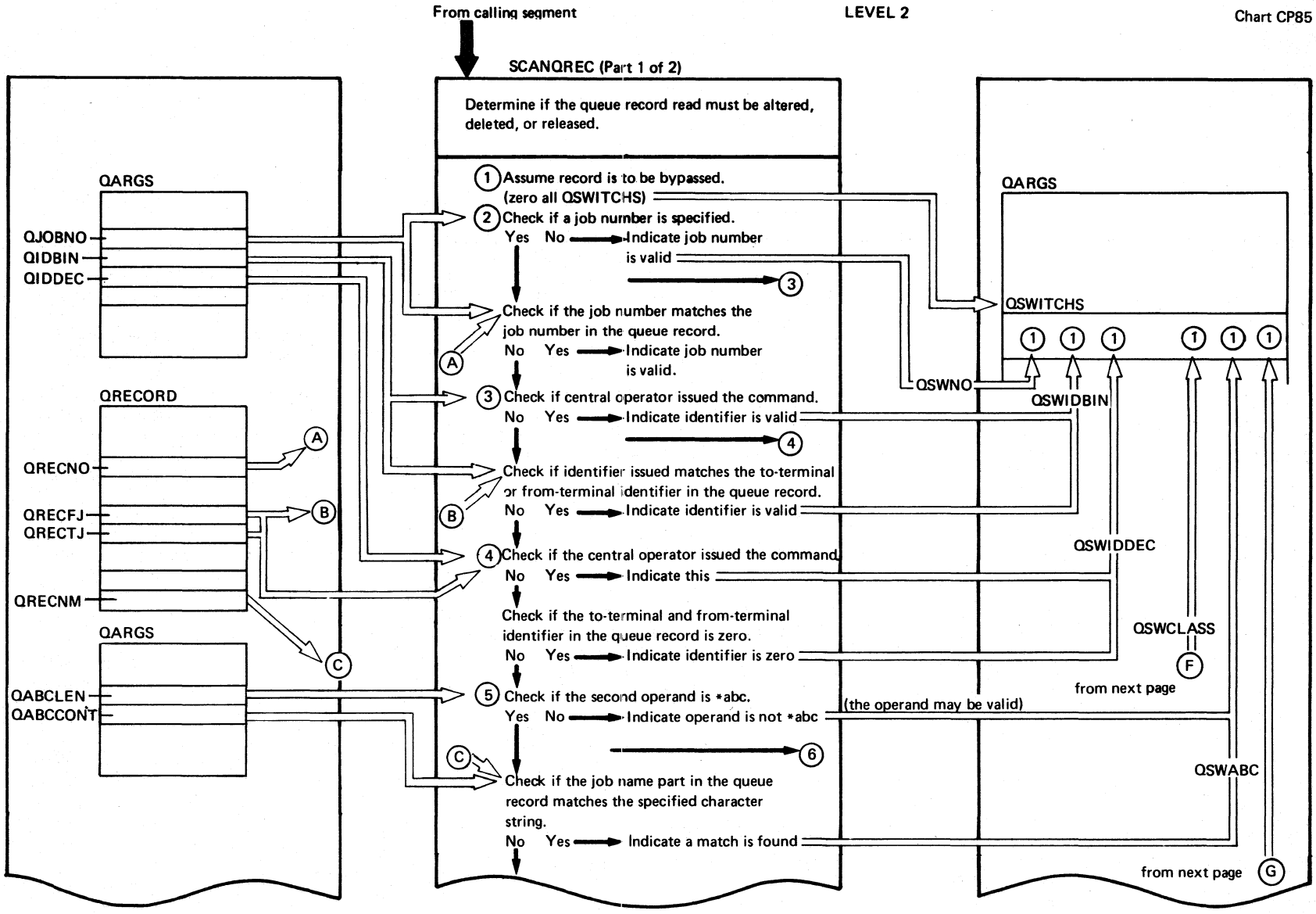
From calling segment

LEVEL 2

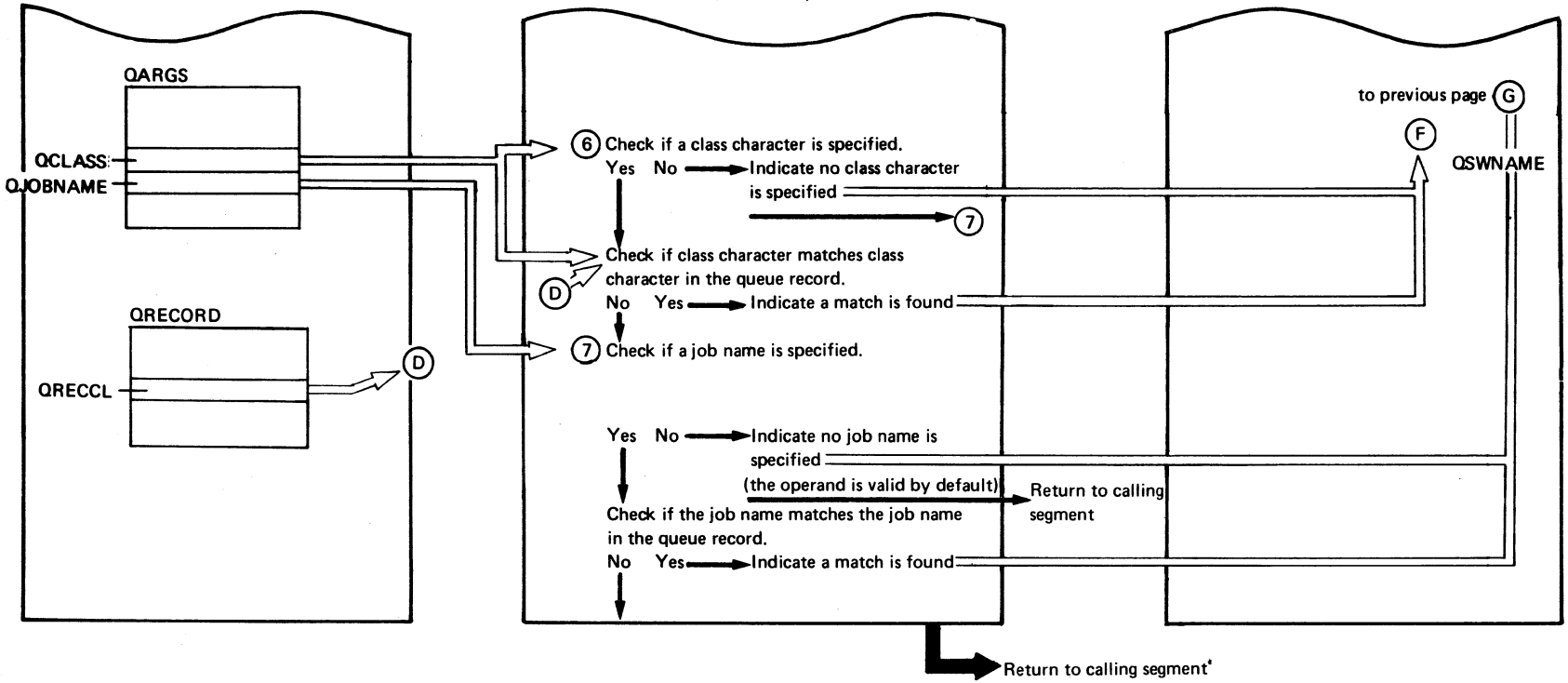
Chart CP84

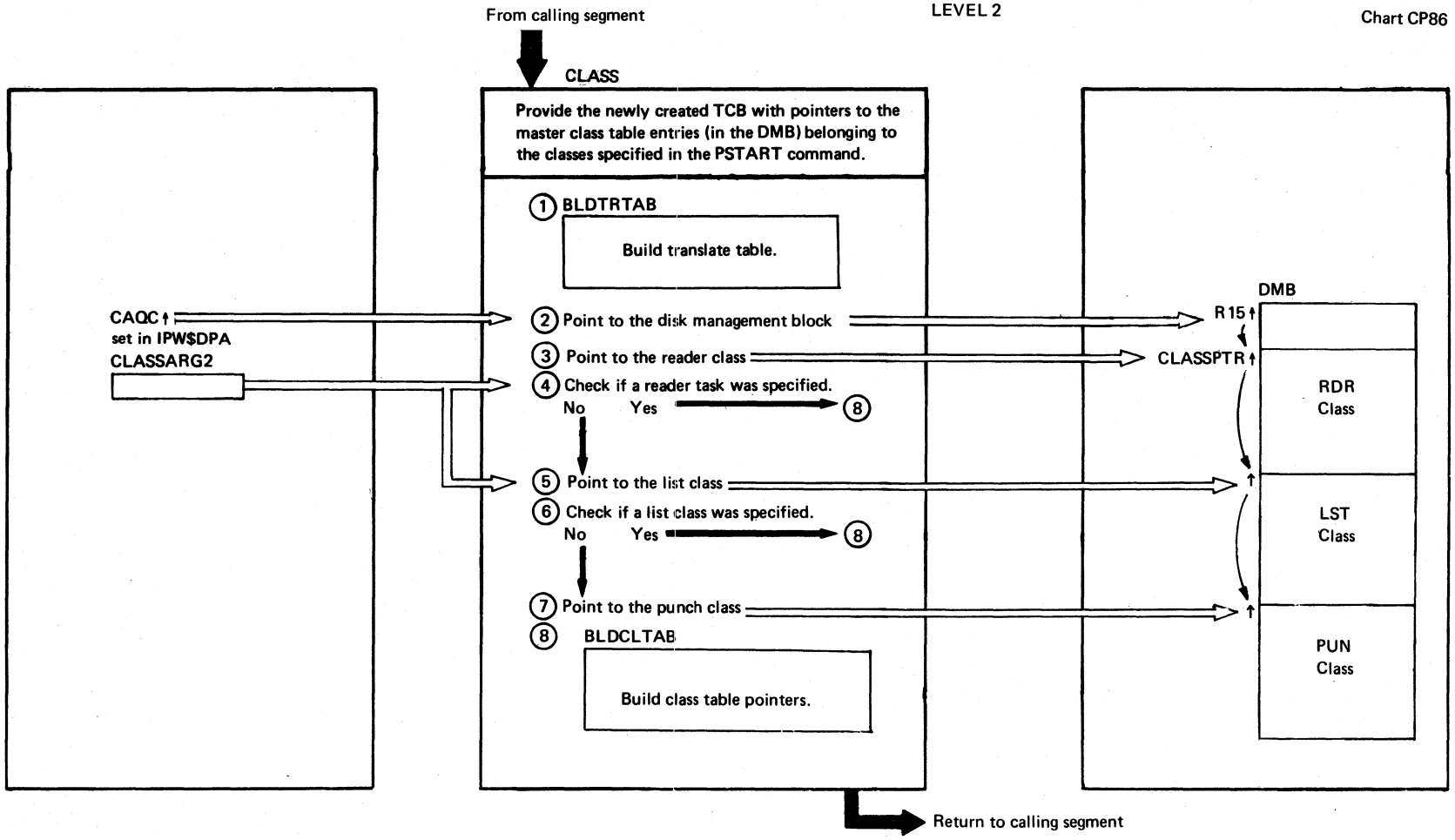


Extended Description	Include Segment	Call Subroutine	Chart
(2) The record number is computed by deviding the relative disk address by the number of records on a track and adding 1. The 1 is added since the first record is the master record.			
(3) The absolute head number (HH) equals the ralative head number plus the starting head number (MCBLH).			
(4) The absolute cylinder number (CC) equals the relative cylinder number plus the starting cylinder number (MCBLC).			


From calling segment
LEVEL 2
Chart CP85
SCANQREC (Part 1 of 2)
Determine if the queue record read must be altered, deleted, or released.
QARGS
QJOBNO
QIDBIN
QIDDEC
QRECORD
QRECNO
QRECFJ
QRECTJ
QRECNM
QARGS
QABCLEN
QABCCONT
1 Assume record is to be bypassed. (zero all QSWITCHS)
2 Check if a job number is specified.
Yes No Indicate job number is valid
3
Check if the job number matches the job number in the queue record.
No Yes Indicate job number is valid.
3 Check if central operator issued the command.
No Yes Indicate identifier is valid
4
Check if identifier issued matches the to-terminal or from-terminal identifier in the queue record.
No Yes Indicate identifier is valid
4 Check if the central operator issued the command.
No Yes Indicate this
Check if the to-terminal and from-terminal identifier in the queue record is zero.
No Yes Indicate identifier is zero
5 Check if the second operand is *abc.
Yes No Indicate operand is not *abc
6
Check if the job name part in the queue record matches the specified character string.
No Yes Indicate a match is found
QARGS
QSWITCHS
QSWNO
QSWIDBIN
QSWIDDEC
QSWCLASS
QSWABC
(the operand may be valid)
from next page
from next page
Continued on next page

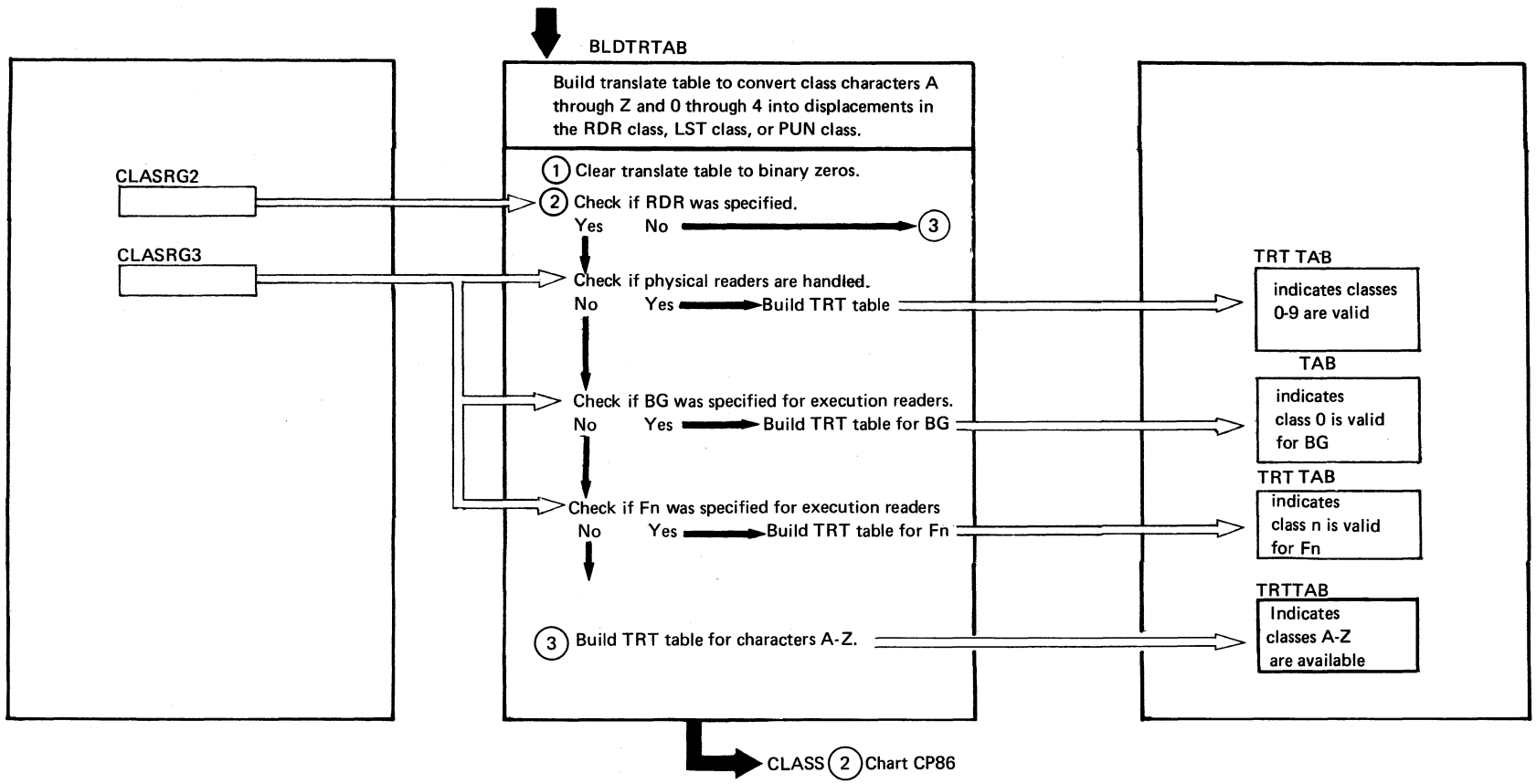
SCANQREC (Part 2 of 2)





Extended Description	Include Segment	Call Subroutine	Chart
(1)	BLDTRTAB		CP87
(2) and (3) Register 15 is used to address the disk management block. The displacement for the master class table in the DMB is added in register 15. Register 15 then points to the start of the RDR class.			
(8)	BLDCLTAB		CP88

Included by CLASS, Chart CP86 LEVEL 3 Chart CP87



Included by CLASS, Chart CP86

LEVEL 3

BLDCLTAB (Part 1 of 2)

Insert one or more pointers to class table entries into the task control block of a new reader or writer task.

CLSARG1

(1) Process all class characters until a blank is found or the maximum number of class characters allowed is reached.

When ready → (2)

For each character take the following steps:

a. Copy the class character

b. Convert class character to a displacement by using TRTTAB

c. Test if the character under investigation is valid

Yes No → Build error message → Call MSG → MSG

Indicate operand is invalid → (1)

CLAS

CLS

MESSAREA

1R54I

SWCANCEL

1

(d.)

TCB

TCBCT(1)

TCBFF(1)

TCBFF(2) FF → TCBCT(2)

TCBFF(3) → TCBCT(3)

TCBFF(4) → TCBCT(4)

CLASSPTR ↑

CLSARG1

d. Insert pointer to this class table entry

e. Insert class character

f. Insert end of class list indicator in next entry of class list in TCB → (1)

CLSARG1

SWCANCEL

On or Off

(2) Check if the maximum number of class characters allowed were processed and still more were specified and no errors were found previously

No Yes → Build error message → Call MSG → MSG

MESSAREA

1R87I

CLASS (end), Chart CP86

Extended Description	Include Segment	Call Subroutine	Chart
(1) b. The CLS field containing the converted class character is at the same location as the class character (CLAS).			
c. 1R54I CLASS x INVALID x: class character		MSG	CP75
f. The end of class list indicator (FF) is placed in the first byte of the next class list entry for each class character that was processed.			
(2) 1R87I TOO MANY CLASSES, FIRST x PROCESSED x: maximum number of class characters allowed		MSG	CP75

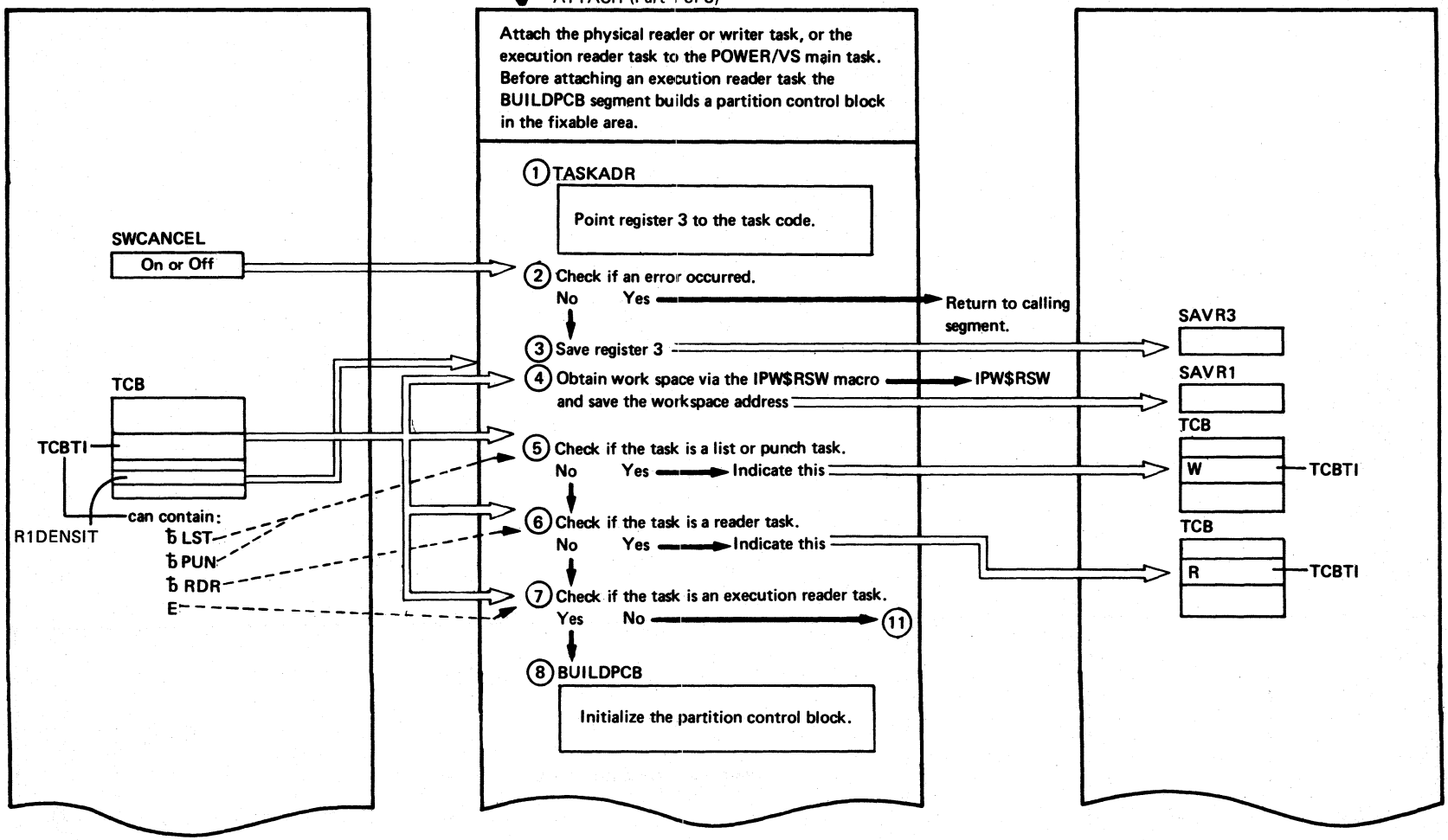
From calling segment

LEVEL 2

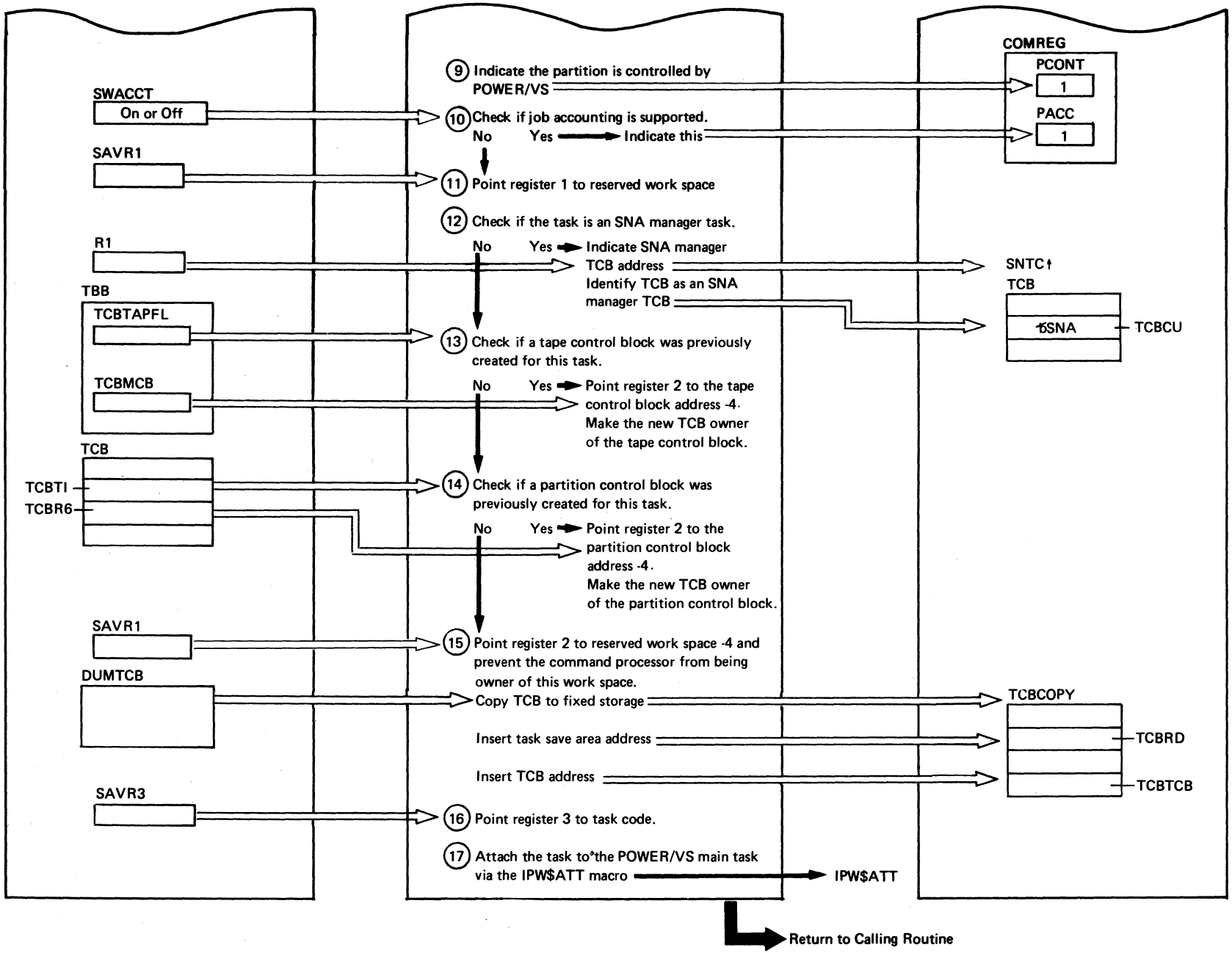
Chart CP89

ATTACH (Part 1 of 3)

Attach the physical reader or writer task, or the execution reader task to the POWER/VS main task. Before attaching an execution reader task the BUILDPCB segment builds a partition control block in the fixable area.



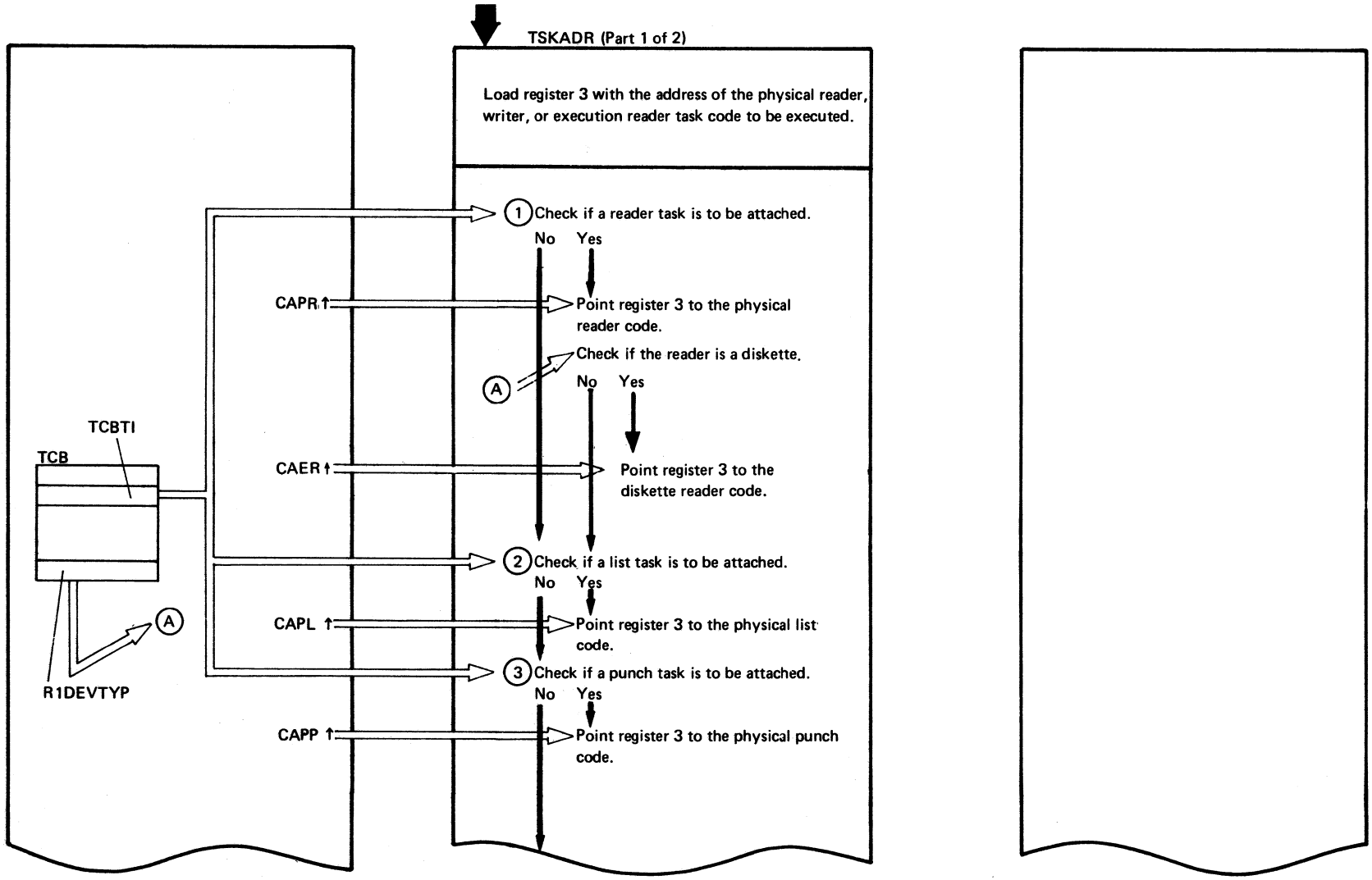
continued on next page


COMREG
PCONT
1
PACC
1
SWACCT
On or Off
SAVR1
R1
TBB
TCBTAPFL
TCBMCB
TCB
TCBTI
TCBR6
SAVR1
DUMTCB
SAVR3
9 Indicate the partition is controlled by POWER/VS
10 Check if job accounting is supported.
No Yes → Indicate this
11 Point register 1 to reserved work space
12 Check if the task is an SNA manager task.
No Yes → Indicate SNA manager TCB address
Identify TCB as an SNA manager TCB
SNTC
TCB
SNA
TCBCU
13 Check if a tape control block was previously created for this task.
No Yes → Point register 2 to the tape control block address -4.
Make the new TCB owner of the tape control block.
14 Check if a partition control block was previously created for this task.
No Yes → Point register 2 to the partition control block address -4.
Make the new TCB owner of the partition control block.
15 Point register 2 to reserved work space -4 and prevent the command processor from being owner of this work space.
Copy TCB to fixed storage
TCBCOPY
Insert task save area address
TCBRD
Insert TCB address
TCBTCB
16 Point register 3 to task code.
17 Attach the task to the POWER/VS main task via the IPW$ATT macro
IPW$ATT
Return to Calling Routine

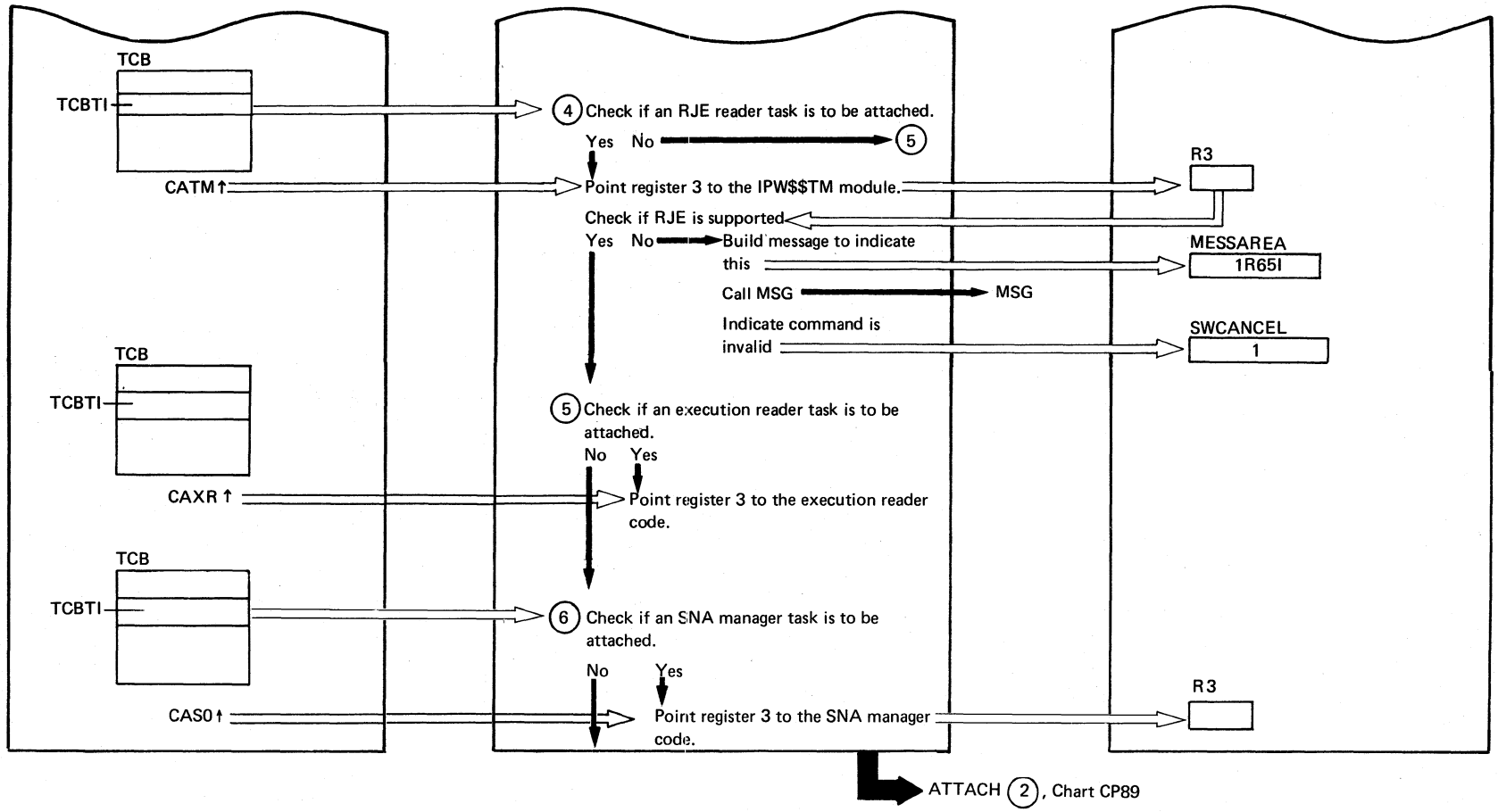
Extended Description	Include Segment	Call Subroutine	Chart
(1)	TASKADR		CP90
(4) When this is list writer task and two data file buffers are used, the TCB storage acquired is 32 bytes longer.			
(8) Registers 0, 1, 2 and 3 are used in this subroutine.	BUILDPCB		CP91

Page of SY33-8577-1, Revised November 24, 1977, By TNL SN33-9241

Included by ATTACH, Chart CP89 LEVEL 3 Chart CP90



continued on next page



Extended Description	Include Segment	Call Subroutine	Chart
(2) 1R65I RJE-BSC NOT SUPPORTED		MSG	CP75

Included by ATTACH, Chart CP89

LEVEL 3

Chart CP91

BUILDPCB (Part 1 of 4)

Build the partition control block for the partition to be started.

TAOC
R11

(1) Check if the command is issued by the initialization routine.

No Yes → Indicate AUTOSTART cards were read → SWAUTOST: 1

(2) a. Build message asking which devices should be spooled → MESSAREA: 1R86D

and initialize counter for number of devices to be spooled → SPOOLNO: 0

Call MSG → MSG

b. Initialize field specifying how many devices for each type may be entered → MAXOP

c. Build prompting message for spooling device specification → MESSAREA: 1R50D

Indicate reply is requested → SWREPLY: 1

Call MSG → MSG

Call the SCANOPND subroutine to convert answer given, to fixed format. → SCANOPND

address of OPS(1) → Point to first device address entered → TOPTR

OPERANDS: OP1, OP2

Assume the device specification is valid → SWAGAIN: 0

d. SCANANSW

Interpret answer given to prompting message.

SWAGAIN
On or Off

e. Check if the answer given specified a valid spooling device address.

Yes No → Keep prompting → (2) c.

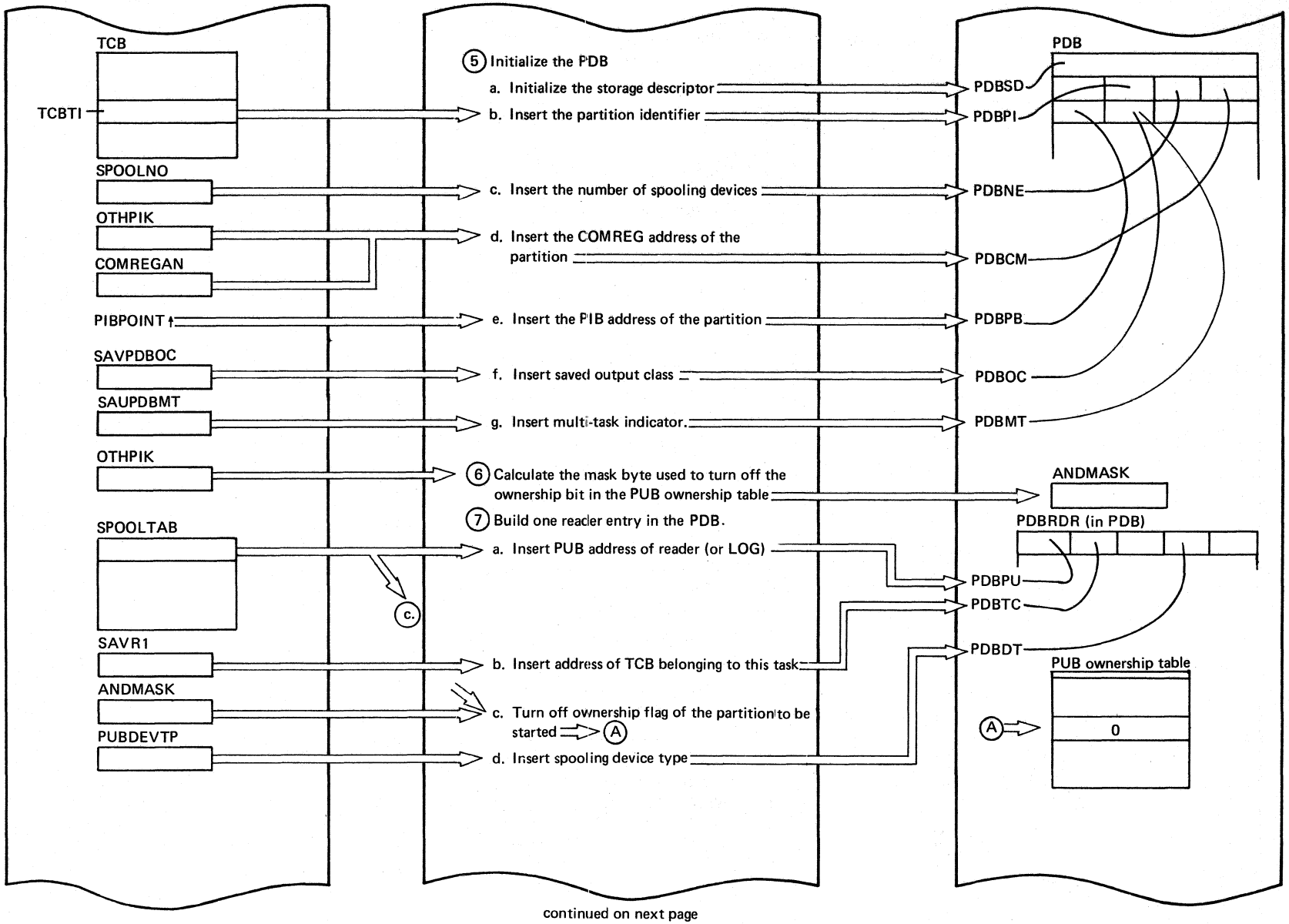
Repeat sequence from (2) b. until reader, printers, and punches have been specified.

(3) Calculate storage requirement for the partition control block → R1

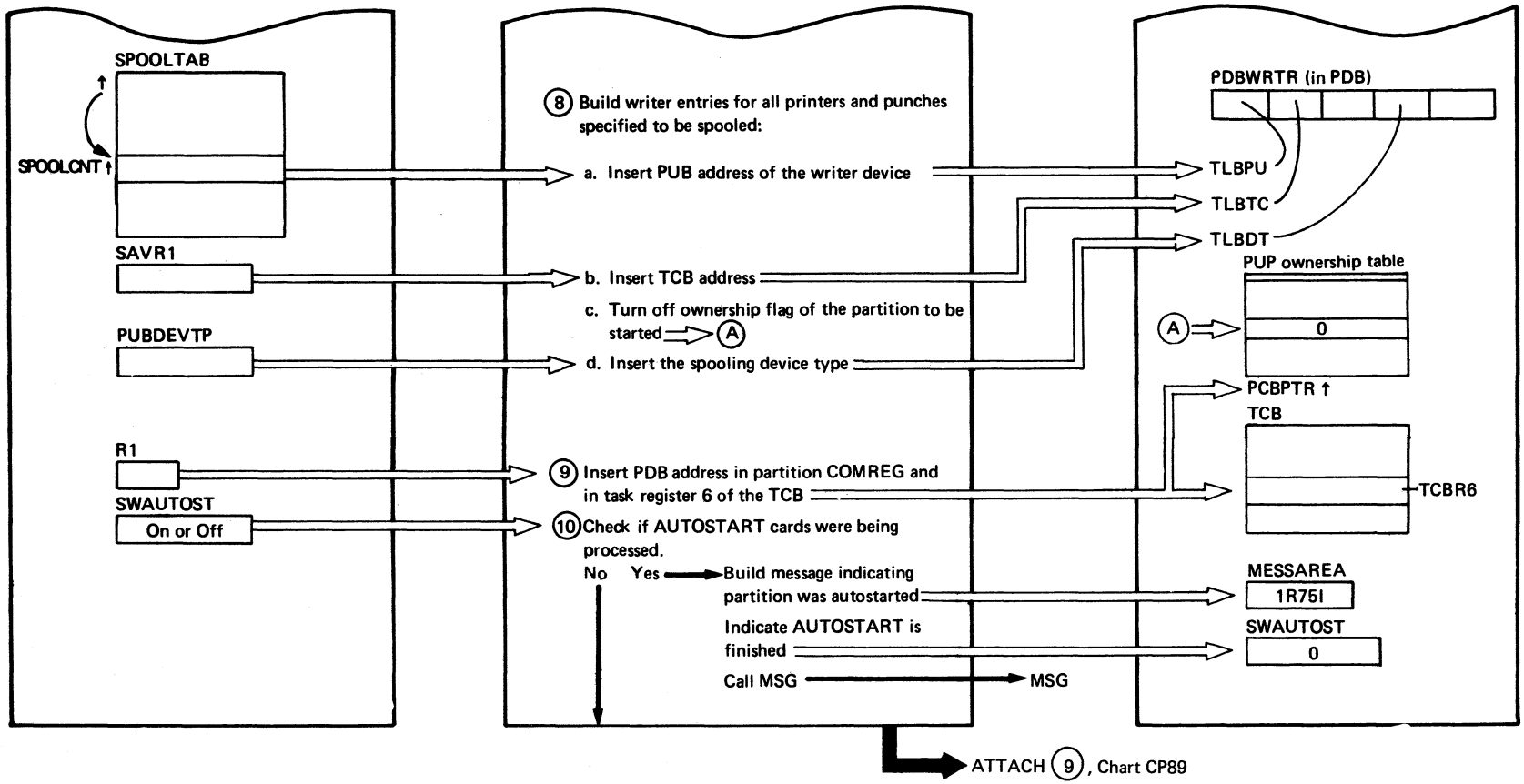
(4) Obtain storage for the PDB via the IPW$RSW macro → IPW$RSW

continued on next page

Chart CP91



continued on next page


SPOOLTAB
SPOOLCNT
SAVR1
PUBDEVTP
R1
SWAUTOST
On or Off
8 Build writer entries for all printers and punches specified to be spooled:
a. Insert PUB address of the writer device
b. Insert TCB address
c. Turn off ownership flag of the partition to be started → A
d. Insert the spooling device type
9 Insert PDB address in partition COMREG and in task register 6 of the TCB
10 Check if AUTOSTART cards were being processed.
No
Yes → Build message indicating partition was autostarted
Indicate AUTOSTART is finished
Call MSG → MSG
ATTACH 9, Chart CP89
PDBWRTR (in PDB)
TLBPU
TLBTC
TLBDT
PUP ownership table
A
0
PCBPTR
TCB
TCBR6
MESSAREA
1R75I
SWAUTOST
0

Extended Description	Include Segment	Call Subroutine	Chart
(1) Register 11 points to the TCB of the task running. TAOC is the address of the permanent command processor TCB. If register 11 is not equal to TAOC, the task is a temporary command processor task so the PSTART command was not issued by the central operator.			
(2) a. 1R86D PLEASE SPECIFY DEVICES TO BE SPOOLED		MSG	CP75
b. MAXOP contains the maximum number of operands allowed, that is MAXOP=DIM (OPS) The types are: Reader : MAXOP=1 only one spooling reader is allowed Printers : MAXOP=8 Punches : MAXOP=8 This step is implemented via a DO LOOP The loop variable is SPOOLCNT: SPOOLCNT=1: prompting for reader SPOOLCNT=2: prompting for printers SPOOLCNT=3: prompting for punches			
c. 1R50D xx READER = or xx PRINTERS = or xx PUNCHES = ↑ partition identifier Depending on the number of writer entries for specified spooling writers register 1 is used in the calculation.		MSG SCANOPND	CP75 CP70
d.	SCANANSW		CP92
(7) SPOOLTAB contains displacements in the PUB table of all specified spooling devices. The first entry is reader or LOG, the next entries are writers (print or punch). SPOOLTAB entries are built by segment SCANANSW for each specified valid spooling device.			
(8) This step is implemented via a DO LOOP: Loop variable is SPOOLCNT, limiting variable is SPOOLNO. SPOOLTAB is an array indexed by SPOOLCNT.			
(10) 1R75I xx AUTOSTARTED ↑ partition identifier			

Included by BUILDPCB, Chart CP91

LEVEL 4

Chart CP92

SCANANSW (Part 1 of 3)

Interpret answers to messages prompting the operator to specify which devices must be spooled for a partition to be started.

OP1

OPLEN

SPOOLCNT

MAXOP

SWAGAIN

On or Off

OP1

OPLEN

OPSWHEX

(1) Check if an answer to the prompting message was entered.

Yes No → Indicate prompting message is to be sent again → SWAGAIN 1

(2) Check if NO was answered for a reader device.

No Yes → Calculate displacement of the SYSLOG LUB.

Insert SYSLOG LUB address → OURLUBPTR ↑

Store PUB number → TRYTAB

→ (5)

(3) Check if NO was answered for a writer device.

No Yes → (5)

(4) Process all entered device addresses up to the maximum number allowed. For each address take the following actions:

a. Check if the device address syntax is valid.

Yes No → Indicate an invalid answer is given → SWAGAIN 1

→ (4)2.

SCANPUBS

Determine whether the device type is valid and store the PUBINDEX in TRYTAB.

b. Check if the device address entered can be used.

Yes No → Build message → MESSAREA 1R71I

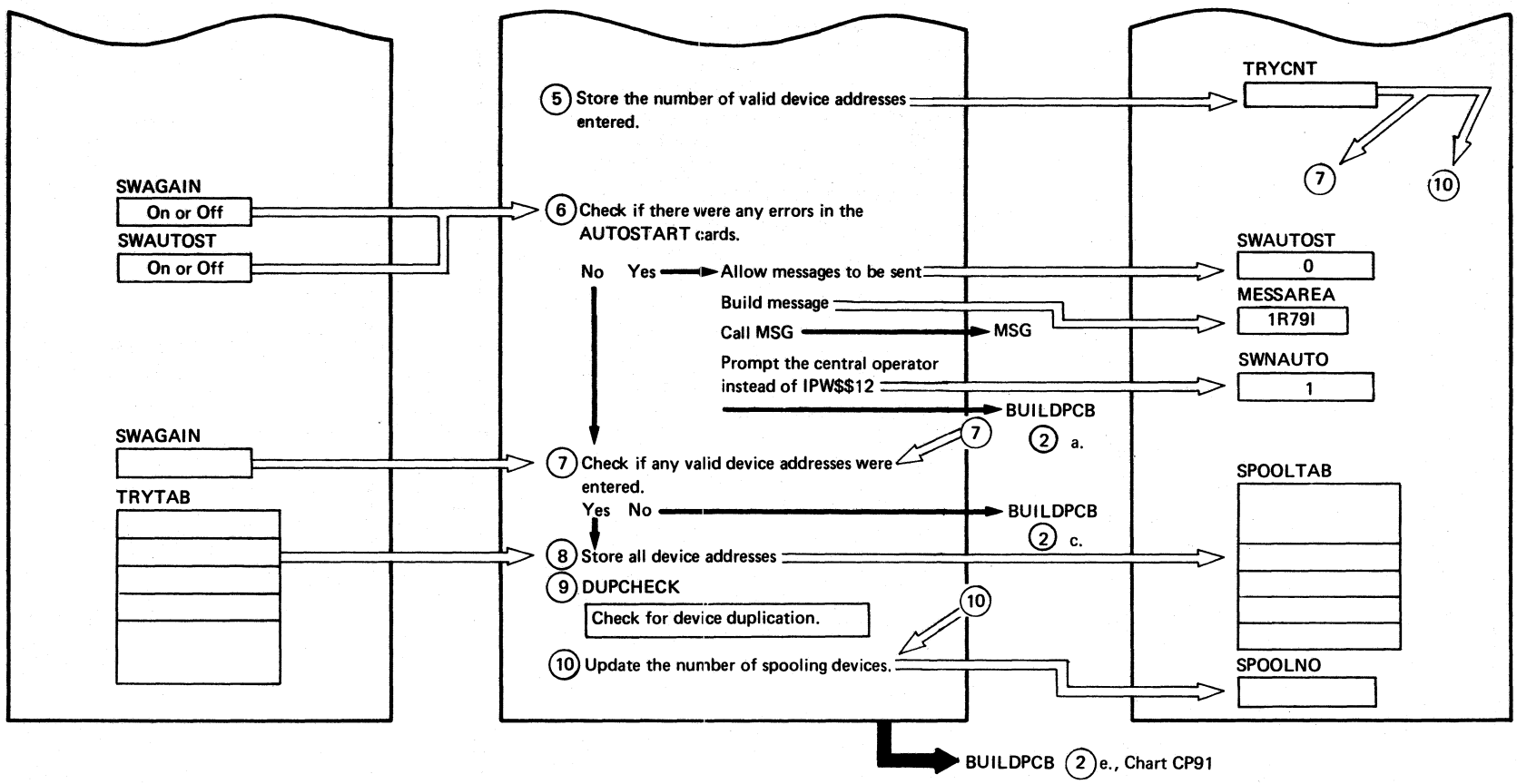
Call MSG → MSG

c. Check if all device addresses have been processed.

Yes No → (5)

Process the next entered address if the maximum number allowed has not yet been reached → (3)

continued on next page

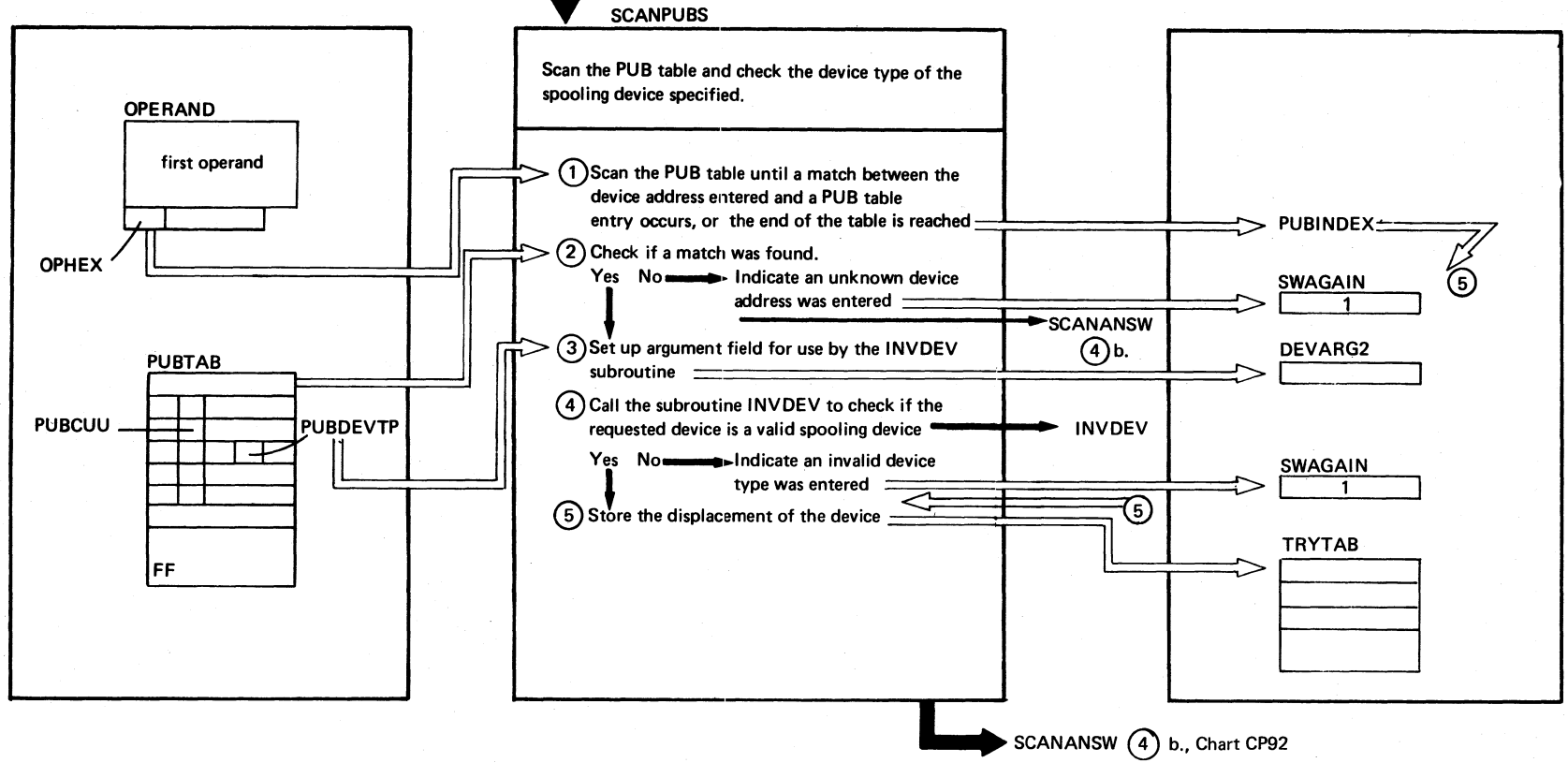

TRYCNT
5 Store the number of valid device addresses entered.
7
10
SWAGAIN
On or Off
SWAUTOST
On or Off
6 Check if there were any errors in the AUTOSTART cards.
SWAUTOST
0
No Yes
Allow messages to be sent
Build message
MESSAREA
1R79I
Call MSG
MSG
Prompt the central operator instead of IPW$$12
SWNAUTO
1
BUILDPCB
2 a.
7
SWAGAIN
7 Check if any valid device addresses were entered.
SPOOLTAB
TRYTAB
Yes No
BUILDPCB
2 c.
8 Store all device addresses
9 DUPCHECK
Check for device duplication.
10
10 Update the number of spooling devices.
SPOOLNO
BUILDPCB 2 e., Chart CP91

Extended Description	Include Segment	Call Subroutine	Chart
(1) The SPOOLCNT field is used to indicate for which spooling device an answer is expected SPOOLCNT = 1 for a reader SPOOLCNT = 2 for a printer SPOOLCNT = 3 for a punch The answers given to the prompting message 1R50D xx READER = PRINTERS = PUNCHES = are ↑ partition identifier hexadecimal addresses of readers, or punches. They are contained in input as OPS(1), OPS(2) etc.			
(1) and (4) The SWAGAIN switch is used to indicate an invalid address was entered and the prompting message should be issued again in segment BUILDPCB The MAXOP field gives the maximum number of device addresses allowed to be specified as spooling devices. MAXOP is set in BUILDPCB and = 1 for reader = 8 (= dimension of OPS) for writers and punches			
(4)1.	SCANPUBS		CP93
(4)2. 1R71I OPERAND x IS NO VALID READER or PRINTER or PUNCH		MSG	CP75
(6) 1R79I ERRONEOUS AUTOSTART CARD (S) READ		MSG	CP75
(9)	DUPCHECK		CP94

Included by SCANANSW, Chart CP92

LEVEL 5

Chart CP93



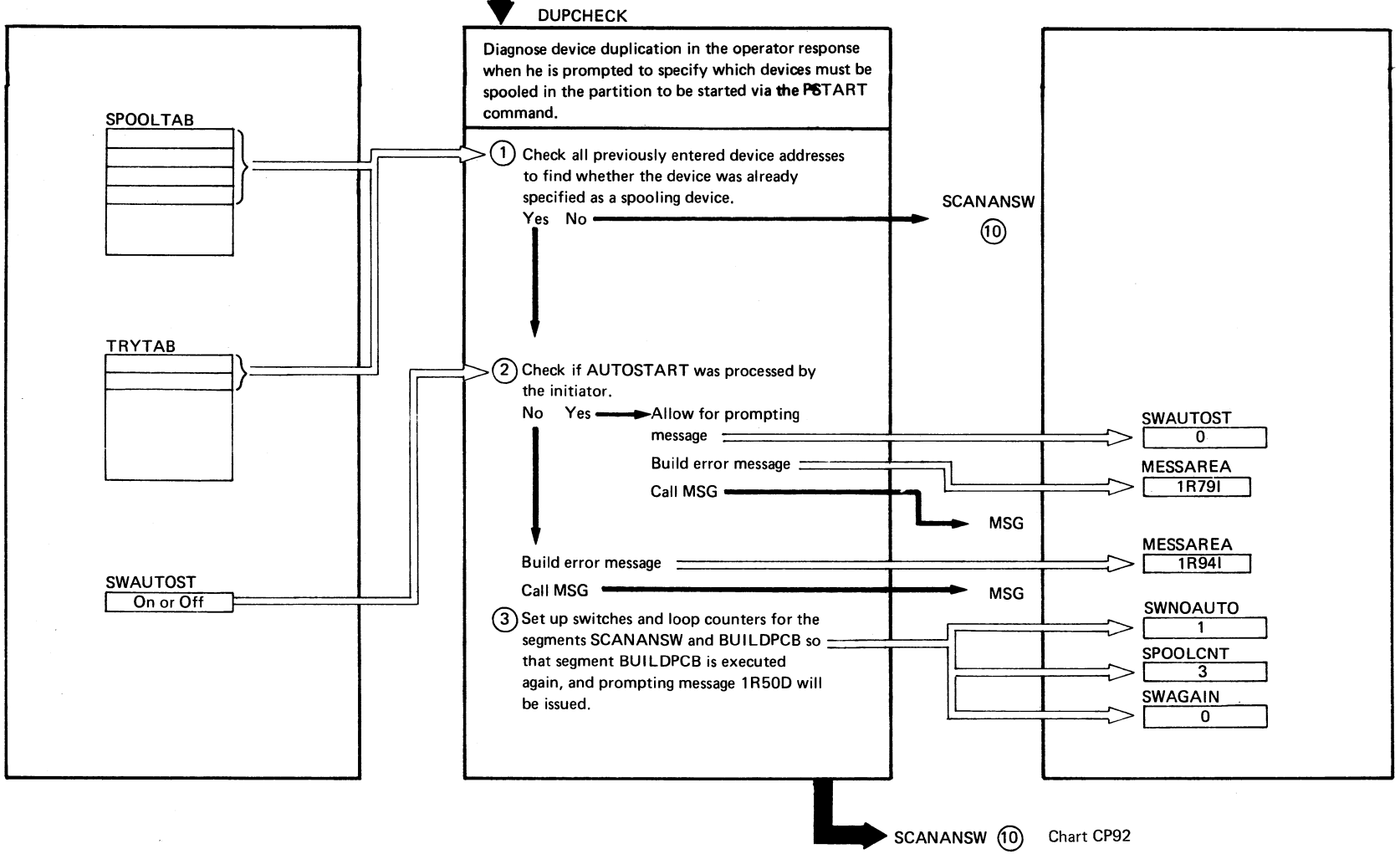
Extended Description	Include Segment	Call Subroutine	Chart
4		INVDEV	CP81

Chart CP93

Included by SCANANSW, Chart CP92

LEVEL 6

Chart CP94



SCANANSW 10 Chart CP92

Extended Description	Include Segment	Call Subroutine	Chart
2 1R79I ERRONEOUS AUTOSTART CARDS (S) READ		MSG	CP75
1R94I INVALID DEVICE DUPLICATION		MSG	CP75

Chart CP94

Included by PDISPLAY, Chart CP35 LEVEL 3 Chart CP95

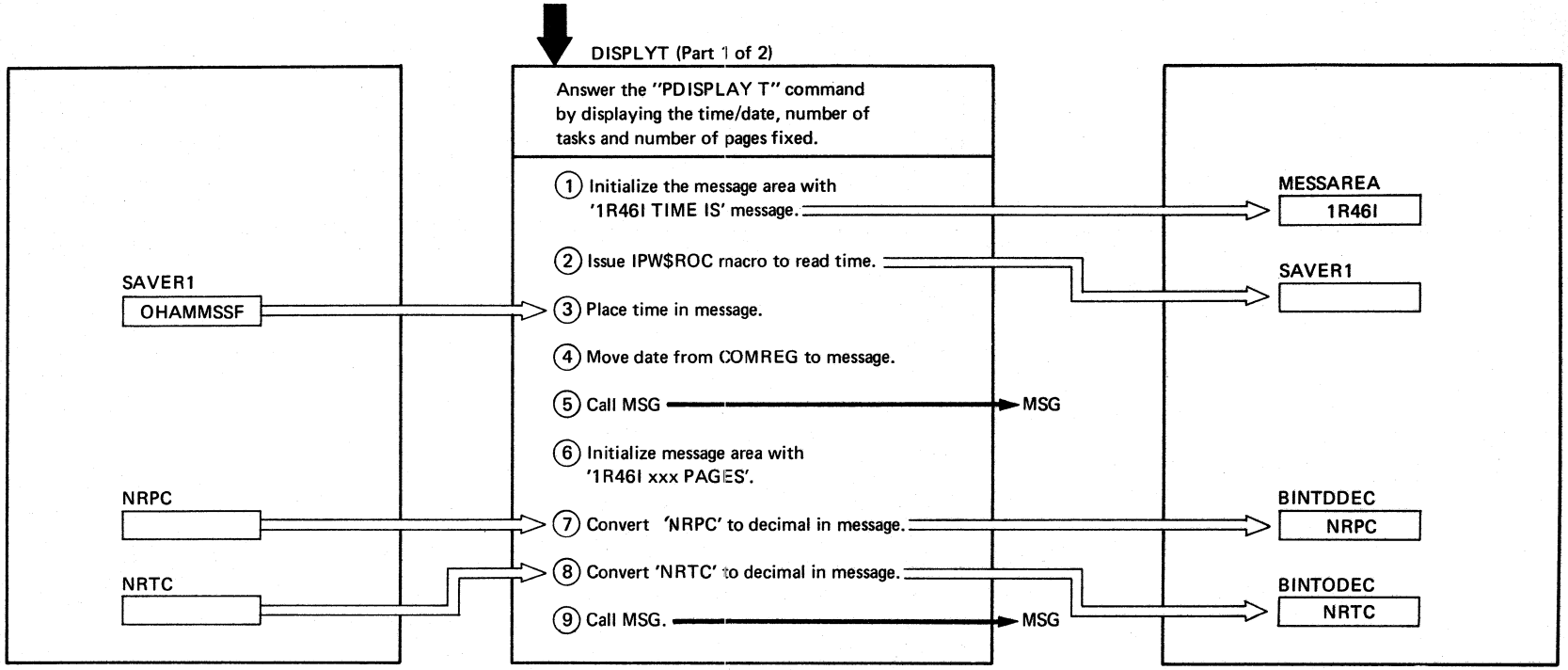
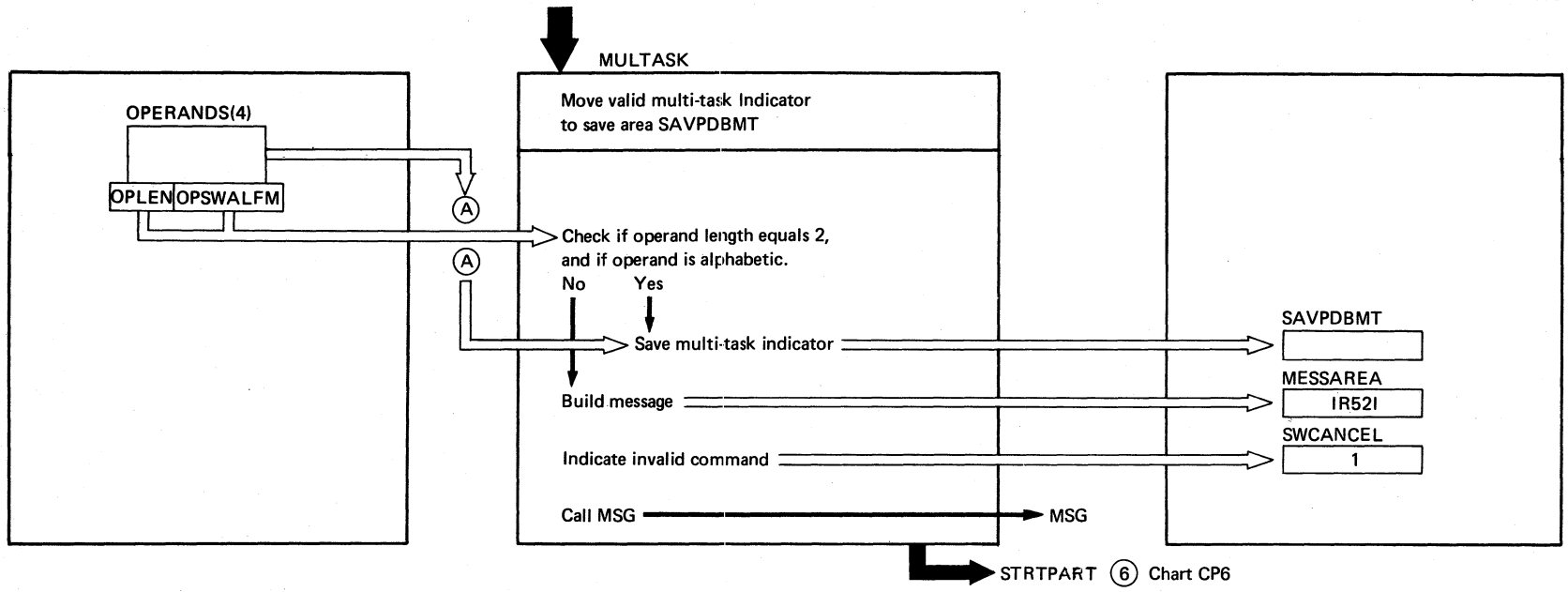


Chart CP95

Extended Description	Include	Call Subroutine	Chart
(2) Read clock and place in register 1			
(3) Edit time from SAVER1 into message			
(4) Get current date from partitions COMREG			
(5) Print line		MSG	CP76
(7) and (8) Convert NRPC and NRTC to printable		BINTODEC	CP84
(9) Print line		MSG	CP76

Included by STRTPART, Chart CP6 LEVEL 4 Chart CP96



Extended Description	Include Segment	Call Subroutin	Chart
IR52I OPENAND 4 IS INVALID KEYWORDS		MSG	CP75

CHART CQ: IPW$$PS - PRINT QUEUE STATUS (23 PARTS)

Chart CQ00: IPW$$PS - Print Queue Status, General Flow and Macro Calls

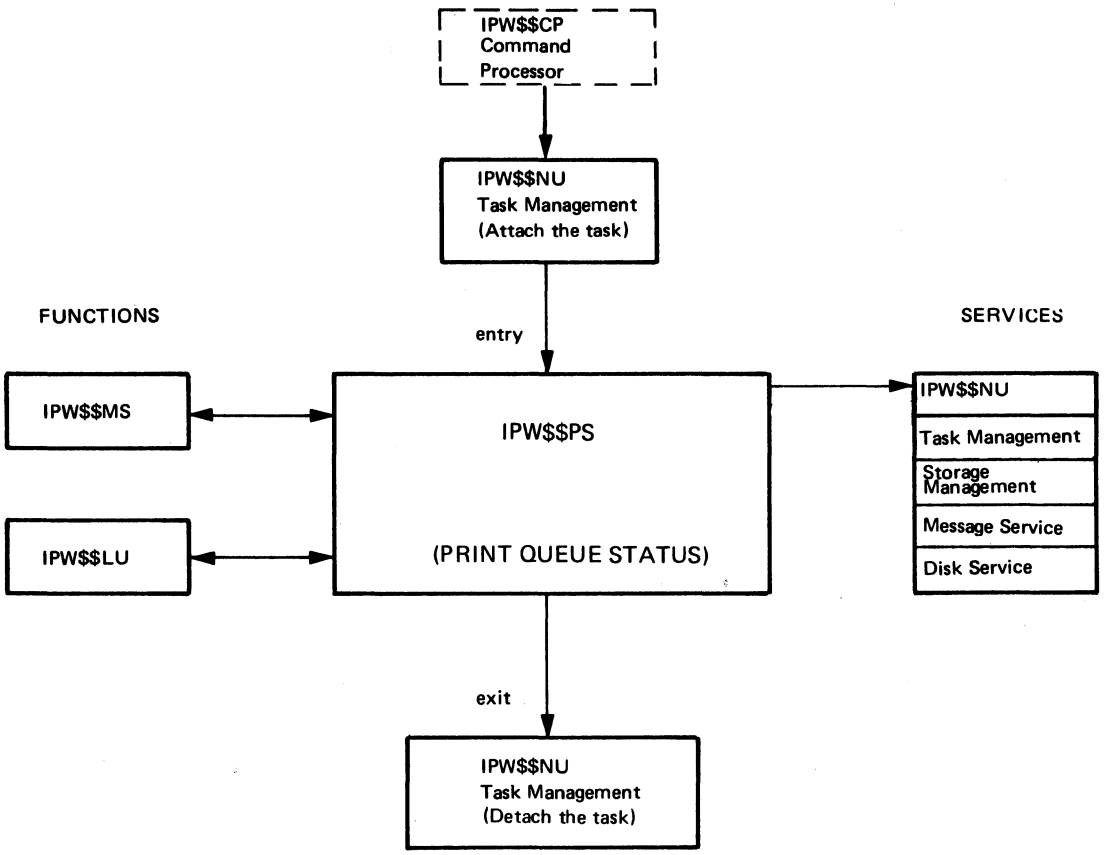


Chart CQ1: IPW$$PS - Organization

Chart CQ1

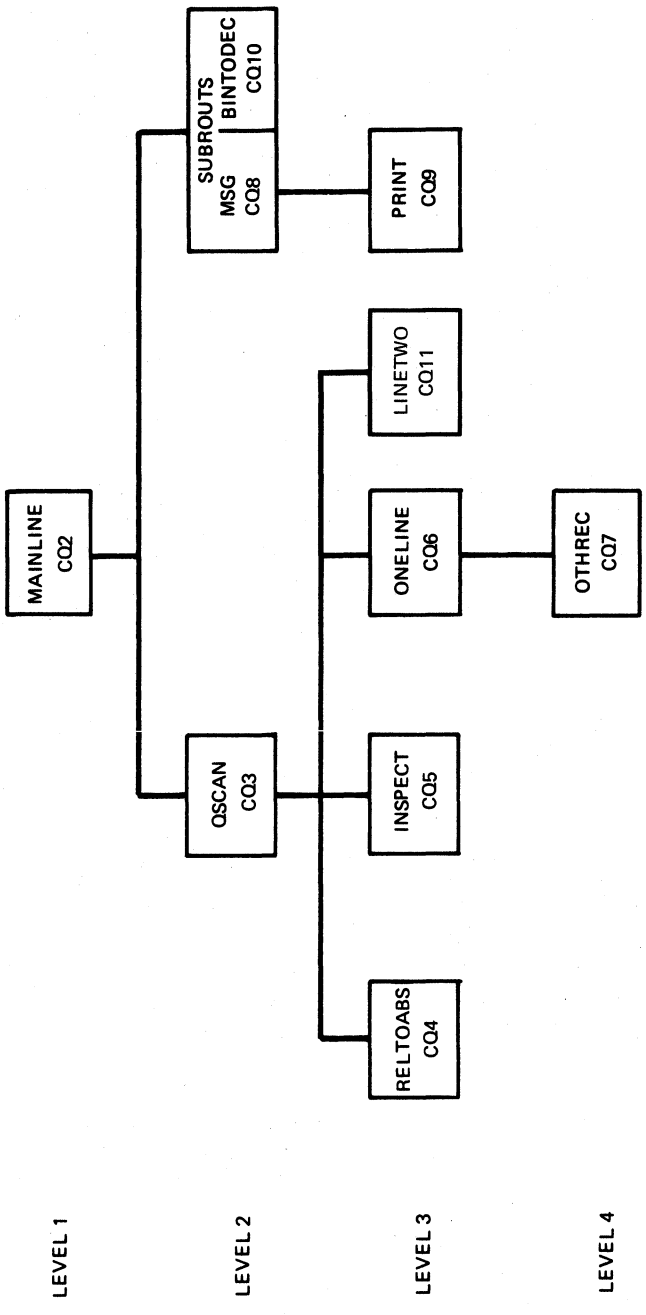


Chart CQ1

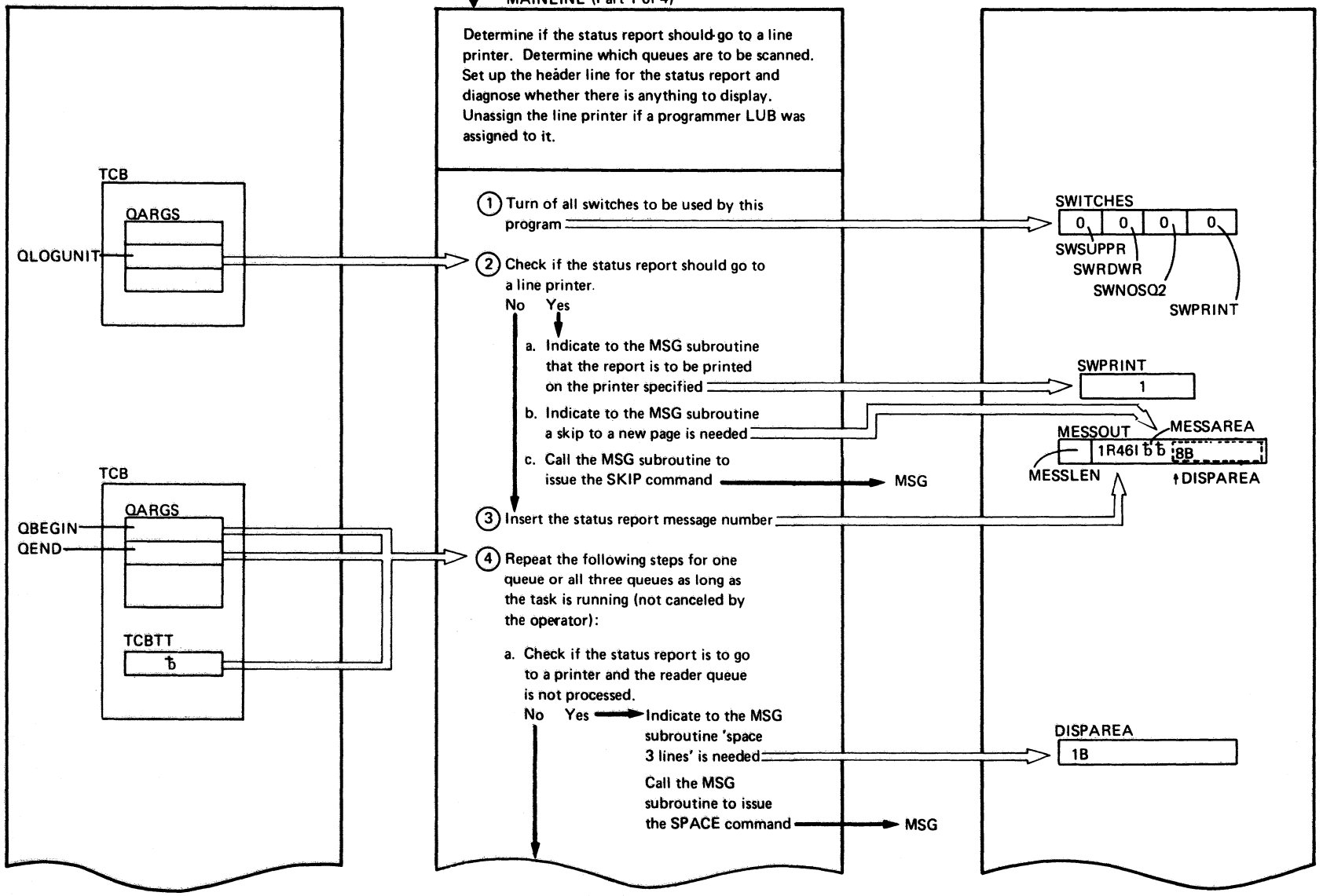
From PDISPLAY (in IPW$$CP) via POWER/VS task management

LEVEL1

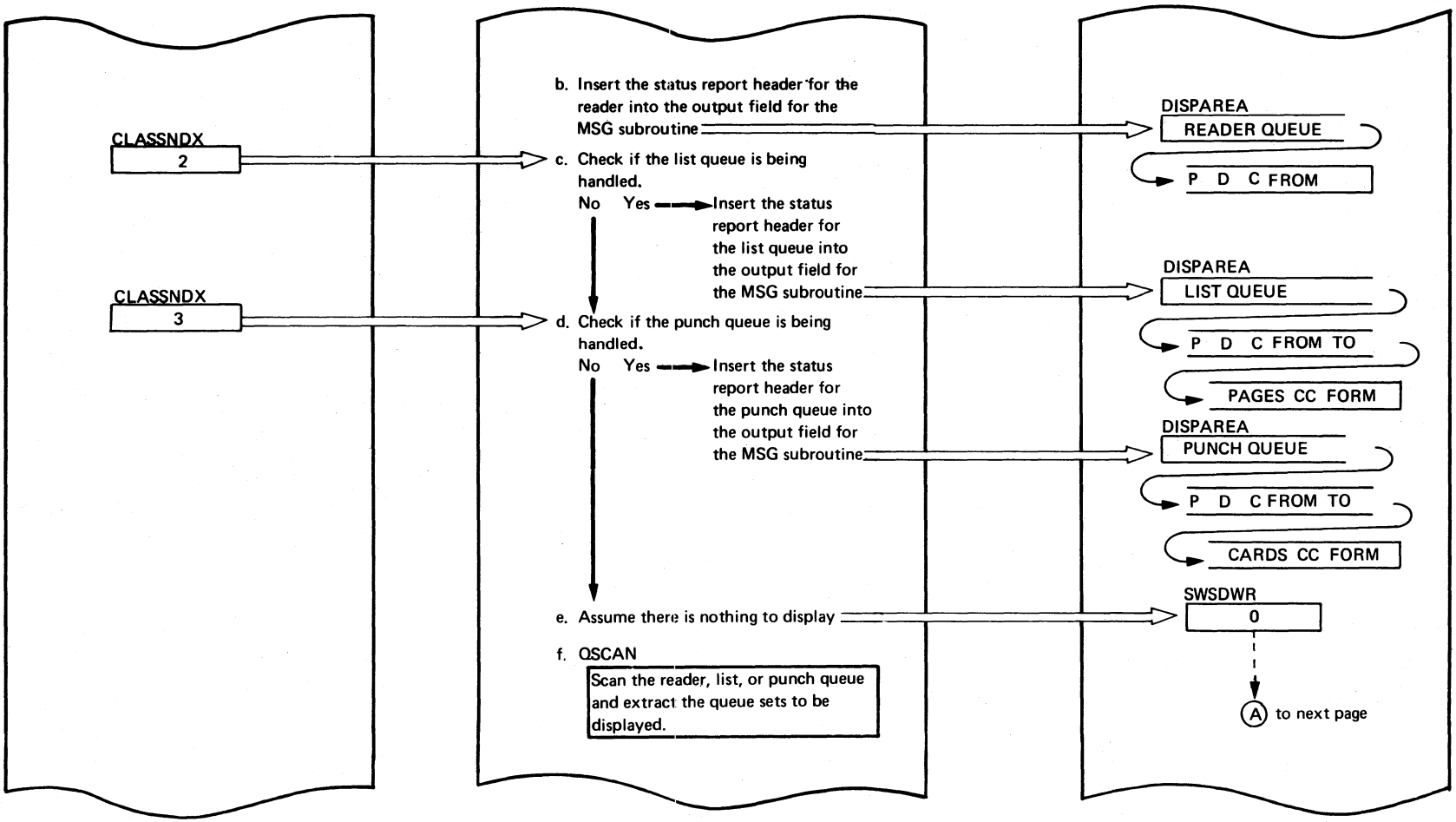
Chart CQ2

MAINLINE (Part 1 of 4)

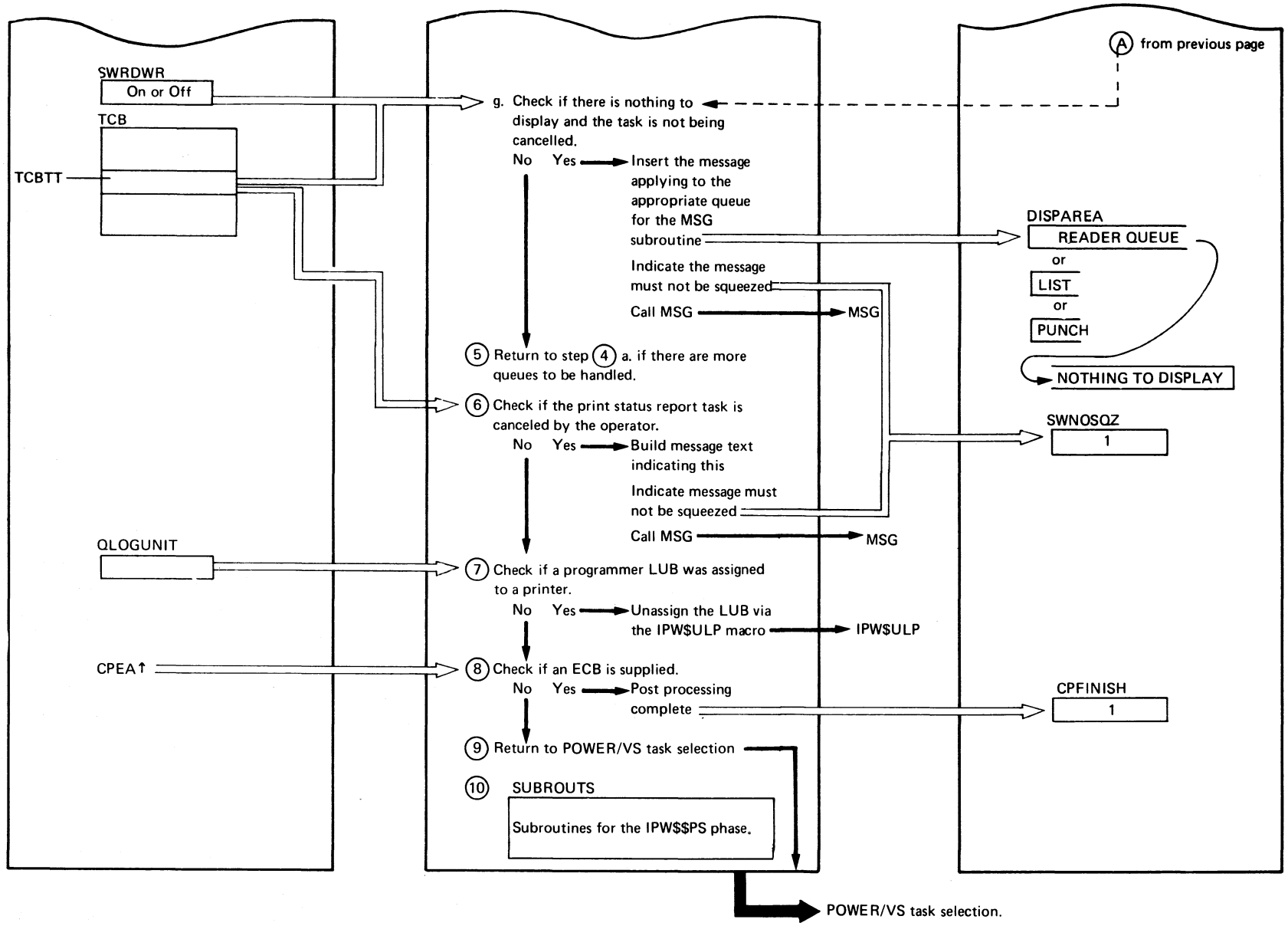
Determine if the status report should go to a line printer. Determine which queues are to be scanned. Set up the header line for the status report and diagnose whether there is anything to display. Unassign the line printer if a programmer LUB was assigned to it.



continued on next page

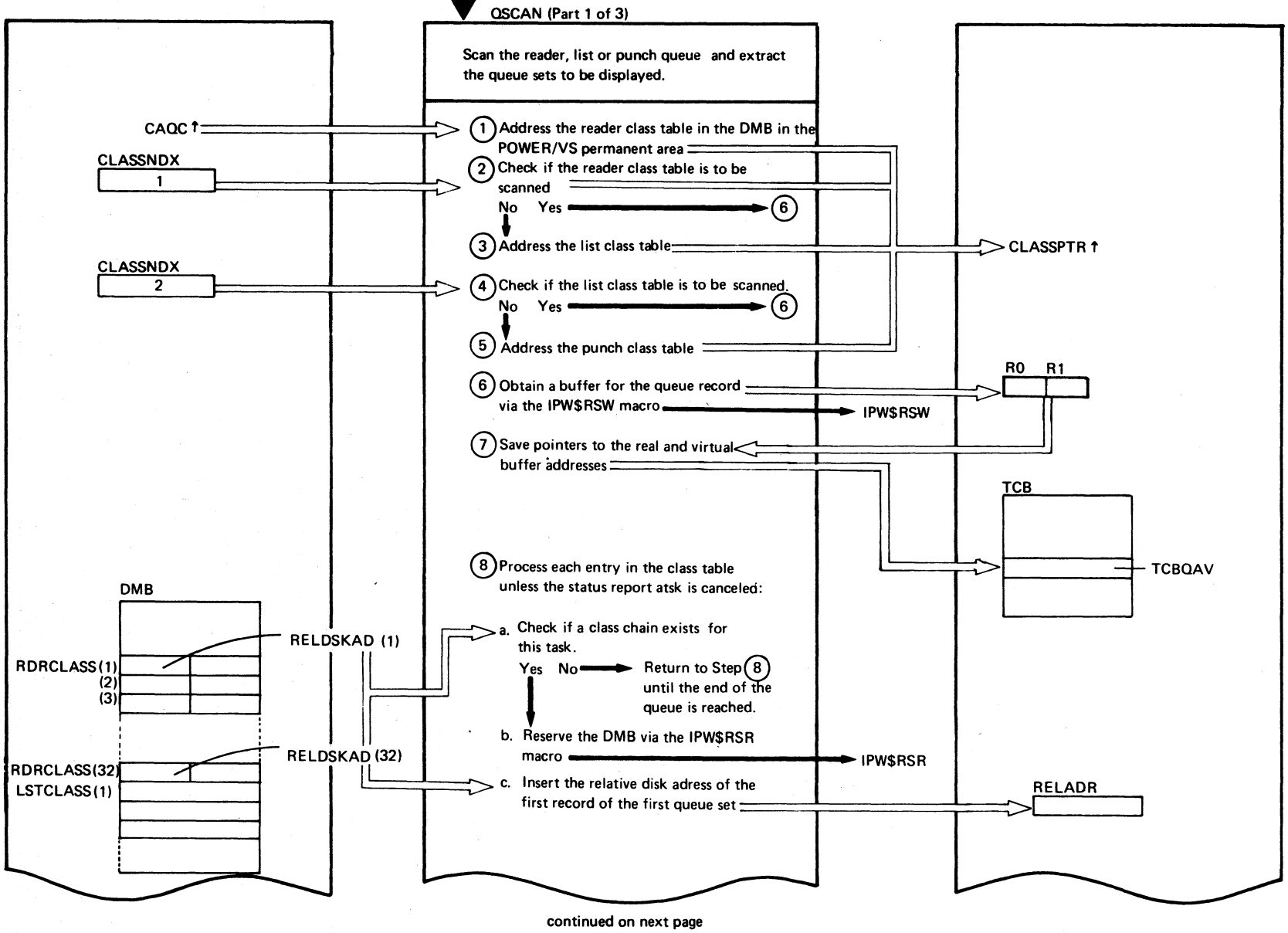


continued on next page

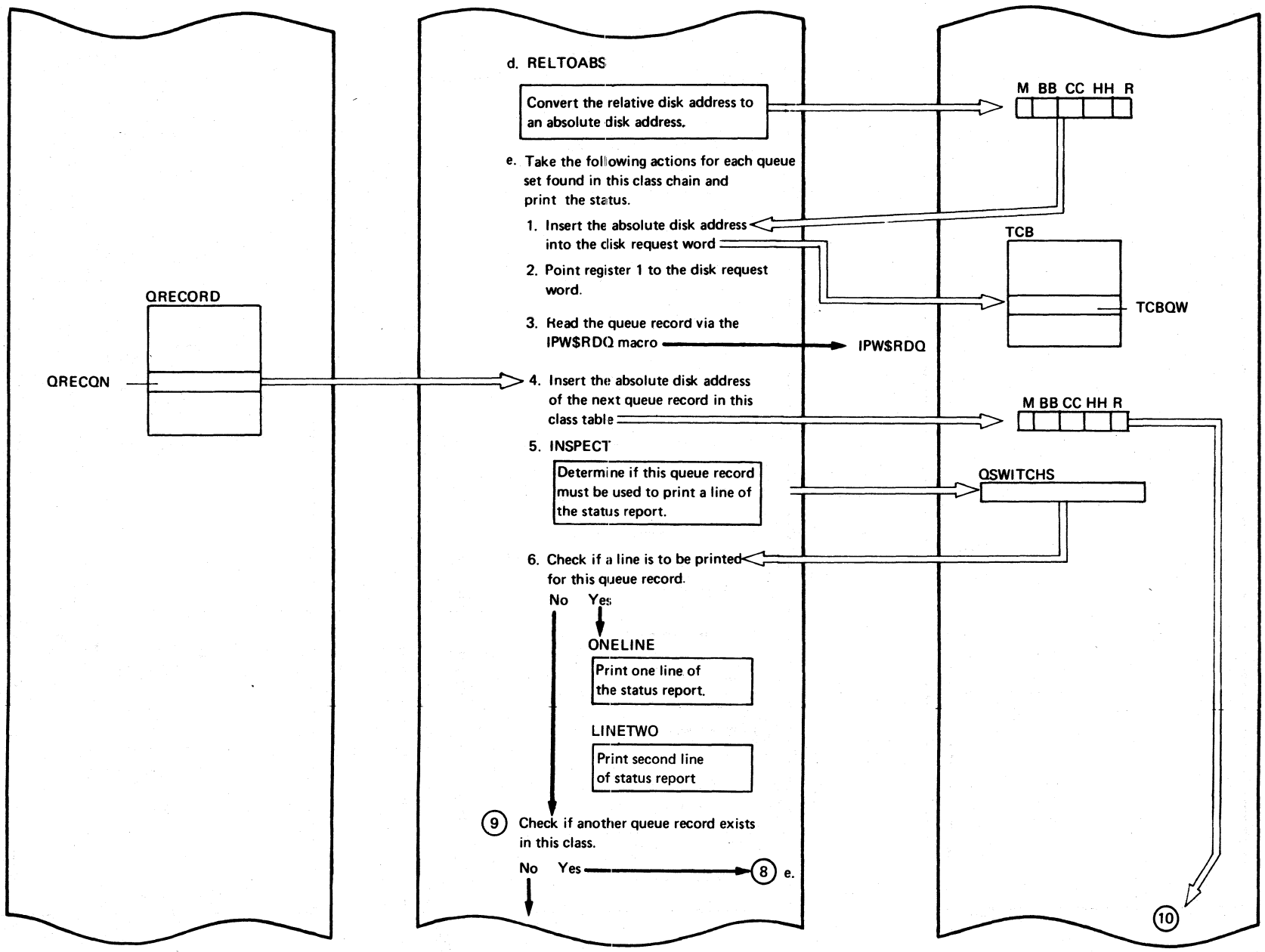

SWRDWR
On or Off
TCB
TCBTT
QLOGUNIT
CPEA↑
(A) from previous page
g. Check if there is nothing to display and the task is not being cancelled.
No Yes → Insert the message applying to the appropriate queue for the MSG subroutine
Indicate the message must not be squeezed
Call MSG → MSG
(5) Return to step (4) a. if there are more queues to be handled.
(6) Check if the print status report task is canceled by the operator.
No Yes → Build message text indicating this
Indicate message must not be squeezed
Call MSG → MSG
(7) Check if a programmer LUB was assigned to a printer.
No Yes → Unassign the LUB via the IPW$ULP macro → IPW$ULP
(8) Check if an ECB is supplied.
No Yes → Post processing complete
(9) Return to POWER/VS task selection
(10) SUBROUTS
Subroutines for the IPW$$PS phase.
DISPAREA
READER QUEUE
or
LIST
or
PUNCH
NOTHING TO DISPLAY
SWNOSQZ
1
CPFINISH
1
POWER/VS task selection.

Extended Description	Include Segment	Call Subroutine	Chart
(1) The print queue status report program is a reenterable program coded in PLS. The PLS/II compiler generates a GETMAIN assembler macro to obtain the dynamic storage required for variables.			
(2) The status report is printed on a local printer, a remote printer, or SYSLOG depending on the command entered. The I/O is done a. for a local printer via an EXCP macro b. for the system LOG via an IPW$WTO macro c. for a remote terminal via an IPW$WTM macro. QARGS was set up in the DSPLSTAT segment (Chart CP45) of the PDISPLAY processor in the IPW$$CP phase and copied into the TCBDW field in the task control block created for use by this program. In IPW$$PS, QARGS is defined upon TCBDW.		MSG	CQ8
(4) a.	QSCAN	MSG	CQ8
f.			CQ3
g.		MSG	CQ8
(6) Registers 0, 1, 2, and 3 are used to unassign the programmer LUB. Register 1 is used to address the PIB pointer of the POWER/VS partition via the COMREG. It is then used to point to the POWER/VS PIB. Register 0 is set to zero to indicate the unassign function. Register 3 points to six bytes before the LUB index in QARGS.		MSG	CQ8
(9) The PLS/II compiler generates a FREEMAIN macro to release the acquired dynamic storage.			
(10)	SUBROUTS		MSG CQ8 BINTODEC CQ10

Included by MAINLINE, Chart CQ2 LEVEL 2 Chart CQ3

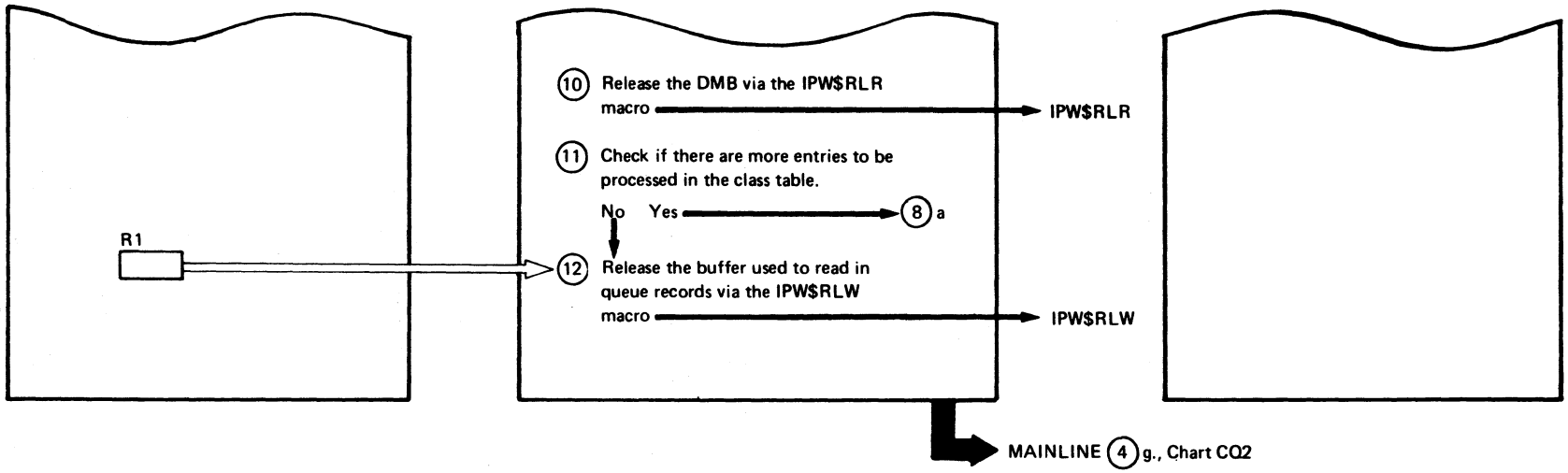


continued on next page



continued on next page

Page of SY33-8577-1, Revised November 24, 1977, By TNL SN33-9241



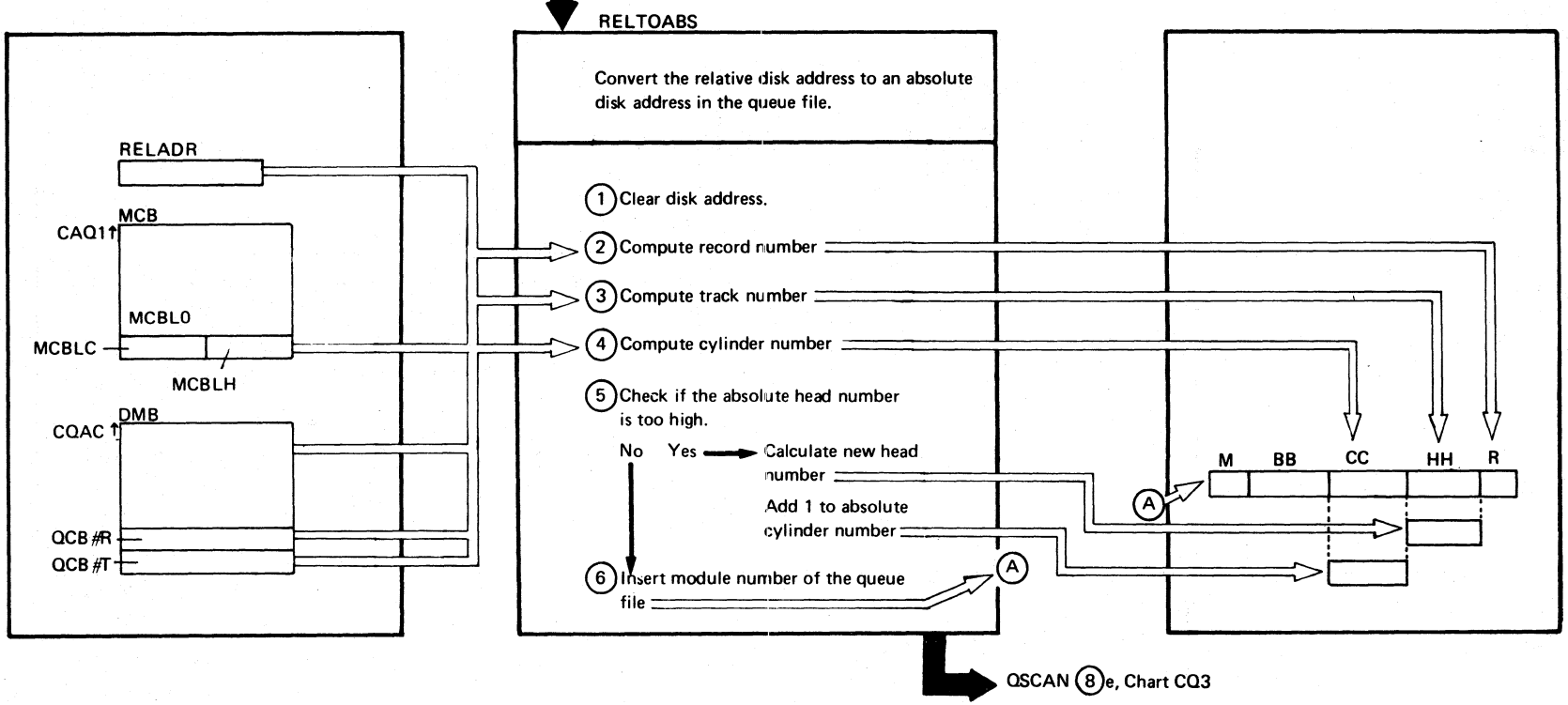
Extended Description	Include Segment	Call Subroutine	Chart
(2) The displacement of the master class table in the DMB is added to register 15 (label QCCT in the DMB). Register 15 can be used after Step (2)			
(6) Register 0 contains the real address of the buffer obtained. Register 1 contains the virtual address of the buffer obtained.			
(8) a. All 37 entries of a class table are handled.			
d.	RELTOABS		CQ4
e. Queue sets belonging to a class entry are chained via the QRECQN field in QRECORD. QRECORD contains the absolute disk address of the first record of the next queue set. For the last queue record this field contains binary zeroes.			
e.5.	INSPECT		CQ5
e.6.	ONELINE		CQ6

Page of SY33-8577-1, Revised November 24, 1977, By TNL SN33-9241

Included by QSCAN, Chart CQ3

LEVEL 3

Chart CQ4



Extended Description	Include Segment	Call Subroutine	Chart
(2) The record number is computed by dividing the relative disk address by the number of records on a track and adding 1. The 1 is added because the first record is the master record. (3) The absolute head number (HH) equals the relative head number plus the starting head number (MCBLH). (4) The absolute cylinder number (CC) equals the relative cylinder number plus the starting cylinder number (MCBLC).			

Included by QSCAN, Chart CQ3

LEVEL 3

Chart CQ5

INSPECT (Part 1 of 3)

Determine if the queue record just read is to be used to print a line of the status report.

(1) Assume no status to be printed for this queue record → QSWITCHS: 0

QJOBNO

(2) Check if the status is requested for a specific job number.

No Yes

QRECNO

Check if this queue record corresponds to the specified job number.

Yes No → (3)

Indicate the status is to be printed for this queue record → QSWNO: 1

QIDBIN

(3) Check if the status is requested for a specific binary RJE user-ID.

No Yes

QRECFJ

QRECTJ

Check if this queue record corresponds to the specified binary RJE user-ID.

Yes No → (4)

Indicate the status is to be printed for for this queue record → QSWIDBIN: 1

QIDDEC: 00

(4) Check if the status is requested for a specific local user-ID.

No Yes

QRECFJ

QRECTJ

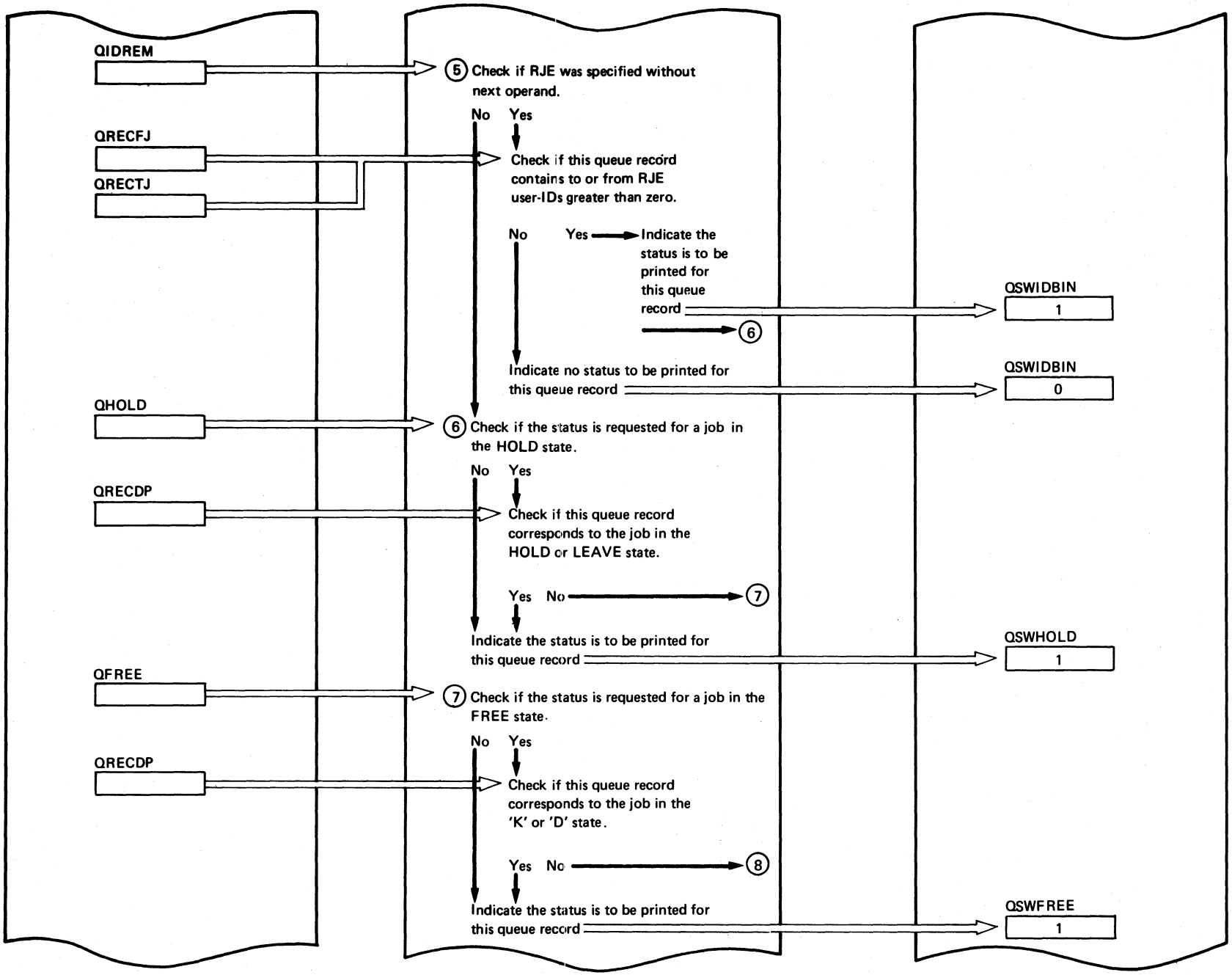
Check if this queue record corresponds to the specified local user-ID.

Yes No → (5)

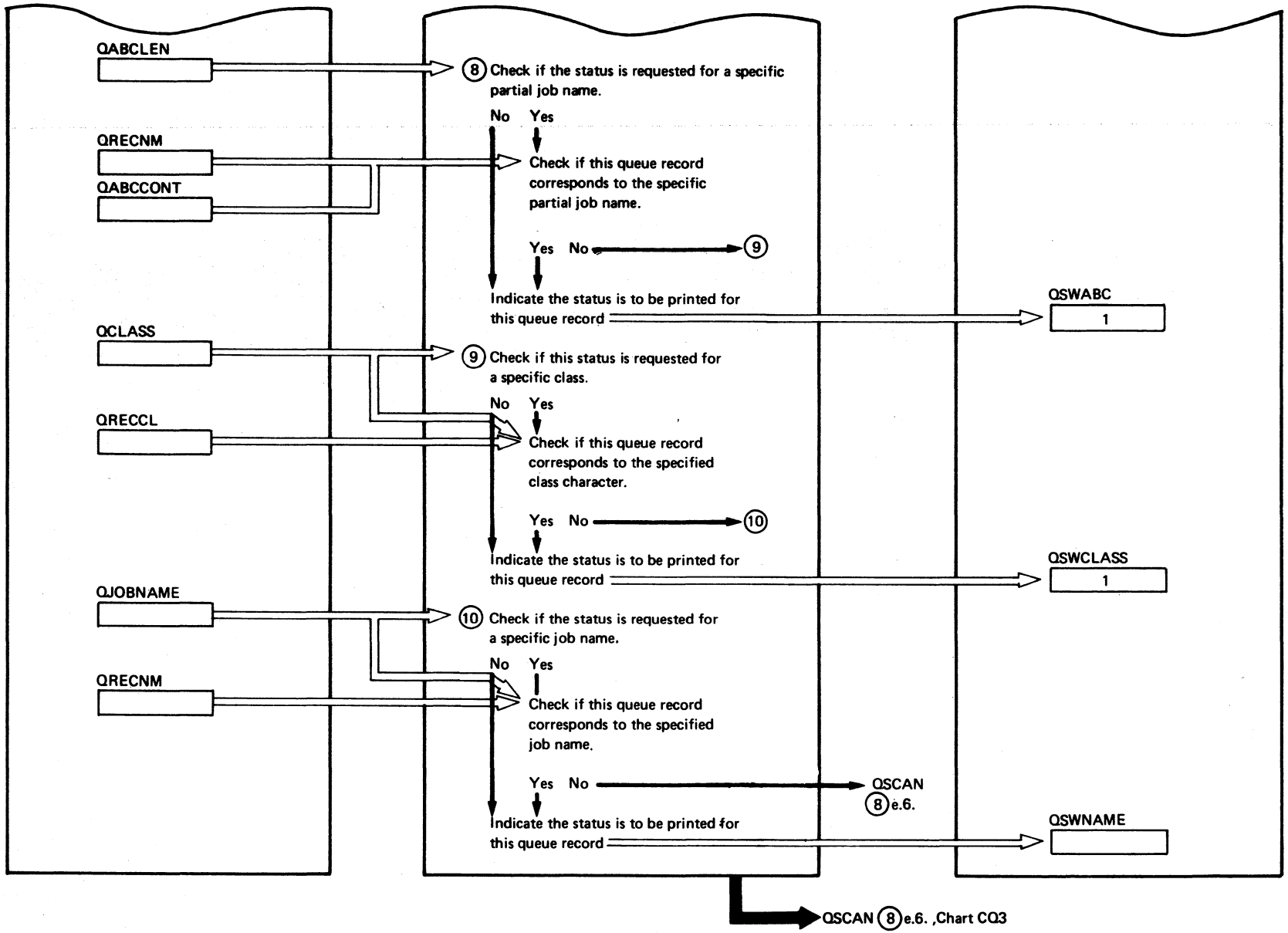
Indicate the status is to be printed for this queue record → QSWIDDEC: 1

continued on next page

Chart CQ5



continued on next page


QABCLEN
8 Check if the status is requested for a specific partial job name.
No Yes
QRECNM
QABCCONT
Check if this queue record corresponds to the specific partial job name.
Yes No 9
Indicate the status is to be printed for this queue record
QSWABC
1
QCLASS
9 Check if this status is requested for a specific class.
No Yes
QRECCL
Check if this queue record corresponds to the specified class character.
Yes No 10
Indicate the status is to be printed for this queue record
QSWCLASS
1
QJOBNAME
10 Check if the status is requested for a specific job name.
No Yes
QRECNM
Check if this queue record corresponds to the specified job name.
Yes No
QSCAN 8 e.6.
Indicate the status is to be printed for this queue record
QSWNAME
QSCAN 8 e.6. ,Chart CQ3

Included by QSCAN, Chart CQ3

LEVEL 3

Chart CQ6

ONELINE (Part 1 of 4)

Set up the contents of a status report line from information found in an approved queue record and call the subroutine MSG to print it

DISPAREA
QUEUE

QRECNM

QRECNO

QRECPY

QRECXS

QRECDP

(1) Indicate there is something to display → SWRDWR: 1

(2) Check if a header is still to be printed.
No / Yes → Call the MSG subroutine to print the header → MSG

(3) Indicate no suppression of leading zeroes → SWSUPPR: 0

(4) Insert the job name in the output line → DISPAREA: DJOBNAME

(5) Set up the input field with the job number for the BINTODEC subroutine → CONVBIN

(6) Call the BINTODEC subroutine to convert the job number to printable decimal form → BINTODEC → CONVDEC

(7) Insert the converted job number in the output line → DISPAREA: DJOBNO

(8) Insert the priority in the output line → DISPAREA: DPRI

(9) Check if the queue entry is being processed by a reader or a writer task.
Yes / No

Insert the disposition in the output line → DISPAREA

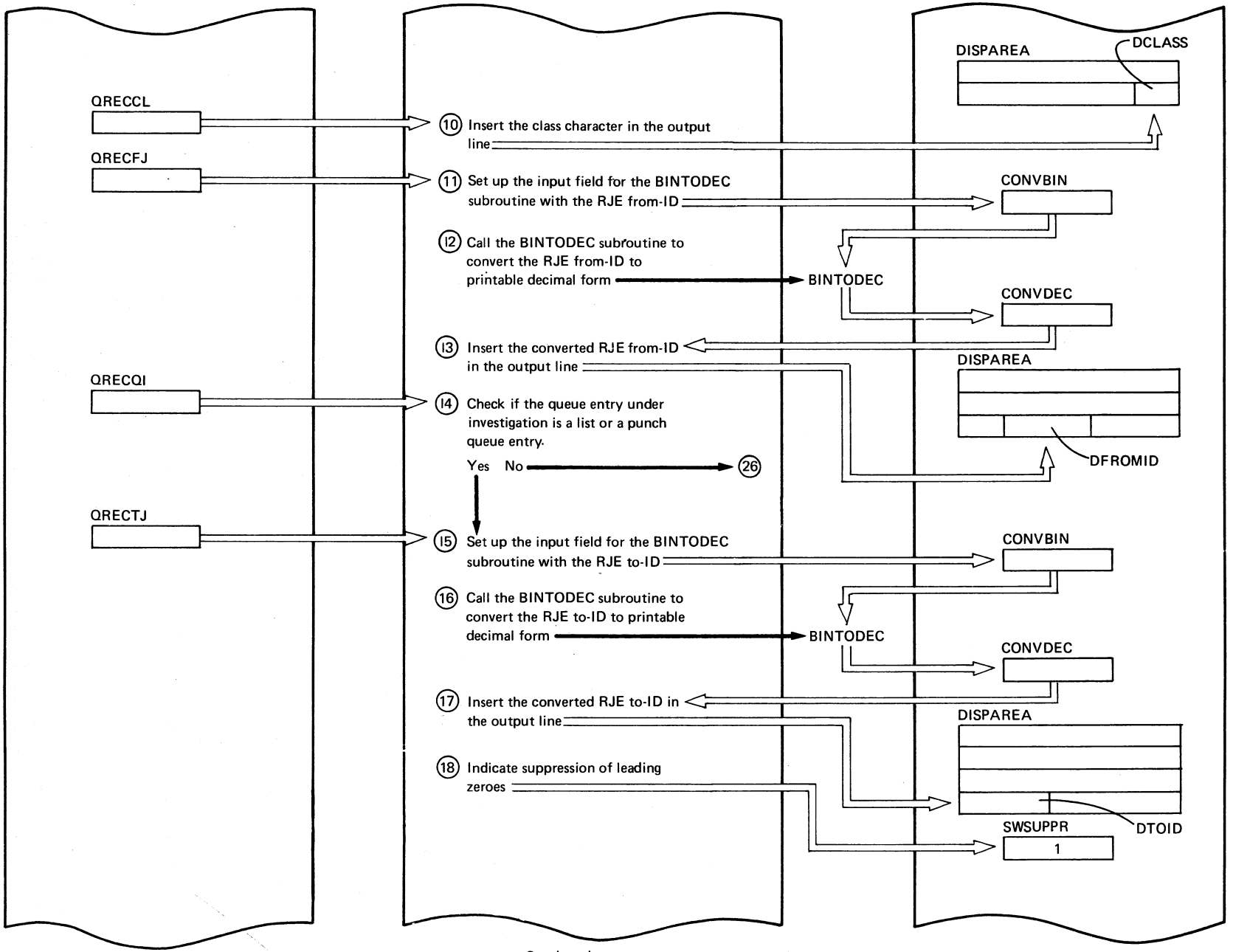
a. Identify the entry being processed → DISPAREA: * DDISP

b. OTHREC

Locate the writer task TCB and calculate the number of copies and pages, or cards still to be written

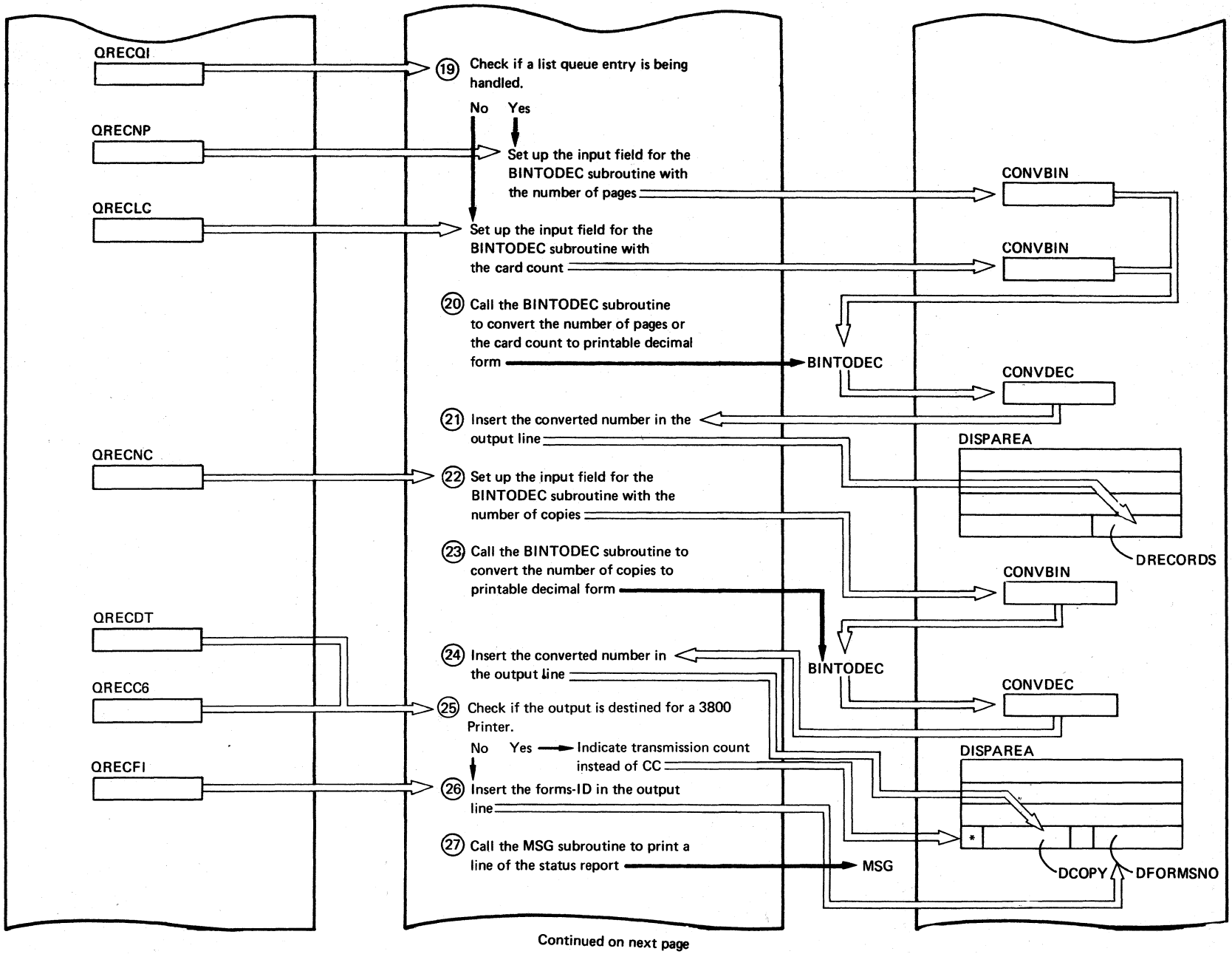
Continued on next page

Chart CQ6



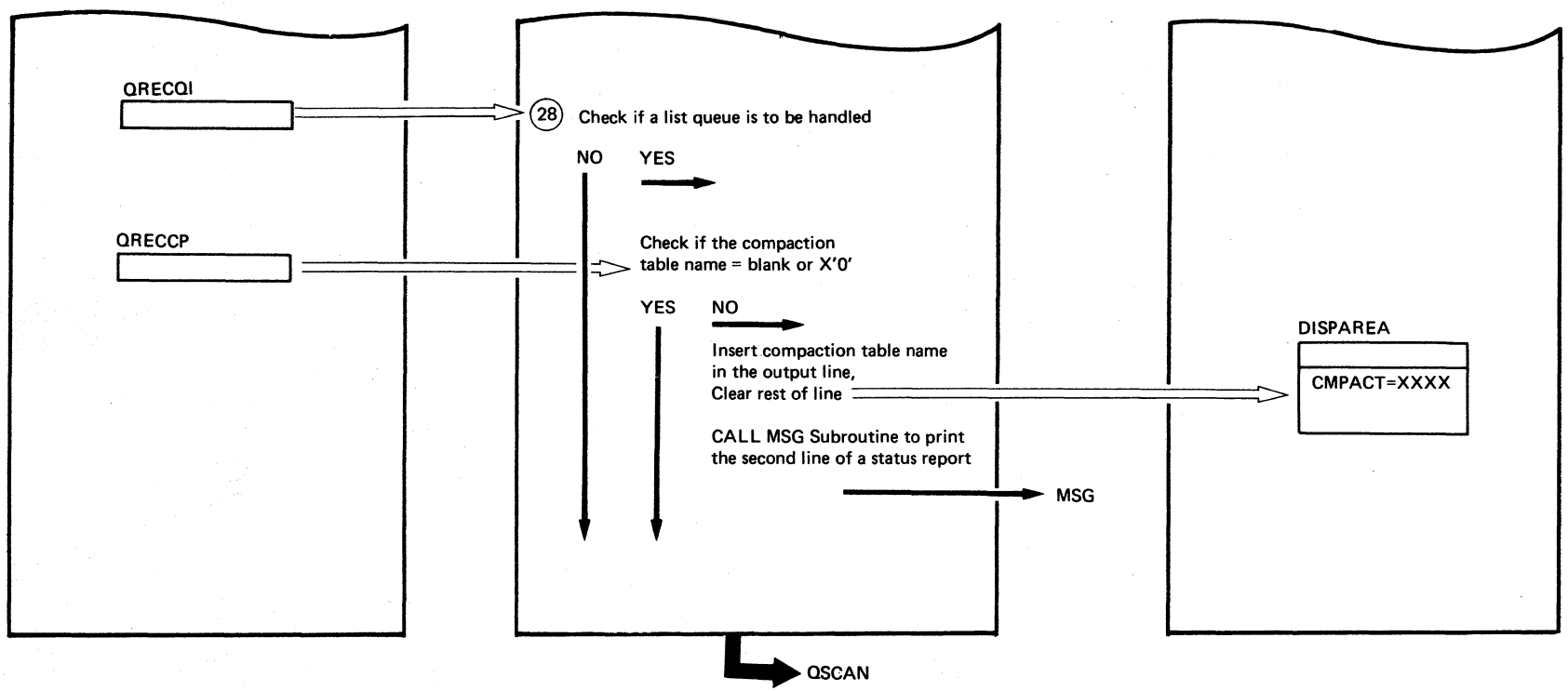
Continued on next page

Page of SY33-8577-1, Revised November 24, 1977, By TNL SN33-9241



Continued on next page

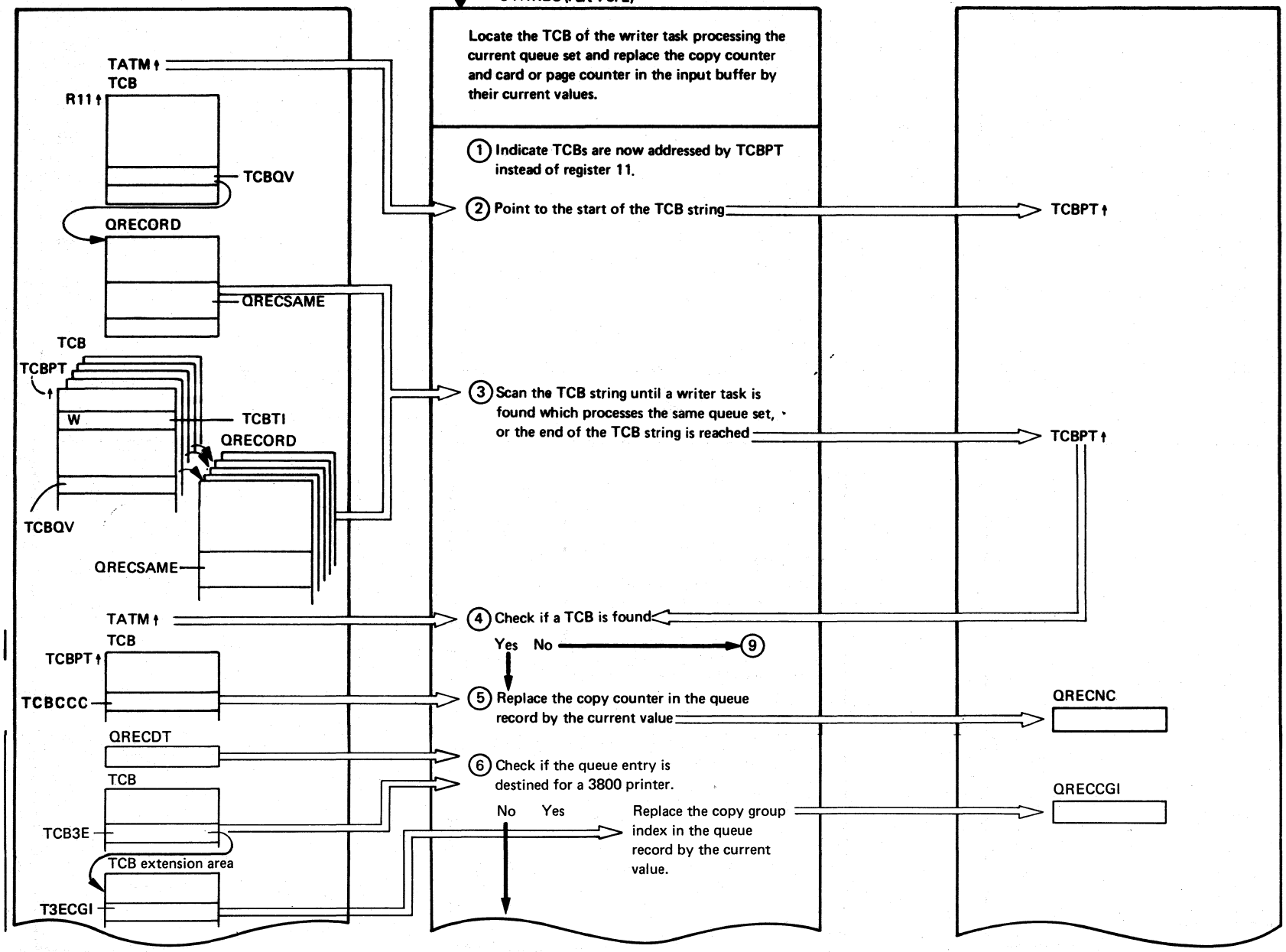
Page of SY33-8577-1, Revised November 24, 1977, By TNL SN33-9241



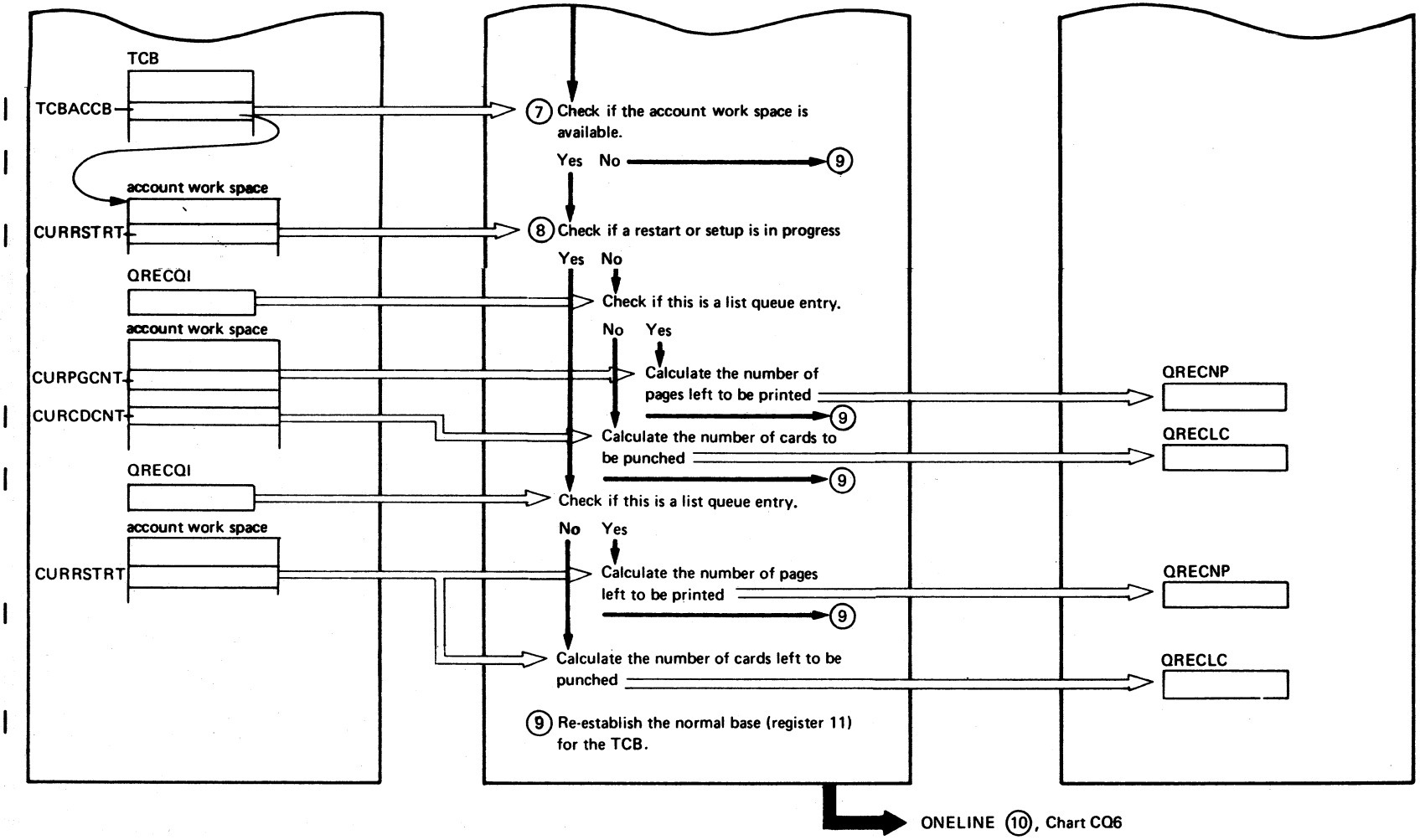
Extended Description	Include Segment	Call Subroutine	Chart
(2)		MSG	PS8
(6)		BINTODEC	PS9
(9) b	OTHREC		PS7
(12)		BINTODEC	PS9
(14)		MSG	PS8
(16)		BINTODEC	PS9
(20)		BINTODEC	PS9
(23)		BINTODEC	PS9
(26)		MSG	PS8

Page of SY33-8577-1, Revised November 24, 1977, By TNL SN33-9241

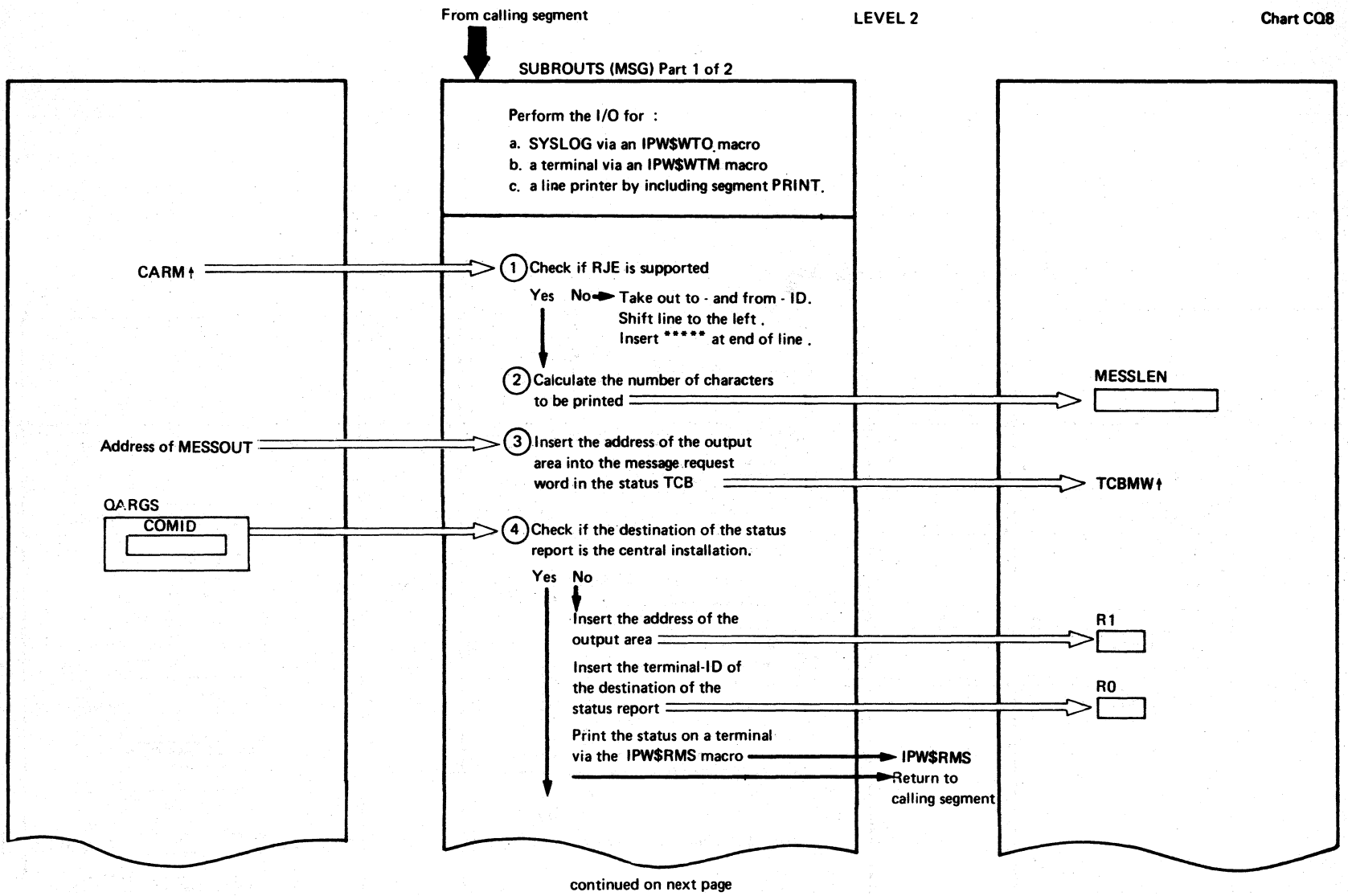
Included by ONELINE, Chart CQ6



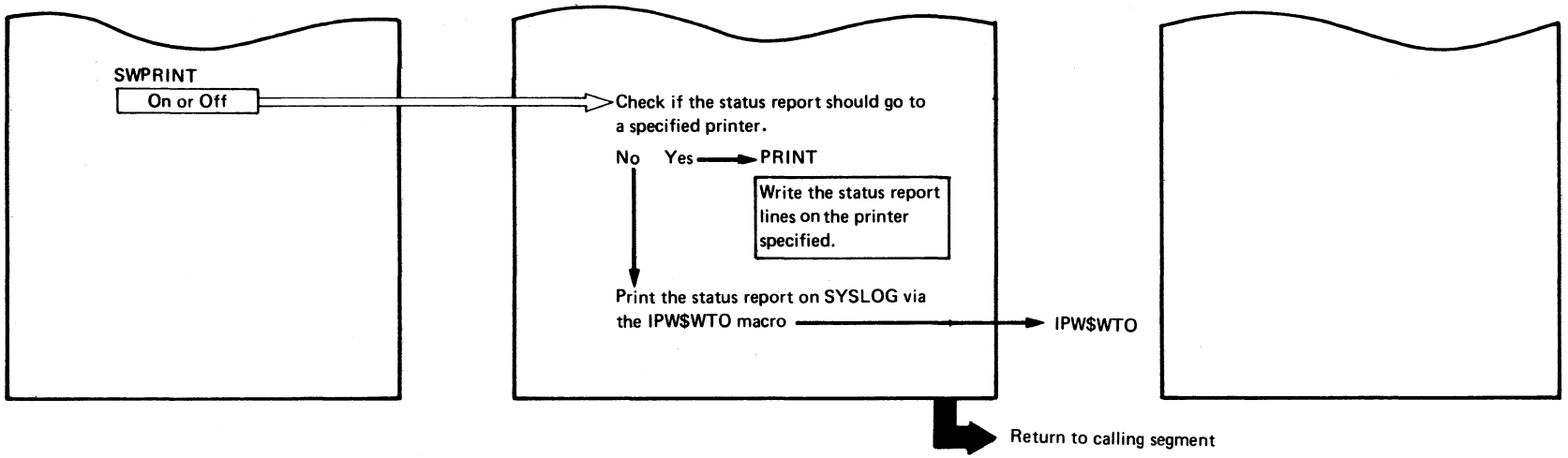
continued on next page



Page of SY33-8577-1, Revised November 24, 1977, By TNL SN33-9241



continued on next page



Extended Description	Include Segment	Call Subroutine	Chart
(1) The format of a status report header in an RJE system is : ŧŧ LIST QUEUE P D C FROM TO PAGES CC FORM The format of a status report header in a non-RJE system is : ŧŧ LIST QUEUE P D C PAGES CC FORM ***** (3) and (4) Registers 0, 1, 2, and 3 are restricted in these steps	PRINT		CQ9

Included by SUBROUTS (MSG), Chart CQ8 LEVEL 3 Chart CQ9

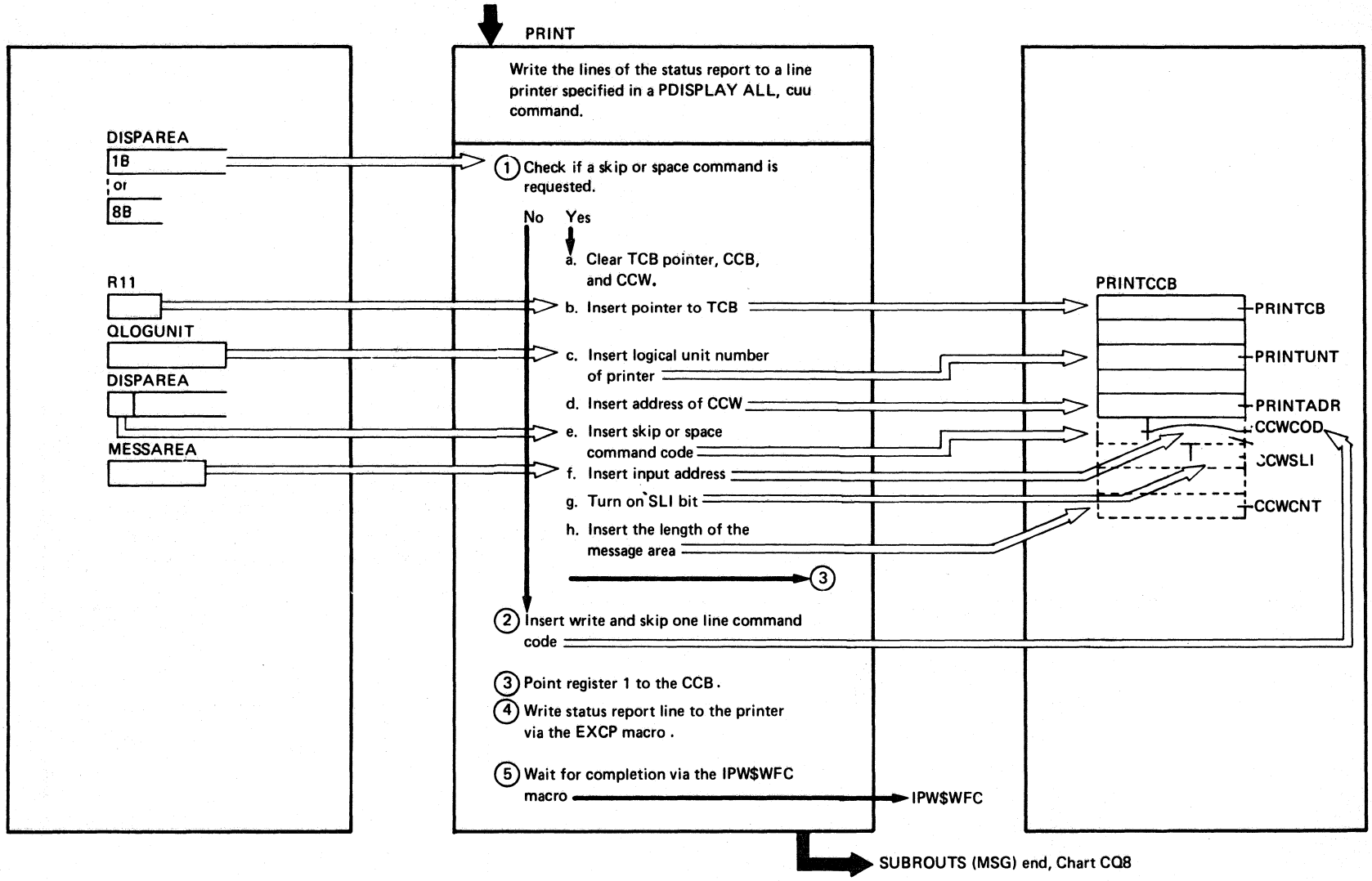


Chart CQ9

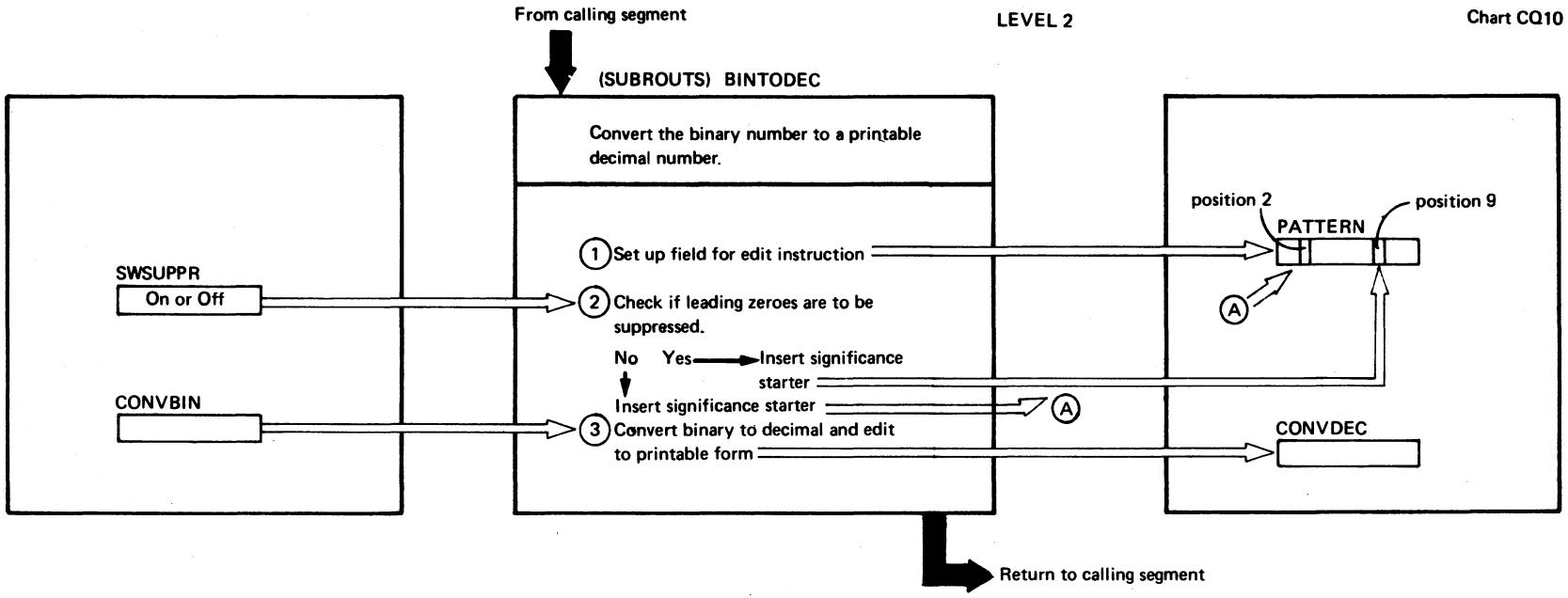

From calling segment
LEVEL 2
Chart CQ10
(SUBROUTS) BINTODEC
Convert the binary number to a printable decimal number.
position 2
position 9
PATTERN
SWSUPPR
On or Off
1 Set up field for edit instruction
2 Check if leading zeroes are to be suppressed.
No
Yes
Insert significance starter
A
CONVBIN
Insert significance starter
3 Convert binary to decimal and edit to printable form
CONVDEC
Return to calling segment

Chart CQ11: IPW$$PS - LINETWO (2 Parts)

Chart CQ11

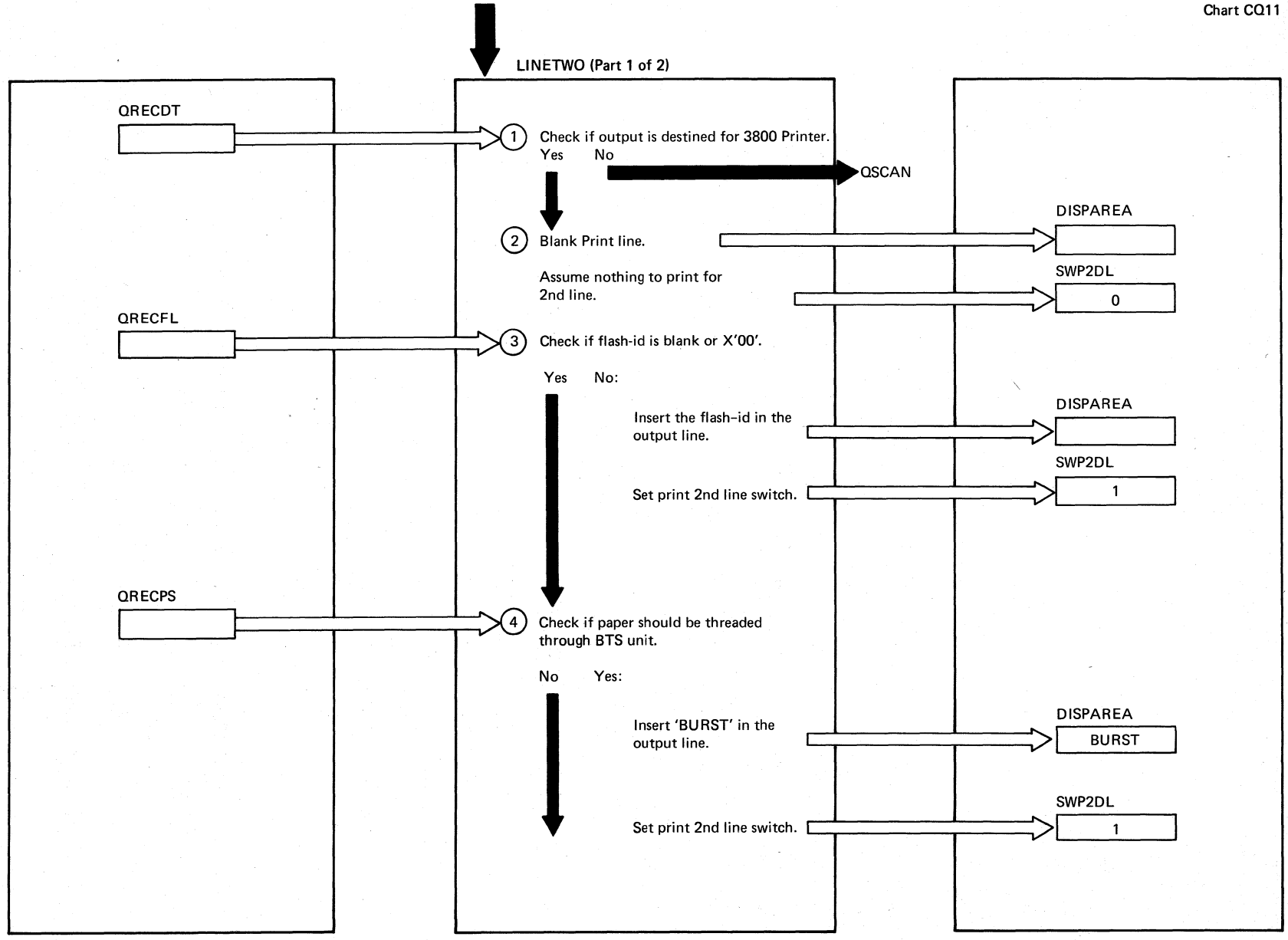
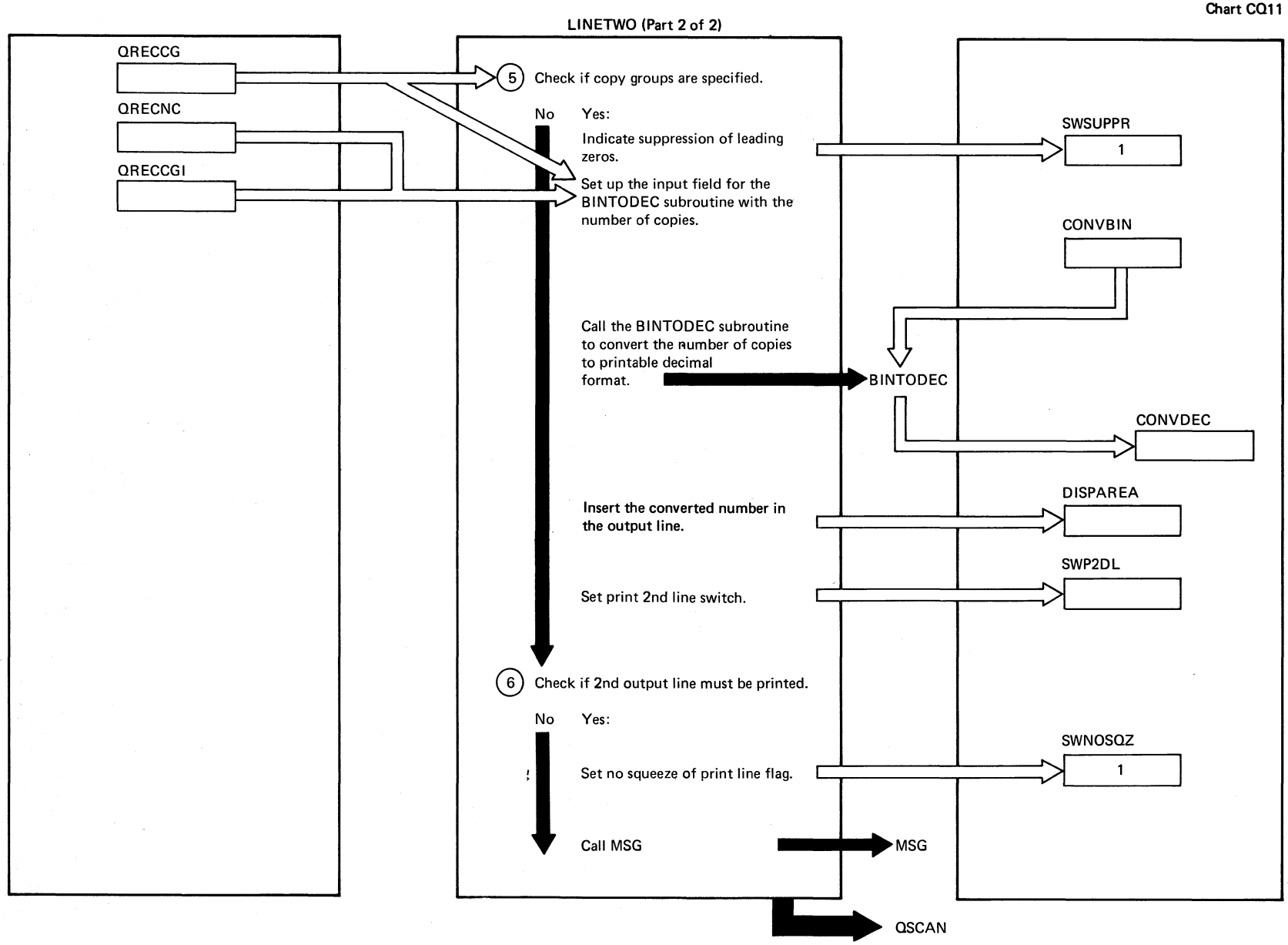


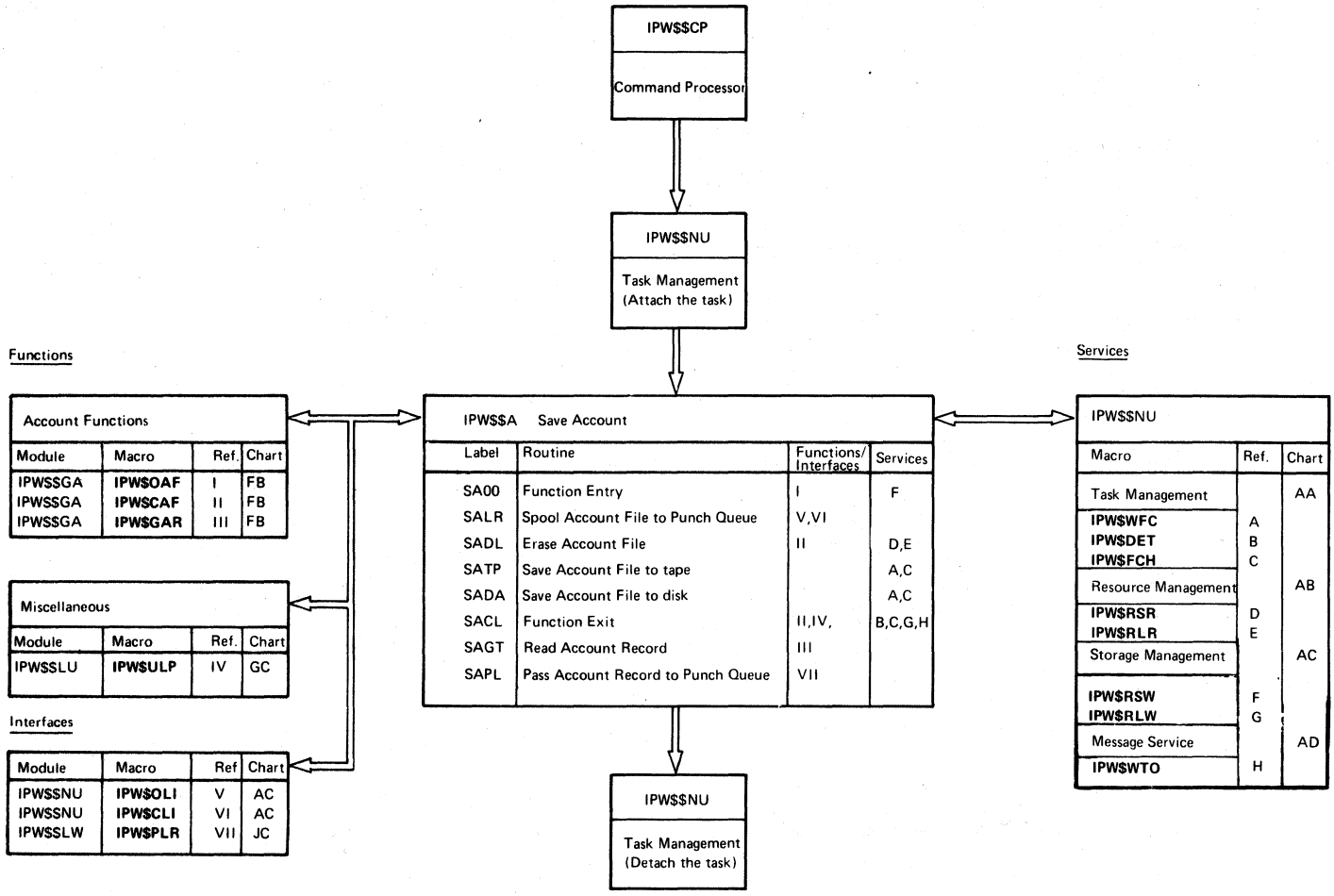
Chart CQ11



Page of SY33-8571-1, Added November 24, 1977, By TNL SN33-9241

CHART CR: IPW$$SA - SAVE ACCOUNT (14 PARTS)

Chart CR00: IPW$$SA - Save Account, General Flow and Macro Calls



Labels	Chart CR01: IPW$$SA - Save Account		Modified Data Fields	Reg. Usage	Calls
SASD	The first 16 bytes constitute the section description: 'CASA V7M0				
	On entry, the following register contents are relevant: 1: Device type and LUB unit output file			R1	
	The following registers are used by IPW$$SA: 8: Account file CCB address 9: SACS base register 10: Address of POWER/VS permanent area 11: Address of TCB 12: Asynchronous address register 13: Address of task save area 14: Link register 15: Base register		 IPW$DPA IPW$DTC	 R8 R9 R10 R11 R12 R13 R14 R15	
SA00	The entry parameter register 1 is saved in register 5.			R5	
	A buffer is reserved to contain the DTFPH for tape or disk output when necessary.				IPW$$RSW Chart AC
	If the Save Device specified is DEL, a branch is made to erase the account file..... >	SADL			
	The account file is opened in GET mode.				IPW$OAF Chart FB
	The address of message 1Q78I is loaded in register 4.			R4	
	The address of close exit SC16 is loaded in link register 14.			R14	
	If open failure, branch..... >	SA04			
	Link to read an account record..... >	SAGT		R2	
	If the file is not empty (zero condition code on return signals end of file), branch..... >	SA08			
	Otherwise, the address of message 1Q83I is loaded in register 4, the address of close exit SC12 is loaded in link register 14.			R4 R14	
	Cancel open tape is indicated in the TCB.		TCTT(IPW$DTC)		

Labels	Chart CR02: IPW$$SA - Account Save		Modified Data Fields	Reg. Usage	Calls
SA04	If device is not a tape, exit..... >	(R14)			
	Otherwise, the LUB index in register 5 is sto red in a dummy CCB, and branch to exit..... >	SC04	PHLU	R8	
SA08	If the account file has to be written to tape, a branch is made..... >	SATP			
	If it has to be saved on disk, a branch is made..... >	SADA			
	Otherwise, the account file is spooled via the logical reader to punch queue.				
	This routine spools the account file via the logical reader to the punch queue.				
SALR	Register 5 is loaded with the address of the IPW$$SA output area.			R5	
	The logical reader interface is opened.				IPW$OLI Chart AC
	Register 3, to be used as a first time switch, is initialized to one.			R3	
	Branch to..... >	LR08			
	(Note: First record is already present. Read in SA00 routine.)				
LR04	A link is made to read an account record..... >	SAGT		R2	
	If EOF is detected (account record length zero in register 0), a branch is made to close the logical reader interface..... >	LR28			
LR08	This routine updates account record length (register 0) and address (register 1) pointers to ignore the DASD record control field part, and passes the updated pointers back in register 6 (true record length) and register 7 (true virtual record address).				
	Account record length is copied to register 6.			R6	
	Account record address is copied to register 7.			R7	
	Using register 1 as a work register, 9 is subtracted from the record length in total, to obtain internal (machine instruction format) length.			R1 R4	

Labels	Chart CR03: IPW$$SA - Account Save		Modified Data Fields	Reg. Usage	Calls
	The account record address is incremented by 8 to point past the DASD control field.			R7	
	The record ID is moved to the output area.		RCID		
	The card sequence field is initialized to (packed decimal) zero. The data length-1 (70 bytes) is loaded in register 1 (machine length).		SQCT	R1	
	If more than one card is required to contain the current account record a branch is made to..... >	LR12			
	If only one card is required to contain the current account record: • The I/O area is blanked out. • The account record, pointed to by register 7, is moved to the I/O area, pointed to by register 5, using the record length in register 6. • Record length is set to zero and made negative to signal end of account record.		ACRD ACRD	R6	
LR12	This routine processes an account record that does not fit in one card.				
	The internal (machine instruction format) length of the 71-bytes account data field that will fit on the current card is loaded in register 1.			R1	
	If the residual account data size is greater than 71 bytes, a branch is made to move 71 bytes of account data..... >	LR16			
	Otherwise, register 1 is reset to the actual residual account data size.			R1	
LR16	The I/O area is blanked, except the first and the last three bytes (record identifier plus sequence number).		ACRD		
	Using the data length in register 1, account data is moved to the I/O area.		DATA		
	Account data pointer register 7 is incremented by the length of the data field just moved to point to the data to be moved next.			R7	
	Residual data length in register 6 is decremented by the length of the data just moved.			R6	

Labels	Chart CR04: IPW$$SA - Save Account		Modified Data Fields	Reg. Usage	Calls
	The card sequence number is incremented by one, and unpacked into the I/O area.		SQCT SQNO		
LR20	Branch if not first spool record... >	LR24		R3	
	Otherwise, provide logical reader with dummy read command (2nd hopper, MFCU 5425).		TCCC(IPW$DTC)		
	Set register 1 to one as minimum record length.			R1	
	A link is made to pass this dummy record..... >	SAPL+4			
	(The address of the output card image is loaded in register 0 by this subroutine SAPL.)				
	On return, the dummy read command is cleared.		TCCC(IPW$DTC)		
	The address of the queue record is loaded in register 2.			R2	
	The address of the DMB is loaded in register 4.			R4	
	The queue record is changed: • Job name PACCOUNT is moved to the queue record. • Record identifier is set to Punch. • Class indicator is set to Punch. • Priority to set to 1. • Disposition is set to HOLD. • The default number of separator cards is moved to the queue record.		QRNM(IPW$DQR) QRQI(IPW$DQR) QRCL(IPW$DQR) PRPY(IPW$DQR) QRCL(IPW$DQR) QRSP(IPW$DQR)		
LR24	A link is made to pass the first and following card image records to the logical reader..... >	SAPL			
	If current account record has been spooled completely, a branch is made to read the next account record.... >	LR04			
	Otherwise, a branch is made to prepare the next card for the current account record..... >	LR12			
LR28	This is the normal exit address (end of account file) for logical reader spooling.				
	The address of message 1Q79I is loaded in register 4.			R4	
	The output area is blanked out.		ACRD		
	'/&' is moved to the output area.		ACRD		

Labels	Chart CR05: IPW$$SA - Save Account		Modified Data Fields	Reg. Usage	Calls
	Card count in queue record is updated with one for this EOJ card, using register 3 as work register.			R3	
	A link is made to pass the EOJ record to the logical reader..... >	SAPL		R2	
	Register 1 is set to zero to simulate EOF condition of the physical reader.			R1	
	A link is made to pass the EOF condition to the logical reader.... >	SAP1		R2	
	The logical reader interface is closed.				IPW$CLI Chart AC
	Normal exit is taken..... >	SC08			
SADL	This routine is entered when the account file is to be erased.				
	The ACB is reserved.				IPW$RSR Chart AB
	A IPW$CAF ERASE is issued to erase the account file.				IPW$CAF Chart FB
	The address of message 1Q80I is loaded in register 4.				
	Exit..... >	SC16		R4	
SATP	This routine is entered if the account file is to be saved on tape.				
	The address of the DTFPH used for tape output control is loaded in register 8.			R8	
	The DTF buffer is initialized with the originally generated tape DTF (reuseability).				
	The DTFPH is prepared for tape output:				
	• Accept unrecoverable I/O error is set ON,		PHCM		
	• No-rewind option is set OFF		PHCS		
	• Rewind-unload option is set ON.		PHCS		
	If no file name has been specified in the PACCOUNT command, and therefore no file name has been posted in the TCB by the command processor, branch to process unlabeled tape..... >	TP04			
	Otherwise, copy the file name into the copied DTFPH in the DTF buffer.		PHNM		
	Branch to bypass setting DTF to NOLABEL..... >	TP08			

Labels	Chart CR06: IPW$$SA - Save Account		Modified Data Fields	Reg. Usage	Calls
TP04	The DTF type indicator (byte 20) is set to NOLABEL.		PHTT		
	The NOLABEL option is switched ON.		PHCS		
TP08	The device-dependent information, passed by the command processor in register 1 and saved in register 5, is now processed.				
	The format of this information is as follows:				
	Byte 0: Density Byte 1: Device type Bytes 2-3: LUB index				
	The density byte is stored in the TCB.		SADY(IPW$DTC)		
	If no density has been specified (density byte contents zero), branch to bypass updating the PUB entry... >	TP12			
	Otherwise, register 2 is loaded with the address of the PUB entry of the tape device concerned.			R2	
	With register 2 as address register, the PUB is now updated to reflect the tape density keyed in by the operator				
	Standard mode is reset.		PUBJCFLG		
	Standard mode is updated according to specified density.		PUBJCFLG		
	The specified density is moved to the PUB.		PUBOPTN		
	Set mode command is initialized.		PUBOPTN		
TP12	The LUB index (programmer logical unit number) is moved to the CCB.		PHLU		
	The address of the DTF is stored in the OPENR parameter field.		DTF		
	The tape DTF is opened, invoking transient $$BOPENR.				IPW$FCH Chart AA
	If unrecoverable I/O error is posted in the TCB (by Terminator routine IPW$$TR), branch to..... >	TP14			
	When not standard labeled tape processing, branch to..... >	TP16			
	Otherwise, test if file labeled information has been found and stored in the DTF. If so, bypass task cancel condition..... >	TP16			

Labels	Chart CR07: IPW$$SA - Save Account		Modified Data Fields	Reg Usage	Calls
TP14	Store task cancel condition in TCB.		TCTT(IPW$DTC)		
	Load the address of message 1Q60I into register 4.			R4	
	Branch to abnormal task termination..... >	SC04			
TP16	The tape DTF open indicator is set on		PHIS		
	Register 5, to be used as an output block counter, is set to zero.			R5	
	Branch..... >	TP24			
	(Note: First record is already present. Read in SA00 routine.)				
TP21	A link is made to read an account record..... >	SAGT			
	On EOF account file (record length in register 0 zero), branch..... >	TP28		R0	
TP24	Otherwise, the block count is incremented by one, and stored in the DTFPH.		PHBC	R5	
	Account record length which was saved in register 6 is copied into the tape write CCW.		TCCW		
	Account record address which was saved in register 7 is copied into the tape write CCW.				
	The account record is written to tape, with suppression of incorrect length.			R8	
	A IPW$WFC POWER/VS wait macro is issued to wait for I/O completion.			R1	IPW$WFC Chart AA
	If no unit exception (EOV) on tape is detected, a branch is made to process the next account record..... >	TP21			
	On EOV:				IPW$FCH Chart AA
	• FEOV switch is set on.		PHIS		
	• EOF switch is set off.		PHIS		
	• End-of-Volume is forced using fetch macro IPW$FCH for $$BCEOV1 transient phase and SVC 2 call.			R0,R1	IPW$FCH Chart AA
	• The alternate tape is opened.				
	• The block count is reset to zero.				
	• The next account record is processed..... >	TP21		R5	

Labels	Chart CR08: IPW$$SA - Save Account		Modified Data Fields	Reg. Usage	Calls
TP28	This is the normal (EOF) exit for tape processing.				
	The address of message 1Q79I is loaded in register 4.			R4	
	A branch is made to exit..... >	SACL			
SADA	This routine is entered if the account file is to be saved on DASD.				
	The address of the ACB is loaded in register 5. The address of the DASD DTFPH is loaded in register 8.			R5 R8	
	The DTF buffer is initialized with the DASD DTFPH (reuseability).				
	The following fields in the DASD channel program are now relocated, using register 1 as a work register:			R1	
	• Seek argument in seek CCW,		SASK		
	• Search argument in search CCW,		SASH		
	• Sector value argument in set sector CCW,		SASS		
	• Reset set sector command X'23'				
	• Sector value plus 1 argument in read sector CCW,		SARS		
	• Search CCW address in TIC CCW,		SATI		
	• Count field address in first write count key data CCW.		SAWC		
	The DTF address in register 8 is stored in the OPENR parameter field.		DTF		
	The DASD device type is copied from the ACB to the DTFPH.		PHDT		
	The file name for the DASD file is copied from the TCB, where it has been posted by the command processor, to the DTFPH.		PHNM		
	The DTFPH is set to accept unrecoverable I/O error.		PHCM		
	DASD CCWs are now chained:				
	Seek CCW to set sector CCW		SASK		
	Set sector CCW to search CCW		SASS		
	Search CCW to TIC CCW		SASH		
	TIC CCW to write count CCW		SATI		
	Write count CCW to write data CCW (command + date)		SAWC		
	Write data CCW to read sector CCW		SAWD		
	The DASD DTFPH is now opened.				IPW$FCH Chart AA

Labels	Chart CR09: IPW$$SA - Save Account		Modified Data Fields	Reg. Usage	Calls
	If unrecoverable I/O error was posted in the TCB, or if open was unsuccessful, register 4 is loaded with 1Q60I message address and a branch is made to..... >	SC12		R4	
	Otherwise, a test for RPS support is made.				
	If RPS is supported, a branch is made to continue..... >	DA04			
	Otherwise, the set sector CCW is overwritten with a TIC * + 8 CCW, and the last CCW (read sector) is unchained.		SASS SAWD		
DA04	Using register 3 as a work register, the extent capacity of the user extent is calculated:			R3	
	Low cylinder number is subtracted from high cylinder number.				
	Number of cylinders available is multiplied by the number of tracks per cylinder.				
	Track number of high limit is added.				
	Track number of low limit is subtracted.				
	One is added to obtain the number of tracks available in the user extent.				
	The number of tracks is multiplied by the track capacity to get the extent capacity.				
	Extent capacity is saved.		SAMC		
	Using register 3 as a work register, the size of the account file is calculated by subtracting residual account file capacity from total account file capacity.			R3	
	If the current account file size is greater than the user extent capacity, a branch is made to error exit..... >	DA28			
	Otherwise, the user's DASD count field is cleared and set to the extent lower limit.		SACF		
	The initial (begin) track number of the IJAFILE to be saved is loaded in register 3.			R3	

Labels	Chart CR10: IPW$$SA - Save Account		Modified Data Fields	Reg. Usage	Calls
	Using register 1 as a work register, the track number of the highest track of a cylinder is calculated and saved.		SAUH		
	Branch..... >	DA12			
	(Note: First record is already present. Read in SA00 routine.)				
DA08	A link is made to read an account record..... >	SAGT		R2	
	On EOF account file (zero record length in register 0)..... >	DA13			
DA12	Otherwise, record length is stored in count field and write data CCW.		SACF		
	Record address is stored in write data CCW.		SAWD		
	Seek and search arguments are copied from count field.		SASA		
DA13	If the track number in the account file search argument has not been changed, a branch is made to update user's DASD record number..... >	DA20		R6	
	Otherwise, a new track has been accessed on the account file and loaded in register 3 and as a consequence the track address of the user DASD file should be incremented too.			R3	
	If the upper head of the current cylinder in the user extent has not been reached yet, branch to update track number..... >	DA16			
	Otherwise, the cylinder number is loaded in register 1.			R1	
	The count field is cleared to set head number and record to zero.		SACF		
	Cylinder number is incremented by one, and stored back into the count field.		SACF	R1	
	Seek and search arguments are copied from the count field.				
	Branch to update record number..... >	DA20			
DA16	Using register 1 as a work register, the head number in the count field is incremented by one.		SACF	R1	
	The record number is set to zero.		SACF		

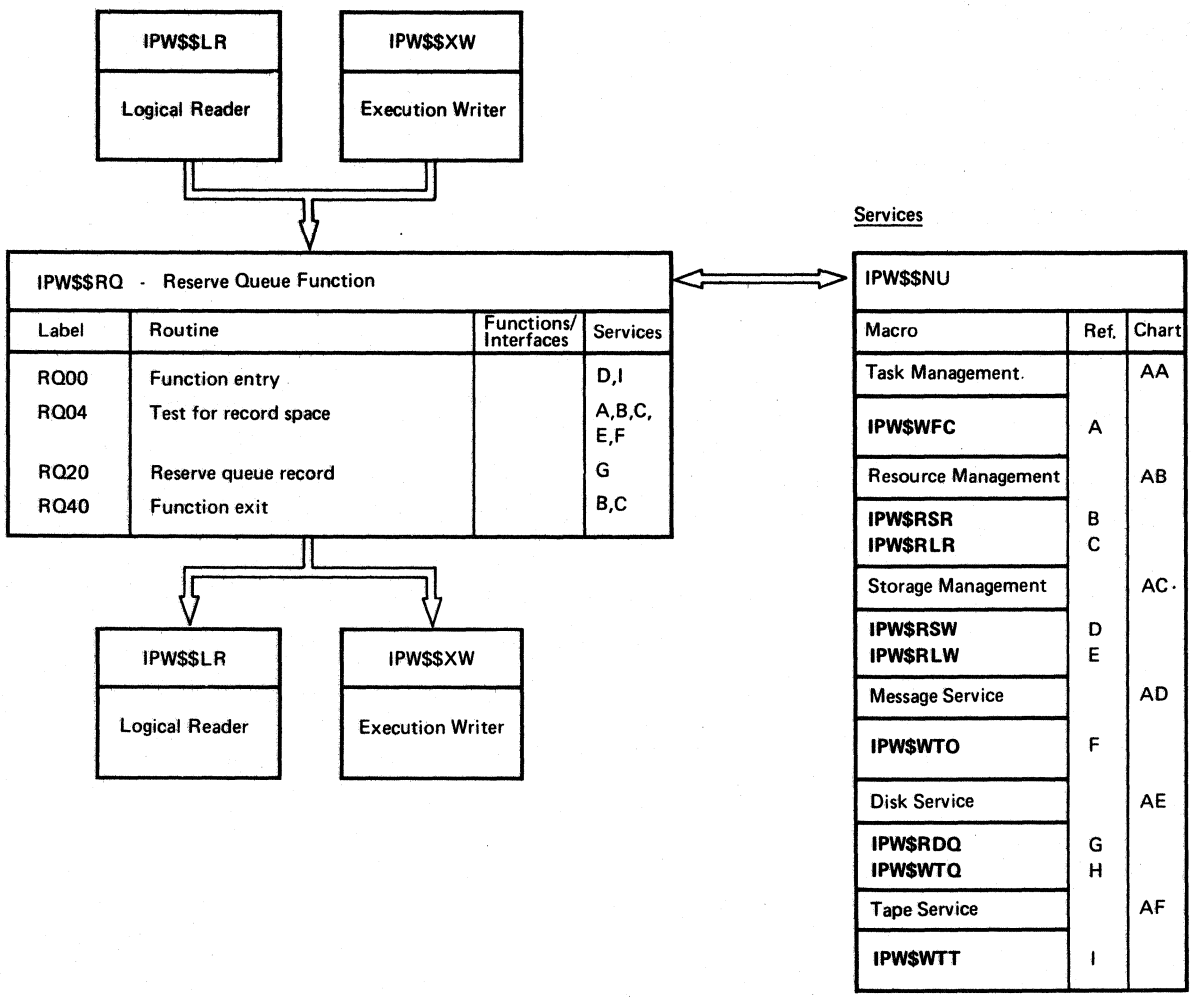
Labels	Chart CR11: IPW$$SA - Save Account		Modified Data Fields	Reg. Usage	Calls
	Seek and search arguments are copied from the count field.		SASA		
DA20	Using register 1 as a work register, record number in count field is incremented by one.		SACF		
	Next sector value is set for the set sector CCW. On EOF account file... >	DA24	SASE		
DA21	The account record is written.				
	A IPW$WFC macro is issued to wait for I/O completion.				IPW$WFC Chart AA
	On completion, branch to get the next account record..... >	DA08			
DA24	This is the DASD normal (EOF) exit routine.				
	The write data CCW is unchained.		SAWC		
	An EXCP is issued to write the EOF record.				
	An IPW$WFC macro is issued to wait for I/O completion.				IPW$WFC Chart AA
	On completion, the address of message 1Q79I is loaded in register 4.			R4	
	A branch is made to exit..... >	SC08			
DA28	This routine is a DASD error exit routine.				
	The address of message 1Q81I is loaded in register 4.			R4	
	The user file name is moved into message 1Q81I.		FLNM		
	Branch to error exit..... >	SC12			
SACL	This routine is the common close and exit routine. The output file is closed; its device is unassigned; if no errors have occurred, the account file is erased; a message is logged stating the action taken by IPW$$SA; the IPW$$SA task is detached.				

Labels	Chart CR12: IPW$$SA - Save Account		Modified Data Fields	Reg. Usage	Calls
	The (tape) output file is closed.				IPW$FCH Chart AA
SC04	This entry point to the exit routine is taken when the account file open procedure failed due to lack of work space, or unrecoverable I/O error, or an attempt was made to save an empty account file.				
	Register 1 is loaded with the POWER/VS PIB address.			R1	
	Register 3 is loaded with the CCB address of the output file and its device is unassigned.			R3	IPW$ULP Chart GC
	If the open tape file procedure was unsuccessful, branch to..... >	SC12			
	If the OPEN account file procedure was unsuccessful, a branch is made to message logging routine (1Q78I).... >	SC16			
SC08	Otherwise, the account file is closed and erased.				IPW$CAF Chart FB
	Branch to continue..... >	SC16			
SC12	An IPW$CAF KEEP is issued to keep the account file and reset and release the ACB, and its work space, if present.				IPW$CAF Chart FB
SC16	The address of the message to be logged, kept in register 4, is stored in the message request word, and the message is issued.		TCMW(IPW$DTC)		IPW$WTO Chart AD
	The DTFPH work space is released.				IPW$RLW Chart AC
SAEX	The TCB account track indicator is set inactive (X'40') and the task is detached.		TCAT(IPW$DTC)		IPW$DET Chart AA
SAGT	This subroutine reads an account record from the account file.				
	An account record is read.				IPW$GAR Chart FB
	IPW$GAR will return the account record length in register 0, and the account record address in register 1.			R0 R1	
	Register 0 is saved in register 6, and the condition code is set to provide a return code for the calling routine condition. Code = 0 indicates EOF otherwise, no EOF.			R6	
	Register 1 is saved in register 7.			R7	
	Control is passed back to the calling routine.			R2	

Labels	Chart CR13: IPW$$SA - Save Account	Modified Data Fields	Reg. Usage	Calls
SAPL	This routine passes a card image record via the logical reader to the punch queue.			
	The card image length is loaded in register 1.		R1	
	The address of the output card image is loaded in register 0.		R0	
SAP1	The card is passed to the logical reader.			IPW$PLR Chart JC
	Control is passed back to the calling routine.		R2	

CHART DA: IPW$$RQ - RESERVE QUEUE RECORD (3 PARTS)

Chart DA00: IPW$$RQ - Reserve Queue Record, General Flow and Macro Calls

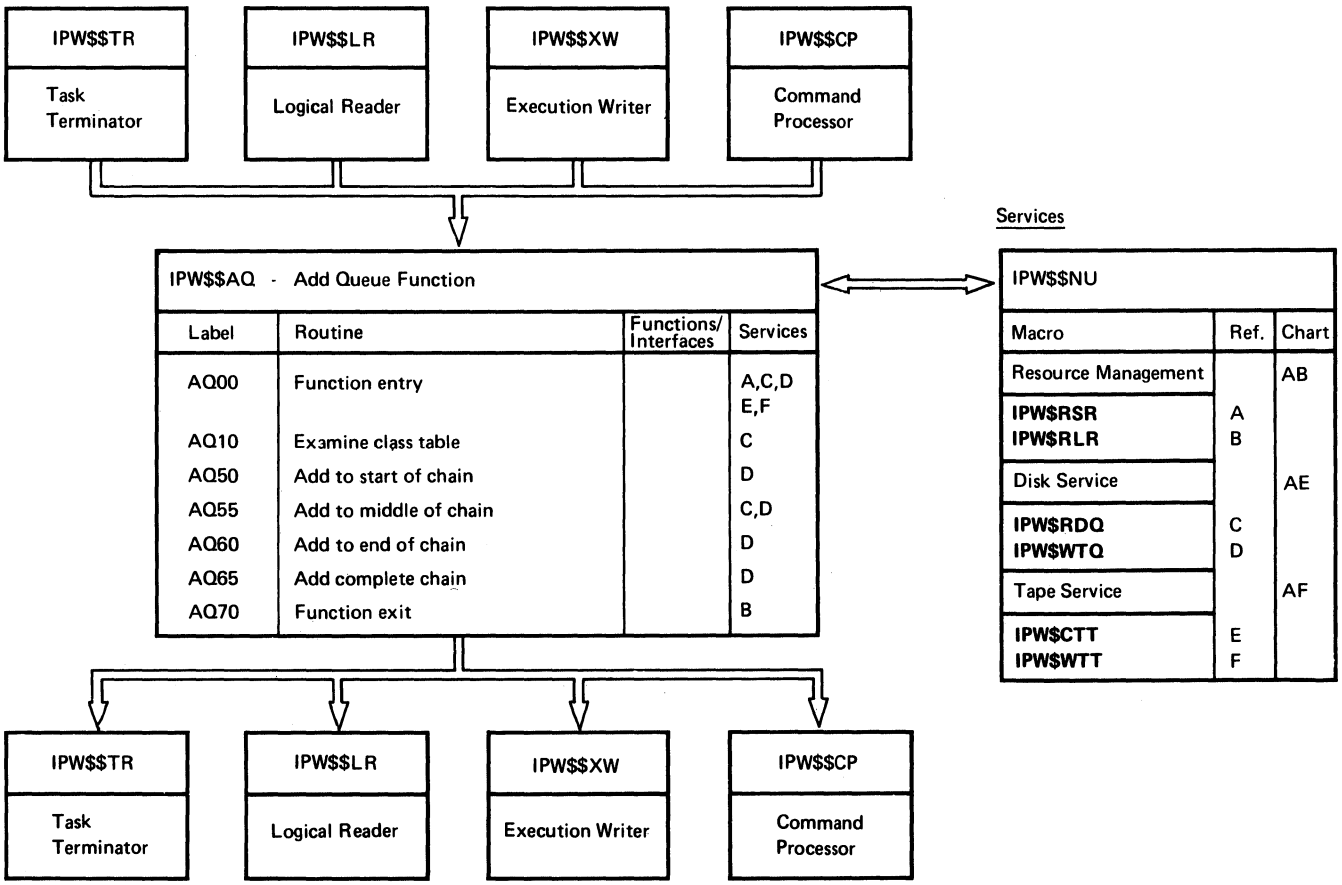


Labels	Chart DA01: IPW$$RQ - Reserve Queue Record		Modified Data Fields	Reg. Usage	Calls
RQSD	The first 16 bytes constitute the section descriptor: 'RQCS V7M0 '				
RQ00	Function Entry Save caller registers 14 through 9 inclusive. Reserve queue record space (if not available). Save addresses.		IPW$DSV TCQA(IPW$DTC) TCQV(IPW$DTC)		IPW$RSW Chart AC
RQ02	If no tape spooling, branch to..... >	RQ04			
	Write initial tapemark. Write the record to tape (queue record area). Exit..... >	 RQ44			IPW$CTT Chart AF IPW$WTT Chart AF
RQ04	Set addressability disk management block (DMB). Lock DMB. Examine Free List Pointer in Master Record			R6	IPW$RSR Chart AB
RQ05	If no queue record available: Issue message 1Q38I NO DASD SPACE AVAILABLE				IPW$WTO Chart AD
RQ10	Unpost ECB in DMB. Unlock DMB. Wait for posting of ECB in DMB. Lock DMB. Examine free list pointer. If zero, branch to retry..... >	 RQ10	QCEB(IPW$DQC)		IPW$RLR Chart AB IPW$WFC Chart AA IPW$RSR Chart AB
RQ20	Reserve Queue Record Read first free queue record from set Update first free queue record to next record. Update queue record field: set first-in-set switch set next-in-set pointer zero set forward chain pointer zero set previous chain pointer to queue record itself		TCQW(IPW$DTC) MRQF(IPW$DQC) QRFS(IPW$DQR) QRNS(IPW$DQR) QRQP(IPW$DQR) QRQN(IPW$DQR)		IPW$RDQ Chart AE

Labels	Chart DA02: IPW$$RQ - Reserve Queue Record		Modified Data Fields	Reg. Usage	Calls
	Update record counters in CAT (statistical information).				
	• decrement number of queue records available		NRQF(IPW$DPA)		
	• maximum number of records in use.		NRQM(IPW$DPA)		
RQ40	Function Exit				
	Write master record back to queue file.		QCMW(IPW$DQC)		IPW$WTQ Chart AE
	Unlock DMB.				IPW$RLR
	Set function track indicator to 0 (open for output) and return to caller.		TCTF(IPW$DTC)		Chart AB
RQ44	Restore registers and return to caller.			R14-R9	

CHART DB: IPW$$AQ - ADD QUEUE SET TO CHAIN (5 PARTS)

Chart DB00: IPW$$AQ - Add Queue Set to Chain, General Flow and Macro Calls



Labels	Chart DB01: IPW$$AQ - Add Queue Set to Chain		Modified Data Fields	Reg. Usage	Calls
AQSD	The first 16 bytes constitute the section descriptor:				
	'AQCS V7M0 '				
AQ00	Function Entry				
	Save caller registers 14 through 9 inclusive.		IPW$DSV		
	Set addressability for queue record.			R5	
	Set function track indicator A (add in progress).		TCFT(IPW$DTC)		
	If tape spooling:				
	• write trailer record to tape (queue record area)				IPW$WTT Chart AF
	• write two tapemarks, backspace one				IPW$CTT
	• exit..... >	AQ89			Chart AF
AQ06	Set addressability disk management block (DMB).			R6	
	Lock DMB.				IPW$RSR Chart AB
	If queue record area does not contain first-in-set:				
	Update record pointers				
	• set forward pointer to zero		QRQN(IPW$DQR)		
	• set next-in-set pointer to zero		QRNS(IPW$DQR)		
	Write the queue record back.				IPW$WTQ Chart AE
	Read first record back into the auxiliary queue area.		QCQW(IPW$DQC)		IPW$RDQ Chart AE
	Copy information in queue record area				
	• overwrite record pointers		QRNS(IPW$DQR)		
	• overwrite control seek address		TCQW(IPW$DTC)		
	• reset first-in-set switch		QRFS(IPW$DQR)		
AQ10	Examine Class Table				
AQ20	Address class table entry:				
	If pointers are zero add complete chain..... >	AQ65			
	Translate relative disk address of last in class pointer.		QCQW(IPW$DQC)		
	• read last-in-class record in auxiliary queue area				IPW$RDQ Chart AE
	• cneck priority for add to end of chain..... >	AQ60			

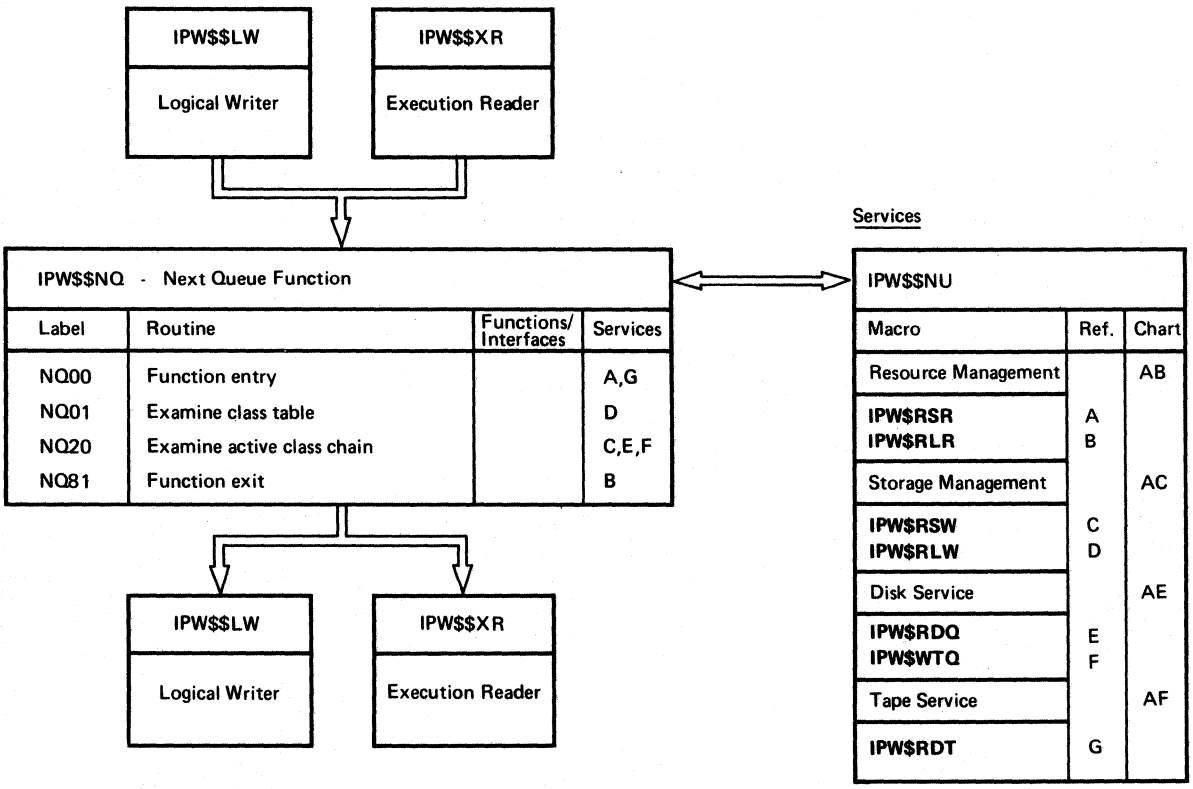
Labels	Chart DB02: IPW$$AQ - Add Queue Set to Chain		Modified Data Fields	Reg. Usage	Calls
AQ25	Scan chain backwards, until priority of new set is equal or lower for add to middle of chain (auxiliary area now contains previous queue record) >	AQ55	QCQW(IPW$DQC)		IPW$RDQ Chart AE
AQ50	Add to Start of Chain				
	Translate and store first-in-class pointer.		CTQF(MCT)		
	Set previous class pointer to zero.		QRQP(IPW$DQR)		
	Set forward pointer to next-in-class.		QRQN(IPW$DQR)		
	Write new record.				IPW$WTQ Chart AE
	Reset backward pointer to first-in-class.		QCQP(IPW$DQC)		
	Write back old record from auxiliary area.				IPW$WTQ Chart AE
AQ55	Add to Middle of Chain				
	Set backward pointer to previous-in-class.		QRQP(IPW$DQR)		
	Set forward pointer to next-in-class.		QRQN(IPW$DQR)		
	Write new record.				IPW$WTQ Chart AE
	Reset forward pointer to next-in-class.		QCQN(IPW$DQC)		
	Write previous record back.				IPW$WTQ
	Read next record.				IPW$RDQ Chart AE
	Reset backward pointer to next-in-class.		QCQP(IPW$DQC)		
	Write next record back.				IPW$WTQ Chart AE
AQ60	Add to End of Chain				
	Translate and store last-in-class pointer.		CTQL(MCT)		
	Set forward class pointer to zero.		QRQN(IPW$DQR)		
AQ63	Set backward pointer to previous-in-class.		QRQP(IPW$DQR)		
	Write new record.				IPW$WTQ
	Reset forward pointer to last-in-class.		QRQN(IPW$DQR)		Chart AE
	Write back old record.				IPW$WTQ Chart AE
AQ65	Add Complete Chain				
	Translate and store first-in-class.		CTQF(MCT)		
	Last-in-class pointers in MCT.		CTQL(MCT)		
	Set backward class pointer to zero.		QRQP(IPW$DQR)		
	Set forward class pointer to zero.		QRQN(IPW$DQR)		
	Write new record.				IPW$WTQ Chart AE

Labels	Chart DB03: IPW$$AQ - Add Queue Set to Chain		Modified Data Fields	Reg. Usage	Calls
	Function Exit				
AQ70	If the queue set is added for a reader task or a non-RJE task, branch to..... >	AQ71			
	If the queue set is added for an RJE task, branch to..... >	AQ73			
AQ71	Post the ECB in the class table entry.		CTQL(IPW$DCT)		
	Branch to exit..... >	AQ88			
AQ73	Set up register 1 as a base register for the LCB chain.			R1	
AQ74	Scan the LCB chain for an LCB associated with the TO terminal user. If not found, branch to..... >	AQ80			
AQ75	Set up a count for scanning the class bytes in register 2, and the values for the list or punch task in registers 3 and 4.			R2,R3, R4	
AQ76	Scan the class bytes in the LCB to check whether the class of the new queue set matches any of them. If so, branch to..... >	AQ77			
	Otherwise, branch to exit..... >	AQ88			
AQ77	Set output switches in the LCB.		LCBOUT(IPW$DLC)		
	Branch to exit..... >	AQ88			
AQ80	Check whether RJE,SNA support has been specified at POWER/VS generation. If not, branch to exit >	AQ88			
	Lock the SNA control block.				IPW$RSR Chart AB
	Establish addressability for the SNA unit control block chain in register 3.			R3	
	If no SNA unit control block is present, branch to..... >	AQ87			

Labels	Chart DB04: IPW$$AQ - Add Queue Set to Chain		Modified Data Fields	Reg. Usage	Calls
AQ81	Scan tne SNA unit control block chain for an SNA unit control block associated with tne TO terminal user. If not found, branch to..... >	AQ87			
AQ83	Point to first (next) device entry. If device is RDR, branch to..... >	AQ87			
	If devices do not match with QRQI, then branch to..... >	AQ83			
	If device is not started or device not available..... >	AQ83			
	Scan for matching class. If not found, branch to..... >	AQ83			
AQ86	Set device not available and output available.		SUL1(IPW$DSU)		
AQ87	Unlock the SNA control block.				IPW$RLR Chart AB
AQ88	Check whether the TCB belongs to a command processor task. If so, branch to..... >	AQ89			
	Unlock the disk management block.				IPW$RLR Chart AB
AQ89	Set the function track byte in the TCB to C'E' to indicate processing complete.		TCFT(IPW$DTC)		
	Return to calling routine.				IPW$RET
	Relative to Absolute Seek Address Conversion Subroutine				
AQ90	The relative queue record address passed in register 1 is converted to an absolute seek address which is stored in the eight byte receiving field addressed by register 2.			R0,R1, R3	
	Return to caller via register 14.			R14	
	Absolute to Relative Seek Address Conversion Subroutine				
AQ95	The absolute seek address addressed by register 2 is converted to a relative record address which is returned to the caller in register 1.			R0,R1	
	Return to caller via register 14.			R14	

CHART DC: IPW$$NQ - GET NEXT QUEUE SET FROM CHAIN (3 PARTS)

Chart DC00: IPW$$NQ - Get Next Queue Set from Chain, General Flow and Macro Calls



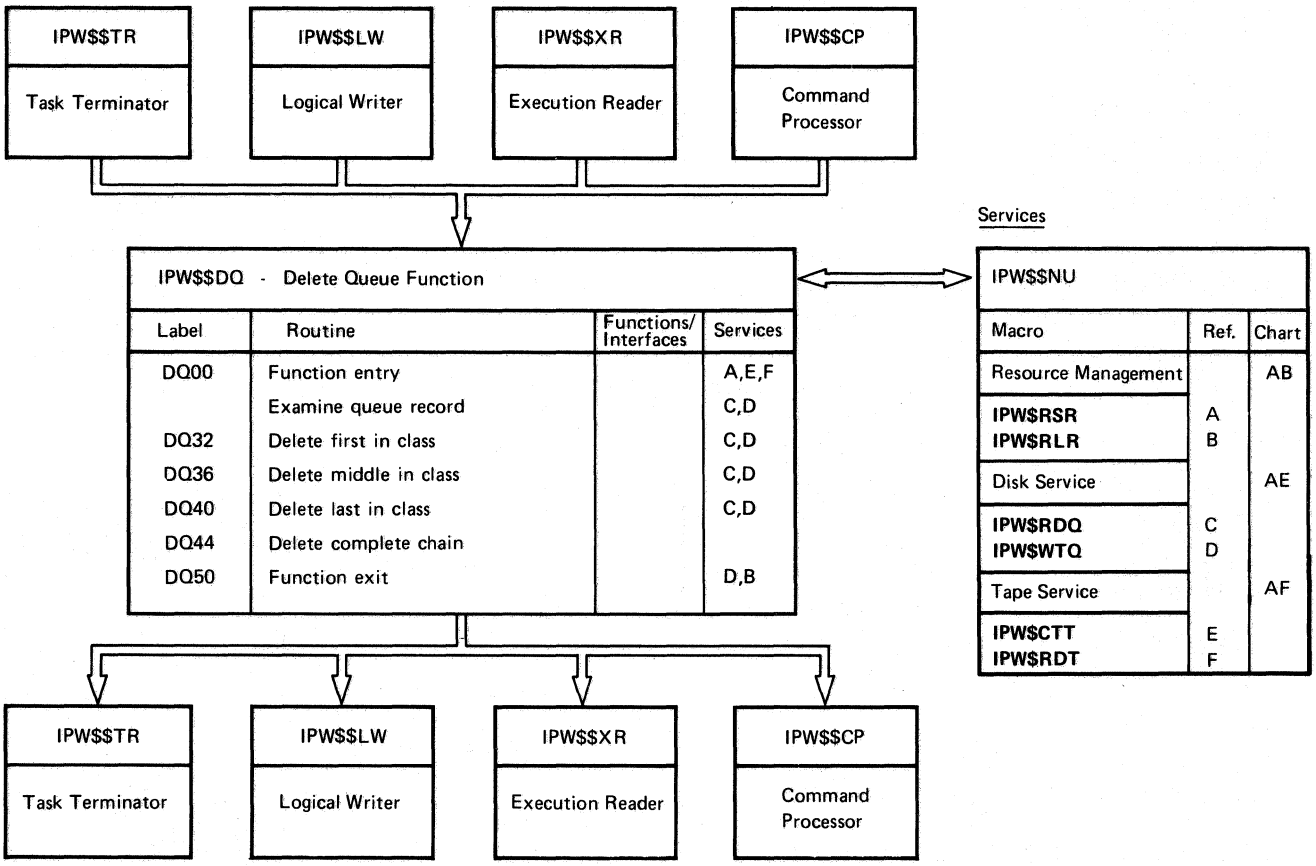
Labels	Chart DC01: IPW$$NQ - Get Next Queue Set from Chain		Modified Data Fields	Reg. Usage	Calls
NQSD	The first 16 bytes constitute the section descriptor: 'NQCS V7M0 '				
NQ00	Function Entry Save caller registers 14 through 9 inclusive.		IPW$DSV		
NQ02	If tape spooling: Read header record into queue record area. Exit..... >	NQ85			IPW$RDT Chart AF
NQ08	Set function track indicator to N (get next in progress). Set addressability for disk management block (DMB). Lock DMB.		TCFT(IPW$DTC)		IPW$RSR Chart AB
NQ10	Examine Class Table Scan task class list in TCB and examine each class task entry in turn If no entry is active (ECB posted): Release queue space Reset space address		TCQA(IPW$DTC)		IPW$RSW Chart AC
NQ14	Reset function track indicator (blank) Exit..... >	NQ82	TCFT(IPW$DTC)		
NQ20	Examine Active Class Chain Translate relative pointer to first-in-class.		QCQW(IPW$DQC)		
NQ30	Read queue record in turn in auxiliary area. Spool management GETSPOOL request? If no, branch to..... >	NQ35			IPW$RDQ Chart AE
	Positioning requested? If no, branch to..... >	NQ35			
	Positioned on Q-record for job given in spool parameter list (SPC)? If no, branch to..... >	NQ40			
	Job for Q-record currently printing? If yes, branch to..... >	NQ40			
	Q-record dispatchable? If yes, branch to..... >	NQ70			

Labels	Chart DC01.1: IPW$$NQ - Get Next Queue Set from Chain		Modified Data Fields	Reg. Usage	Calls
	Indicate that the record is not positioned.		SPR3(SPL)		
NQ35	The record is tested for dispatching.				
NQ40	If no record is dispatchable unpost class entry.	NQ01	CTQF(MCT)		
NQ70	Set execution switch. Write record back.		QCXS(IPW$DQC)		IPW$WTQ Chart AE
	Reserve queue space if necessary and save its real and virtual addresses. Store queue record seek address.		TCQA(IPW$DTC) TCQW(IPW$DTC)		IPW$RSW Chart AC
NQ80	Move queue record from auxiliary area to queue space.		QRDS(IPW$DQR)		

Labels	Chart DC02: IPW$$DQ - Get Next Queue Set from Chain		Modified Data Fields	Reg. Usage	Calls
	Function Exit				
NQ81	Set the function track byte in the TCB to C'I' to indicate open for input.		TCFT(IPW$DTC)		
NQ82	Check whether the TCB belongs to a command processor task. If so, branch to..... >	NQ85			
	Unlock the disk management block.				IPW$RLR Chart AB
NQ85	Check if queue space is present. If so, branch to..... >	NQ89			
	Check if the requesting task is an RJE, SNA writer task. If not, branch to..... >	NQ89			
	Set up the no list, or no punch output available indicator in register 4.			R4	
NQ86	Establish addressability to the LUCB in register 8.				
	Establish addressability to the device entry in the SUCB in register 8.				
	Indicate no output available.		SUL1S(IPW$DSU)		
NQ89	Return to calling routine.				IPW$RET
	Relative to Absolute Seek Address Conversion Subroutine				
NQ90	The relative record address passed in register 1 is converted to an absolute seek address which is stored in the 8-byte field addressed by register 2.			R0,R1, R3	
	Return to caller via register 14.			R14	

CHART DD: IPW$$DQ - DELETE QUEUE SET FROM CHAIN (3 PARTS)

Chart DD00: IPW$$DQ - Delete Queue Set from Chain, General Flow and Macro Calls



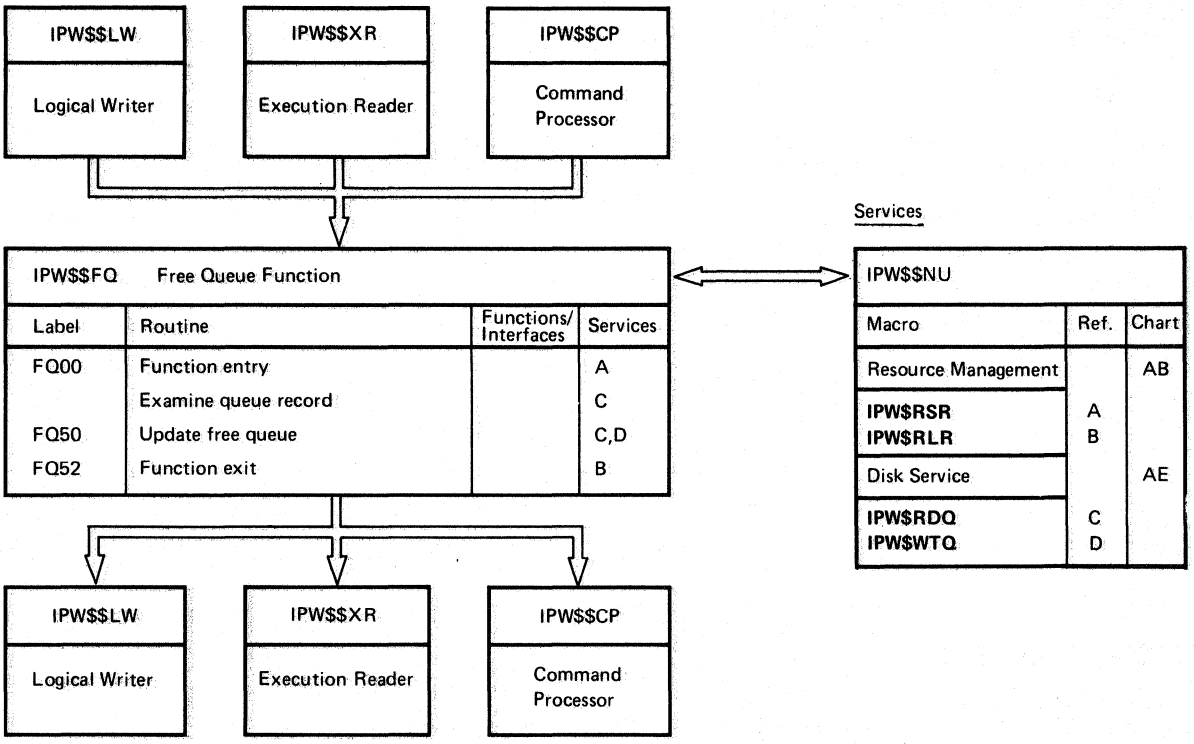
Labels	Chart DD01: IPW$$DQ - Delete Queue Set from Chain		Modified Data Fields	Reg. Usage	Calls
DQCS	The first 16 bytes constitute the section descriptor: 'DQCS V10M0 '				
DQ00	Function Entry Save caller registers 14 through 9 inclusive. Set addressability for queue record area. If no tape spooling, branch to..... >	DQ01	IPW$DSV	R5	
	Read in the trailing queue record. Branch to..... >	DQ55			IPW$CTT Chart AF IPW$RDT Chart AF
DQ01	Set function track indicator to D (delete in progress). Set addressability for disk management block (DMB). Lock DMB.		TCFT(IPW$DTC)	R6	IPW$RSR Chart AB
DQ08	Save "flush hold" indicator from current queue record. Save remaining copy count from current queue record. Save current copy group index Save remaining restart page count from current queue record. (It still exists if PSTOP, RESTART was entered). Examine Queue Record			R4 R4 R4 R8	
	Read first-in-set queue record. If the previous class pointer addresses the record itself, the queue set was not yet added to a class chain..... >	DQ52			IPW$RDQ Chart AE
	If request from command processor.. >	DQ20			
	If 'keep' disposition, change it to 'leave'. If FLUSH,HOLD was entered reset saved indicator and set saved copy count. Set saved copy group index Branch..... >	DQ16	QRDP(IPW$DQR) QRDI(IPW$DQR) QRCR(IPW$DQR) QRCI(IPW$DQR)	R4	
	If 'flush hold' disposition, change it to 'hold', reset saved "flush hold" indicator and set saved copy count.		QRDP(IPW$DQR) QRDI(IPW$DQR) QRCR(IPW$DQR)	R4	
DQ16	Reset execution switch. Set saved remaining restart page count.		QRXS(IPW$DQR) QRRR(IPW$DQR)	R8	

Labels	Chart DD01.1: IPW$$DQ - Delete Queue Set from Chain		Modified Data Fields	Reg. Usage	Calls
	Reset execution switch.		QRXS(IPW$DQR)		
	Rewrite the record.				IPW$WTQ Chart AE
	Exit..... >	DQ52			
DQ20	Determine position in class chain by examining the class chain pointers:				
	Delete first in class..... >	DQ32			
	Delete middle in class..... >	DQ36			
	Delete last in class..... >	DQ40			
	Delete complete chain..... >	DQ44			
DQ32	Delete First-in-Class Set				
	Read next first-in-set queue record in the auxiliary record area of the DMB.				IPW$RDQ Chart AE
	Zero its previous class pointer.		QCQP(IPW$DQC)		
	Rewrite this record.				IPW$WTQ Chart AE
	Store new forward pointer in class table entry.		CTQF(MCT)		
	Exit..... >	DQ50			

Labels	Chart DD02: IPW$$DQ - Delete Queue Set from Chain		Modified Data Fields	Reg. Usage	Calls
DQ36	Delete Middle-in-Class Set				
	Read previous and next first-in-set queue records in turn.				IPW$RDQ Chart AE
	Exchange class pointers.				
	previous record - forward pointer		QCQN(IPW$DQC)		
	next record - previous pointer		QCQP(IPW$DQC)		
	Write the records back in turn.				IPW$WTQ Chart AE
	Exit..... >	DQ50			
DQ40	Delete Last-in-Class Set				
	Read previous first-in-set queue record.				IPW$RDQ Chart AE
	Zero its next class pointer.		QCQN(IPW$DQC)		
	Rewrite this record.				IPW$WTQ Chart AE
	Store new backward pointer in class table entry.		CTQL(MCT)		
	Set live bit (class ECB).		CTQL(MCT)		
	Exit..... >	DQ50			
DQ44	Delete Complete Chain				
	Zero forward and backward pointers in the appropriate class table entry.		CTQF(MCT) CTQL(MCT)		
	Function Exit				
DQ50	Set previous set pointer.		QRQP(IPW$DQR)		
	Reset execution switch.		QRXS(IPW$DQR)		
	Rewrite the record.				IPW$WTQ Chart AE
DQ52	Set function track indicator to C'U' (unchained).		TCFT(IPW$DTC)		
	Unlock DMB.				IPW$RLR Chart AB
DQ55	Restore caller registers and return.			R14-R9	
	Absolute to Relative Seek Address Conversion Subroutine				
DQ90	The absolute seek address addressed by register 2 is converted to a relative record address which is returned to the caller in register 1.			R0, R1 R3	
	Return to caller via register 14.			R14	

CHART DE: IPW$$FQ - FREE QUEUE SET STORAGE (2 PARTS)

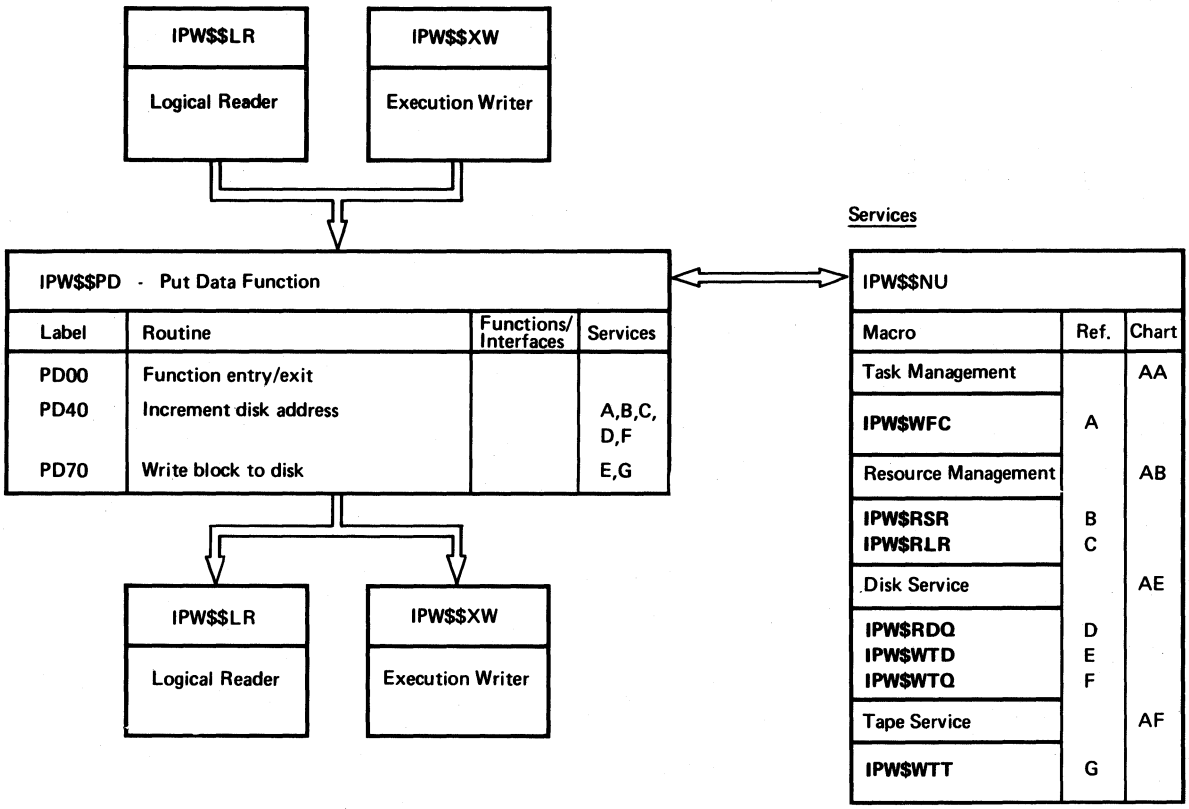
Chart DE00: IPW$$FQ - Free Queue Set Storage, General Flow and Macro Calls



Labels	Chart DE01: IPW$$FQ - Free Queue Set Storage		Modified Data Fields	Reg. Usage	Calls
FQSD	The first 16 bytes constitute the section descriptor:				
	'FQCS V7M0 '				
	Function Entry				
	Save caller registers 14 through 9 inclusive.		IPW$DSV		
	If tape spooling for this task, exit to..... >	FQ65			
FQ02	Set track action indicator F		TCFT(IPW$DTC)		
	Set addressability for:				
	Queue record area (QRA)			R5	
	Disk management block (DMB)			R6	
	Lock DMB.				IPW$RSR Chart AB
	Examine Queue Record				
	If 'leave' or 'hold' disposition, exit..... >	FQ60			
	If the queue set was not yet added. >	FQ50			
	Read first-in-set queue record (if any).				IPW$RDQ Chart AE
	Update Free Queue				
FQ50	Clear auxiliary queue record area.		QCBF(IPW$DQC)		
	Set free record identifier.		QCQI(IPW$DQC)		
FQ51	Copy forward pointer from master record.		QCNS(IPW$DQC)		
	Copy track group pointer from QRA.		QCDF(IPW$DQC)		
	Set seek address in disk request word in DMB.		QCQW(IPW$DQC)		
	Write new first-in-set in free queue.				IPW$WTQ Chart AE
	Update master record with first in queue pointer.		MRQF(IPW$DQC)		
	Increment number of free queue record and CAT.		NRQF(IPW$DPA)		
	If last-in-set, exit..... >	FQ52			
	Read next-in-set queue record, and branch back..... >	FQ51			IPW$RDQ Chart AE
	Function Exit				
FQ52	Post ECB in DMB.		QCEB(IPW$DQC)		
FQ60	Set function track indicator to C'E' (processing complete).		TCFT(IPW$DTC)		
	Unlock DMB.				IPW$RLR Chart AB
FQ65	Restore caller registers and return.			R14-R9	

CHART EA: IPW$$PD - PUT DATA RECORD (4 PARTS)

Chart EA00: IPW$$PD - Put Data Record, General Flow and Macro Calls



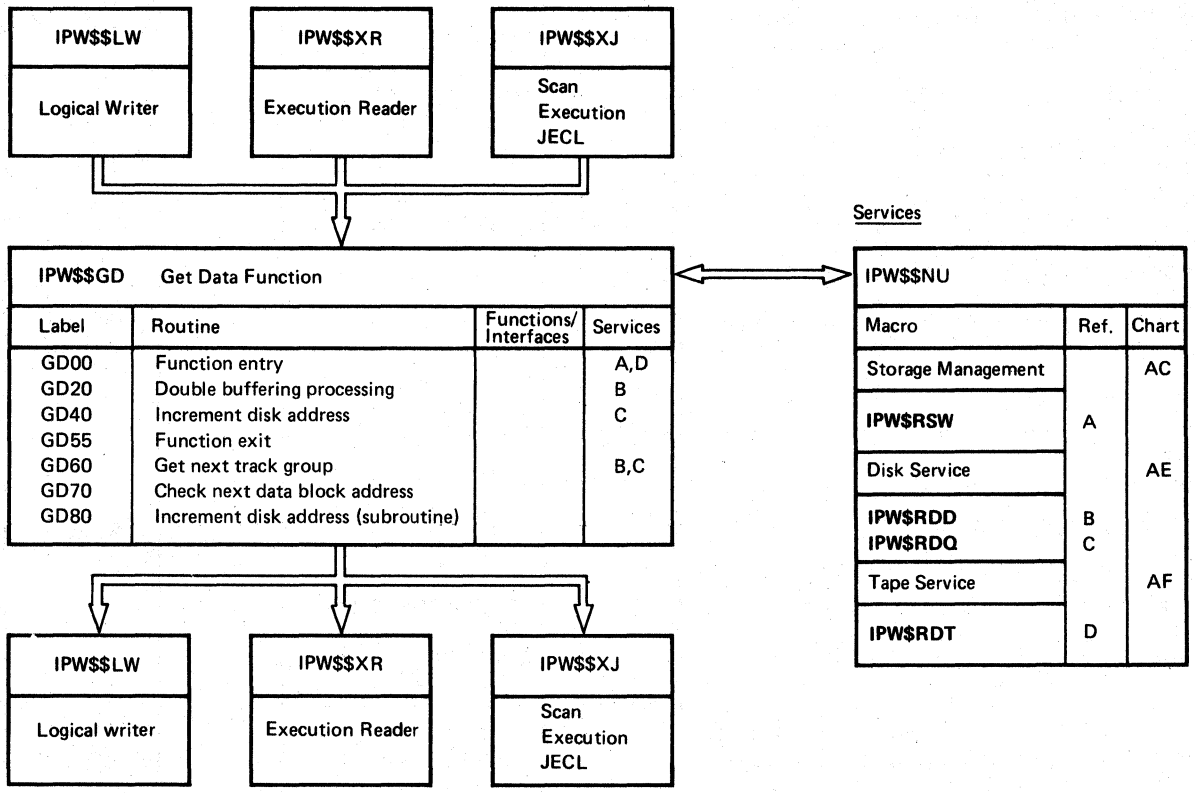
Labels	Chart EA01: IPW$$PD - Put Data Record		Modified Data Fields	Reg. Usage	Calls
PDSD	The first 16 bytes constitute the section descriptor:				
	'PDCS V7M0 '				
PD00	Function Entry				
	Save registers 14 through 9 inclusive		IPW$DSV		
	If end-of-block posted in TCGP..... >	PD19			
	Suppress trailing blanks in record and reset record length.		TCRL(IPW$DTC)		
PD10	Store address of this record.		TCPR(IPW$DTC)		
	If record does not fit in current block..... >	PD30			
	Format record within current block:				
	store record length		DRRL(IPW$DDR)		
	store general purpose byte		DRGP(IPW$DDR)		
	store command code		DRCC(IPW$DDR)		
	move record to block		DRDT(IPW$DDR)		
	Store remaining capacity in current block.		TCBC(IPW$DTC)		
	Update record counter.		QRNR(IPW$DQR)		
PD17	If normal record, restore registers and return to caller.			R14-R9	
	If data break record..... >	PD20			
	If end of data record..... >	PD25			
	Handling of Special Conditions				
PD19	Unexpected end of input:				
	Indicate end of block in previous record if available		DRGP(IPW$DDR)		
	Write data block..... >	PD70			
	Update disk request word..... >	PD40			
	Return to caller..... >	PD17			
PD20	Data break:				
	Write data block..... >	PD70			
	Update disk request word..... >	PD40			
	Return to caller..... >	PD17			
PD25	End of data.				
	Write data block..... >	PD70			
	Return to caller..... >	PD17			

Labels	Chart EA02: IPW$$PD - Put Data Record		Modified Data Fields	Reg. Usage	Calls
PD30	No room in current block:				
	Indicate end-of-block in previous record.		DRGP(IPW$DDR)		
	Write data block..... >	PD70			
	Update disk request word..... >	PD40			
	Handle current record..... >	PD10			
PD40	Increment disk address subroutine.				
	If disposition field indicates tape spooling, return..... >	PD17			
	Set function track indicator to R.		TCFT(IPW$DTC)		
	Set addressability for DMB.			R6	
	Update seek address record number.		TCDW(IPW$DTC)		
	If track group is exhausted:				
	Lock DMB.				IPW$RSR
	If no queue record available (MRQF=0)				Chart AB
	• issue warning message 1Q38I				
	• unpost ECB in DMB		QCEB(IPW$DQC)		
	• unlock DMB				IPW$RLR
					Chart AB
	• wait for queue record (posting of ECB)				IPW$WFC
					Chart AA
	• lock DMB				IPW$RSR
					Chart AB
	Address free queue record and read it into the auxiliary area.		QCQW(IPW$DQC)		IPW$RDQ
			QCDS(IPW$DQC)		Chart AE
	Update next-in-set pointer and write current queue record.		QRNS(IPW$DQR)		IPW$WTQ
					Chart AE
	Update statistical information in CAT				
	• decrement number of available queue records		NRQR(IPW$DPA)		
	• calculate maximum queue records in use		NRQM(IPW$DPA)		
	Update queue record area:				
	• set first-in-set switch off		QRFS(IPW$DQR)		
	• set next-in-set pointer to zero		QRNS(IPW$DQR)		
	• copy track group pointer from auxiliary		QRDF(IPW$DQR)		
	Update logical work space in TCB:				
	• queue record seek address		TCQW(IPW$DTC)		
	• data record seek address		TCDW(IPW$DTC)		
	Write master record back.				IPW$WTQ
	Unlock DMB.				IPW$RLR
	Set function track indicator to O.		TCFT(IPW$DTC)		Chart AB
	Return to caller.				

Labels	Chart EA03: IPW$$PD - Put Data Record		Modified Data Fields	Reg. Usage	Calls
PD70	Write Block Subroutine				
	If tape spooling in disposition field in queue record, write tape record..... >	PD77			IPW$WTT Chart AF
	Set function track indicator to P.		TCFT(IPW$DTC)		
PD75	If this is not execution processor, branch to..... >	PD76			
	Turn off double buffering ECB.		TCED(IPW$DTC)		
	If TCDB not equal C'2', branch to.. >	PD78			
	Wait for second buffer to be written.				IPW$WFC Chart AA
PD78	If this is the end of data, then branch to..... >	PD76			
	Set double buffer switch.		TCDB(IPW$DTC)		
PD76	Write block to disk.				IPW$WTQ Chart AE
PD77	Set function track indicator to O.		TCFT(IPW$DTC)		
PD77	The output block area is reset to zeros.		IPW$DDR		
	The remaining block length is reset. The previous record pointer is reset to start of block.		TCBC(IPW$DTC) TCPR(IPW$DTC)		

CHART EB: IPW$$GD - GET DATA RECORD (4 PARTS)

Chart EB00: IPW$$GD - Get Data Record, General Flow and Macro Calls



Labels	Chart EB01: IPW$$GD - Get Data Record		Modified Data Fields	Reg. Usage	Calls
GDSD	The first 16 bytes constitute the section descriptor: 'GDCS V10M1 '				
GD00	Function Entry				
	Save caller registers 14 through 9 inclusive (IPW$SAV).		IPW$DSV		
	Set up addressability for the queue record.			R5	
	If a data buffer is present, branch to..... >	GD02			
	Reserve a new data buffer.				IPW$RSW Chart AC
	Store buffer pointers.		TCDA(IPW$DTC)		
	Force end of buffer.		TCPR(IPW$DTC)		
GD02	Check the record address. If the buffer is empty, branch to..... >	GD40		R8	
	If the data file is located on disk, branch to..... >	GD03			
	Set the function track byte to C'G'.		TCFT(IPW$DTC)		
	Read tape block.				IPW$RDT Chart AF
	Branch to continue..... >	GD40			
GD03	Set addressability for DMB			R6	
	If double buffer processing, branch to..... >	GD20			
	Load address of disk request word in Register 3.			R3	
	Check if the current data block still belongs to the track group; branch to..... >	GD70			
	If yes, branch to..... >	GD05			
	Otherwise, get next queue record and first data block; branch to..... >	GD60			
	Branch to continue..... >	GD40			
GD05	Get next data block, branch to..... >	GD65			
	Continue; branch to..... >	GD40			

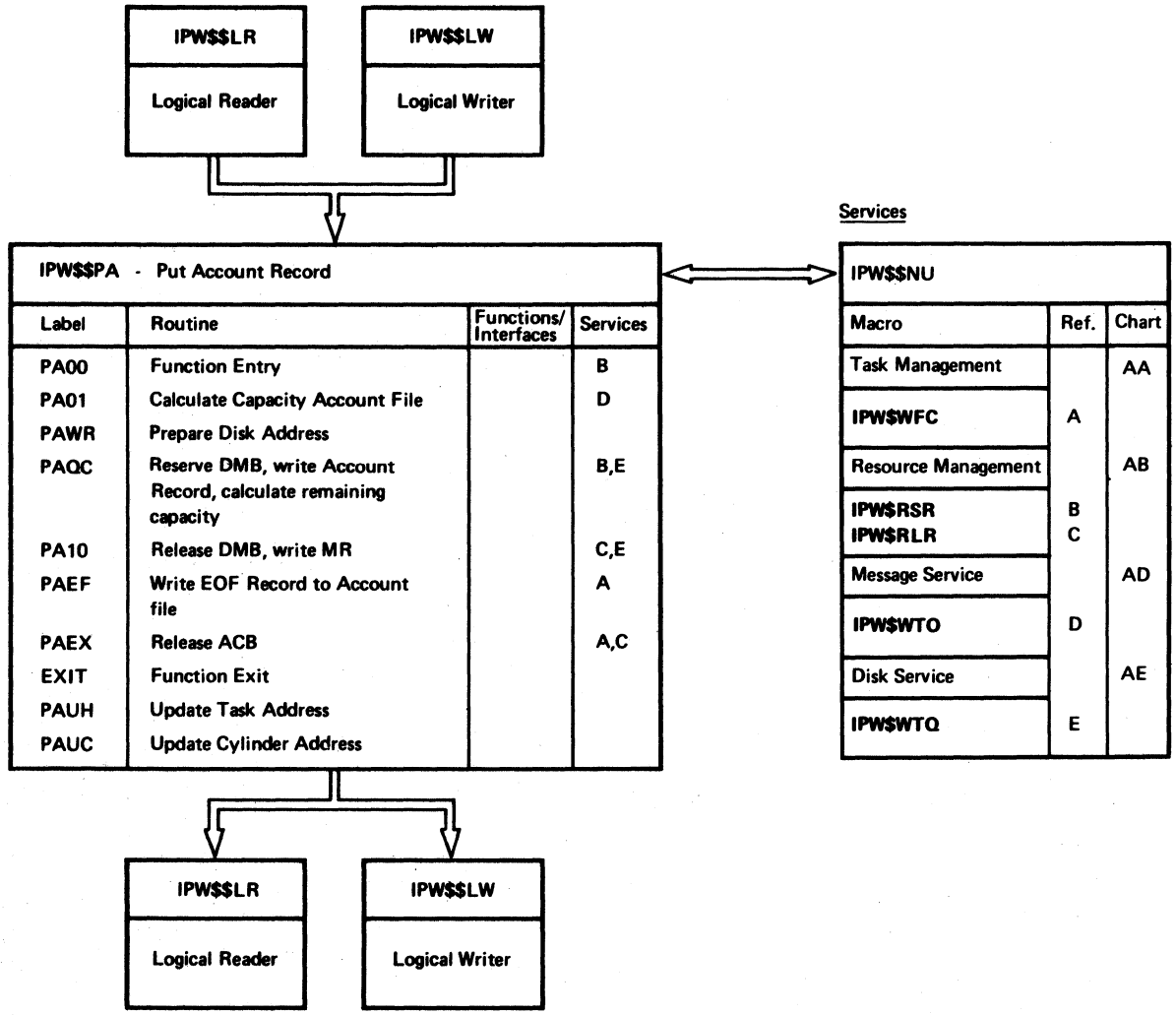
Labels	Chart EB02: IPW$$GD - Get Data Record		Modified Data Fields	Reg. Usage	Calls
	Double Buffer Processing				
GD20	Load register 14 with return address.			RE	
	If the next data block belongs to the next track group, branch to..... >	GD60			
	If this is the first time through, branch to..... >	GD65			
	Swap the buffers pointers.		TCDW(IPW$DTC) TC2DW(IPW$DTC)	R0,R1	
	Indicate that the buffer is empty.		TCPR(IPW$DTC)	R8	
GD24	Address the disk request word by using Register 3.				
	Increment disk address (update the record address to the next record on the track), branch to..... >	GD80	TC2DW(IPW$DTC)	R1 RE	
	Check if the next data block belongs to the same track group, branch to. >	GD70		RE	
	If yes, branch to..... >	GD30			
	Set function track byte to C'G'. Update disk request word with old seek address.		TCFT(IPW$DTC) TC2DW(IPW$DTC		
	Read old data block to make sure that previous I/O is completed.				IPW$RDD Chart: AE
	Indicate next block is not read in branch..... >	GD40	TC2DW(IPW$DTC)		
GD30	Set function track byte to C'G' Read in next data block on the second data buffer. Branch to..... >	GD40	TCFT(IPW$DTC)		IPW$RDD Chart: AE
GD40	Locate the next record within the block and set the appropriate values in the record control word: • Command code. • General purpose byte. • Record address. • Record length. • Record pointer.		TCCC(IPW$DTC) TCGP(IPW$DTC) TCRV(IPW$DTC) TCRL(IPW$DTC) TCPR(IPW$DTC)		
	Check for a break condition or end of block. If not, branch to..... >	GD55			
	Increment Disk Address				
	Reset record pointer to start of block.		TCPR(IPW$DTC)		
	In case of tape spooling, exit..... >	GD55			

Labels	Chart EB02: IPW$$GD - Get Data Record		Modified Data Fields	Reg. Usage	Calls
	In case of double buffer processing, branch to..... >	GD55			
	Update the record address to the next record on the track; branch to >	GD80	TCDW(IPW$DTC)	R1,R3	
GD55	Function Exit				
	Reset function track indicator to I.		TCFT(IPW$DTC)		
	Restore registers and return to caller.			R14-R9	
	Get next trackgroup				
	This subroutine reads the next queue record in set and/or reads in the next data block respectively.				
GD60	Get the next queue record in set.		TCQW(IPW$DTC)		
	Set function track byte to C'N'.		TCFT(IPW$DTC)		
	Save the job suffix number in register 4.			R4	
	Read in next queue record.				IPW$RDQ
	Restore the job suffix number.				Chart: AE
	Update the disk address to the first data block of the new track group		TC2DW(IPW$DTC)		
GD65	Set the function track byte to C'G'.		TCFT(IPW$DTC)		
	Read in the data block.				IPW$RDD
	Return to caller..... >	RE			Chart: AE

Labels	Chart EB03: IPW$$GD - Get Data Record		Modified Data Fields	Reg. Usage	Calls
	Check Next Data Block Address This subroutine checks if the address (seek addr.) of the next data block to be read in belongs to the current track group. Registers on entry: R3 addresses the disk request word RE return address of caller Register upon exit: R1 destroyed Exits: 0(RE) taken when data block fits 4(RE) taken when data block does not fit.				
GD70	If the new data block belongs to the same track, return to caller..... >	RE			
	If the next block is on the next track, update the disk address (track number). If the next track belongs to the same track group, return to caller..... >	RE			
	Otherwise, return to caller via register 14 with a displacement of 4.	4(RE)			
	Increment Disk Address Subroutine This subroutine increments the current data block seek address to address the next data block. No check is made if the data block fits in the track group. Registers on entry: R3 addresses disk request word RE return address of caller Register upon exit: R1 destroyed				
GD80	Update the seek address of the data block: The record number is incremented by one. Return to caller..... >	RE			

CHART FA: IPW$$PA - PUT ACCOUNT RECORD (8 PARTS)

Chart FA00: IPW$$PA - Put Account Record, General Flow and Macro Calls



Labels	Chart FA01: IPW$$PA - Put Account Record		Modified Data Fields	Reg. Usage	Calls
	The first 16 bytes constitute the section descriptor:				
PACS	'PACS V7M0 '				
	On entry, the following register contents are relevant:				
	0: Account record length			R0	
	1: Virtual address account record area			R1	
	The following registers will be used by IPW$$PA:				
	4: Track capacity required			R4	
	5: Address of ACB		IPW$DAC	R5	
	6: Address of DMB		IPW$DQC	R6	
	8: Save account record length			R8	
	9: Save account record area address			R9	
	10: Address of permanent area		IPW$DPA	R10	
	11: Address of TCB		IPW$DTC	R11	
	12: Asynchronous address register			R12	
	13: Address of task save area			R13	
	14: Link register			R14	
	15: Function base address			R15	
PA00	Indicate in TCB that write account function is active (required by termination routine).		TCFT(IPW$DTC)		
	The task identifier field in the TCB is examined. If it is still initialization time (autostart record), or PACCOUNT processing is going on, control is passed back to the caller..... >	EXIT			
	The BG communication region is examined if job accounting support is still active. If yes, branch to... >	PA00A		R2	
	Issue message 'IQ84I' and branch to exit..... >	EXIT			IPW$WTO Chart AB
	Otherwise, indicate no accounting support in module load table		CAPA(IPW$DPA)		
PA00A	Register 5 is loaded with the ACB address.			R5	

Labels	Chart FA01: IPW$$PA - Put Account Record		Modified Data Fields	Reg. Usage	Calls
	Register 6 is loaded with the DMB address.			R6	
	Account record length is saved in register 8.			R8	
	Virtual account record address is saved in register 9.			R9	
PAAC	The ACB is reserved for updating.				IPW$RSR Chart AB
	Account record length is examined.			R8	
	If not zero, a branch is made to continue..... >	PA01			
	If zero, indicating EOF, register 4 is initialized at 1, and a branch is made to write an EOF record..... >	PAEF		R4	

Labels	Chart FA02: IPW$$PA - Put Account Record		Modified Data Fields	Reg. Usage	Calls
PA01	The record capacity is calculated:				
	Register 4 is loaded with the account record length plus 8.			R4	
	This value is multiplied by the tolerance value in the ACB, and divided by 512.			R4	
	The overhead value is added.			R4	
	If the 80 percent limit has already been passed, a branch is made to check for 100 percent..... >	PA02		R4	
	Otherwise, the residual IJAFILE capacity is loaded in register 2 and compared to the 80 percent limit value.				
	If residual capacity is more than 20 percent, a branch is made to continue..... >	PA02			
	Otherwise, the high order bit of the 20 percent limit value is set to 1 to bypass the above comparison on a following occasion.		ACEC(IPW$DAC)		
	Message 1Q31I is issued:				
	Using register 2 as a work register, the message address is stored in the message request word.		TCMW(IPW$DTC)	R2	
	Message 1Q31I is logged.				IPW$WTO Chart AD
PA02	If the account file upper limit has not yet been reached, a branch is made to write the account record... >	PAWR			
	Otherwise, register 4 is incremented by one to examine if the current account record plus an EOF record will fit on the current track, which is the last of the extent.			R4	
	The contents of register 4 are compared to the residual capacity of the current track.		ACLC(IPW$DAC)	R4	
	If not enough space is left on the current track to write account record plus EOF record, a branch is made to notify the operator..... >	PA03			
	Otherwise, register 4 is decremented by one again.			R4	
	A branch is made to write the account record..... >	PAWR			

Labels	Chart FA03: IPW$$PA - Put Account Record		Modified Data Fields	Reg. Usage	Calls
PA03	This routine issues a message to the operator, notifying him that the account file is full.				
	Register 4 is set to zero to indicate wait for full IJAFILE.			R4	
	Using register 2 as a work register, the address of message 1Q32I is			R2 R2	
	stored in the message request word, and message 1Q32I is logged.		TCMW(IPW$DTC)		IPW$WTO Chart AD
	The soft wait bit in the ECB is reset		ACEB(IPW$DAC)		
	A branch is made to release the ACB..... >	PAEX			
PAWR	If the account record to be written fits on the current track, a branch is made to continue..... >	PA04			
	Otherwise, using register 2 as a work register, the residual total capacity			R2	
	is decremented by the (now useless) residual capacity of the current track.		ACAC(IPW$DAC)		
	The account record length (plus 8 for disk data management) is loaded in register 1 and compared to the residual capacity of the current track.			R1	
	If the account record fits as the last record, a branch is made to write the account record..... >	PA04			
	Otherwise, the current track capacity is reset to maximum track capacity and a link is made to update track address..... >	PAUH	ACLC(IPW$DAC)	R14	
PA04	Register 7 is loaded with the address of the account record count field.			R7	
	The current record address is moved to the count field.				
	Using register 2 as a work register, the record number is incremented by one to point to the next account record.		CNTF CTRN	R2	
	Data length plus 8 is stored in the count field, and physical record length in register 2 is stored in the write data CCW.		CTRL ACRW(IPW$DAC)	R2	

Labels	Chart FA04: IPW$$PA - Put Account Record		Modified Data Fields	Reg. Usage	Calls
	Actually, the sequence Write Count and Write Data CCW consists of two Write Count-Key-Data CCWs because the count field part resides in fixed storage (ACB), while the data part address is variable: • The Write Count CCW has a byte count of 8, causing writing of the Count Field only. • The Write Data CCW has a byte count equal to the physical record length of the account record. When chained together, they act as one Write CKD CCW; unchained, the first CCW serves to write an EOF record.)				
PAQC	The DMB is reserved for updating.				IPW$RSR Chart AB
	If the account record ID is not that of an execution processor, a branch is made to continue..... >	PA05			
	Register 9 is set to point 8 bytes before the account record to provide for the control field space.			R9	
	Register 3 is loaded with this address and used as a parameter register in the subsequent REALAD call.			R3	
	The corresponding real address is stored in the write data CCW.		ACRW(IPW$DAC)		
	The control field is zeroed out, and, using register 2 as a work register, the logical record length field LL is set up as account record length plus 4, and the block length as LL plus 4.		AEBL AELL AEBL	R2	
	A branch is made to write the account record..... >	PA06			
PA05	To handle a non-execution-processor account record, the control field located in the DMB is cleared.		ACPR(IPW$DQC)		
	The account record is moved to the I/O area in the DMB.				
	Using register 2 as a work register, logical record length is set to account record length plus 4, and block length to logical record length plus 4.		ACPR(IPW$DQC)	R2	

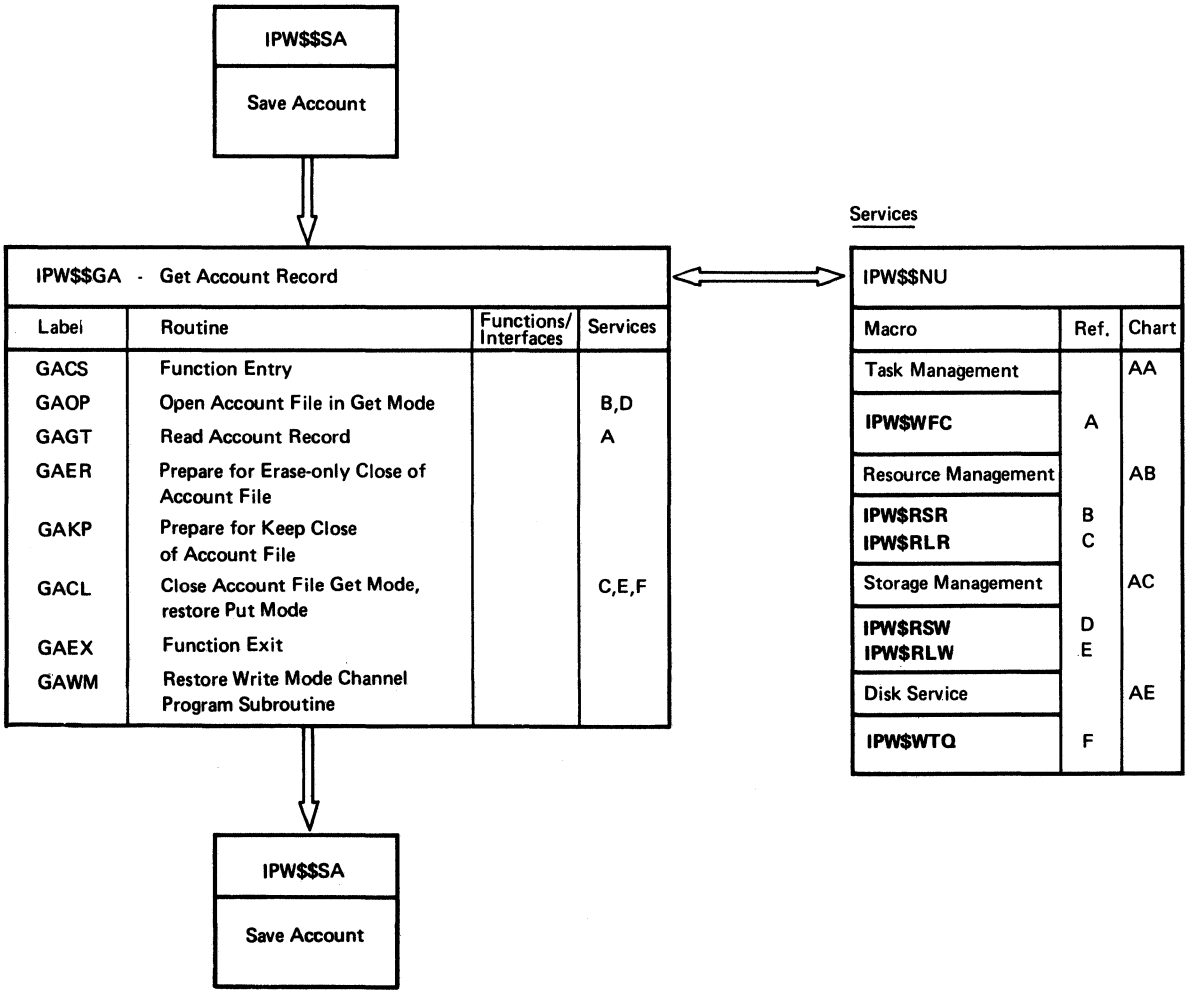
Labels	Chart FA05: IPW$$PA - Put Account Record		Modified Data Fields	Reg. Usage	Calls
	Register 3 is loaded with the real address of the account record and decremented by 8 to point to the beginning of the I/O area.			R3	
	The real address of the I/O area is now stored in the write data CCW.		ACRW(IPW$DAC)		
PA06	Register 1 is loaded with the account CCB address in the ACB.			R1	
	An EXCP is issued to write the account record to the account file.				
	An IPW$WFC call is issued to wait for I/O completion.				IPW$WFC Chart AA
PA07	Register 3 is loaded with the residual track capacity and decremented with the capacity last used as contained in register 4.			R3	
	If negative (in case enough space remained on the track to write just a physical record, whereas register 4 contains physical record length recalibrated into track capacity), branch..... >	PA08			
	Otherwise, the updated residual track capacity is stored back and, using register 3 as a work register, the residual total capacity is decremented by the amount last used.		ACLC(IPW$DAC) ACAC(IPW$DAC)		
	Update seek argument in ACB with current record address (CCHHR from count field ACCF of current record).		ACSA(IPW$DAC)		
	A branch is made to bypass track address updating..... >	PA09			
PA08	A link is made to update track address..... >	PAUH		R14	
	The residual track capacity is reset to maximum value.		ACLC(IPW$DAC)		
PA09	The current record address is moved to the master record.		MRAS(IPW$DQC)		
	The sector value is updated for the next cycle.		ACSE(IPW$DAC)		
	If the record size is not greater than all previous ones, a branch is made to bypass updating maximum record size..... >	PA10			
	Otherwise, the current record size is made the maximum.		MRAZ(IPW$DQC)		

Labels	Chart FA06: IPW$$PA - Put Account Record		Modified Data Fields	Reg. Usage	Calls
PA10	The DMB is released.				IPW$RLR Chart AB
	The updated master record is written to the queue file.				IPW$WTQ Chart AE
	A branch is made to return to the caller..... >	PAEX			
PAEF	This routine is entered when a PEND command has been issued, it writes an EOF record on the account file.				
	Register 7 is loaded with the count field address of the account record.			R7	
	Using register 2 as a work register, the record number is incremented by one.		CTRN	R2	
	Key length and data length are set to zero.		CTKL		
	Write CCW flags are set to zero to unchain CCW string.		ACRW(IPW$DAC)		
	The account CCB address is loaded in register 1.			R1	
	An EXCP is issued to write the EOF record.				
	An IPW$WFC is issued to wait for I/O completion.				IPW$WFC Chart AA
	On completion, the command and data chain flags are set again.		ACRW(IPW$DAC)		
	The sector value is set for the next cycle.		ACSE(IPW$DAC)		
	A branch is made to exit..... >	PAEX			
PAEX	This routine releases the ACB, and, if the account file is full, reissues the write account record after return from saving the account file.				
	The ACB is released.				IPW$RLR Chart AB
	If the account file is not full (WAIT indicator register 4 (set by PA03) not zero), branch is made to return to the caller..... >	EXIT		R4	
	Otherwise, the account event control block is prepared (bit 16 was set to zero by PA03).		ACEB(IPW$DAC)		
	The task is set in internal wait state.				IPW$WFC Chart AA

Labels	Chart FA07: IPW$$PA - Put Account Record		Modified Data Fields	Reg. Usage	Calls
	On return from this wait state after the account file has been saved, a branch is made to restart write account record..... >	PAAC			
EXIT	Indicate in TCB that write account function is finished (General requirement for Task Termination Routine).		TCFT(IPW$DTC)		
	Set save area for register 15 to zero to set caller's register 15 to zero on restore (specific requirement of Task Termination Routine indicating write account function is finished).		SVRF(IPW$DSV)		
	Caller registers are restored and a branch is made back to the calling routine.			R14	
PAUH	This routine updates the seek argument of the last record written.				
	The highest head address is loaded in register 2.			R2	
	If the highest track of the current cylinder has already been reached, a branch is made to update both cylinder and track address..... >	PAUC			
	Otherwise, the current track address is incremented by one, using register 2 as a work register.		ACSA(IPW$DAC)	R2	
	Record number and sector value are zeroed.		ACSA(IPW$DAC) ACSE(IPW$DAC)		
	Control is passed back to the calling routine.			(R14)	
PAUC	The current cylinder address is incremented by one, using register 2 as a work register.		ACSA(IPW$DAC)	R2	
	Current track address and record number are set to zero.		ACSA(IPW$DAC)		
	The sector value is set to zero.		ACSE(IPW$DAC)		
	Control is passed back to the calling routine.			R14	

CHART FB: IPW$$GA - GET ACCOUNT RECORD (9 PARTS)

Chart FB00: IPW$$GA - Get Account Record, General Flow and Macro Calls



Labels	Chart FB01: IPW$$GA - Get Account Record		Modified Data Fields	Reg. Usage	Calls
GASD	The first 16 bytes constitute the Section Descriptor:				
	'GACS V7M0 '				
	On exit, the following register contents are relevant:				
	0: Account record length			R0	
	1: Virtual address account record area			R1	
	The following registers will be used by IPW$$GA:				
	4: Highest track address of cylinder			R4	
	5: ACB address		IPW$DAC	R5	
	6: DMB address		IPW$DQC	R6	
	7: ERASE/KEEP switch			R7	
	10: Address of permanent area			R10	
	11: Address of TCB			R11	
	12: Asynchronous address register			R12	
	13: Address of task save area			R13	
	14: Link register			R14	
	15: Function base address			R15	
	Immediately behind the section descriptor, a branch table entered by the calling routine through the macros IPWOAF, IPWGAR and IPW$CAF, provide the correct subroutine address:				
	0: To open get mode routine....... >	GAOP			
	4: To get mode routine............ >	GAGT			
	8: To close get mode routine..... >	GACL			
	12: To erase only routine.......... >	GAER			
	16: To keep IJAFILE routine....... >	GAKP			
GAOP	This routine is entered if IPW$$GA is invoked by an IPW$OAF macro.				
	Open functions are:				
	• Signal OPEN running to TCB.		TCAT(IPW$DTC)		
	• Reserve ACB,				
	• Reserve real I/O buffer space,				
	• Set ACB from (normal) PUT mode to GET mode.				
	Caller's registers are saved.				IPW$SAV
	Address of ACB is loaded in register 5.			R5	
	Address of DMB is loaded in register 6.			R6	
	The ACB is reserved.				IPW$RSR Chart AB

Labels	Chart FB02: IPW$$GA - Get Account Record		Modified Data Fields	Reg. Usage	Calls
	Using register 1 as a work register, the amount of buffer space is calculated. The maximum record size is rounded to the next higher multiple of 32.			R1	
	The value 8 (1 doubleword) is subtracted from register 1 contents.				
	Work space is reserved using register 1 as parameter register.				IPW$RSW Chart AC
	Since the work space actually obtained is 8 bytes more than the value in the parameter register, the value in the caller's register 0 save area will now agree with the actual buffer size.				
	If work space is available (register 0 = nonzero), branch..... >	GA01			
	Otherwise, indicate immediate stop in TCB for save account function, and reset caller into active status to service terminator routine.		TCTT(IPW$DTC) TCAT(IPW$DTC)		
	Release ACB and return control..... >	GAC7			
GA01	The (virtual) buffer space address is saved in the ACB.		ACWA(IPW$DAC)		
	If the account channel program is already in read mode, a branch is made to bypass switching to read mode..... >	GA02			
	Otherwise, the channel program is switched from write to read mode by swapping the contents of the ACRW and ACPM channel programs.		ACRW(IPW$DAC) ACPM(IPW$DAC)		
GA02	The real buffer address is stored in the read data CCW.		ACRW(IPW$DAC)		
	Suppress incorrect length indication bit is on by default.				
	Maximum account record size is moved to the read data CCW.		ACRW(IPW$DAC)		
	Seek address is set to zero.		ACSA(IPW$DAC)		
	Sector value is set to zero.		ACSE(IPW$DAC)		
	Seek address is initialized at the extent lower limit.		ACSA(IPW$DAC)		
	Record number of search argument is initialized at 1.		ACSA(IPW$DAC)		

Labels	Chart FB03: IPW$$GA - Get Account Record		Modified Data Fields	Reg. Usage	Calls
	The account file is now ready to be accessed in GET mode.				
	A branch is made to return to the caller..... >	GAEX			
GAGT	This routine is entered if IPW$$GA is invoked by an IPW$GAR macro. It reads an account record, and passes its length and (virtual) address to the caller in register 0 and register 1, respectively.				
	Signal GET function running to TCB.		TCAT(IPW$DTC)		
	The caller's registers are saved.				IPW$SAV
	The address of the ACB is loaded in register 5.			R5	
	The address of the DMB is loaded in register 6.			R6	
	The address of the buffer space is loaded in register 8.			R8	
	The caller's register 0 save area is initialized at zero (EOF).		SVR0(IPW$DSV)		
	Head and record count in seek address are copied from the count field.		ACSA(IPW$DAC)		
	If not yet end of extent or end of cylinder, branch..... >	GAG1			
	If not end of cylinder, (so end of extent), branch..... >	GAG3			
	If upper limit reached, branch..... >	GAG6			
	Otherwise, using register 1 as a work register, cylinder number in seek address is incremented by one, and head and record number are set to zero.		ACSA(IPW$DAC)	R1	
	Multitrack bit in read count CCW is set on.		ACRW(IPW$DAC)		
	Sector value is set to zero.		ACSE(IPW$DAC)		
GAG1	If upper limit reached, branch..... >	GAG2			
	Otherwise, using register 4 as a work register, the current head address is compared with the upper head address of a cylinder.			R4	
	If upper head not reached yet, branch to skip resetting the multitrack bit..... >	GAG3			

Labels	Chart FB04: IPW$$GA - Get Account Record		Modified Data Fields	Reg. Usage	Calls
GAG2	The multitrack bit is set to zero.		ACRW(IPW$DAC)		
GAG3	Sector value next cycle is set.		ACSE(IPW$DAC)		
	EXCP parameter register 1 is loaded with the account file CCB address.			R1	
	An EXCP is issued to read the next account record.				
	An IPW$WFC is issued to wait for I/O completion.				IPW$WFC Chart AA
	On completion, if an EOF record was read, branch..... >	GAG6			
GAG5	The account record block length (logical record length + 8) is stored in the caller's register 0 save area.		SVR0(IPW$DSV)		
GAG6	The virtual address of the account record is stored in the caller's register 1 save area.		SVR1(IPW$DSV)		
	Reset caller to active status to service terminator routine.		TCAT(IPW$DTC)		
	A branch is made to return to the caller..... >	GAEX			
GAER	This routine is entered if IPW$$GA is invoked by a IPW$CAF ERASE macro. It is assumed that the caller has made available the ACB and will also release it.				
	Signal ERASE running to TCB.		TCAT(IPW$DTC)		
	The caller's registers are saved.				IPW$SAV
	register 7 is set up to point to the ERASE function indicator.			R7	
	A branch is made to continue..... >	GAC0			
GAKP	This routine is entered if IPW$$GA is invoked by a IPW$CAF KEEP macro, indicating that, on an unrecoverable I/O error, the caller has decided to keep the account file.				
	Signal KEEP running to TCB.		TCAT(IPW$DTC)		
	The caller's registers are saved. register 7 is set up to point to the KEEP function indicator.			R7	IPW$SAV
	A branch is made to continue..... >	GAC0			

Labels	Chart FB05: IPW$$GA - Get Account Record		Modified Data Fields	Reg. Usage	Calls
GACL	This routine is entered if IPW$$GA is invoked by a IPW$CAF macro. Close functions are: 1. Switch account channel program back to PUT mode. 2. Erase account file and write EOF record on each track. 3. Reinitialize several DMB and ACB fields. 4. Release buffer space and ACB. Exceptions: • ERASE will skip function 4, assuming that the caller will release the ACB. • KEEP will skip function 2. Signal CLOSE running to TCB. The caller's registers are saved. Zero function indicator register 7.		TCAT(IPW$DTC)	R7	IPW$SAV
GAC0	Register 5 is loaded with the address of the ACB. Register 6 is loaded with the address of the DMB. If no error have occurred, and IPW$CAF KEEP has therefore not been issued, a branch is made to continue ..... >	GAC1		R5 R6	
	Otherwise, a link is made to reset the ACB to PUT mode..... >	GAWM		R14	
	On return, a branch is made to exit..... >	GAC6			
GAC1	The address of the count field is loaded in register 8. The count field is set to zero. The extent lower limit cylinder and track is moved into the count field. Using register 1 as a work register, the initial record number is set to 1. A link is made to reset the ACB to PUT mode..... >	GAWM	CNTF CCHH	R8 R1 R14	
	On return, command and data chain flags in the first account CCW are reset.				

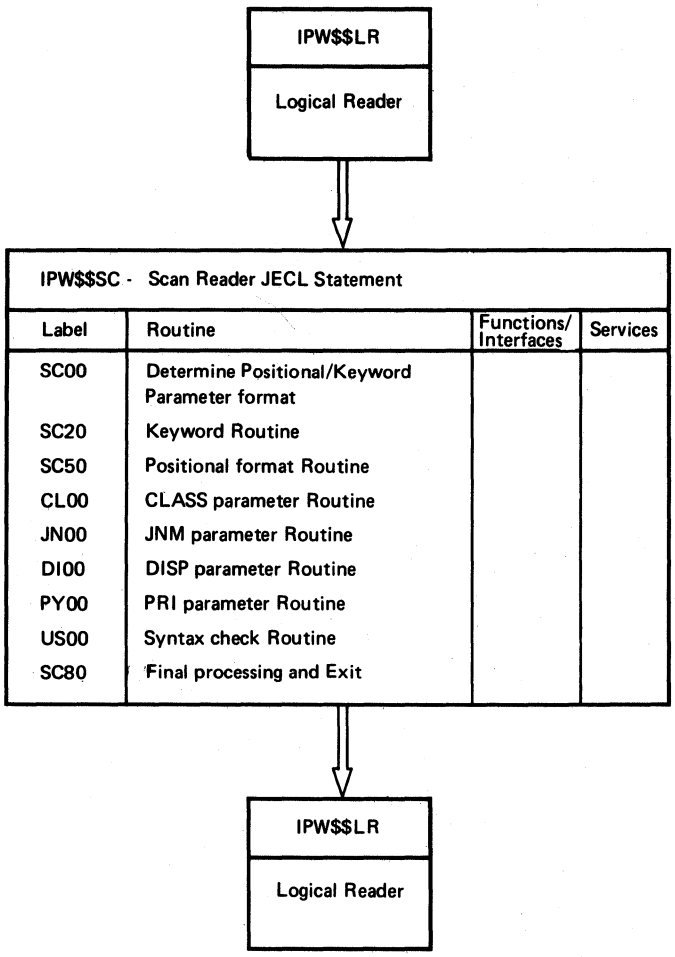
Labels	Chart FB06: IPW$$GA - Get Account Record		Modified Data Fields	Reg. Usage	Calls
	The track number of the highest track of a cylinder is loaded in register 4.		ACRW(IPW$DAC)	R4	
	Record number (search argument) is set to zero.		ACSA(IPW$DAC)		
GAC2	Update track and cylinder address (seek argument) using contents of count field, which was set in preceding cycle or initialized.		ACSA(IPW$DAC)		
	Sector value is set to zero.		ACSE(IPW$DAC)		
	Register 1 is loaded with the address of the account CCB.			R1	
	An EXCP is issued to write an EOF record.				
	The task is put in a wait state until I/O completion.				IPW$WFC Chart AA
	If the account file extent upper limit has been reached, a branch is made to exit..... >	GAC5			
	If the highest track of the current cylinder has been reached, a branch is made to increment the cylinder address..... >	GAC4			
	Otherwise, the track address is incremented by one, using register 3 as a work register.		CTHH	R3	
	A branch is made to write the EOF record on the next track..... >	GAC2			
GAC4	The track address is set to zero.		CTHH		
	Using register 3 as a work register, the cylinder address is incremented by one.			R3	
	A branch is made to write the EOF record on the next track..... >	GAC2			
GAC5	Seek argument in the ACB is set to zero.		ACSA(IPW$DAC)		
	Sector value is set to zero.		ACSE(IPW$DAC)		
	Using register 3 as a work register, the maximum record size is initialized at 1.		MRAZ(IPW$DQC)	R3	
	Current seek address in the master record is set to zero.		MRAS(IPW$DQC)		

Labels	Chart FB07: IPW$$GA - Get Account Record		Modified Data Fields	Reg. Usage	Calls
	Current seek addresses in master record and ACB are initialized at the extent lower limit.		MRAS(IPW$DQC) ACSA(IPW$DAC)		
	Cylinder and track address in the account record count field is reset to the extent lower limit.		CCHH		
	Current account file capacity is reset to maximum.		ACAC(IPW$DAC)		
	Current track capacity is set to maximum.		ACLC(IPW$DAC)		
	The 20 percent residual account file capacity is made positive.		ACEC(IPW$DAC)		
	EOF record writing is finished, so command and data chain flags are set on to chain the write data CCW to the write count CCW.		ACRW(IPW$DAC)		
	Actually, both so-called Write Count and Write Data CCWs are Write Count-Key-Data CCWs; the Write Count CCW having a byte count of 8, providing for the count field only, the Write Data CCW having a byte count equal to the physical record length. Chained, they act as one write CKD CCW; isolated, the Write Count CCW serves to write an EOF record.				
	An IPW$WTQ macro is issued to rewrite the master record.				IPW$WTQ Chart AE
	ACB update is now complete, therefore the account ECB is posted.		ACEB(IPW$DAC)		
GAC6	Neutralize account trace indicator in TCB (signal GET function ended).		TCAT(IPW$DTC)		
	If ERASE has been requested, a branch is made to exit..... >	GAEX			
	Otherwise, register 1 is loaded with the address of the buffer space.			R1	
	If no buffer space was reserved (register 1 = zero), branch to..... >	GAC7			
	Otherwise, release buffer space.				IPW$RLW Chart AC
GAC7	The ACB is released.				IPW$RLR Chart AB
GAEX	This routine is the common IPW$$GA exit routine to the caller.				
	The caller's registers are restored, and control is returned to the caller.			R14	IPW$RET

Labels	Chart FB08: IPW$$GA - Get Account Record		Modified Data Fields	Reg. Usage	Calls
GAWM	This subroutine resets ACB and account channel program to PUT mode.				
	If the ACB is found to be in PUT mode, a branch is made to bypass ACB reset..... >	GAW1			
	Otherwise, the multitrack indicator, which may have been switched off by the IPW$GAR function, is set ON.		ACRW(IPW$DAC)		
	The account channel program is set to write by swapping read and write channel programs.		ACRW(IPW$DAC) ACPM(IPW$DAC)		
	The ACB seek address is set to the disk address of the last record in the account file, which was saved and kept in the master record.		ACSA(IPW$DAC)		
	The sector value is set to zero.		ACSE(IPW$DAC)		
	A possible unit exception indication in the account CCB is reset.		ACST(IPW$DAC)		
GAW1	Control is passed back to the caller.			R14	

CHART GA: IPW$$SC - SCAN READER JECL STATEMENT (21 PARTS)

Chart GA00: IPW$$SC - Scan Reader JECL Statement, General Flow and Macro Calls



Labels	Chart GA01: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
SASD	The first 16 bytes constitute the section descriptor:				
	'SCCS V7M0 '				
	The following register contents are relevant at entry:				
	0: address of end of statement to be checked			R0	
	1: address of parameter to be checked			R1	
	5: address of queue record		IPW$DQR	R5	
	10: address of POWER/VS permanent area		IPW$DPA	R10	
	11: address of TCB		IPW$DTC	R11	
	13: address of task save area		IPW$DSV	R13	
	15: base address IPW$$SC			R15	
	Define Format				
SC00	Caller's registers are saved. Parameter registers 0 and 1 are saved in registers 6 and 7, respectively.			R6 R7	IPW$SAV
SC02	Register 4, to be used in subsequent error indication, is set to 0.			R4	
	Register 3 is set up to contain the machine length of the field to be scanned for the current parameter, being the field starting at the parameter address passed and ending at the statement end.			R3	
	With a translate and test instruction, the delimiter of the parameter is found.			R1,R2	
	If no delimiter found, branch to check form switch.................. >	SC12			
	The value, stored in register 2 by the TRT instruction, is now used as a branch index to the appropriate routine:				
SC05	4: '=' delimiter, branch to process keyword form.................. >	SC20			
	8: ',' delimiter, branch to process positional form................ >	SC50			
	12: ' ' delimiter, branch to process last parameter................. >	SC10			
SC10	The last parameter switch is turned ON.		LWPI(IPW$DTC)		
	A branch is then made to process positional parameter................ >	SC50			

Labels	Chart GA02: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
SC12	This routine is entered if no delimiter was found before the end of the statement. Therefore, register 1 is not set to the delimiter address by the TRT instruction.				
	Register 1 is set to the end of statement address.			R1	
	If the form switch has been set to keyword form, branch to diagnose an invalid statement..... >	SC86			
	Otherwise, if register 4 is negative, indicating a parameter in error, branch to handle the parameter before exit..... >	SC70			
	If the parameter is not in error, the end of statement address is saved in register 9.			R9	
	The last parameter switch is set ON, and a branch is made to return to the caller..... >	SC92	LWPI(IPW$DTC)		
	Keyword Form Handling Routine				
SC20	Temporarily saving the delimiter address in register 3, the form switch, as previously set, is examined using a translate and test instruction, which loads register 2 with the appropriate branch index.			R3 R1 R2	
	Using register 2 as branch index register, a branch is now made to the appropriate routine:				
SC22	4: 'keyword', branch to continue. >	SC30			
	8: 'positional' branch to process error situation..... >	SC26			
	12: blank (not set), branch to process first keyword parameter..... >	SC24			
SC24	Forms switch is set to 'keyword'.		LWFS(IPW$DTC)		
	Branch to join keyword parameter processing..... >	SC30			
SC26	Register 1 is incremented by one to point after the delimiter.			R1	
	Register 4 is made negative to indicate error.			R4	
	A branch is made to scan next delimiter..... >	SC02			

Labels	Chart GA03: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
SC30	Register 3 is set up to contain the length of the keyword to be examined.			R3	
	If zero, branch to indicate error.. >	SC80			
	Otherwise, register 3 is decremented by one to contain machine length.			R3	
	Register 8 is initialized with the start address of the keyword table.			R8	
SC32	The keyword parameter is now checked for validity by comparing it to the valid keyword entries in the keyword table.				
	If end of table has been reached, branch to indicate invalid keyword..... >	SC80			
	Otherwise, if the keyword parameter is equal to the keyword table entry, branch to continue..... >	SC33			
	If not equal, register 8 is incremented to point to the next keyword table entry.			R8	
	Branch back to check with next entry..... >	SC32			
SC33	Register 3 is incremented by one to get the real keyword length again.			R3	
	If unequal to the keyword length of the matching entry in the keyword table, branch to indicate invalid keyword..... >	SC80			
SC35	Preparations are made for the parameter value check.				
	The branch index used to branch to the appropriate parameter check routine, is moved from keyword table entry to the TCB.		LWBI(IPW$DTC)		
	Register 7 is set to point to the start of the parameter value.			R7	
SC36	Register 1 is set to point the '=' delimiter to be used to address the parameter value field about to be scanned.			R1	
	Register 3 is set up to contain the machine length of the remainder of the statement to be scanned.			R3	

Labels	Chart GA04: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
	If the real length is zero, branch to set last parameter switch..... >	SC38			
	Otherwise, scan for the next delimiter, which must be a comma or a blank this time.			R1,R2	
	If not found before end of statement, branch to indicate last parameter.. >	SC38			
	The scan with a translate and test instruction has set a branch index in register 2, to be used to branch to the appropriate routine through the following branch table:				
SC37	4: '=' delimiter, branch to continue scan..... >	SC36			
	8: ',' delimiter, branch to continue..... >	SC70			
	12: ' ' delimiter, branch to indicate last parameter..... >	SC40			
SC38	Register 1 is set to point to the end of the statement to be scanned.			R1	
SC40	The last parameter switch is set ON.		LWPI(IPW$DTC)		
	A branch is made to the appropriate parameter routine..... >	SC70			
	Positional Form Routine				
SC50	The form switch is checked and, if necessary, set.				
	Saving parameter address in register 1 temporarily in register 3, the forms switch is tested using a TRT instruction.			R1,R3	
	The value left in register 2 by this instruction is used as a branch index to branch to the appropriate routine through the following branch table:			R2	
SC54	4: '=' delimiter, branch to indicate invalid parameter in keyword mode..... >	SC80			
	8: ',' delimiter, branch to continue check..... >	SC70			
	12: ' ' (not set, first parameter), branch to set forms switch to positional..... >	SC56			
SC56	The forms switch is set to positional form.		LWFS(IPW$DTC)		

Labels	Chart GA05: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Parameter Syntax Check Routines				
SC70	Parameter address in register 1 is saved. Branch index is loaded in register 2 from the TCB.			R1 R2	
	The branch index will now control branching to the appropriate parameter check routine:				
SC72	4: branch to CLASS parameter routine..... >	CL00			
	8: branch to JNM parameter routine..... >	JN00			
	12: branch to DISP parameter routine..... >	DI00			
	16: branch to PRI parameter routine..... >	PY00			
	20: branch to USER parameter routine..... >	US00			
	24: branch to DEV parameter routine>	DE00			
	28: branch to FID parameter routine>	FI00			
	32: branch to NOD parameter routine>	NO00			
	36: branch to VSC parameter routine>	VS00			
	40: branch to VER parameter routine>	VE00			
	44: branch to FEED parameter routine..... >	FD00			

Labels	Chart GA05.1: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
	CLASS Parameter Syntax Check Routine				
CL00	If current statement is not a JOB statement, branch to check for CTL statement..... >	CL02			
	If statement has keyword form, branch to continue..... >	CL05			
	Otherwise, set the last parameter switch ON, and branch to continue..... >	CL05	LWPI(IPW$DTC)		
CL02	If current statement is not a CTL statement, branch to flag invalid parameter..... >	SC90			
CL05	If CLASS parameter not already specified in current statement, branch to continue..... >	CL10			
	If already specified, and current statement is CTL statement, branch to flag invalid parameter..... >	SC90			
	If current statement is JOB statement, reset CLASS value in queue record to default.		QRCL(IPW$DQR)		
	Register 7 is loaded with address of start of keyword, and branch to flag invalid parameter..... >	SC90		R7	
CL10	CLASS parameter identifier switch is set ON.		LWPI(IPW$DTC)		
	The parameter length is calculated in register 3.			R3	
	If parameter length zero, branch to bypass check..... >	SC92			

Labels	Chart GA06: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Otherwise, if more than one, branch to check two characters..... >	CL30			
	A one-character CLASS parameter should be any of the following characters: A-Z, or 0-9.				
	If CLASS parameter indicates autostart class, branch to change default class indicator in queue record accordingly..... >	CL20			
	If CLASS parameter is not alphameric, branch to flag invalid parameter... >	SC90			
	If the parameter is lower than 'A' branch to indicate valid parameter. >	SC90			
	If parameter is numeric higher than 4, branch to indicate invalid parameter..... >	SC90			
	If current statement is not a CTL statement, branch to change default class indicator in queue record to value specified..... >	CL20			
	If current statement is a CTL statement, change default class value in TCB.		TCCT(IPW$DTC)		
	Branch to return..... >	SC92			
CL20	Default class value in current queue record is set according to class value specified.		QRCL(IPW$DQR)		
	Branch to return..... >	SC92			
	If the CLASS parameter is two characters long, it should be BG, F1, F2, ... Fn. This CLASS parameter format is invalid in a CTL statement, and also in positional form.				
CL30	If CLASS parameter is longer than 2 characters, branch to flag invalid parameter..... >	SC90			
	If current statement no JOB statement, branch to flag invalid parameter..... >	SC90			

Labels	Chart GA07: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
	If current statement is in positional form, branch to flag invalid parameter..... >	SC90			
	If parameter is not "BG", branch to check for F1-Fn..... >	CL40			
	Otherwise, change default value in queue record to '0' (BG).		QRCL(IPW$DQR)		
	Branch to return..... >	SC92			
CL40	If parameter is lower than F1, branch to flag invalid parameter..... >	SC90			
	If parameter is higher than Fn, branch to flag invalid parameter... >	SC90			
	Else, change default class value in queue record to '1' (for F1) - 'n' (for Fn).		QRCL(IPW$DQR)		
	Branch to return..... >	SC92			
	JNM Parameter Syntax Check Routine				
JN00	If current statement not JOB statement, branch to flag invalid parameter..... >	SC90			
	Otherwise, the branch index in the TCB is set for DISP, the next parameter if positional form.		LWBI(IPW$DTC)		
	If JNM parameter not already specified, branch to check JNM parameter..... >	JN10			
	Otherwise, using register 3 to address the DMB, the default job name is copied from DMB to queue record.		QRNM(IPW$DQR)	R3	
	Register 7 is loaded with the start address of the JNM keyword in error.			R7	
	Branch to flag invalid parameter... >	SC90			
JN10	The JNM parameter switch is set ON.		LWPI(IPW$DTC)		
	The parameter length is calculated in register 3.			R3	
	If zero (omitted), branch to return..... >	SC92			
	Using register 2 as a work register, the parameter length is compared with the maximum allowable length of 8 bytes.			R2	

Labels	Chart GA08: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
	If longer than 8 bytes, branch to flag invalid parameter..... >	SC90			
	Register 3 is decremented by one to contain the parameter length in machine format, and the parameter is scanned for non-alphamerics.			R3	
	If non-alphameric characters present, branch to flag invalid parameter... >	SC90			
	Otherwise, the job name field in the queue record is blanked out.		QRNM(IPW$DQR)		
	The JNM parameter is copied into the job name field.		QRNM(IPW$DQR)		
	Branch to return..... >	SC92			
	DISP Parameter Syntax Check Routine				
DI00	If current statement is not a JOB statement, branch to flag invalid parameter..... >	SC90			
	Branch index in TCB is set for PRI, which is the next positional JOB parameter.		LWBI(IPW$DTC)		
	If DISP parameter not already specified in this statement, branch to continue..... >	DI10			
	Otherwise, default value 'D' (for dispatchable) is set in the queue record.		QRDP(IPW$DQR)		
	The address of the start of keyword DISP is loaded in register 7, and a branch is made to flag invalid parameter..... >	SC90		R7	
DI10	The DISP parameter identifier switch is set ON.		LWPI(IPW$DTC)		
	The parameter length is calculated in register 3.			R3	
	If zero (parameter omitted), branch to return..... >	SC92			
	If greater than one, branch to flag invalid parameter..... >	SC90			
	Otherwise, if the DISP parameter is D (for dispatchable), H (for hold), K (for keep), or L (for leave), branch to continue..... >	DI20			
	Otherwise, branch to flag invalid parameter..... >	SC90			

Labels	Chart GA09: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
DI20	The DISP parameter value is moved to the queue record.		QRDP(IPW$DQR)		
	Branch to return..... >	SC92			
	PRI Parameter Syntax Check Routine				
PY00	If current statement is not a JOB statement, branch to flag invalid parameter..... >	SC90			
	Otherwise, the branch index in the TCB is set for CLASS, being the next and last positional JOB parameter.		LWBI(IPW$DTC)		
	If PRI parameter not already specified, branch to continue..... >	PY10			
	Otherwise, using register 3 to address the DMB, the default priority value is copied from DMB to queue record.		QRPY(IPW$DQR)	R3	
	Register 7 is decremented to point to the keyword start, and a branch is made to flag invalid parameter..... >	SC90		R7	
PY10	The PRI parameter identifier switch is set ON.		LWPI(IPW$DTC)		
	The parameter length is calculated in register 3.			R3	
	If zero (omitted), branch to return..... >	SC92			
	If greater than 1, branch to flag invalid parameter..... >	SC90			
	If parameter less than '0', branch to flag invalid parameter..... >	SC90			
	If parameter greater than '9', branch to flag invalid parameter..... >	SC90			
	Otherwise, the priority value specified is copied into the queue record, and a branch is made to return..... >	SC92	QRPY(IPW$DQR)		
	USER Information Parameter Syntax Check Routine				
	Valid in keyword form only.				
	This routine is divided into two segments:				
	One to handle unquoted user information, and one to handle user information within quotes.				

Labels	Chart GA10: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
US00	If first character of USER parameter is not an apostrophe, branch to handle unquoted USER information string..... >	US50			
	Otherwise, register 4 is loaded with the address of the user information field in the queue record.			R4	
	Register 2 is set to 16 (maximum string length), and a link is made, passing these register values to the string scan routine..... >	SS00		R2 R3	
	This string scan routine scans the quoted USER information string, and copies a valid string to the queue record. It passes control back to the caller normally if the scanned string is found valid. If invalid, return is made to an address 4 bytes past the normal point.				
	If USER string is valid, branch to continue..... >	US10			
	If invalid, branch to diagnose error..... >	US20			
US10	Parameter pointer register 1 is saved in register 9.			R1 R9	
	If current statement is not a JOB statement, branch to flag invalid parameter..... >	SC90			
	If USER parameter already specified, branch to flag invalid parameter... >	US30			
	Otherwise, USER parameter identifier switch is set ON, and branch to return..... >	SC92	LWPI(IPW$DTC)		
US20	Parameter pointer register 1 is saved in register 9.			R1 R9	
	If USER parameter not already specified, branch to bypass setting keyword pointer register 7..... >	US40			
US30	Register 7 is decremented to point to the keyword start address.			R7	
US40	User information field in the queue record is blanked out again.		QRUI(IPW$DQR)		
	USER parameter identifier switch is set ON, and a branch is made to flag invalid parameter..... >	SC90	LWPI(IPW$DTC)		

Labels	Chart GA11: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
US50	This routine handles an unquoted user information string.				
	If current statement is not a JOB statement, branch to flag invalid parameter..... >	SC90			
	Otherwise, if the USER parameter has not already been specified in this statement, branch to continue..... >	US60			
	If this is the second USER parameter, register 7 is decremented to point to the keyword start address, and a branch is made to flag invalid parameter..... >	SC90		R7	
US60	The parameter length is calculated in register 3.			R3	
	If zero, branch to return..... >	SC92			
	Using register 2 as a work register, parameter length is compared with 16, being the maximum allowable length.			R2	
	If longer than 16 bytes, branch to flag invalid parameter..... >	SC90			
	Otherwise, register 3 is decremented by one to contain parameter length in machine format.			R3	
	The user information string is copied to the queue record.		QRUI(IPW$DQR)		
	Branch to return..... >	SC92			
	DEV Parameter Syntax Check Routine				
DE00	Load register 5 with the address of the physical work space saved in the linkage save area of the TCB.			R5	
	If the current statement is not a RDR statement, branch to flag invalid parameter..... >	SC90			
	Otherwise, the branch index in the TCB is set for FID, being the next positional RDR parameter.		LWBI(IPW$DTC)		
	Indicate data file mode processing in the TCB.		LWER(IPW$DTC)		
	If the DEV parameter has not already been specified, branch to continue >	DE10			
	If the queue record has not been reserved, branch to..... >	DE05			

Labels	Chart GA12: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Otherwise, SYSIN mode processing is indicated in the queue record or the current job.		QRER(IPW$DQR)		
DE05	Register 7 is decremented to point to the keyword start, and a branch is made to flag invalid parameter..... >	SC90		R7	
DE10	The DEV parameter identifier switch is set on.		LWPI(IPW$DTC)		
	The parameter length is calculated in register 3.			R3	
	If zero (omitted), branch to return >	DE15			
	If the parameter length is not equal to 6 bytes, branch to test for a length of 3 bytes..... >	DE20			
	Registers 2 and 4 are loaded with the address of the cuu field in the DEV parameter, and a branch is made to scan the cuu..... >	DE30		R2,R4	
DE15	Reset the data file indication in the TCB and branch to return..... >	SC92	LWER(IPW$DTC)		
DE20	If the parameter length is not equal to 3 bytes, branch to flag invalid parameter..... >	SC90			
	Register 2 is loaded with the address of the cuu field.			R2	
	Register 3 is set to 3 for the loop count.			R3	
	Register 4 is loaded with the address of the cuu.			R4	
DE40	If the value is less than 'A', branch to flag invalid parameter..... >	SC90			
	If the value is not greater than 'F', branch to check the next character >	DE50			
	If the value is less than '0' or more than '9', branch to flag invalid parameter..... >	SC90			
DE50	Point register 2 to the next character.			R2	
	If all 3 characters have not yet been checked, return to..... >	DE40			
	Otherwise, move the parameter value into the PWS.		PEDW(IPW$DPW)		

Labels	Chart GA13: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Translate the EBCDIC value into packed hexadecimal form (X'0cuu'), and get it into register 2.		PEDW(IPW$DPW)	R2	
	If the queue record has not yet been reserved, return to..... >	SC92			
	If the value calculated has already been set in the queue record, branch to return..... >	SC92			
	If the queue record already contains a physical device address (from a preceding RDR statement), branch to flag invalid parameter..... >	SC90			
	Otherwise, the packed hexadecimal value is saved in the queue record of the current job, which is addressed by register 5.		QRER(IPW$DQR)		
	Branch to return..... >	SC92			
	FID Parameter Syntax Check Routine				
FI00	Load register 5 with the address of the physical work space saved in the linkage save area of the TCB.			R5	
	If the current statement is not a RDR statement, branch to flag invalid parameter..... >	SC90			
	Otherwise, the branch index in the TCB is set for NOD, being the next positional RDR parameter.		LWBI(IPW$DTC)		
	If the first character of the FID parameter is not an apostrophe, branch to flag invalid parameter... >	SC90			
	Otherwise, register 4 is loaded with the address of the file ID in the physical work space, register 2 is set to 8 (maximum string length), and a link is made to pass these register values to the string scan routine.. >	SS00		R2,R4	
	The string scan routine scans the quoted FID information string, and copies a valid string to the physical work space. It passes control back to the caller if the string is valid. If it is invalid, a return is made to an address 4 bytes passed the normal point.				
	If the FID string is valid, branch to continue..... >	FI10			
	If it is invalid, branch to diagnose the error..... >	FI20			

Labels	Chart GA14: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
FI10	The parameter pointer register 1 is saved in register 9.			R9	
	If the FID parameter has already been specified, branch to flag invalid parameter..... >	FI30			
	Otherwise, the FID parameter identifier switch is set on, and a branch is made to return..... >	SC92	LWPI(IPW$DTC)		
FI20	The parameter pointer register 1 is saved in register 9.			R9	
	If the FID parameter has not already been specified, branch to bypass setting the keyword pointer register 7..... >	FI40			
FI30	Register 7 is decremented to point to the keyword start address.			R7	
FI40	The file identification field in the physical work space is cleared.		PEFI(IPW$DPW)		
	The FID parameter identifier switch is set on, and a branch is made to flag invalid parameter..... >	SC90	LWPI(IPW$DTC)		
	NOD Parameter Syntax Check Routine				
NO00	Load register 5 with the address of the physical work space saved in the linkage save area of the TCB.			R5	
	If the current statement is not a RDR statement, branch to flag invalid parameter..... >	SC90			
	Otherwise, the branch index in the TCB is set for VSC, being the next positional RDR parameter.		LWBI(IPW$DTC)		
	If the NOD parameter has not already been specified, branch to continue >	NO10			
	Otherwise, set the NOD information field in the PWS to default 1.		PEND(IPW$DPW)		
	Register 7 is decremented to point to the keyword start address and a branch is made to flag invalid parameter..... >	SC90		R7	
NO10	The NOD parameter identifier switch is set on.		LWPI(IPW$DTC)		
	The parameter length is calculated in register 3.			R3	
	If the parameter length is zero (omitted), branch to return..... >	SC92			

Labels	Chart GA15: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
	If the parameter length is greater than 3, branch to flag invalid parameter..... >	SC90			
	Otherwise, copy register 3 into register 4 and decrement register 4 by one to contain the parameter length in machine format.			R4	
	Copy the parameter pointer register 7 into register 2.			R2	
NO20	If the value of the character addressed by register 2 is less than '0' or greater than '9', branch to flag invalid parameter..... >	SC90			
	Register 2 is incremented by one to point to the next character.			R2	
	Register 3 is decremented by one.			R3	
	If the loop count register 3 is not zero, branch to continue..... >	NO20			
	Convert the NOD parameter value from EBCDIC into packed decimal in the 8 byte work field of the physical work space.		PEDW(IPW$DPW)		
	Convert the packed decimal value of the NOD parameter in the physical work space into binary, and load it into register 2.			R2	
	If the binary value is greater than 255, branch to flag invalid parameter ..... >	SC90			
	Set the NOD parameter value just obtained in register 2 in the physical work space.		PEND(IPW$DPW)		
	Branch to return..... >	SC92			
	VSC Parameter Syntax Check Routine				
VS00	Load register 5 with the address of the physical work space saved in the linkage save area of the TCB.			R5	
	If the current statement is not a RDR statement, branch to flag invalid parameter..... >	SC90			
	If the parameters are not specified in positional format but in keyword format, branch to continue..... >	VS10			
	Otherwise, set the last parameter switch in the TCB.		LWPI(IPW$DTC)		

Labels	Chart GA16: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
VS10	If the VSC parameter has not already been specified, branch to continue. >	VS20			
	Otherwise, set the VSC information field in the physical work space to the default value.		PESC(IPW$DPW)		
	Register 7 is decremented to point to the keyword start address, and a branch is made to flag invalid parameter..... >	SC90		R7	
VS20	The VSC parameter identifier switch is set on.		LWPI(IPW$DTC)		
	The parameter length is calculated in register 3.			R3	
	If the length is zero (omitted), branch to return..... >	SC92			
	If the length is not one, branch to continue..... >	VS30			
	If the parameters are passed in keyword format, branch to flag invalid parameter..... >	SC90			
	If the parameter value (positional format) is not 'S', branch to flag invalid parameter..... >	SC90			
	Otherwise, the volume sequence check specified is copied into the physical work space, and a branch is made to return..... >	SC92	PESC(IPW$DPW)		
VS30	If the parameters are passed in positional format, branch to flag invalid parameter..... >	SC90			
	If the length is not two, branch to continue..... >	VS40			
	If the parameter is not 'NO', branch to flag invalid parameter..... >	SC90			
	Leave the default value in the physical work space and branch to return..... >	SC92			
VS40	If the parameter length is not three characters, branch to flag invalid parameter..... >	SC90			
	If the parameter is not 'YES', branch to flag invalid parameter..... >	SC90			
	Otherwise, the volume sequence check field in the physical work space is set to 'S' and a branch is made to return..... >	SC92	PESC(IPW$DPW)		

Labels	Chart GA17: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
	VER Parameter Syntax Check Routine				
VE00	Load register 5 with the address of the physical work space saved in the linkage save area in the TCB.			R5	
	If the current statement is not a RDR statement, branch to flag invalid parameter..... >	SC90			
	If the VER parameter has not already been specified, branch to continue >	VE10			
	Otherwise, set the VER information field in the physical work space to the default value.		PEVE(IPW$DPW)		
	Register 7 is decremented to point to the keyword start address, and a branch is made to flag invalid parameter..... >	SC90		R7	
VE10	The VER parameter identifier switch is set on.		LWPI(IPW$DTC)		
	The parameter length is calculated in register 3.			R3	
	If the length is zero (omitted), branch to return..... >	SC92			
	If the length is not equal to one, branch to continue..... >	VE30			
	Otherwise, branch to flag invalid parameter..... >	SC90			
	If the parameter length is not two, branch to continue..... >	VE30			
	If the parameter value is not 'NO', branch to flag invalid parameter... >	SC90			
	Otherwise, leave the default value in the physical work space, and branch to return..... >	SC92			
VE30	If the parameter length is not three, branch to flag invalid parameter... >	SC90			
	If the parameter value is not 'YES', branch to flag invalid parameter... >	SC90			
	Otherwise, the verify field in the physical work space is set to 'V', and a branch is made to return..... >	SC92	PEVE(IPW$DPW)		

Labels	Chart GA17.1: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
	FEED Parameter Syntax Check Routine				
FD00	Load register 5 with the address of the PWS from TCB.			R5	
	If the current statement is not a RDR statement, branch to..... >	SC90			
	If the FEED parameter has not already been specified, branch to..... >	FD10			
	Otherwise load the address of the master record in register 3.			R3	
	Clear the option FEED in PWS.		PEOP(IPW$DPW)		
	If the option FEED in the master record is not on, branch to..... >	FD08			
	Set the option FEED in the master record.		PEOP(IPW$DPW)		
FD08	Register 7 is decremented to the point of the keyword start address, and branch to..... >	SC90		R7	
FD10	The FEED parameter identifier switch is set on.		LWPI(IPU$DTC)		
	In register 3 is the length of the parameter.			R3	
	If the length is zero, branch to... >	SC92			
FD20	If the length is greater than two characters, branch to..... >	FD30			
	If the parameter is not 'NO', branch to..... >	SC90			
	Clear option FEED in PWS and branch to..... >	SC92	PEOP(IPW$DPW)		
FD30	If the length of the parameter is greater than 3, branch to..... >	SC90			
	If the parameter is not 'YES', branch to..... >	SC90			
	Set the option FEED in PWS and branch to.....	SC92	PEOP(IPW$DPW)		

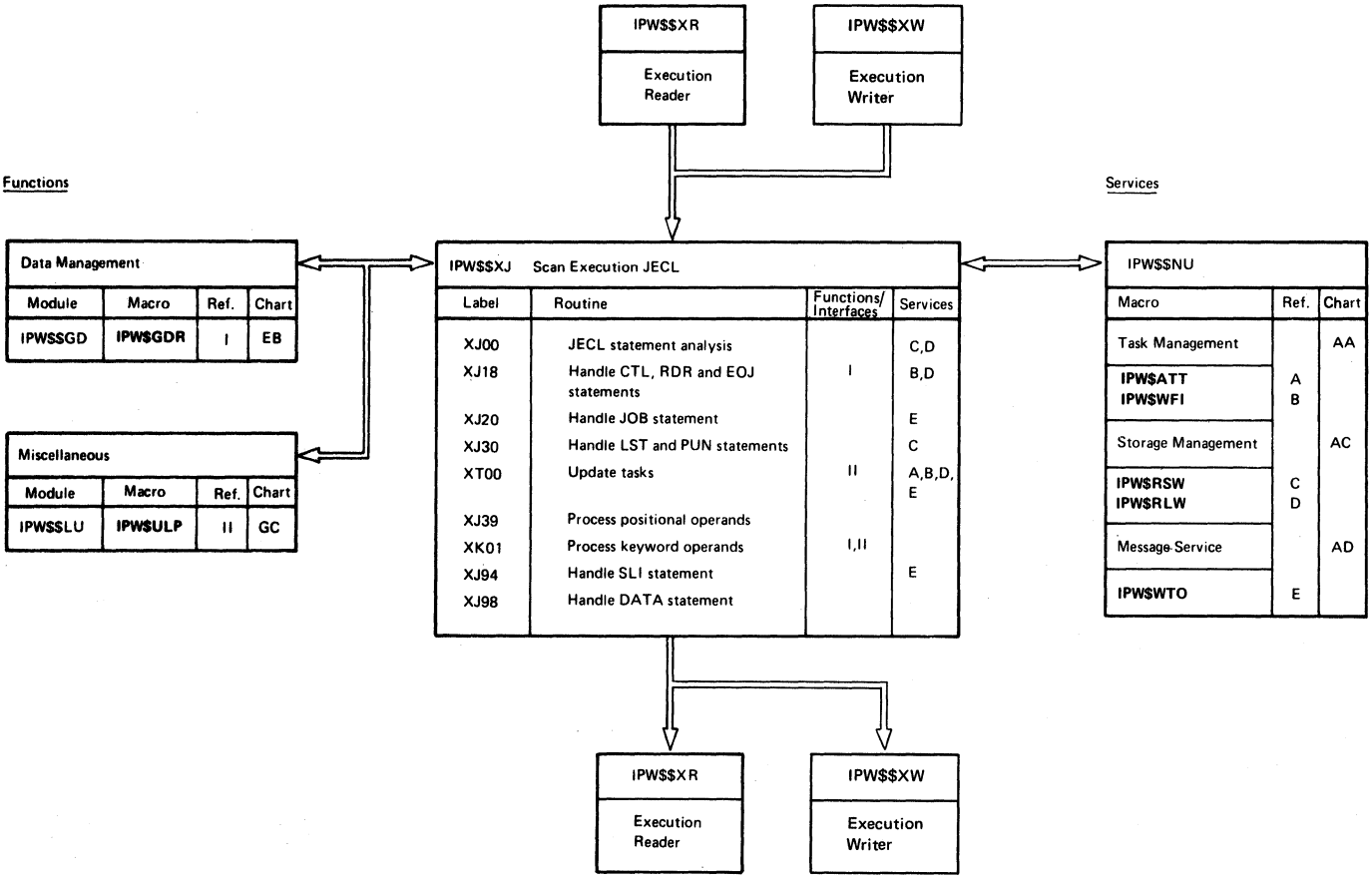
Labels	Chart GA18: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Quoted String Scan Routine				
SS00	Register 1 is loaded with the start address of the string.			R1	
	The last parameter switch is reset, because a possible blank delimiter found may be part of the string to be scanned in stead of a real delimiter.		LWPI(IPW$DTC)		
	Length counter register register 8 is initialized at zero.			R8	
SS10	Register 1 is incremented by one to point to the next character (the first time executed register 1 will point to the first character past the opening apostrophe).			R1	
	The address of the byte in the user information field in the queue record, that is to receive the next USER string character, is incremented by one.			R4	
	Length counter is incremented by one.			R8	
	If the next character is an apostrophe, branch to check for end of string..... >	SS30			
	Otherwise, if end of statement reached before end of string, error return to caller..... >	(R3)+4			
SS20	If string length greater than maximum allowed, branch to flush string.... >	SS10			
	Otherwise, move current character to queue record, and branch to get next character..... >	SS10	QRUI(IPW$DQR)		
SS30	Register 1 is incremented by one to point to the next character.			R1	
	If next character is an apostrophe too, branch to move one apostrophe to queue record user information field..... >	SS20			
	Otherwise, the end of string address is saved in register 4.			R4	
SS40	If end of string is end of statement, branch to set last parameter switch on..... >	SS50			
	If character following string is a comma, branch to continue..... >	SS60			

Labels	Chart GA19: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
	If character following string is a blank, branch to set last parameter switch on..... >	SS50			
	Otherwise (invalid delimiter), scan is continued: length counter is incremented by one.			R8	
	Register 1 is incremented by one to point to the next character, and a branch is made to continue scan for valid delimiter..... >	SS40		R1	
SS50	The last parameter is set ON.		LWPI(IPW$DTC)		
SS60	If string is not followed immediately by a valid delimiter, error return..... >	(R3)+4			
	Register 8 is incremented by one to contain the true string length.			R8	
	If string length zero, return..... >	(R3)			
	If string length greater than maximum allowed, error return..... >	(R3)+4			
	Otherwise, normal return..... >	(R3)			
	Final Processing Routine				
SC80	Invalid parameter handling.				
	The length of the rest of the statement scanned for delimiter is calculated in register 3.			R3	
	If zero (end of statement reached), branch..... >	SC84			
	A scan is made for the next delimiter.			R1,R2	
	If no delimiter before end of statement, branch..... >	SC84			
	The value, stored in register 2 by the TRT scan, is used as a branch index:				
SC82	4: '=', branch to continue scan for comma..... >	SC80			
	8: ',', branch to continue..... >	SC86			
	12: ' ', branch to set last parameter switch..... >	SC84			
SC84	Register 1 is set to point to end of statement.			R1	
	Last parameter switch is set ON.		LWPI(IPW$DTC)		

Labels	Chart GA20: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
SC86	Register 1 is saved in register 9.			R9	
	Branch to flag invalid statement... >	SC90			
SC90	Error exit: address of parameter in error is loaded in register 0 and made negative to signal error.			R0	
	Register 1 is set to next delimiter, as saved in register 9.			R9	
	Branch to exit..... >	SC94			
SC92	Normal exit: parameter address in register 7 is loaded in register 0.			R0	
	Address of next delimiter is loaded in register 1.			R1	
SC94	Return parameter registers are stored in save area.		SCR0(IPW$DSV)		
	Control is passed back to the caller.				IPW$RET

CHART GB: IPW$$XJ - SCAN EXECUTION JECL STATEMENT (16 PARTS)

Chart GB00: IPW$$XJ - Scan Execution JECL Statement, General Flow and Macro Call



Labels	Chart GB01: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	This routine is entered whenever a JECL statement is encountered in the input stream.				
XJCS	CSECT name				
XJSD	The first 16 bytes constitute the section descriptor: 'XJCS V10M1 '				
	General register usage				
	0: **** - Service work register			R0	
	1: **** - Service work register			R1	
	2: **** - Service work register			R2	
	3: **** - Service work register			R3	
	4: **** - Reserved			R4	
	5: IPW$DQR - Queue record space			R5	
	6: IPW$DPB - PDB			R6	
	7: IPW$DCB - User command control block			R7	
	8: IPW$DCW - User channel command word			R8	
	9: **** - Second base register			R9	
	10: IPW$DPA - POWER/VS nucleus			R10	
	11: IPW$DTC - Task control block			R11	
	12: **** - Reserved for nucleus use			R12	
	13: IPW$DSV - Task save area			R13	
	14: **** - Function linkage register			R14	
	15: XJCS - Function base register			R15	
	Note that the usage of registers 0-8 in the analysis routines starting at label XJ39 may differ from the above.				
	JECL Statement Analysis				
XJ00	Save caller registers.				IPW$SAV
	Establish second base register using register 9.			R9	
	Reserve space for a second function register save area.				IPW$RSW Chart AC
	Establish addressability for the new save area in register 13.			R13	
	Check whether a continuation card from a writer-only partition is being processed. If so, branch to..... >	XS38			
XJ02	The address of the statement is loaded into register 9 to determine the operation code. If no operation code is found, return to caller without processing the statement... >	XJ06			
	Save the address of the operation code, which is contained in register 1, in register 7.			R1,R7	

Labels	Chart GB02: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
XJ04	Match the operation code against the entries in the JECL statement routing table to determine which handling routine to branch to. If a match is found, branch to the appropriate handling routine via table XJ12..... >	XJ12			
	If no match is found, return to caller without processing the statement. Return to caller without having processed the statement				
XJ06	If operator corrected JECL statement, branch to..... >	XJ10			
XJ07	Return the additional save area space to the storage pool. Return to IPW$$XR via link register 14, to process the statement. Return to caller after having processed the statement			R14	IPW$RLW Chart AC (IPW$$XR) Chart KA
XJ08	Branch and link to the 'bypass statement' routine..... >	XJ18			
XJ10	If not operator correction JECL, branch to..... >	XJ11			
	Release space.				PW$RLW
	Set up request word.		TCRW(IPW$OTC)		
XJ11	Return the additional save area space to the storage pool. Return to IPW$$XR via link register 14, to process the statement.			R14	IPW$RLW Chart AC (IPW$$XR) Chart KA

Labels	Chart GB02.1: IPW$$SC - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	JECL statement routing table				
XJ12	CTL..... >	XJ08			
	JOB..... >	XJ30			
	EOJ..... >	XJ24			
	RDR..... >	XJ08			
	LST..... >	XJ60			
	PRT..... >	XJ60			
XJ14	PUN..... >	XJ60			
XJ16	SLI..... >	XJ78			
	DATA..... >	XJ86			
	Handle CTL, RDR, and EOJ Statements				
	Since CTL and RDR statements need not be processed at execution time, these statements are bypassed together with any continuation cards relating to them.				
XJ18	If there are no continuation cards, return to caller via link register 14.			R14	
	If the continuation card is for a writer-only partition, branch and link to the writer-only partition routine..... >	XS36			

Labels	Chart GB03: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
XJ20	Get the continuation card.				IPW$GDR Chart EB
XJ22	Restore the base register and the return address.			R14,R15	
	Load the address of the new statement into register 9.			R9	
	Branch to check for more continuation cards..... >	XJ18			
	Handle EOJ Statement				
XJ24	Release the SLI work space, if present.				IPW$RLW Chart AC
XJ25	Check whether the EOJ statement is for a writer-only partition. If not, the statement need not be processed. Branch to..... >	XJ08			
	Set job boundary switch to X'80' to ignore output till next $$JOB.		TCJB(IPW$DTC)		
	Set up register 4 for scanning the entries in the PDB to shut down any subordinate writer tasks.			R4	
	Establish addressability for the TCB of the subordinate task (IPW$DTC) in register 2.			R2	
XJ28	Check the entries in the PDB and if a task has been started:				
	• Reset the ownership in the device list entry in the PDB.		TLTC(IPW$DTL)		
	• Set the 'stop' termination type code.		TNTT(IPW$DTC)		
	• Post the event control block.		TNEB+2(IPW$DTC)		
	Exit to task selection.				IPW$WFI Chart AA
	Branch to..... >	XJ28			

Labels	Chart GB03.1: Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Handle JOB Statement				
XJ30	Check whether the JOB statement is for a writer-only partition, so that the contents can be used to construct the queue record for the ensuing job. If so, branch to..... >	XJ31			
	If the caller is an execution reader, branch to..... >	XJ56			
XJ31	Set the job boundary switch to X'FF' to indicate job in progress.		TCJB(IPW$DTC)		
	Check for a record length of 72 or more. If the record length is smaller, it is set to 71.				

Labels	Chart GB04: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Branch and link to get the statement operand to..... >	XA00			
	• If there are no operands branch to..... >	XJ56			
	• If there are keyword operands branch to..... >	XJ36			
	• If there are positional operands branch to..... >	XJ32			
XJ32	Load the address of error message 1Q51I INVALID JNM PARAMETER in register 7.			R7	
	Branch and link to scan the job name to..... >	XS24			
	Move the job name into the queue record.		QRNM(IPW$DQR)		
	If there are no more operands to be processed branch to..... >	XJ56			
	If there are more operands to be processed branch to..... >	XJ54			
XJ36	Set up registers 1 and 4 for scanning the keyword table and determine which routine to branch to.			R1,R4	
	JNM=..... >	XJ32			
	USER=..... >	XJ40			
	DISP=..... >	XJ52			
	PRI=..... >	XJ52			
	CLASS=..... >	XJ52			
XJ40	Load the address of error message 1Q51I INVALID USER PARAMETER in register 7.			R7	
	Get the user information, which may have a maximum length of 17 characters, from the statement and move it into the queue record. If the user information has been specified incorrectly, issue message 1Q51I.		QRUI(IPW$DQR)		
XJ52	Set up registers 1 and 3 for scanning the remaining keywords.			R1,R3	
XJ54	Check for more keywords.				
	• If there are more keywords branch to..... >	XJ36			
	• If the statement is in error issue error message 1Q49I INVALID DELIMITER.				IPW$WTO Chart AD
	• Otherwise, branch to..... >	XJ56			

Labels	Chart GB05: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
XJ56	Check the job logging switch in the disk management block to see whether the statement is to be logged on the console. If so, load the address of message 1Q47I into register 1, and issue the job log message.			R1	IPW$WTO Chart AD
	Branch to..... >	XJ08			
	Handle LST, PUN, and PRT Statements				
	Task Initiation/Termination				
XJ60	If partition canceling, then branch to..... >	XJ06			
	Reserve storage for the TCB of the new task.				Chart AC
	Set up register 8 as the base register for the new TCB.			R8	
	Initialize the storage descriptor with the information of the storage descriptor of the execution reader TCB.		TNSD(IPW$DTC)		
	Reserve storage for the queue records to be generated by the new task.				IPW$RSW Chart AC
	Initialize the new queue record with system defaults and with information from the execution reader task:				
	• Insert values from the reader queue record.		QNBF(IPW$DQR)		
	• Insert blanks for default compaction name.		QNCP(IPW$DQR)		
	• Set class identifier to C'A'.		QNCL(IPW$DQR)		
	• Set forms identifier to blanks.		QNFI(IPW$DQR)		
	• Set flash identifier to blanks.		QNFL(IPW$DQR)		
	• Set disposition to C'D'.		QNDP(IPW$DQR)		
	• Set number of copies to X'01'.		QNNC(IPW$DQR)		
	• Set counts to zero.		QNNA(IPW$DQR)		
	• Set more counts to zero.		QNNR(IPW$DQR)		

Labels	Chart GB05.1: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Set up register 2 as base register for the disk management block.			R2	
	Check the statement to determine whether a LST or PUN queue record is to be formed. If a list record is to be formed branch to..... >	XJ64			
	Set punch defaults: • Set record identifier to C'P' to indicate punch records.		QNQI(IPW$DQR)		
	• Insert punch default values.		QNSP(IPW$DQR)		
	Branch to..... >	XJ66			
XJ64	Set printer defaults: • Set record identifier to C'L' to indicate list record.		QNQI(IPW$DQR)		
	• Insert printer default values.		QNSP(IPW$DQR)		
	• Insert default line table.		TNGW(IPW$DTC)		
	• Blank out the phase names.		TNCT(IPW$DTC)		
	• Indicate no phase to load.		TNCT(IPW$DTC)		
	• Insert the default UCS options.		TNRS(IPW$DTC)		
	• Insert option byte.		QNOP(IPW$DQR)		

Labels	Chart GB06: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
XJ65	If RJE is present in the system, set the remote 'to' ID in the queue record.		QNTJ(IPW$DQR)		
XJ68	Set up register 4 for scanning the entries in the PDB.			R4	
XJ70	Scan the entries in the PDB to locate the first printer or punch device defined.				
	In case of error, issue message 1Q48I NO MATCHING SPOOL DEVICE.	XT50			IPW$WTO Chart AD
XJ74	Move the physical device address in the queue record.		QNCU(IPW$DQR)		
	Reserve storage for the TCB extension area of the new task.				IPW$RSW Chart AC
	Initialize the 3800 printer setup portion of the extension area with system defaults:				
	• Set initialize printer flag.		SPLFLAG1		
	• Set TRC=NO flag.		SPLFLAG1		
	• Set DEBUG=NORM flag.		SPLFLAG2		
	• Set forms identifier to blanks.		SPLFORMS		
	• Set flash identifier to blanks.		SPLFLASH		
	• Set number of copies to be flashed to 255.		SPLFLSHC		
	• Set 1st copy group value to one.		SPLCOPYG		
	• Set copy group index to one.		SPLCINDX		
	• Set correct length of SETPRT parm-list.		SPLLNGTH (IPW$DTE)		
	• Set default required flag.		TE38RQB		
	Note: The defaults of the PUB2 area related to the device are taken at a later time.				
	Branch and link to the statement analysis routine..... >	XA00			
	If there are no operands branch to..... >	XT00			
	If there are keyword operands branch to..... >	XK02			
	If there are positional operands branch to..... >	XA08			
	Update Tasks				
XT00	Reset registers 5 and 6 as base registers for the queue record and the PDB (IPWDQR, IPWDPD).			R5,R6	
	If the additional count value is not specified, it is set to the maximum value.		QNBN(IPW$DQR)		

Labels	Chart GB06: IPW$$XJ - Scan Execution JECL Statement	Modified Data Fields	Reg. Usage	Calls
XT05	Copy the device address from the queue record into the storage descriptor of the new TCB to determine which execution writer task, if any, the new task will replace. Set up registers 0 and 4 for scanning the entries in the PDB.	TNCU(IPW$DTC)	R0,R4	
XT10	Scan the entries in the PDB to locate the entry relating to the nominated device and determine whether a subordinate writer task exists for it. If DISP=N was specified for the related entry, reset the do-not-interrupt indication, and release ownership of the PUB specified by that entry.			

Labels	Chart GB07: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
XT20	Save the address of the entry in the new TCB.		TNR4(IPW$DTC)		
	Set the device type.		QNDT(IPW$DQR)		
	If no subordinate task is running branch to..... >	XT25			
	Terminate the subordinate task:				
	• Set 'stop' termination type code.		TNTT(IPW$DTC)		
	• Post event control block.		TNEB+2(IPW$DTC)		
	• Exit to task selection.				IPW$WFC Chart AA
XT25	Check the device class within the device entry in the PDB whether or not to intercept requests for this device. If not, branch to..... >	XT35			
	Check if the disposition is T and, if so, set the number of copies to one and the copy groupings to zero.		QNNC(IPW$DQR) QNCG(IPW$DQR)		
XT27	Check if the disposition is I and, if so, test if this is punch output. If not, change disposition to D.		QNDP(IPW$DQR)		
	Set up device-dependent FCB name				
XT28	If the first four characters of the FCB phase name are '$$$$', the four $s are substituted by standard prefixes, depending on the printer device being serviced. Following prefixes are used: FCB1 for 3800 FCB2 for 3211 FCB3 for 3203 FCB4 for 5203		TNCT(IPW$DTC)		
XT30	If the device to be serviced is a 3800 printer, branch to..... >	XT32			
	Otherwise, release the TCB extension area.			R1	IPW$RLW Chart AC
XT32	Attach the new writer task into the system.				IPW$ATT Chart AA
	Return to IPW$$XR..... >	XJ10			
XT35	Get ownership.				IPW$ULP Chart GC
	If no ownership is received:				
	• Set disposition to C'D'.		QNDP(IPW$DQR)		
	• Issue message 1Q46I DISP FORCED TO D FOR				IPW$WTO Chart AD
	• Branch to..... >	XT25			
XT40	Set the device type code in the device list entry in the PDB to C'N' to indicate no interception of requests for this device.		TLCL(IPW$DTL)		

Labels	Chart GB07.1: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
XT45	Return the storage of the TCB to the storage pool, if present.				IPW$RLW Chart AC
	Release the queue space.				IPW$RLW Chart AC
	Release the 3800 TCB extension area, if present.				IPW$RLW Chart AC
XT85	Write JECL statement that is in error on SYSLOC.			R4	IPW$WTO
	Write appropriate error message.			R7	IPW$WTO
	If the incorrect statement is not in a continuation card, branch to..... >	XT86B			
XT86A	Write message '1R33D ERROR IN CONTINUATION CARD -CORRECT FULL STATEMENT'				IPW$WTO
	Bypass any continuation card and branch to..... >	XT86C			
XT86B	Write message '1R33D CORRECT FULL STATEMENT'.				IPW$WTO
XT86C	If a TCB has not been acquired, branch to..... >	XT86			
	Release queue space.				IPW$RLW
	Release 3800 TCB extension area.				IPW$RLW Chart AC
	Release TCB space.				IPW$RLW
XT86	If error statement from operator, branch to..... >	XT87			
	Reserve space for correction.				IPW$RWS
XT87	Set up space for operator correction.				
	Write a blank and ask for operator correction.				IPW$WTR
	If operator replied EOB, branch to. >	XJ10			
	If FLUSH entered, branch to..... >	XT89			
	If the corrected card is not "* $$ LST" or "* $$ PRT" or "* $$ PUN", branch to..... >	XT87A			
	Branch to process card..... >	XJ02			
XT87A	Write message '1R33D NO VALID CORRECTION' and branch to..... >	XT86B			IPW$WTO
XT89	Set cancel code.		TCTT (IPW$DTC)		

Labels	Chart GB07.2: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Set off dump option.		CMRG		
	Branch to release work space..... >	XT45			
	Statement Analysis				
	Additional register usage in the statement analysis routines:				
	0: End of (current) statement			R0	
	1: Translate and test work register			R1	
	2: Translate and test work register			R2	
	3: Start of (current) (next) field			R3	
	4: (Expected) length of field			R4	
	5: New queue record			R5	
	6: General work register			R6	
	7: General work register			R7	
	8: New task control block			R8	

Labels	Chart GB08: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
XA00	The statement is checked to determine whether there are: • No operands. • Keyword operands. • Positional operands.				
	If the statement is in error, issue message 1Q49I INVALID DELIMITER.... >	XT55			IPW$WTO Chart AD
	Return to caller via register 14.			R14	
	Process Positional Operands				
XA08	First Positional Operand				
	This operand represents the disposition attribute and the class attribute to be assigned to the queue entry.			R0,R1 R3,R7	
	If the disposition attribute is specified, a branch and link is made to the disposition validation subroutine..... >	XS30			
	The disposition indicator is reset.		QNDP(IPW$DQR)		
	If the class attribute is specified, the 'disposition/class' indicator is reset.		QNCL(IPW$DQR)		
	In case of error, issue message 1Q51I INVALID CLASS PARAMETER..... >	XT70			IPW$WTO Chart AD
XA10	Branch and link to get the next operand..... >	XS40			
	Second Positional Operand				
	This operand represents the forms identification of the stationery or card stock to be used in processing the output.			R0,R1, R2,R3,	
	If the operand is specified, the 'forms' indicator is reset.		QNFI(IPW$DQR)	R7	
	In case of error, issue message 1Q51I INVALID FNO PARAMETER..... >	XT85			IPW$WTO Chart AD
XA18	The TCB extension area is addressed using register 4 as base. The forms number indicator is moved into it.		SPLFORMS (IPW$DTE)	R4	
	Branch and link to get the next operand..... >	XS40			

Labels	Chart GB09: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Third Positional Operand				
	This operand represents either the number of copies to be produced, or the address of the tape unit to which tape output is to be directed.			R2,R4, R7	
	If a tape unit is specified, the address of error message 1Q51I INVALID TADDR PARAMETER is loaded into register 7 and a branch and link is made to the tape subroutine..... >	XS28		R7	
	If no tape address is specified, branch to..... >	XA20			
XA20	If the number of copies is specified, the address of error message 1Q51I INVALID COPY PARAMETER is loaded into register 7 and a branch and link is made to the numeric operand processing subroutine to convert the operand..... >	XS00		R7	
	The 'number of copies' indicator is reset.		QNNC(IPW$DQR)		
XA22	Branch and link to get the next operand..... >	XS40			
	Fourth Positional Operand				
	This operand represents the number of output records to be handled before the warning message 1Q51I is issued to the operator.			R2,R4, R7	
	If the operand is specified, the address of error message 1Q51I INVALID RBM PARAMETER is loaded into register 7 and a branch and link is made to the numeric operand processing subroutine to convert the operand..... >	XS00		R7	
	The RBM indicator is reset.		QNBM(IPW$DQR)		
	Branch and link to get the next operand..... >	XS40			
	Fifth Positional Operand				
	This operand, which may only be present in a LST (or PRT) statement, represents the line table to be used for the emulation of carriage channel 9 and carriage channel 12 overflow.				
	A branch and link is made to the line table subroutine..... >	XS10			
	Branch to start a new task..... >	XT00			

Labels	Chart GB10: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Process Keyword Operands				
XK00	Branch and link to get the next operand..... >	XS40			
XK02	Load the address of the keyword routing table into register 1.			R1	
XK04	Match the keyword against the entries in the keyword table to determine which routine to branch to for processing this keyword.			R1,R2, R3,R4	
	If no match is found, issue message 1Q50I UNKNOWN KEYWORD. Otherwise, branch to one of the following processing routines.				IPW$WTO Chart AD
	Disposition Operand (DISP=)				
XK10	Branch and link to the disposition validation subroutine..... >	XS30			
	Reset the disposition indicator.		QNDP(IPW$DQR)		
	Branch to get the next operand and process it..... >	XK00			
	Class Operand (CLASS=)				
XK12	In case of error, issue message 1Q51I INVALID CLASS PARAMETER.				IPW$WTO Chart AD
	Reset the class indicator.		QNCL(IPW$DQR)		
	Branch to get the next operand and process it..... >	XK00			
	Remote Operand (REMOTE=)				
XK14	Load the address of error message 1Q51I INVALID REMOTE PARAMETER into register 7 and branch and link to the numeric operand processing subroutine to convert the operand..... >	XS00		R7	
	Reset the remote indicator.		QNTJ(IPW$DQR)		
	Branch to get the next operand and process it..... >	XK00			
	Forms Number Operand (FNO=)				
XK16	In case of error, issue message 1Q51I INVALID FNO PARAMETER..... >	XT85			IPW$WTO Chart AD
	Reset the forms number indicator.		QNFI(IPW$DQR)		
XK20	The TCB extension area is addressed using register 4 as base. The forms number indicator is moved into it.		SPLFORMS (IPW$DTE)	R4	

Labels	Chart GB10.1: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Branch to get the next operand and process it..... >	XK00			
	Job Separator Operand (JSEP=)				
XK24	Set the maximum permissible length in register 4.			R4	
	Load the address of error message '1Q51D invalid JSEP parameter...' into register 7.			R7	
	If the operand is enclosed in parentheses, branch to..... >	XK25			
	Branch to the numeric operand processing subroutine and convert the operand..... >	XS00		R2,RE	
	Reset the job separator indicator.		QNSP(IPW$DQR)		
	Branch to get next operand..... >	XK00			
XK25	Bypass opening parenthesis and address first suboperand.			R3	
	Branch to the numeric operand processing subroutine and convert the suboperand..... >	XS00		R2,RE	
	If nothing is specified, branch to. > Otherwise, reset the job separator indicator.	XK2A	QNSP(IPW$DQR)	R2	
XK2A	If the delimiter is not a comma, branch to..... >	XK2D			
	Address the 2nd suboperand and assume no suppression of separator pages between copies desired.		QNOP(IPW$DQR)		
	If 'N' is specified, branch to..... >	XK2C			
	If 'Y' is specified, set suppress pages between copies flag in QR. Otherwise, branch to..... >	XT85	QNOP(IPW$DQR)		
XK2C	Address delimiter			R3	
XK2D	Test for closing parenthesis. If not, branch to..... >	XT85			
	Address next possible operand and branch to handle it..... >	XK00		R3	

Labels	Chart GB11: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Copies Operand (COPY=)				
XK26	Load the address of error message 1Q51I INVALID COPY PARAMETER into register 7 and branch and link to the numeric operand processing subroutine to convert the operand..... >	XS00		R7	
	Reset the number of copies indicator.		QNNC(IPW$DQR)		
	Branch to get the next operand and process it..... >	XK00			
	Tape Address Operand (TADDR=)				
XK28	Load the address of error message 1Q51I INVALID TADDR PARAMETER into register 7 and branch and link to the tape subroutine..... >	XS28		R7	
	If a mode is specified, it is moved into the new TCB.		TNQW+4(IPW$DTC)		
	Indicate tape disposition.		QNDP(IPW$DQR)		
	Branch to get the next operand and process it..... >	XK00			
	Records before Message Operand (RBM=)				
XK40	Reset the records before message indicators.		QNBM(IPW$DQR) QNBN(IPW$DQR)		
	In case of error, issue message 1Q51I INVALID RBM PARAMETER.				IPW$WTO Chart AD
	Branch to get the next operand and process it..... >	XK00			
	Line Table Operand (LTAB=)				
XK48	Branch and link to the line table subroutine..... >	XS10			
	Branch to get the next operand and process it..... >	XK00			
	Records before Split Operand (RBS=)				
XK50	Load the address of error message 1Q51I INVALID RBS PARAMETER into register 7 and branch and link to the numeric operand processing subroutine to convert the operand..... >	XS00		R7	

Labels	Chart GB12: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Reset the records before split indicator.		QNBS(IPW$DQR)		
	Branch to get the next operand and process it..... >	XK00			
	UCS Buffer Operand (UCS=)				
XK52	Load the address of error message 1Q51I INVALID UCS PARAMETER into register 7 and branch and link to the alphameric phase name subroutine... >	XS24		R7	
	Move the phase name to the new TCB.		TNCT+8(IPW$DTC)		
	If the FOLD or CHECK option was specified, the option is set in the TCB.		TNRS(IPW$DTC)		
	If the CHARS operand is not specified, the UCS specification is treated as if the UCS image specification has been specified for CHARS.				
XK65	The TCB extension area is addressed using register 4 as base. If CHARS is specified, branch to... >	XK00		R4	
	Otherwise, move the UCS phase name into the TCB extension area.		SPLCHAR1 (IPW$DTE)		
	Branch to get the next operand and process it..... >	XK00			
	FCB Buffer Operand (FCB=)				
XK72	Load the address of error message 1Q51I INVALID FCB PARAMETER into register 7 and branch and link to the alphameric phase name subroutine... >	XS24		R7	
	Move the phase name to the new TCB.		TNCT(IPW$DTC)		
	The TCB extension area is addressed using register 4 as a base. The fifth through eighth character of FCB name is moved into the TCB extension area.		SPLFCB (IPW$DTE)	R4	
	Branch to get the next operand and process it..... >	XK00			
	List Device Operand (LST= or PUN=)				
XK80	In case of error, issue message 1Q51I INVALID LST/PUN PARAMETER.				IPW$ULP Chart GC IPW$WTO Chart AD
	Store the channel and unit number.		QNCU(IPW$DQR)		
	Branch to get the next operand and process it..... >	XK00			

Labels	Chart GB12.1: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Priority Operand Handler (PRI=)				
XK94	In case of error, issue message 1Q51I.				
	Check parameter for numeric..... >	XS00			
	Store priority parameter.		QNPY(IPW$DQR)		
	Branch to get next operand and process it..... >	XK00			
	Compact Operand Handler (CMPACT=)				
XK96	In case of error, issue message "1Q51I INVALID COMPACT PARAM."			R7	
	Check parameter for alphameric and first character only alphabetic.				
	Move the compact name.		QNCP(IPW$DQR)		
	Branch to get next operand and process it..... >	XK00			
	FLASH operand (FLASH=)				
XL00	Load the address of error message '1Q51I INVALID FLASH PARAMETER ..' into register 7.			R7	
	If opening parenthesis, branch to.. > Set the maximum permissible length in register 4.	XL05		R4	
	Branch to validate the operand..... >	XS60		RE	
	Reset the flash identifier.		QNFL(IPW$DQR) SPLFLASH (IPW$DTE)		
	Let register 3 point to the next possible operand.			R3,R1	
	Branch to get next operand..... >	XK00			
XL05	Bypass opening parenthesis.			R3	
	Set the maximum permissible length in register 4.			R4	
	Branch to validate the operand..... >	XS60		RE	
	Reset the flash identifier.		QNFL(IPW$DQR) SPLFLASH (IPW$DTE)		
XL07	Check if the delimiter is a comma or closing parenthesis. If not, branch to..... >	XT85		R1	
	Address second value (flash count).			R3	

Labels	Chart: GB12.1 IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Set maximum permissible length in register 4 and register 2 to zero.			R4,R2	
	Branch to validation routine..... >	XS00		RE	
	If value is greater than 255, branch to..... >	XT85		R2	
	Otherwise, reset flash count.		SPLFSLHC (IPW$DTE)		
	If the delimiter is not a closing parenthesis, branch to..... >	XT85		R1,R3	
	Branch to get next operand..... >	XK00			
	BURST operand handler (BURST=)				
XL15	Load address of error message '1Q51I' into register 7.			R7	
	Make 3800 TCB extension area addressable using register 4 as base.			R4	
	Assume BURST=N is specified.		SPLFLAG1 (IPW$DTE)		
	Examine operand: If 'Y' is specified, branch to..... > If 'N' is specified, branch to..... >	 XL16 XL18			
	Otherwise, it is an error; branch to..... >	XT85			
XL16	Set burst flag in queue record.		QNPS(IPW$DQR)		
	Set burst flag in 3800 TCB extension area.		SPLFLAG1 (IPW$DTE)		
XL18	Address next possible operand.			R3	
	Branch to get next operand..... >	XK00			
	DFLT operand handler (DFLT=)				
XL20	Load address of error message '1Q51I' in register 7.			R7	
	Examine operand: If 'Y' is specified, branch to..... > If 'N' is not specified, branch to. >	 XL22 XT85			
	Address 3800 TCB extension area using register 4 as base.			R4	
	Indicate no default requested.		TE38RQB(IPW$DTE)		
XL22	Address next possible operand.			R3	
	Branch to get next operand..... >	XK00			

Labels	Chart: GB12.1 IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	COPYG operand (COPYG=				
	Load address of error message '1Q51I' into register 7.			R3	
	If operand is enclosed in parentheses, branch to..... >	XL32			
	Set maximum permissible length in register 4.			R4	
	Branch to validate and convert..... >	XS00		RE	
	If nothing is specified, branch to. >	XT85		R2	
	If value is greater than 255, branch to..... >	XT85		R2	
	Store copy group index in queue record.		QNCG(IPW$DQR)	R2	
	Set transmission count to one. Branch to..... >	XL39	QNTC(IPW$DQR)		
XL34	Clear out work register and set maximum permissible length in register 4.			R2 R4	
	Branch to validate and convert..... >	XS00		RE	
	If value is zero, branch to..... >	XT85		R2	
	If more than 8 copy groups are specified, branch to..... >	XT85		R2,R4	
	If value is greater than 255, branch to..... >	XT85		R2	
XL36	Store copy group in queue record and address next sub operand.		QNCG(IPW$DQR)	R2	
XL38	If no next suboperand, calculate total value of all copy groups specified.				
	If total value greater than 255, branch to..... >	XT85			
XL39	Update the 3800 TCB extension area.		SPLCOPYG (IPW$DTE)		
	Branch to get next operand..... >	XK00			
	MODIFY operand (MODIFY=)				
XL40	Load the error message'1Q51I ..' address into register 7.			R7	
	If opening parenthesis, branch to.. >	XL42		R3	
	Get maximum permissible length in register 4.			R4	

Labels	Chart GB12.1 IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Branch to validate operand..... >	XS60		RE	
	If nothing is specified, branch to. >	XT85		R4	
	Address the 3800 TCB extension area and update the copy modification field.		SPLCPMOD (IPW$DTE)	R2	
	Branch to get the next operand..... >	XK00			
XL42	Bypass the opening parenthesis and set the maximum permissible length in register 4.			R3,R4	
	Branch to validate the first specification..... >	XS60		RE	
	If nothing is specified, branch to. >	XT85		R4	
	Address the 3800 TCB extension area and update the copy modification field.		SPLCPMOD (IPW$DTE)	R2	
	If no continuation, branch to..... >	XL43		R1	
	Address the second part of the operand and set the maximum permissible length in register 4.			R3,R4	
	Branch to validate operand..... >	XS60		RE	
	If nothing is specified, branch to. >	XT85		R4	
	Otherwise, address the 3800 TCB extension area and update it.		SPLCMCHR (IPW$DTE)	R2	
XL43	If the delimiter is not a closing parenthesis, branch to..... >	XT85		R1	
	Branch to get next operand..... >	XK00			
	CHARS operand (CHARS=)				
XL50	Load the error message address into register 7.			R7	
	If opening parenthesis, branch to.. >	XL54			
	Set the maximum permissible length in register 4.			R4	
	Branch to validate operand..... >	XS60		RE	
	If nothing is specified, branch to. >	XL52		R4	
	Address the TCB extension area, using register 2.			R2	
	Move in character arrangement table value.		SPLCHAR1 (IPW$$DTE)		
XL52	Address delimiter and branch to.... >	XL56		R3	

Labels	Chart: GB12.1 IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
XL54	Store address of first character arrangement table field in TCB extension area.		TE38GW (IPW$DTE)	R2,R4	
XL55	Set the maximum permissible length in register 4.			R4	
	Branch to validate operand..... >	XS60		RE	
	If nothing is specified, branch to. >	XT85		R2	
	If more than four CATs are specified, branch to..... >	XT85			
	Load address of the current CAT field in the TCB extension area in register 14.			RE	
	Move CAT value into TCB extension area.		SPLCHAR(IPW$DTE)		
	Bump to next CAT field in TCB extension area and save it.		TE38GW(IPW$DTE)		
	Point register 3 to delimiter.			R3	
	If continuation, branch to..... >	XL55			
	If not closing parenthesis, branch to..... >	XT85			
XL56	Examine if the UCS operand is specified. If so, branch to..... >	XK00			
	Otherwise, take the first CHARS value as UCS specification.		TNCT(IPW$DTC)		
	Branch to get next operand..... >	XK00			
	Numeric Operand Processing Subroutine				
XS00	On entry to this routine, register 3 contains the address of the first byte of the field to be scanned and register 4 contains the maximum permissible length of the field.			R3,R4	
	Scan the operand field. If the field is in error, issue the error message, the address of which is contained in register 7.			R7	IPW$WTO Chart AD
	Convert the value of the field to binary in register 2.			R2	
	Return to caller via register 14.			R14	

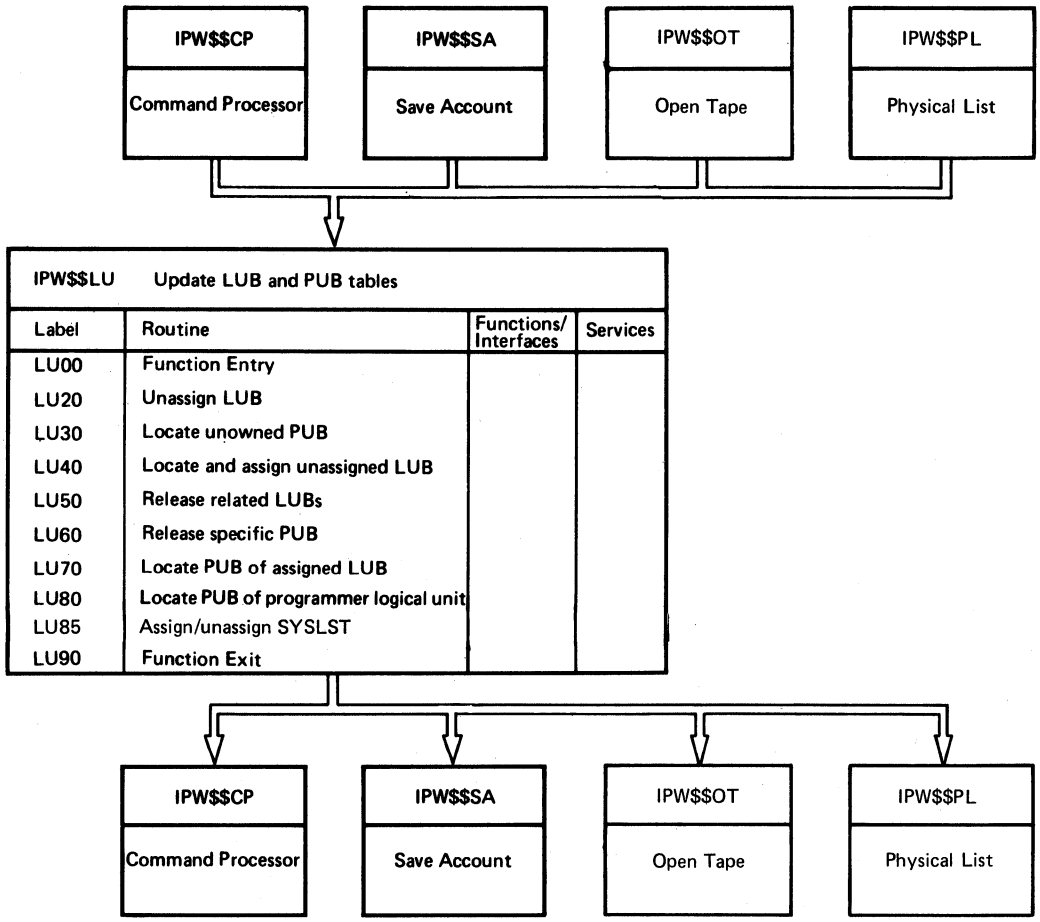
Labels	Chart GB13: IPW$$XJ - Scan Execution JECL Statement	Modified Data Fields	Reg. Usage	Calls
	Line Table Entry Subroutine			
XS10	On entry to this routine, register 3 contains the address of the first byte of the field to be scanned.		R3	
	Scan the operand field to insure that it consists of numeric characters only and that it has a length of 26. If the field is correct, each duplet is converted to binary and stored in the line table field of the TCB.	TCGW(IPW$DTC)		
	In case of error, issue message 1Q51I INVALID LTAB PARAMETER.			IPW$WTO Chart AD
	Return to caller via register 14.		R14	
	Alphameric Phase Name Subroutine			
XS24	On entry to this routine, register 3 contains the address of the first byte of the phase name.		R3	
	Scan the phase name to insure that it consists of alphmeric characters.			
	In case of error, issue the error message, the address of which is contained in register 7.		R7	IPW$WTO Chart AD
	Get the machine length of the phase name in register 4.		R4	
	Return to caller via register 14.		R14	
	Tape Subroutine			
XS28	Set the default mode set (X'C3') in the new TCB.	TNQW(IPW$DTC)		
	Check the tape unit address in register 3 for hexadecimal characters. If it is invalid, issue the error message, the address of which is contained in register 7.		R7	IPW$WTO Chart AD
	Move the tape address to the new TCB.	TNQW+5(IPW$DTC)		
	Return to caller via register 14.		R14	

Labels	Chart GB14: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Disposition Validation Subroutine				
XS30	Get the address of the disposition table in register 1.			R1	
	Match the disposition against the entries in the table. If an invalid disposition was specified, issue error message 1Q51I INVALID DISP PARAMETER.				IPW$WTO Chart AD
	Return to caller via register 14.			R14	
	Get Continuation Card for Writer-only Partition				
XS36	If a continuation card is to be read for a writer-only partition, a return is made to IPW$$XR.				
	Check register 8 to see if a new TCB has been acquired. If not return to caller via register 14.			R8 R14	
	Save registers 14 through 5.				
	Store the TCB address in the restart indicator of the calling TCB.		TCRS(IPW$DTC)		
	Return to IPW$$XR..... >	XJ10			
XS38	Clear the restart indicator in the TCB.		TCRS(IPW$DTC)		
	Restore registers 14 through 5.			R14-R5	
	Reload this task's registers.				
	Return to caller via register 14.			R14	
	Get Next Operand Subroutine				
XS40	On entry to this routine, register 3 contains the address of the field delimiter of the current (processed) operand.			R3	
	If there are no more operands to be processed return to start a new task..... >	XT00			
XS42	If the delimiter is invalid, issue message 1Q49I INVALID DELIMITER..... >	XT55			IPW$WTO Chart AD
	If the remaining operands to be processed are on a continuation card, read the continuation card.				IPW$GDR Chart EB
	Load the address of the new operand into register 3.			R3	
	Return to caller via register 14.			R14	

Labels	Chart GB15: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Alphameric Operand Processing Subroutine				
XS60	On entry to this routine, register 3 contains the address of the first byte of the field to be examined. Register 4 contains the maximum permissible length of the field.			R3,R4	
	Scan the operand field; if the field is not alphameric, branch to..... >	XT85		R1,R2	
	Examine the delimiter: If a comma (,) or blank ( ) or a closing parenthesis, branch to..... >	XS62		R1	
	If not stopped by end of statement, branch to..... >	XT85		R0	
XS62	Get the machine length of the operand in register 4.			R4	
	Return to caller via register 14.			RE	
	Handle SLI Statement				
XJ78	If SLI is not supported or in progress, branch to issue a message..... >	XJ84			
	Scan the operand field for the book name.				
XJ82	Branch and link to IPW$$SL source stateement library inclusion.				IPW$$SL Chart GE
	Return to IPW$$XR..... >	XJ10			
XJ84	Issue message 1Q45I SLI STATEMENT NOT SUPPORTED.				IPW$WTO Chart AD
	Set termination type in the TCB to C'C'.		TCTT(IPW$DTC)		
	Return to IPW$$XR..... >	XJ08			
	Handle DATA Statement				
XJ86	If no SLI workspace is available, return to IPW$$RR..... >	XJ06			
	Reset reader switch to C'R' in the SLI work space.		SLRR(IPW$DSL)		
	Return to IPW$$XR..... >	XJ08			

CHART GC: IPW$$LU - UPDATE LUB AND PUB TABLES (12 PARTS)

Chart GC00: IPW$$LU - Update LUB and PUB Tables, General Flow and Macro Calls



Labels	Chart GC01: IPW$$LU - Update LUB and PUB Tables	Modifier Data Fields	Reg. Usage	Calls
	The first 16 bytes constitute the section descriptor: 'LUCS V10M1 ' On entry, the following register contents are relevant: 0: branch index to required function: 0: unassign LUB 4: locate unowned PUB 8: assign LUB 12: release related LUBs 16: release specific PUB 20: identify PUB for specific LUB 24: release related programmer LUBs 28: identify PUB of logical unit 32: assign SYSLST to printer 36: unassign SYSLST		R0	
	1: PIB address		R1	
	2: device address in EBCDIC: C'cuux', where x may be - R for Reader P for Punch L for List T for Tape D for Disk For the unassign function register 2 contents are ignored, and for the release functions the device type 'X' is ignored.		R2	
	3: CCB address For the locate and release functions register 3 contents are ignored.		R3	
	10: POWER/VS permanent area address	IPW$DPA	R10	
	11: TCB address	IPW$DTC	R11	
	13: Task save area address	IPW$DSV	R13	
	14: caller's return address		R14	
	15: Function base address		R15	

Labels	Chart GC02: IPW$$LU - Update LUB and PUB Tables		Modifier Data Fields	Reg. Usage	Calls
	Function Entry				
LU00	Caller's registers are saved. Because the subsequent functions will (or may) change system I/O tables, any concurrent I/O handling activity may produce unpredictable results. Therefore, the system is 'seized' (monopolized) for the duration of function execution through an SVC 22. Parameter register 0 will contain 255 (X'FF') to signal that interrupts are allowed.				IPW$SAV
	Register 5, to be used as a base address for COMREG access, is loaded with the COMREG address.			R5	
	The PIK of the partition concerned is placed in register 6 from the PIB extension of the PIB addressed in register 1.				
	The function branch index passed in register 0 is now loaded in register 4 and used to branch to the appropriate function through the following branch table.			R4	
LU10	Index 0: Unassign LUB..... >	LU20			
	Index 4: Locate PUB..... >	LU30			
	Index 8: Assign LUB..... >	LU40			
	Index 12: Release related LUBs.... >	LU50			
	Index 16: Release specific PUB.... >	LU60			
	Index 20: Identify PUB of specific LUB..... >	LU70			
	Index 24: Release related programmer LUBs..... >	LU50			
	Index 28: Identify PUB of programmer logical unit..... >	LU80			
	Index 32: Assign SYSLST LUB to specified PUB..... >	LU85			
	Index 36: Unassign SYSLST LUB from specified PUB..... >	LU85			
	Unassign LUB Routine				
LU20	The address of the start of the programmer LUB table part of the partition concerned is now calculated in register 7.			R7	
	The PIK in register 6 is converted to the appropriate index in the FICL, which is then added to the FICL address to address the first-in-class index required. This index is multiplied by 2 (LUB entry size) and added to the LUB address of the partition.			R6 R8	

Labels	Chart GC03: IPW$$LU - Update LUB and PUB Tables		Modifier Data Fields	Reg. Usage	Calls
	If the logical unit found in the specified CCB is a programmer logical unit, LUB table pointer register 7 is positioned correctly and branch to continue..... >	LU22			
	Otherwise, register 7 must be 'backspaced' by the number of system LUBs for the partition to point to the first system LUB in stead of the first programmer LUB.			R7	
	Using register 8 as a work register, the number of system LUBs is retrieved from the NICL, and, after multiplication by 2, subtracted from the first programmer LUB address in register 7.			R8	
LU22	Using register 8 as a work register, the logical unit number concerned is retrieved from the CCB.			R8	
	The logical unit number and programmer LUB indicator in the CCB are reset.		CBLN(IPW$DCB) CBLC(IPW$DCB)		
	Register 7 is updated to point to the LUB entry concerned.			R7	
	If LUB currently unassigned, return..... >	LU90			
	Otherwise, the PUB pointer is loaded in register 8.			R8	
LU23	If the device is a 3800 printer, a branch is made to reset the printer setup with the hardware/system defaults..... >	LUP1		RE	
LU25	Check for DASD. If so, branch to.. >	LU90			
	The address of the PUB ownership table entry for the PUB concerned is loaded in register 4.			R4	
	The ownership code for the partition involved is loaded in register 8, using the partition index in register 6 to address the applicable byte of the internal ownership reference table LU3Z.			R8	
	If not owned by partition concerned, return..... >	LU90			
	Otherwise, the ownership is released, and return..... >	LU90			

Labels	Chart GC03.1: IPW$$LU - Update LUB and PUB Tables		Modifier Data Fields	Reg. Usage	Calls
	Locate Unowned PUB				
LU30	The PIB address is saved in register 9 because the subsequent check for valid hexadecimal digits in the device address specification, passed in register 2, may destroy register 1 contents.			R9	
	The device address as stored in register 2 save area location is checked.				
	PIB address restored in register 1.			R1	
	If specification error, branch to diagnose..... >	LU3X			
	Otherwise, the device address is converted from EBCDIC to hexadecimal.		SVR2(IPW$DSV)		
	PUB table address is loaded in register 7, and PUB ownership table address is loaded in register 8.			R7 R8	
	In the following loop the PUB table is scanned for the specified device address.				

Labels	Chart GC04: IPW$$LU - Update LUB and PUB Tables		Modifier Data Fields	Reg. Usage	Calls
LU32	If end of PUB table reached and PUB not found, branch..... >	LU3X			
	If device address is the one specified, branch to process..... >	LU34			
	Otherwise, register 8 is incremented by 2 to point to the next PUB ownership entry.			R8	
	Register 7 is incremented by 8 to point to the next PUB entry, and branch back to check next PUB entry..... >	LU32		R7	
LU34	If PUB ownership entry indicates that the device is already owned, branch..... >	LU3X			
	If PUB ownership entry indicates that the device is waiting for volume to be mounted, branch..... >	LU3X			
	If PUB entry indicates 'device down', branch..... >	LU3X			
	Otherwise, PUB entry address is loaded in register 2.			R2	
	The device type code from the PUB is inserted in the high order byte of register 2, and the device type as passed in register 2, having been destroyed by the conversion to hexadecimal, is restored to its original EBCDIC representation.		SVR2(IPW$DSV)	R2	
	Register 9 is loaded with the address of the internal EBCDIC device type table LU3Y.			R9	
	In the following loop the device type as specified is matched against device type in the internal table.				
LU36	If device type specified matches device type in current table entry, branch to continue check..... >	LU38			
LU35	If end of table, branch to error exit..... >	LU3X			
	Otherwise, register 9 is incremented by 2 to point to the next table entry, and branch back to continue scan..... >	LU36		R9	

Labels	Chart GC05: IPW$$LU - Update LUB and PUB Tables		Modifier Data Fields	Reg. Usage	Calls
	The high order hexadecimal digit of the one byte internal device type in the PUB is now used to check for the correct device. Since there are two families of reader devices and of punch devices, each of which has a different high-order hexadecimal digit as its characteristic, the following routine is used to determine to which particular reader/punch device family this particular device belongs.				
LU38	The internal device type is loaded in register 4 from the PUB, the low-order 4 bits are set to zero to make direct comparison possible, and, if correct reader/punch type not yet found, branch back to continue table scan..... >	LU35		R4	
	Otherwise, if device tape cartridge reader, error return..... >	LU3X			
	Otherwise, the partition ownership is set for the applicable PUB.				
	Register 6 is shifted to convert the PIK to the appropriate index to the internal PUB ownership mask table LU3Z, and the mask is moved to the PUB ownership table entry.			R6	
	The return value for register 2 (device type + device address) is stored in the task save area location for register 2, and return..... >	LU90	SVR2(IPW$DSV)		
	<u>Locate and Assign LUB Routine</u>				
LU40	The PIK in register 6 is converted to the correct NICL/FICL index for the partition concerned.			R6	
	Using register 4 as a work register, the programmer LUB index is loaded in register 8, and the number of programmer LUBs is loaded in register 9.			R4 R8 R9	
	The address of the first programmer LUB of the partition is then calculated in register 8, and saved in register 7.			R8,R7	
	In the following loop, the first available programmer LUB of the partition is searched. If found, it will be assigned.				

Labels	Chart GC06: IPW$$LU - Update LUB and PUB Tables		Modifier Data Fields	Reg. Usage	Calls
LU44	If LUB to be checked is currently unassigned, branch to assign..... >	LU46			
	Otherwise, register 8 is incremented by 2 to point to the next LUB, and, if not yet all LUBs for this partition have been scanned, branch back to check next programmer LUB.. >	LU44		R8	
	If no LUB available, exit..... >	LU49			
LU46	The high-order byte of register 2 is cleared to strip internal device type, leaving just the PUB address passed by the caller.			R2	
	The start address of the PUB table is subtracted and the remaining displacement is divided by 8 to produce the appropriate PUB pointer, which is stored in the located free LUB, thereby assigning it to the PUB concerned.				
	The LUB displacement is calculated in register 8 and divided by 2 to produce the correct programmer logical unit number, which is then stored in the caller's CCB, pointed to by register 3.			R8	
	The function is completed by setting the 'programmer logical unit' and 'EXCP Real' switches in the caller's CCB.				
	Branch to function exit..... >	LU90			
LU49	Since no LUB is available, LUB number and 'programmer logical unit' switch in the caller's CCB are reset to 0.				
	Branch to function exit..... >	LU90			
	Release Related LUBs Routine				
LU50	The PIB address in register 1 is saved in register 9, because it might be destroyed in the subsequent examination of the device address passed in register 2.			R9	
	The device address is checked for valid hexadecimal digits in EBCDIC representation.				
	The PIB address is restored, and if device address in error, branch..... >	LU3X		R1	

Labels	Chart GC07: IPW$$LU - Update LUB and PUB Tables		Modifier Data Fields	Reg. Usage	Calls
	Otherwise, the device address is converted from EBCDIC to true hexadecimal representation.		SVR2(IPW$DSV)		
	Register 2 is initialized with the PUB table address.			R2	
	The following loop scans the PUB table for the specified device address.				
LU52	If end of PUB table reached, branch to indicate error..... >	LU3X			
	Otherwise, if current PUB is the PUB for the device address requested, branch to process..... >	LU54			
	Otherwise, register 2 is incremented by 8 to point to the next PUB, and branch back to check next PUB..... >	LU52		R2	
LU54	The PUB pointer associated with this PUB is calculated in register 2.			R2	
	The NICL/FICL index is calculated by shifting PIK in register 6.			R6	
	Using register 4 as a work register:			R4	
	• The programmer LUB index is loaded in register 7,			R7	
	• the number of system LUBs is loaded in register 8, and			R8	
	• the number of programmer LUBs is loaded in register 9.			R9	
	The total number of LUBs is then calculated in register 9, and the system LUB index in register 7.			R9 R7	
	The address of the first LUB of the partition concerned is calculated in register 7.			R7	
	In the following loop, all partition LUBs are scanned for a PUB pointer to the specified device. If the device is a 3800 printer, a branch is made to reset the printer setup with the hardware/system defaults.....> If found, the LUB concerned will be unassigned.	LUP1		R2, R4	
LU56	If the LUB to be examined is assigned to the PUB specified, the assignment is reset.				
LU58	Register 8 is incremented by 2 to point to the next LUB.			R8	
	If not all partition LUBs have been scanned yet, branch back to check next LUB..... >	LU56		R9	

Labels	Chart GC08: IPW$$LU - Update LUB and PUB Tables		Modifier Data Fields	Reg. Usage	Calls
	The PUB pointer in register 2 is multiplied by 2 (PUB ownership table entry width) to provide the proper table index.			R2	
	The address of the PUB ownership table entry for the PUB concerned is calculated in register 4.			R4	
	Using the partition index in register 6, the address of the correct ownership mask is loaded in register 2.			R2	
	If the PUB is not owned by the partition concerned, branch to function exit..... >	LU90			
	Otherwise, ownership is released.				
	Branch to function exit..... >	LU90			
	Release Specific PUB Routine				
LU60	The device address passed in register 2 in EBCDIC representation is checked for valid hexadecimal digits.				
	If device address specification in error, branch..... >	LU6X			
	Otherwise, the device address is converted to true hexadecimal.		SVR2(IPW$DSV)		
	The PUB ownership table address is loaded in register 4.			R4	
	The PUB table address is loaded in register 7.			R7	
	The following loop scans the PUB table for the PUB with the specified device address.				
LU62	If the PUB to be checked contains the specified device address, branch to release ownership..... >	LU64			
	Otherwise, register 7 is incremented by 8 to point to the next PUB, and register 4 is incremented by 2 to point to the associated PUB ownership table.			R7 R4	
	If end of PUB table, branch to indicate not found..... >	LU6X			
	Otherwise, branch back to check next PUB..... >	LU62			

Labels	Chart GC09: IPW$$LU - Update LUB and PUB Tables		Modifier Data Fields	Reg. Usage	Calls
LU64	The PIK in register 6 is shifted to convert it to a partition index.			R6	
	Using this index, the address of the correct ownership mask is loaded in register 8, and, if the device concerned is not owned by this partition, a branch is made to the function exit..... >	LU90		R8	
	Otherwise, the ownership is released, and branch to function exit..... >	LU90			
	Identify PUB of Specified LUB				
LU70	The LUB specified, as passed in register 2 in EBCDIC representation, is checked to determine if LST (SYSLST) has been specified. If so, branch..... >	LU74			
	Otherwise, if SYSPCH has been specified, branch..... >	LU72			
	The PIB address in register 1 is saved in register 9 to allow for the subsequent translate and test instruction.			R9	
	The LUB specification passed is checked for valid numerics.				
	PIB address is restored.			R1	
	If LUB specification in error, branch to diagnose..... >	LU7Z			
	Otherwise, the LUB is packed, loaded in register 9, converted to negative to indicate programmer LUB.		SVR2(IPW$DSV)	R7	
	Branch to continue..... >	LU76			
LU72	SYSPCH index (2) is loaded in register 7 (positive to indicate system LUB).			R7	
	Branch to continue..... >	LU76			
LU74	SYSLST index (3) is loaded in register 7 (positive to indicate system LUB).			R7	
LU76	The PIK in register 6 is converted to the proper partition index for FICL/NICL access, and used to load the proper programmer LUB index in register 8, as well as in byte 1 of task save area for register 2.		SVR2(IPW$DSV)	R6 R8	

Labels	Chart GC10: IPW$$LU - Update LUB and PUB Tables		Modifier Data Fields	Reg. Usage	Calls
	If LUB concerned is programmer LUB, branch..... >	LU78			
	Otherwise, using register 9 as a work register, the number of system LUBs is stored in the first halfword of task register 2 save area, and subtracted from the programmer LUB pointer in register 8 to have register 8 contain the LUB index for the first system LUB.		SVR2(IPW$DSV)	R9 R8	
	Branch to continue..... >	LU7A			
LU78	The programmer LUB number is made positive again.			R7	
	Register 9 is set up to point to the number of programmer LUBs for the partition concerned.			R9	
	If the programmer LUB number specified is too high for this partition, branch to diagnose..... >	LU7Z			
LU7A	The (programmer or system) LUB number is multiplied by 2 to obtain the proper LUB displacement.			R7	
	The (programmer or system) first LUB index is multiplied by 2, and both values are added to the address of the start of the LUB table to obtain the address of the LUB concerned.			R8 R7	
	If this LUB is not assigned, branch to diagnose..... >	LU7Z			
	Otherwise, using register 9 as a work register, the PUB pointer is converted to a PUB address in register 8.			R9 R8	
	The device address of this PUB is converted to EBCDIC in the 3 high order bytes of the task register 2 save area.		SVR2(IPW$DSV)		
	The device type is copied from the PUB to the low-order byte of task register 2 save area.		SVR2(IPW$DSV)		
	Branch to function exit..... >	LU90			
LU7Z	Return parameter is made zero by clearing task register 2 save area, and branch to exit..... >	LU90	SVR2(IPW$DSV)		

Labels	Chart GC11: IPW$$LU - Update LUB and PUB Tables		Modifier Data Fields	Reg. Usage	Calls
	Identify PUB and Device Address of System/Programmer Logical Unit				
LU80	Register 7 is loaded with the logical unit and the type code byte is cleared.			R7	
	If the logical unit is a system unit, branch to define the cuu..... >	LU76			
	Otherwise, make the contents of register 7 negative, and branch to define the cuu..... >	LU76		R7	
	Assign/Unassign SYSLST Routine				
LU85	The PIB address in register 1 is saved in register 9, because it might be destroyed in the subsequent examination of the device address passed in register 2.			R9	
	The device address is checked for valid hexadecimal digits in EBCDIC representation.				
	The PIB address is restored and, if device address in error, branch.... >	LU3Z		R1	
	Otherwise, the device address is converted from EBCDIC to true hexadecimal representation.		SVR2(IPW$DSV)		
	Register 2 is initialized with the PUB table address.			R2	
	The following loop scans the PUB table for the specified device address.				
LU87	If end of PUB table is reached, branch to indicate error..... >	LU3Z			
	Otherwise, if current PUB is the PUB for the device address requested, branch to process..... >	LU88			
	Otherwise, register 2 is incremented by 8 to point to the next PUB, and branch back to check next PUB..... >	LU87		R2	
LU88	The start address of the PUB table is subtracted and the remaining displacement is divided by 8 to produce the appropriate PUB pointer.			R2	
	The PIK in register 6 is converted to the proper partition index for FICL/NICL access.			R6	

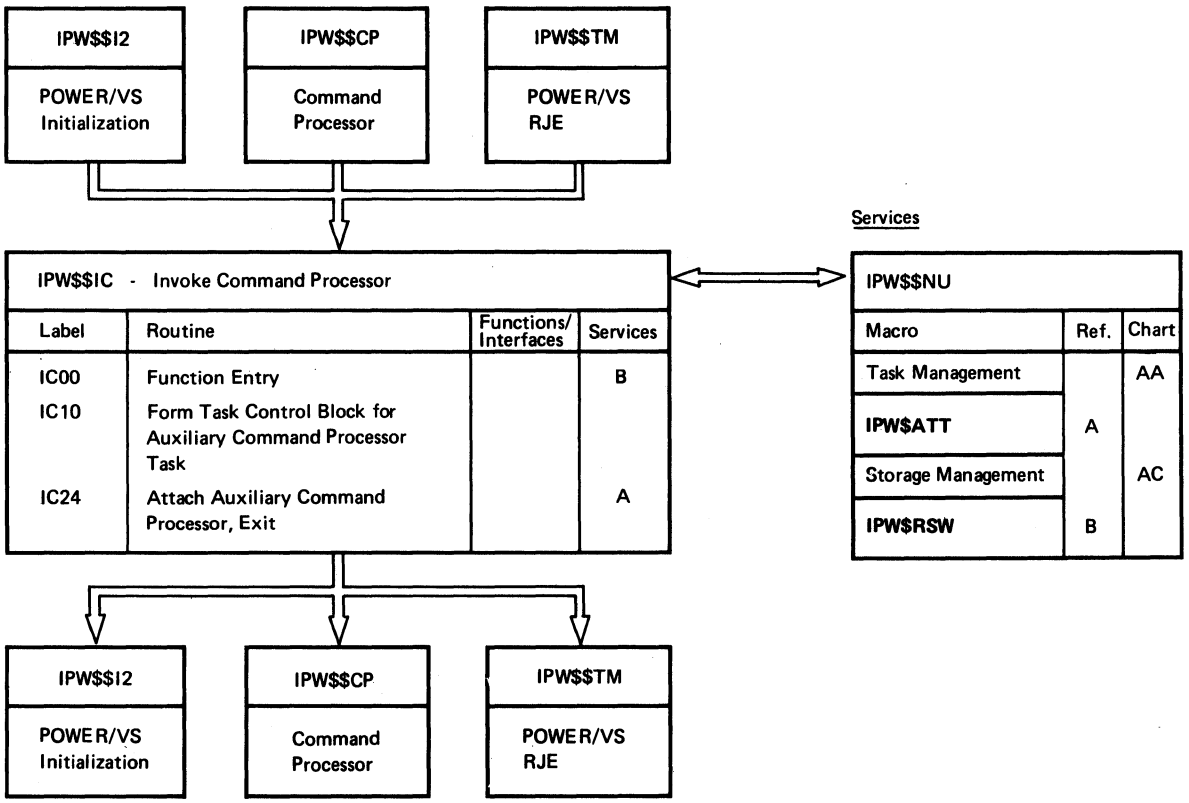
Labels	Chart GC12: IPW$$LU Update LUB and PUB Tables		Modified Data Fields	Reg. Usage	Calls
	Using register 4 as a work register:			R4	
	- the programmer LUB index is loaded into register 7.			R7	
	- the number of system LUBs is loaded into register 8.			R8	
	The index to the system LUBs is calculated in register 7.			R7,R8	
	The address of the SYSLST LUB of the partition concerned is calculated in register 7.			R7,R8	
	If assign function was requested, branch to..... >	LU89			
	If the LUB to be examined is assigned, the assignment is reset. Branch to function exit..... >	LU90			
LU89	If the SYSLST LUB is already assigned, branch to..... >	LU3Z			
	Otherwise, the low-order byte of register 2, containing the PUB index, is stored in the SYSLST LUB, thereby assigning it to the PUB concerned.				
	Function Exit Routine				
LU90	The system is now released by the PUB/LUB update routines:				
	Register 0 is loaded with 255 (X'FF') to signal enabled state, and an SVC 22 is issued to release the system.			R0	
	Caller's registers are restored and control is passed back to the caller.				IPW$RET
	3800 printer setup				
	On entry the following register contents are relevant: 2 index to PUB entry 7 address of LUB 14 return address				
LUP1	Check whether the TCB belongs to the command processor task. If so, return to caller via register 14.			RE	
	If the TCB belongs to the Initiator/ Terminator task or the print status task, return is made to the caller via register 14.			RE	
	The address of the PUB ownership table entry for the PUB concerned is loaded in register 3.			R3	

Labels	Chart GC12.1: IPWSSLU - Update LUB and PUB Tables		Modified Data Fields	Reg. Usage	Calls
	The ownership code for the partition involved is loaded into register 1, using the partition index byte in register 6 to address the applicable byte of the internal ownership reference table LU3Z.				
	If not owned by the partition concerned, return to caller via register 14.				
	Otherwise, register 0 is loaded with 255 (X'FF') to signal enabled state and an SVC 22 is issued to release the system.			R0	
	Load the length of the temporary work space in register 1 and reserve work space.			R1	IPW$RSW Chart AC
	Set up register 4 as a base to the work space area.			R4	
	Copy the storage descriptor into the work space area.		LUWS(LUWS)		
	Save registers 14 and 15.		LUGR(LUWS)		
	The model SETPRT parameter list and the LUB address are moved into the work space.		LUSP(LUWS)	R1	
	Storage is reserved for the service request block (SRB).				IPW$RSW Chart AC
	Register 1 is set up as a base to the SRB.			R1	
	SETPRT request is indicated, and the parm field in the SRB is set up to point to the SETPRT parameter list.		SRBREQ,SRBPARM (IPW$DSR)	R2	
	Temporarily assign a new register save area.		LUSV(LUWS)	RD	
	Examine if the task already uses asynchronous service. If so, branch to..... >	LUP3			
	Otherwise, attach the DOS/VS subtask via IPW$IAS TYPE=ATTACH.				IPW$$AS

Labels	Chart GC12.2: IPW$$LU - Update LUB and PUB Tables	Modified Data Fields	Reg Usage	Calls
LUP3	Pass the SETPRT request to asynchronous service for processing IPW$IAS TYPE=SERVICE.			IPW$$AS Chart
	Unassign the temporary register save area and reassign the original.		RD	
	Restore registers 14 and 15.		RE,RF	
	Release the service request block.			IPW$RLW Chart AC
	Release the work space.			
	Seize the system again for further execution of the function through an SVC 22. Parameter register 0 will contain 255 (X'FF') to signal that interrupts are allowed.		R0	
	Return to the caller via register 14.			

CHART GD: IPW$$IC - INVOKE COMMAND PROCESSOR (4 PARTS)

Chart GD00: IPW$$IC - Invoke Command Processor, General Flow and Macro Calls



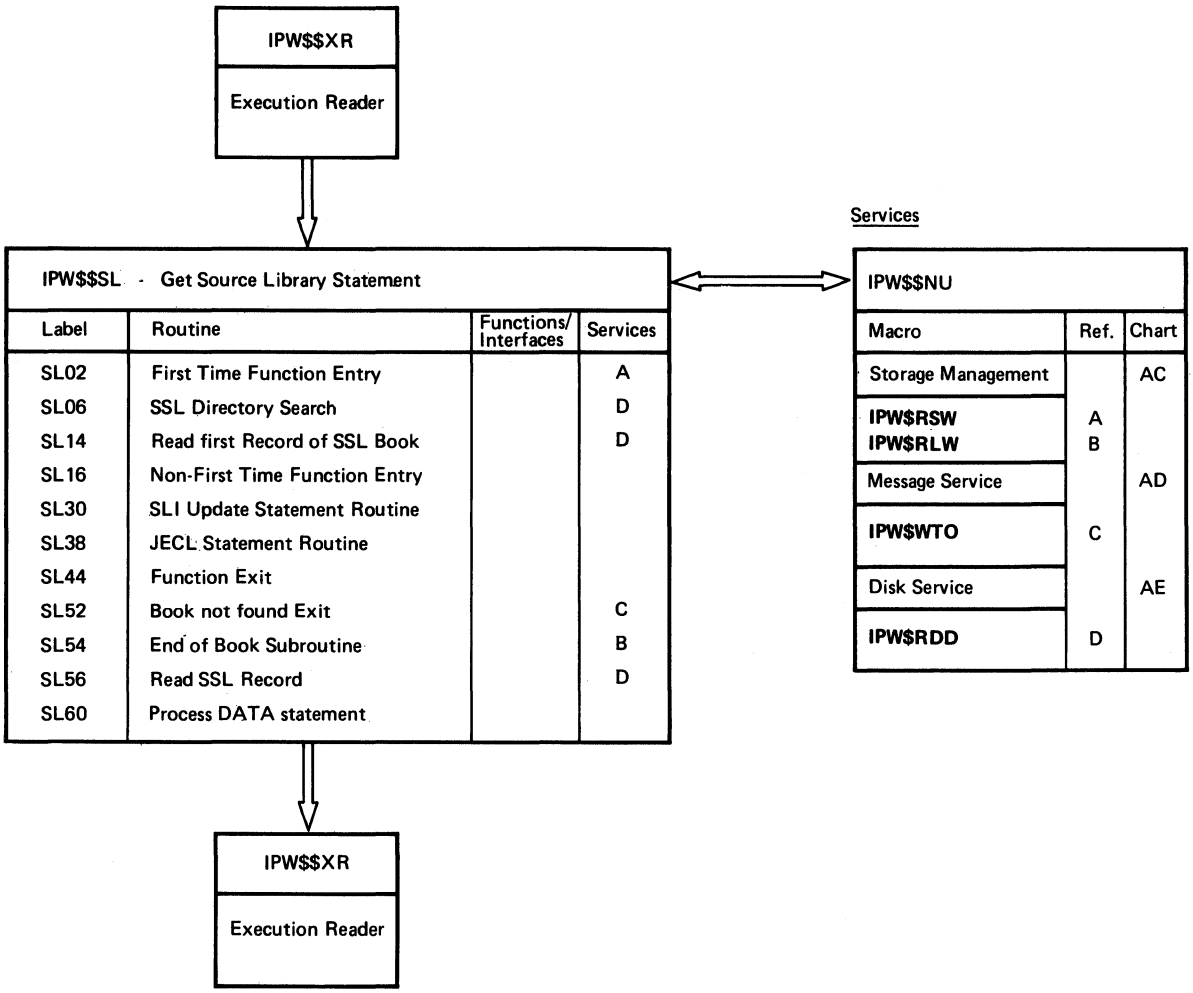
Labels	Chart GD01: IPW$$IC - Invoke Command Processor		Modified Data Fields	Reg. Usage	Calls
	The first 16 bytes constitute the section descriptor: 'ICSD V7M0 ' On entry, the following register contents are relevant: 0: (If nonzero) address of event control block to be posted on function completion 1: address of buffer containing the command to be processed 10: address of POWER/VS permanent area 11: address of TCB 13: address of task save area 14: return address caller 15: function base address		IPW$DPA IPW$DTC IPW$DSV	R0 R1 R10 R11 R13 R14 R15	
	Function Entry				
IC00	Caller's registers are saved.				IPW$SAV
	Form Task Control Block				
	A command processor TCB is reserved.				IPW$RSW Chart AC
	The address of the new TCB is loaded in register 8. Initialization of the new TCB is started: • The storage descriptor is moved in. • The RJE identifier is set from the caller's TCB. • Receiving fields for command and operands are set to blanks. • The caller's ECB address is set.		CPSD(IPW$DTC) CPID(IPW$DTC) CPCM(IPW$DTC) CPOP(TCDS) CPEA(IPW$DTC)	R8	
	The address of the 72 byte buffer containing the command is loaded in register 3. The address of the second-last byte of the buffer is loaded in register 0. If the first 4 bytes of the buffer do not contain an RJE prefix ('*..'), a branch is made to process central operator command..... >	IC10		R3 R0	
	Otherwise, the remote sequence number is moved from buffer to TCB, and register 3 is incremented by four to point past the RJE prefix.		CPNO(IPW$DTC)	R3	

Labels	Chart GD02: IPW$$IC - Invoke Command Processor		Modified Data Fields	Reg. Usage	Calls
IC10	The length of the buffer to be scanned is calculated in register 4, and the buffer is scanned for the command operation code.			R4	
				R1,R2	
	If no operation code is found, a branch is made to bypass further statement checking..... >	IC20			
	Otherwise, if the first character of the operation code is not a P, branch to bypass incrementing operation code pointer register 1 (set by translate and test instruction)..... >	IC15			
	If the first character is a P, register 1 is incremented by one to point past the P.			R1	
IC15	The operation code address is loaded in register 3.			R3	
	Register 1 is initialized with the address of the last operation code byte if the operation code has the maximum length of 8 bytes, in case the subsequent translate and test instruction would not encounter a blank before the end of the field.			R1	
	The buffer is scanned for the end of the operation code.			R1,R2	
	The machine length of the operation code is calculated in register 4			R4	
	The operation code is copied into the new TCB.		CPCM(IPW$DTC)		
	Register 3 is set to point past the operation code.			R3	
	The machine length of the the remaining field to be scanned for operands is calculated in register 4.			R4	
	The buffer is scanned for the first non-blank position, indicating the start of the operand field.			R1,R2	
	If no operand is found, branch to bypass further statement scan..... >	IC20			
	Otherwise, the address of the start of the operand field is loaded in register 3.			R3	
	The length of the remaining operand field is calculated in register 4, and the operand field is moved to the new TCB.		CPOP(IPW$DTC)	R4	

Labels	Chart GD03: IPW$$IC - Invoke Command Processor	Modified Data Fields	Reg. Usage	Calls
	Auxiliary Processor			
IC20	The address of the permanent command processor TCB is in register 3.		R3	
	The command processor TCB storage descriptor is copied to the new TCB.	TCSD(IPW$DTC)		
	The termination code in the new TCB is initialized.	TCTT(IPW$DTC)		
	Using register 1 as a work register, the new task's register 7 is set to address the command processor control block.	TCR7(IPW$DTC)	R1	
	Using register 3 as a work register, the new task's register 8 is set to address the command processor.	TCR8(IPW$DTC)	R3	
	The address to the spool management parameter list (SPL) is initialized if present.	TCPL(IPW$DTC)		
	The address of the new TCB is now loaded in register 1, and the command processor is attached.		R1	IPW$ATT Chart AA
	After attaching command processor, control is returned to the caller.			IPW$RET

CHART GE: IPW$$SL - GET SOURCE STATEMENT LIBRARY RECORD (11 PARTS)

Chart GE00: IPW$$SL - Get Source Statement Library Record, General Flow and Macro Calls



Labels	Chart GE01: IPW$$SL - Get Source Statement Library Record		Modified Fields	Reg. Usage	Calls
SLSD	The first 16 bytes constitute the section descriptor: 'SLCS V7M0 '				
	On entry, the following register contents are relevant:				
	0: address of SSL book name			R0	
	1: book name length			R1	
	10: permanent area address		IPW$DPA	R10	
	11: TCB address		IPW$DTC	R11	
	13: task save area		IPW$DSV	R13	
SL00	This entry point is taken if entry is not made for the first time: branch to continue..... >	SL16			
SL02	This entry point is taken if entry is made for the first time. Caller's registers are saved.				IPW$SAV
	SSL book name pointers are loaded in register 7 (book name address) and			R7	
	register 8 (book name length).			R8	
	Address of DMB is loaded in register 5, and SLI work space is reserved.			R5	IPW$RSW Chart AC
	The virtual address of SLI work space is stored in the partition control		PDSL(IPW$DPD)		
	table, and copied to register 4 to address work space.			R4	
	The book name field in the SLI work space is blanked, and the default		SLBM(IPW$DSL) SLSL(IPW$DSL)		
	sublibrary indicator is copied from the DMB. If a sublibrary has not been specified in the book name passed, branch..... >	SL04			
	Otherwise, the sublibrary specified is moved to SLI work space, register		SLSL(IPW$DSL)		
	7 is incremented by two to point past the sublibrary indicator, and regis-			R7	
	ter 8 is decremented by two to contain the length of the book name only.			R8	
	If register 8 is negative (bookname=S.), a branch is made to >	SL52			
SL04	The book name is moved to the SLI work space. If the first two characters of the book are "$$", the second "$" is changed to "B" or "1" to "4" depending on the partition being serviced.		SLBM(IPW$DSL)		
SL05	Using registers 1 and 2 as work registers, the real address of the SLI I/O area in the SLI work space is			R1,R2	
	calculated in register 0, and stored in the DRW within SLI work space.		SLSA(IPW$DSL)	R0	

Labels	Chart GE02: IPW$$SL - Get Source Statement Library Record		Modified Fields	Reg. Usage	Calls
	The skeleton seek address for a private SSL is moved to the DRW.		SLSW(IPW$DSL)		
	The MCB address of the private SSL is loaded in register 7.			R7	
	If a private SSL is present, branch to continue..... >	SL08			
	Otherwise, the address of the system SSL MCB is loaded in register 7.			R7	
SL06	The skeleton seek address for a system SSL is moved to the DRW.		SLSW(IPW$DSL)		
SL08	The start address of the SSL is moved from SSL MCB to the seek address, and register 8 is set to point 80 bytes past the beginning of the SSL data block, to skip the first five 16-byte entries.		SLSW+3	R8	
	The first SSL data block is read in.				IPW$RDD Chart AE
SL10	The SSL directory entries, pointed to by register 8, are now scanned:				
	If end of directory reached, branch..... >	SL12			
	If current directory entry is book name requested, branch..... >	SL14			
	Otherwise, register 8 is incremented by 16 to point to the next entry.			R8	
	If the buffer has not yet been exhausted, branch back to check the next entry..... >	SL10		R2	
	Otherwise, link to read the next SSL block..... >	SL56		R14	
	Branch to continue scan..... >	SL10			
SL12	The MCB address of the system SSL is loaded in register 1, and if register 7 was pointing already to the system SSL, implying that a possible private SSL directory has been scanned already, and the requested book name has not been found, branch to exit..... >	SL52		R1 R7	
	Otherwise, the system SSL MCB address is loaded in register 7, and branch back to scan system SSL..... >	SL06		R7	

Labels	Chart GE03: IPW$$SL - Get Source Statement Library Record		Modified Fields	Reg. Usage	Calls
SL14	The DRW is now prepared to address the beginning of the requested book rather than its directory entry:				
	Cylinder, head and record address are copied from the directory entry to seek address in the DRW.		SLSW(IPW$DSL)		
	The two high-order bits in the head address byte are set to zero.		SLSW(IPW$DSL)		
	Using register 2 as a work register, the C1 byte is isolated and moved to the DRW.		SLSW(IPW$DSL)	R2	
	The first data block of the SSL book is now read in.				IPW$RDD Chart AE
	Register 8 is loaded with the data area address.			R8	
	Read reader switch and read SSL book switch are both set to R.		SLRR(IPW$DSL) SLRS(IPW$DSL)		
	Branch to take normal exit..... >	SL50			
	This entry is taken whenever a request is received for a logical record from an SSL book that is already being read.				
	The next sequential record is provided and eventual SLI update processing is taken care of.				
SL16	Caller's registers are saved.				IPW$SAV
	Registers 4, 8 and 5 are set up to address the SLI work space, the SSL logical record, and the DMB, respectively.			R4,R8 R5	
	Save general purpose byte, and set it to X'00' to indicate normal record.		SLGP(IPW$DSL) TCGP(IPW$DTC)		
	If not in data mode (implying read from SSL), branch..... >	SL20			
	Otherwise, if last data record read is '/*' (end of data), branch to terminate data mode..... >	SL18			
	Otherwise, if not '/*', branch to exit and continue data mode processing..... >	SL46			
SL18	The first byte of the logical record is set to blank to force end of data mode.		SLLR(IPW$DSL)		
	Branch to exit..... >	SL46			

Labels	Chart GE04: IPW$$SL - Get Source Statement Library Record		Modified Fields	Reg. Usage	Calls
SL20	Read reader switch is set to blank.		SLRR(IPW$DSL)		
	If read SSL switch is not R, indicating that the previous SSL record was not processed due to an SLI update statement, branch to process the next SLI update statement..... >	SL30			
	Otherwise, the read SSL switch is reset to blank.		SLRS(IPW$DSL)		
	Branch to obtain the next SSL record..... >	SL22			
	This routine builds a decompressed logical record in SLI work space from the available physical block.				
SL22	The logical record area is blanked out.		SLLR(IPW$DSL)		
	Register 9 is initialized with the logical record address.			R9	
SL24	The byte pointed to by the physical record pointer register 8 is checked for binary zero, indicating the end of block.				
	If end of block not yet reached, branch to continue..... >	SL26			
	Otherwise, if also end of SSL book, branch to exit..... >	SL54			
	If end of block, but not yet end of book, link to obtain next SSL block..... >	SL56			
	Branch back to scan next block..... >	SL24			
SL26	The number of non-blanks in the next string is loaded in register 2, and the number of consecutive compressed blanks following it is loaded in register 3.			R2 R3	
	If the length of the non-blank string is zero, branch to bypass moving the string to the logical record..... >	SL28			
	Otherwise, the string is moved to the logical record.				
SL28	Physical record pointer register 8 and logical record pointer register 9 are incremented to point to the next non-blank/blank sequence.			R8 R9	

Labels	Chart GE05: IPW$$SL - Get Source Statement Library Record		Modified Fields	Reg. Usage	Calls
	If end of logical record not yet reached, branch back to continue logical record build..... >	SL24		R1	
	This routine processes possible $SLI update statements.				
SL30	Using register 1 to address the ID-sequence field of the reader record last read in, a check is made for a $SLI update statement in the reader.			R1	
	If the reader record last read in is not a $SLI update statement, branch to process the SSL record read in.. >	SL38			
	Otherwise, if no 'delete' statement, branch to continue..... >	SL32			
	If 'delete' statement, the sequence numbers of the $SLI update statement and the current SSL statement are compared.				
	If the sequence number of the $SLI update statement is high, implying that the statement to be deleted has not yet been encountered, branch to process SSL record..... >	SL38			
	Otherwise, the read reader switch is set to R to cause the next reader record to be read.		SLRR(IPW$DSL)		
	If equal, branch back to obtain the next SSL record, hereby effectively deleting the current SSL record..... >	SL22			
	Otherwise, branch to process current SSL record..... >	SL38			
SL32	If the $SLI update statement is not an 'insert after' statement, branch to continue check..... >	SL34			
	Otherwise, if the sequence field of the $SLI update statement is not lower than that of the current SSL statement, branch to process the current SSL record..... >	SL38			
	Otherwise, branch to pass the reader record first..... >	SL46			

Labels	Chart GE06: IPW$$SL - Get Source Statement Library Record		Modified Fields	Reg. Usage	Calls
SL34	If the $SLI update statement is not an 'insert before' statement, and must therefore be a 'replace' statement, branch..... >	SL36			
	Otherwise, if the sequence number of the $SLI statement is higher than that of the current SSL statement, branch to process the current SSL statement..... >	SL38			
	Otherwise, branch to pass the $SLI statement to be inserted first..... >	SL46			
SL36	If the SSL statement to be replaced has not yet been reached, branch to process the current SSL record..... >	SL38			
	If it has been passed already, branch to pass the $SLI statement first... >	SL46			
	If the current SSL statement is to be replaced, the read SSL switch is set to R.		SLRS(IPW$DSL)		
	Branch to pass the $SLI statement.. >	SL46			
	This routine processes and passes the current SSL record.				
SL38	If the current SSL record is not a JECL statement, branch to pass it to the caller..... >	SL44			
	Otherwise, the JECL statement is scanned for the operation code.			R1	
	If not found, branch to pass current SSL record..... >	SL44			
	If not POWER/VS end of data record ('* $$/*'), branch..... >	SL40			
	Otherwise, the record is changed to normal DOS/VS end of data record ('/*').		SLLR(IPW$DSL)		
	Branch to pass modified SSL record..... >	SL44			
SL40	If the current SSL JECL statement is not a POWER/VS end of job statement ('* $$/&'), branch to check for '* $$DATA' statement..... >	SL42			
	Otherwise, the statement is changed to normal DOS/VS EOJ statement.		SLLR(IPW$DSL)		
	Branch to pass SSL record..... >	SL44			

Labels	Chart GE07: IPW$$SL - Get Source Statement Library Record		Modified Fields	Reg. Usage	Calls
SL42	Parameter registers 0 and 1 are loaded with address and length of the SSL record, respectively.			R0,R1	
	Link to check for '* $$DATA' statement..... >	SL60		R14	
	On return, if register 1 contents (data name length) are zero, indicating no '* $$DATA' statement, branch to pass SSL record..... >	SL44			
	Otherwise, data name address in register 0 is copied to register 9.			R0,R9	
	Data name length is copied to register 7.			R7	
	Parameter registers 0 and 1 are now loaded with the address and length of the reader record last read in.			R0,R1	
	Link to check reader record for DATA statement..... >	SL60		R14	
	On return, if no DATA statement read, branch to ignore SSL DATA record... >	SL22			
	Otherwise, SSL and reader DATA names are compared, using register 2 and register 7 as work registers.			R2,R7	
	If not equal, branch to ignore SSL data record..... >	SL22			
	Otherwise, the read reader switch is set to I to cause the reader record to be ignored.		SLRR(IPW$DSL)		
	Branch to exit..... >	SL48			
SL44	The SSL record is passed:				
	The SSL record length is loaded in register 1.			R1	
	SSL record address is loaded in register 0.			R0	
	Read SSL switch is set to R.		SLRS(IPW$DSL)		
	Branch to return to caller..... >	SL50			
SL46	The reader switch is set to R.		SLRR(IPW$DSL)		

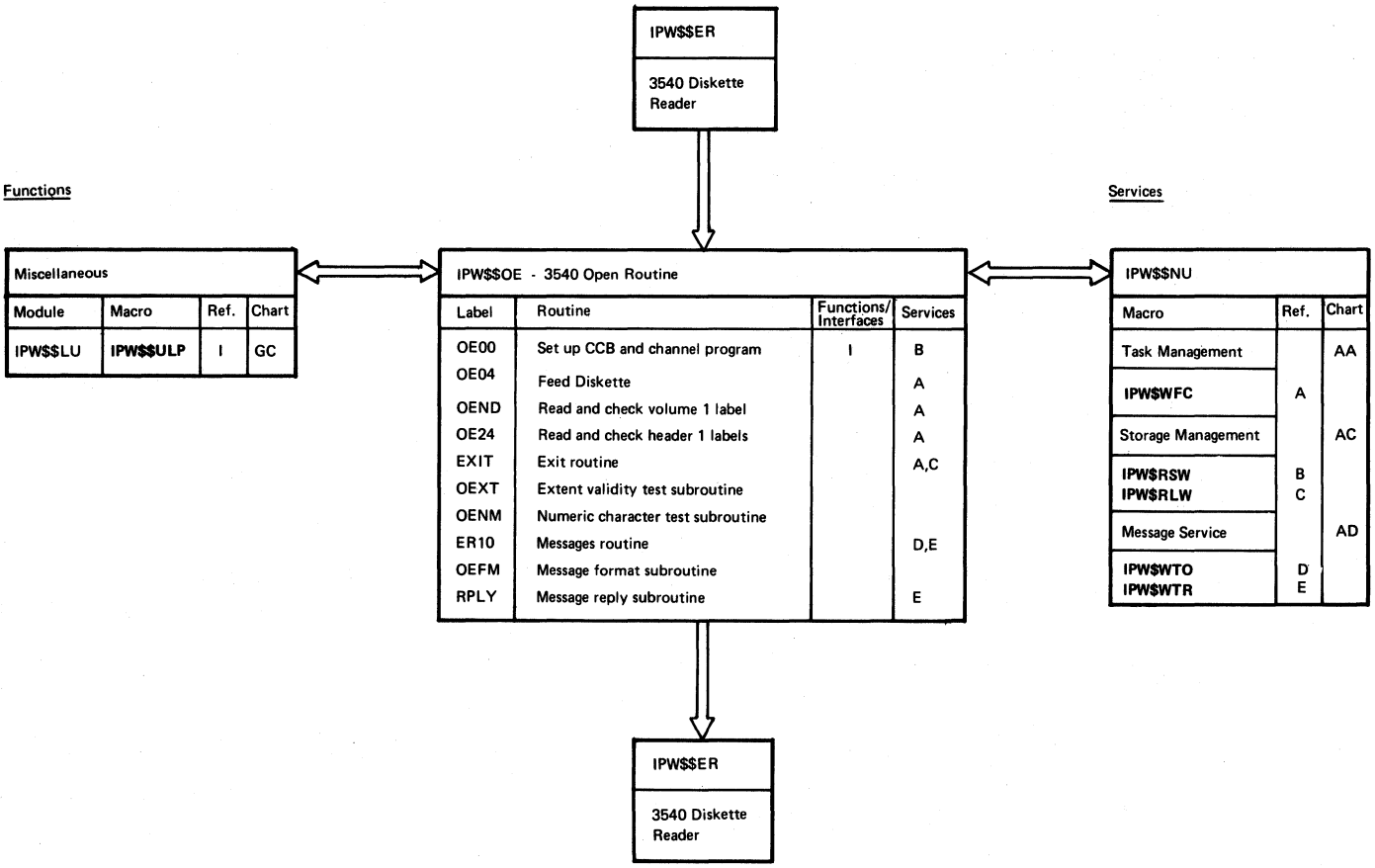
Labels	Chart GE08: IPW$$SL - Get Source Statement Library Record		Modified Fields	Reg. Usage	Calls
SL48	The reader record is passed to the caller:				
SL48	Reader record address is loaded in register 0.			R0	
	Reader record length is loaded in register 1.			R1	
	The general purpose byte is restored.		TCGP(IPW$DTC)		
SL50	The record pointer registers 0 and 1 are stored in the task save area.		SVR0(IPW$DSV)	R0,R1	
	SSL current block pointer register 8 is saved in the SLI work space.		SLR8(IPW$DSV)	R8	
	Return is made to the caller.				IPW$RET
	This routine is entered if the specified book is not found.				
	Diagnostic message 1Q44I is issued and SLI processing is terminated.				
SL52	Using register 2 as a work register, the address of message 1Q44I is moved to the message request word.		TCMW(IPW$DTC)	R2	
	Message 1Q44I is issued.				IPW$WTO Chart AD
	Branch to terminate SLI processing..... >	SL54			
	This routine is entered if a requested SSL book is not found or if the end of the SSL book being processed is encountered. SLI processing is terminated by releasing SLI work space and on exit the current reader record is passed to the caller.				
SL54	IPW$RDW parameter register 1 is loaded with the SLI work space address.			R1	
	The general purpose byte is restored.		TCGP(IPW$DTC)		
	SLI work space is released.				IPW$RLW Chart AC
	Work space pointer in the partition control block is set to zero.		PDSL(DPD)		
	Current reader record address is loaded in register 0.			R0	
	Current reader record length is loaded in register 1.			R1	

Labels	Chart GE09: IPW$$SL - Get Source Statement Library Record		Modified Fields	Reg. Usage	Calls
	Record pointer registers are saved in the task save area.		SVR0(IPW$DSV)		
	Return is made to the caller.				IPW$RET
	This routine reads the next sequential SSL data block.				
SL56	The disk address of the current SSL data block is incremented:				
	Record number from DRW is loaded in register 1, incremented by one, and stored back into DRW.		SLSW(IPW$DSL)	R1 R1	
	If updated record number is not higher than the number of records per track, branch to read next block... >	SL58			
	Otherwise, the record number is reset to one.		SLSW(IPW$DSL)		
	Using register 1 as a work register, the current head number is incremented by one.		SLSW(IPW$DSL)	R1	
	If end of cylinder not yet reached, branch to read next block..... >	SL58			
	Otherwise, the head number is reset to zero.		SLSW(IPW$DSL)		
	Using register 1 as a work register, the current cylinder number is incremented by one.		SLSW(IPW$DSL)	R1	
SL58	The next SSL data block is read in.				IPW$RDD Chart AE
	SSL block data pointer register 8 is loaded with the address of the first byte of the SSL block just read in.			R8	
	Branch to return to caller..... >	(R14)			
	This routine checks whether the logical record passed to it through registers 0 (record address) and 1 (record length) is a '* $$DATA' statement.				
	Presence of the JECL '* $$' prefix is assumed, no test is made to check for this prefix.				
	If the current record is indeed a DATA statement, the DATA name is passed to the caller through register 0 (DATA name address) and register 1 (DATA name length).				

Labels	Chart GE10: IPW$$SL - Get Source Statement Library Record		Modified Fields	Reg. Usaage	Calls
	If the current record is not a DATA record, a zero DATA name length is passed back to the caller in register 1.				
SL60	The address of the last byte of the logical record passed is loaded in register 3.			R3	
	Skipping the possible '* $$' prefix, the rest of the statement is scanned for the operation code.			R1 R2	
	If not found, branch to signal 'not found'..... >	SL64			
	Otherwise, if not DATA, branch to signal 'not found'..... >	SL64			
	If DATA, the remainder of the statement is scanned for the start of the DATA name.			R1 R2	
	If not found, branch to signal 'not found'..... >	SL64			
	Otherwise, the start of the DATA name is saved in register 0.			R0	
	The name is scanned to obtain its ending address in register 1.			R1	
	If not found, branch to assume name end is statement end..... >	SL62			
	The start address is subtracted to obtain the name length in register 1.			R1	
	Branch to return to caller..... >	(R14)			
SL62	Assuming statement end is name end, name length is calculated in register 1.			R3,R1	
	Branch back to caller..... >	(R14)			
SL64	Register 1 is set to zero to indicate that no DATA name has been found.			R1	
	Branch back to caller..... >	(R14)			

CHART GF: IPW$$OE - 3540 OPEN ROUTINE (14 PARTS)

Chart GF00: IPW$$OE - 3540 Open Routine, General Flow and Macro Calls



Labels	Chart GF01: IPW$$OE - 3540 Open Routine	Modified Data Fields	Reg. Usage	Calls
	This routine is entered when the 3540 Diskette Reader (IPW$$ER) issues an IPW$OEF macro.			
OECS	CSECT name.			
OESD	The first 16 bytes constitute the section descriptor: 'OECS V7M0 '			
	General Register Usage			
	0 - **** - Service work register		R0	
	1 - **** - Service work register		R1	
	2 - **** - Service work register		R2	
	3 - **** - Service work register		R3	
	4 - **** - Work register		R4	
	5 - **** - Not used		R5	
	6 - **** - Internal link register in message/reply routine		R6	
	7 - **** - Pointer to work space buffer		R7	
	8 - IPW$DPW - Physical work space		R8	
	9 - **** - Not used		R9	
	10 - IPW$DPA - POWER/VS nucleus		R10	
	11 - IPW$DTC - Task control block		R11	
	12 - **** - Asynchronous address register		R12	
	13 - IPW$DSV - Task save area		R13	
	14 - **** - Link register		R14	
	15 - OECS - Base register		R15	
	Set Up CCB and Channel Program			
OE00	Save caller's registers.		R14-R9	IPW$SAV
	Load the length of the 3540 work space buffer into register 1 and reserve the work space.		R1	IPW$RSW Chart AC
	Set up register 7 as a base to the work space area.		R7	
	Calculate the negative displacement between the real and virtual work space addresses in register 4.		R4	
	Copy the storage descriptor into the work space area.	OEWS(OEWS)		
	Build the 3540 CCB:			
	• Set the communication byte to X'94' to indicate transmit information	OECM(OEWS)		
	• Set device type and logical unit	OELU(OEWS)		
	• Set the mode bits off	OELU(OEWS)		
	• Set the CCB in real mode	OELU(OEWS)		
	• Store the real address of the channel program	OECA(OEWS)		

Labels	Chart GF02: IPW$$OE - 3540 Open Routine		Modified Data Fields	Reg. Usage	Calls
	Build the 3540 channel program:				
	• Move the CCWs		OECP(OEWS)		
	• Relocate and store the real area addresses		OEDO(OEWS) OESK(OEWS) OERD(OEWS)		
	Initialize and store the length of the reply buffer area.		OERL(OEWS)		
	Load the POWER/VS partition PIB address into register 1 and the 3540 programmer logical unit into register 2.			R1,R2	
	Temporarily assign a new register save area (OESV).		OESV(OEWS)	R4	
	Determine the 3540 device address (cuu).				IPW$ULP Chart GC
	Unassign the temporary register save area and reassign the original.				
	Move the device address into the storage descriptor.		OEWS+13		
	The following indicators are copied from the physical work space to prevent them from destruction if open is unsuccessful:				
	• number of 3540 volumes		WEND(OEWS)		
	• record length		WERL(OEWS)		
	• sequence identification		WESI(OEWS)		
	• number of opened diskettes		WEOD(OEWS)		
	Feed Diskette				
OE04	Check for more than one volume left. If so, branch to..... >	OE08			
	Check for single volume file. If not, branch to..... >	OEFD			
OE08	Check if the volume has already been opened. If not, branch to read the volume label..... >	OEND			
OEFD	Insert the feed command.		OEDO(OEWS)		
	Unchain the CCW string.		OEDO+4(OEWS)		
	Feed the diskette via SVC 0 and wait for completion.				IPW$WFC Chart AA
	Reset the define operations command.		OEDO(OEWS)		
	Chain the CCW to the string again.		OEDO+4(OEWS)		

Labels	Chart GF03: IPW$$OE - 3540 Open Routine		Modified Data Fields	Reg. Usage	Calls
	Read and Check Volume 1 Label				
	Read Volume 1 Label				
OEND	Reset the seek argument to the initial value X'00000007'.		OESA(OEWS)		
	Load the virtual CCB address into register 1, read the volume label via SVC 0 and wait for completion.			R1	IPW$WFC Chart AA
	Set the define operations command to a NOP command.		OEDO(OEWS)		
	Check for unit check. If so, branch to..... >	OE12			
	Check the volume 1 label. If valid, branch to..... >	OE16			
OE12	Set up for issuing message '1Q91D VOL1 LABEL ERROR OR NOT FOUND R=', and branch to..... >	ER10	OML2(OEWS) OMT2(OEWS)		
OE16	Check for label standard version (W). If so, branch to..... >	OE12			
	Set up for issuing message '1Q91D LABEL STANDARD VERSION VIOLATION R=', and branch to..... >	ER10	OML2(OEWS) OMT2(OEWS)		
OE20	Check for volume security. If so, branch and link for operator intervention..... >	ER30		R14	
	Check for basic exchange volume. If so, branch to..... >	OE24			
	Set up for issuing message '1Q91D NON-BASIC EXCHANGE DISKETTE TYPE R=', and branch to..... >	ER10	OML2(OEWS) OMT2(OEWS)		
	Read and Check Header 1 Labels				
OE24	Check if the last sector has already been read. If not, branch to..... >	OE26			
	If multivolume file, give message.. >	ER20			
	If no automatic feed is already given, feed next diskette by branching to..... >	OEFD	OEAF(OEWS)		
	Otherwise, reset automatic feed indicator and give message..... >	ER20	OEAF(OEWS)		
OE26	Using register 1 as a work register, update the seek argument to the next sector.		OESA(OEWS)	R1	
	Load the virtual CCB address into register 1, read the header 1 label via SVC 0, and wait for completion.			R1	IPW$WFC Chart AA
	Check the header 1 label. If not valid, branch to read the next label..... >	OE24			

Labels	Chart GF04: IPW$$OE - 3540 Open Routine		Modified Data Fields	Reg. Usage	Calls
	Check for a matching file ID. If so, branch to..... >	OE28			
	If the file ID has been omitted, and if the file is secured, branch to read the next label..... >	OE24			
OE28	Otherwise, check for basic exchange data. If so, branch to..... >	OE32			
	Set up for issuing message 1Q91D NON-BASIC EXCHANGE ffffffff FILE R=', and branch to..... >	ER12	OML2(OEWS) OMT2(OEWS)		
OE32	Check if the file has to be bypassed. If not, branch to..... >	OE36			
	Set up for issuing message 1Q91D ffffffff BYPASS REQUIRED R=, and branch to..... >	ER12	OML2(OEWS) OMT2(OEWS)		
OE36	Check if the file is secured. If so, branch and link for operator intervention..... >	ER30		R14	
OE40	Check if data/file verification is required. If not, branch to..... >	OE44			
	If the data/file has not been verified, branch to..... >	ER50			
OE44	Copy the multivolume indicator into the work space.		WEMI(OEWS)		
	Check for a continuation file. If not, branch to..... >	OE48			
	Check if more volumes are to be expected. If so, branch to..... >	OE56			
	Branch and link to..... >	ER90		R14	
	Branch to continue..... >	OE56			
OE48	Set the multivolume indicator to X'40'.		WEMI(OEWS)		
	If the multivolume indicator was not specified in the header 1 label, or if the last volume has been encountered, branch to..... >	OE52			
	Set up for issuing message '1Q91D MULTIVOLUME IND NOT C, L, OR BLANK R=', and branch to..... >	ER10	OML2(OEWS) OMT2(OEWS)		
OE52	Check if more volumes are to be expected. If so, branch and link to..... >	ER80		R14	
	Set the multivolume indicator to C'L' to simulate last volume.		WEMI(OEWS)		

Labels	Chart GF05: IPW$$OE - 3540 Open Routine		Modified Data Fields	Reg. Usage	Calls
OE56	Load the length of the volume sequence number into register 0 and the address into register 1, and branch and link to the numeric character test subroutine..... >	OENM		R0,R1, R14	
	If the volume sequence number is invalid set up for issuing message '1Q91D VOL SEQ NO. ERR HDR1 LABEL ( ) R=', and branch to..... >	ER10	OML2(OEWS) OMT2(OEWS)		
OE60	If sequence checking is not required, branch to..... >	OE64			
	Check if the sequence numbers match. If not, branch and link to..... >	ER40		R0,R1, R14	
OE64	Store the new sequence number.		WESN(OEWS)		
	Load the length of the block length field into register 0, and the address into register 1.			R0,R1	
	Set up for issuing message '1Q91D ---------- ERR HDR1 LABEL ( ) R='.		OML2(OEWS) OMT2(OEWS)		
	Branch and link to the numeric character test subroutine..... >	OENM		R14	
	If the block length is invalid, branch to..... >	OE72			
	Check if the file is a data file or a SYSIN file. If SYSIN file, branch to..... >	OE68			
	Check the limits of the data file (L=1-128):				
	If invalid, branch to..... >	OE72			
	Otherwise, branch to..... >	OE76			
OE68	Check the limits of the SYSIN file (L=80-81):				
	If valid, branch to..... >	OE76			
OE72	Set up for issuing message '1Q91D BLOCKLENGTH ERR HDR1 LABEL ( ) R=', and branch to..... >	ER10	OMT2(OEWS)		
OE76	Store the correct block length.		WERL(OEWS)		
	Load the begin extent address into register 1, and branch and link to the extent validity test subroutine..... >	OEXT			
	If the address is invalid, set up for issuing message '1Q91D BEGINEXTENT ERR HDR1 LABEL ( ) R=', and branch to..... >	ER10	OMT2(OEWS)		

Labels	Chart GF06: IPW$$OE - 3540 Open Routine		Modified Data Fields	Reg. Usage	Calls
OE80	Store the correct begin extent address.		PELO(IPW$DPW)		
	Load the end extent address into register 1, and branch and link to the extent validity test subroutine >	OEXT		R1,R14	
	If the address is invalid, set up for issuing message '1Q91D END EXTENT ERR HDR1 LABEL ( ) R=', and branch to..... >	ER10	OMT2(OEWS)		
OE84	Save the correct end extent address in register 4.			R4	
	If the end extent is below the begin extent, set up for issuing message '1Q91D ffffffff END XTNT BELOW BEGIN XTNT R=' and branch to..... >	ER12			
OE86	Load the end-of-data address into register 1, and branch and link to the extent validity test subroutine..... >	OEXT		R1,R14	
	If the end-of-data address is invalid, set up for issuing message '1Q91D END-OF-DATA ERR HDR1 LABEL ( ) R=', and branch to..... >	ER10	OMT2(OEWS)		
OE88	Check if the EOD address is higher than the upper limit extent, and if so, then not more than one sector higher.				
	If the EOD address is correct, branch to..... >	OE96			
OE92	Set up for issuing message '1Q91D END-OF-DATA ERR HDR1 LABEL ( ) R=', and branch to..... >	ER10	OMT2(OEWS)		
OE96	Check if the EOD address is below the begin extent. If so, set up for issuing message '1Q91D EOD ADDR BELOW BEGIN XTNT R=', and branch to >	ER12			
	Check for an empty file (EOD=beginning of file). If so, branch and link to..... >	ER60		R14	
	Update the physical work space:				
	• EOD address		PEED(IPW$DPW)		
	• Updated number of volumes to be read		PEND(IPW$DPW)		
	• Record length		PERL(IPW$DPW)		
	• Sequence ID		PESI(IPW$DPW)		
	• Number of successfully opened volumes		PEOD(IPW$DPW)		
	• Open return code (set to C'0')		PEOC(IPW$DPW)		

Labels	Chart GF07: IPW$$OE - 3540 Open Routine		Modified Data Fields	Reg. Usage	Calls
	Exit Routine				
EXIT	Change the command code to a define operations command.		OEDO(OEWS)		
	Unchain the CCW from the string.		OEDO+4(OEWS)		
	Set the mode to read mode and enable unit checks.		OESM+1(OEWS)		
	Load the virtual CCB address into register 1, issue the define operations command via SVC 0 and wait for completion.			R1	IPW$WFC Chart AA
	Release the work space.				IPW$RLW Chart AC
	Return to caller (IPW$$ER) via link register 14.			R14	IPW$RET
	Extent Validity Test Subroutine				
OEXT	Save the return address.		OESV(OEWS)		
	Load the length into register 0, and branch and link to the numeric character test subroutine..... >	OENM		R0,R14	
	If the field is in error, restore the return address and return to caller via link register 14 with a displacement of 4.			R14	
XT04	Check the validity of the cylinder address:				
	If valid, branch to..... >	XT08			
	Otherwise, return to caller via link register 14 with a displacement of 4.			R14	
XT08	Save the cylinder number.		OECC(OEWS)		
	Check the validity of the track address which must be zero:				
	If invalid, return to caller via link register 14 with a displacement of 4.			R14	
	Load the length of the sector field into register 0 and branch and link to the numeric character test subroutine..... >	OENM		R0,R14	
	If the field is in error, restore the return address and return to caller via link register 14 with a displacement of 4.			R14	

Labels	Chart GF08: IPW$$OE - 3540 Open Routine		Modified Data Fields	Reg. Usage	Calls
XT12	Check the validity of the sector address:				
	If invalid, return to caller via link register 14 with a displacement of 4.			R14	
	Otherwise, store the converted extent address (00CC00RR) into register 0 and return to caller via link register 14.			R0,R14	
	Numeric Character Test Subroutine				
OENM	Set up work registers 2 and 3 for examining the characters for numeric validity (C'0' - C'9').			R2,R3	
NM04	Check for a blank character. If not, branch to..... >	NM08			
	If all characters are blank, return to caller with reg. 0 set to zero.			R0,R14	
NM08	Check the characters for numeric validity.				
	If invalid, return to caller via link register 14 with a displacement of 4.			R14	
	Otherwise, convert the field into hexadecimal in register 0, and return to caller via link register 14.			R0,R14	
	Messages Routines				
ER10	Clear register 2 to indicate to the message format subroutine that no file ID substitution is required.			R2	
ER12	Move the message text of message 1Q91D to the output area.		OML1(OEWS)		
	Move the message identity to the second line of the message.		OMI2(OEWS)		
	Load the address of the cuu field to be substituted in the message text into register 1.			R1	
	Branch and link to the message format subroutine..... >	OEFM		R6	
	Issue the message.				IPW$WTO Chart AD
	Branch and link to the message reply routine..... >	RPLY		R6	
	If the reply is 'IGNORE', which is not accepted as a valid response, return to the reply routine via register 6.			R6	

Labels	Chart GF09: IPW$$OE - 3540 Open Routine		Modified Data Fields	Reg. Usage	Calls
ER20	Move the message text of message 1Q92D to the output area.		OML2(OEWS)		
	Load the address of the cuu field to be substituted in the message text into register 1 and the address of the file ID into register 2.			R1,R2	
	Branch and link to the message format subroutine..... >	OEFM		R6	
	Branch and link to the message reply routine..... >	RPLY		R6	
	If the reply is 'IGNORE', which is not accepted as a valid response, return to the reply routine via register 6.			R6	
ER30	Move the message text of message 1Q93D to the output area.		OML2(OEWS)		
	Load the address of the cuu field to be substituted in the message text into register 1.			R1	
	Clear register 2 to indicate no file ID substitution.			R2	
	Branch and link to the message format subroutine..... >	OEFM		R6	
	If it is a volume label, change FILE into VOLUME in the message text.		OMS2(OEWS)		
ER32	Branch and link to the message reply routine..... >	RPLY		R6	
	If the reply is 'IGNORE', return to caller via link register 14.			R14	
ER40	Move the message text of message 1Q94D to the output area.		OML2(OEWS)		
	Move the sequence number, converted to decimal, to the message text.		OMS2(OEWS)		
	Load the address of the cuu field to be substituted in the message text into register 1.			R1	
	Clear register 2 to indicate no file ID substitution.			R2	
	Branch and link to the message format subroutine..... >	OEFM		R6	
	Branch and link to the message reply routine..... >	RPLY			
	If the reply is 'IGNORE', return to caller via link register 14.			R14	

Labels	Chart GF10: IPW$$OE - 3540 Open Routine		Modified Data Fields	Reg. Usage	Calls
ER50	Move the message text of message 1Q95D to the output area.		OML2(OEWS)		
	Load the address of the cuu field to be substituted in the message text into register 1 and the address of the file ID into register 2.			R1,R2	
	Branch and link to the message format subroutine..... >	OEFM		R6	
	Branch and link to the message reply routine..... >	RPLY		R6	
	If the reply is 'IGNORE', which is not accepted as a valid response, return to the reply routine via register 6.			R6	
ER60	Save registers 0-3 in the work space.		OESV(OEWS)	R0-R3	
	Move the text of message 1Q96I to the output area.		OML1(OEWS)		
	Load the address of the cuu field to be substituted in the message text into register 1, and the address of the file ID into register 2.			R1,R2	
	Branch and link to the message format subroutine..... >	OEFM		R6	
	Issue the message.				IPW$WTO Chart AD
	Restore registers 0-3 and return to caller via link register 14.			R0,R3, R14	
ER80	Move the message text of message 1Q97D to the output area.		OML2(OEWS)		
	Load the address of the cuu field to be substituted in the message text into register 1.			R1	
	Clear register 2 to indicate no file ID substitution.			R2	
	Branch and link to the message format subroutine..... >	OEFM		R6	
	Branch and link to the message reply routine..... >	RPLY		R6	
	If the reply is 'IGNORE', which is a valid response, meaning 'last' volume encountered, return to caller via link register 14, after the diskette volume counter has been reset to one.		WEND(OEWS)	R14	

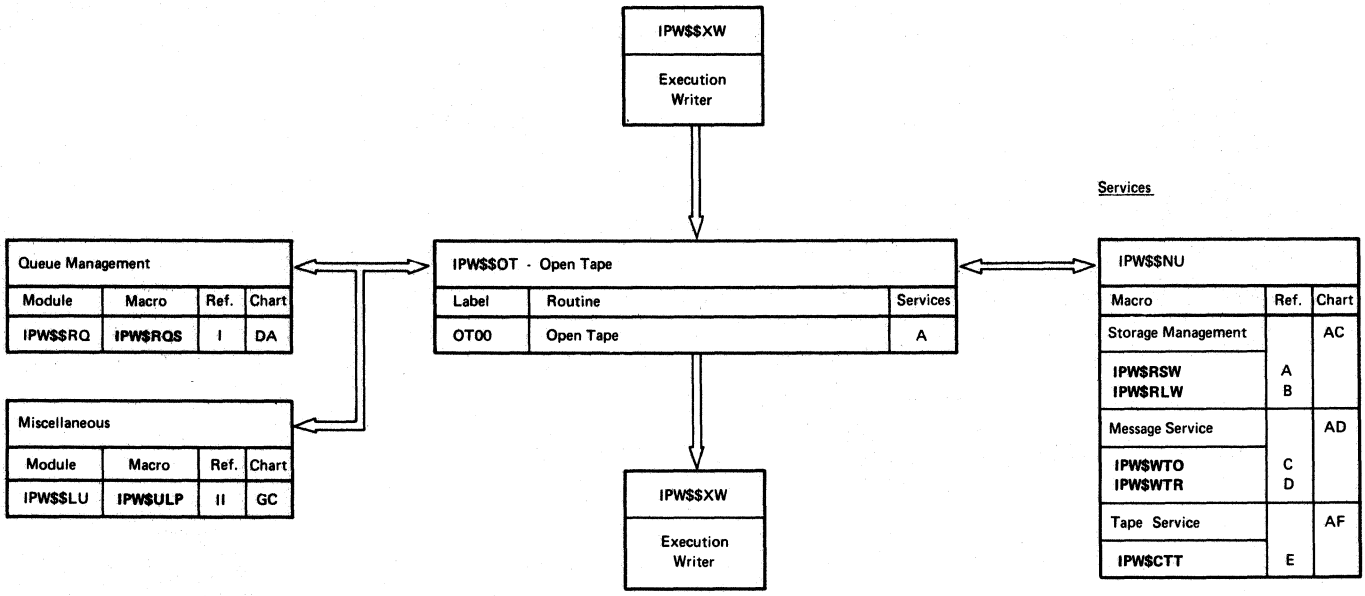
Labels	Chart GF11: IPW$$OE - 3540 Open Routine		Modified Data Fields	Reg. Usage	Calls
ER90	Move the message text of message 1Q98D to the output area.		OML1(OEWS)		
	Load the address of the cuu field to be substituted in the message text into register 1 and the address of the file ID into register 2.			R1,R2	
	Branch and link to the message format subroutine..... >	OEFM		R6	
ER92	Issue the message and wait for reply.				IPW$WTR Chart AD
	If the operator's reply is EOB, or IGNORE, branch to..... >	ER96			
	If the operator's reply is FEED, branch to..... >	OEFD			
	If the operator's reply is NEWPAC, branch to..... >	OEND			
	If the operator's reply is CANCEL, signal immediate stop C'S' via the open indicator and return to caller >	EXIT	PEOC(IPW$DPW)		
ER93	Save the return address in the work space.		OESV(OEWS)		
	Calculate the length of the reply in register 0.			R0	
ER94	Branch and link to the numeric character test subroutine..... >	OENM		R14	
ER95	If the reply is invalid, set up for issuing message 'INVALID RESPONSE R=', and branch to..... >	ER92	OML1(OEWS) OMT1(OEWS)		
ER96	If the reply was zero, branch to... >	ER97			
	Update the number of diskettes to be read and store the new number.		WEND(OEWS)	R0	
	Set the multivolume indicator to C'C' to indicate current volume.		WEMI(OEWS)		
	Restore the link register and return to caller.			R14	
ER97	Set the multivolume indicator to C'L' to indicate last volume.		WEMI(OEWS)		
	Return the caller via link register 14.			R14	

Labels	Chart GF12: IPW$$OE - 3540 Open Routine		Modified Data Fields	Reg. Usage	Calls
	Message Format Subroutine				
OEFM	Check if this is a card reader task with a 3540. If not, branch to.... >	FM02			
	Move the device address into the message text, and branch to substitute the file ID..... >	FM04			
FM02	Clear the device address in the message text.				
FM04	Check if the file ID has to be substituted in the message text. If not, return to caller via link register 6.			R6	
	Check if the file ID has been specified. If not, branch to..... >	FM06			
	Move the file ID into the message text, and return to caller via link register 6.			R6	
FM06	Check if this is a data file. If not, return to caller via link register 6.				
	Move (NONAME) to indicate unidentified file into the message text and return to caller via link register 6.			R6	
	Message Reply Subroutine				
RPLY	Save registers 0-3 in the work space.		OESV(OEWS)		
	Issue the message and wait for reply.				IPW$WTR Chart AD
	Restore registers 0-3.			R0-R3	
	If the reply is 'IGNORE', return to caller via link register 6.			R6	
	If the reply is FEED, branch to feed the next diskette..... >	OEFD			
	Check if a new volume has been mounted. If so, branch to read the volume 1 label..... >	OEND			
	Check if the reply was EOF. If not, branch to..... >	RP08			
	Check if the preceeding volume existed. If not, branch to..... >	RP04			
	Set the open return code to C'E' to indicate end of file.		PEOC(IPW$DPW)		

Labels	Chart GF13: IPW$$OE - 3540 Open Routine		Modified Data Fields	Reg. Usage	Calls
	Set the multivolume indicator to C'L' to indicate last volume.		PEMI(IPW$DPW)		
	Return to caller (IPW$$ER)..... >	EXIT			
RP04	Set up for issuing message 'NO PRECEEDING VOL, INCONSIST RESP R=', and branch to issue the message.... >	RPLY+4	OML2(OEWS) OMT2(OEWS)		
RP08	Check if the task has to be canceled. If not, branch to..... >	RP12			
	Set the stop code to C'S' to indicate stop immediate.		PEOC(IPW$DPW)		
	Return to caller (IPW$$ER)..... >	EXIT			
RP12	Set up for issuing message 'INVALID RESPONSE R=', and branch to issue the message..... >	RPLY+4			

CHART GG: IPW$$OT - OPEN TAPE (4 PARTS)

Chart GG00: IPW$$OT - Open Tape, General Flow and Macro Calls



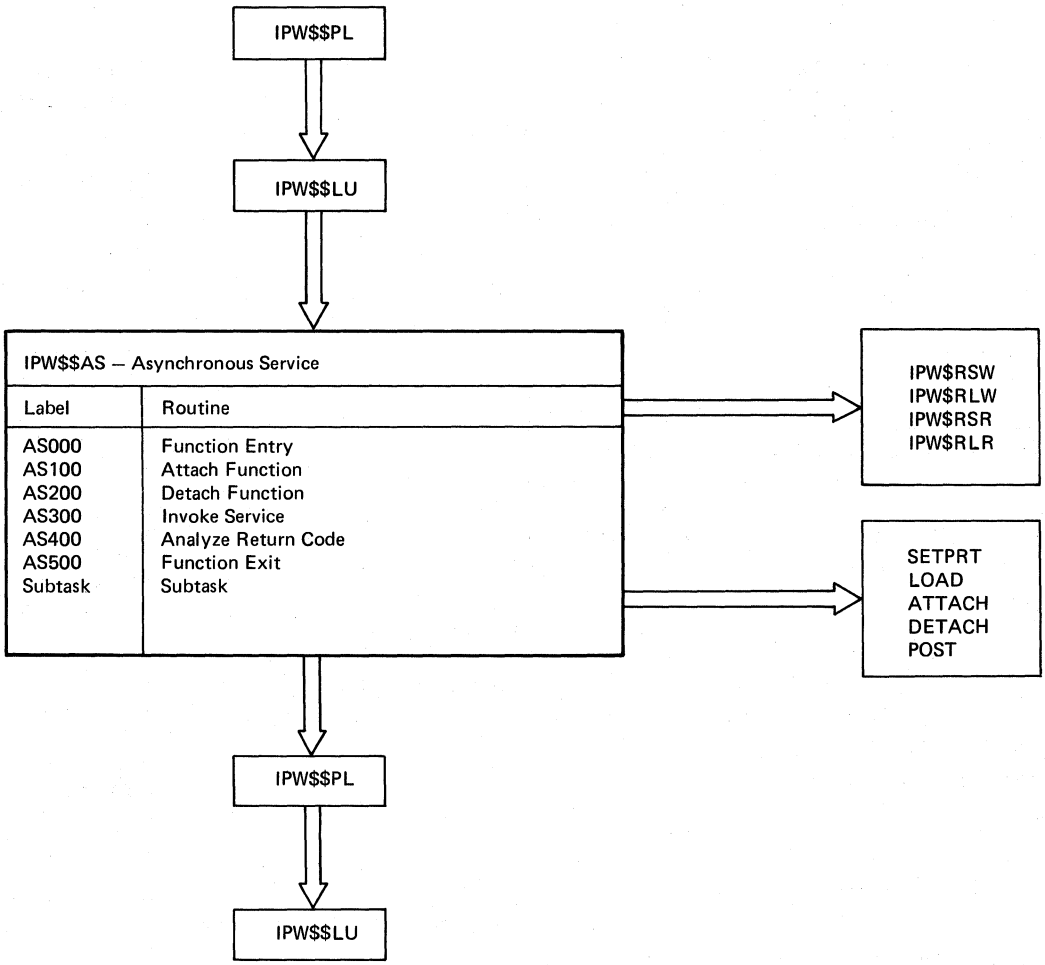
Labels	Chart GG01: IPW$$OT - Open Tape		Modified Data Fields	Reg. Usage	Calls
OTSD	The first 16 bytes constitute the section descriptor: 'OTCS V7M0 '				
OT00	Save caller registers 14 through 9 inclusive (IPW$SAV).		IPW$DSV		
	Reserve work space for a second function register save area.				IPW$RSW Chart AC
	Chain new save area.				
	If tape to be closed, branch to.... >	OT56			
OT08	Reserve space for the tape control block.				IPW$RSW Chart AC
	Set up register 8 as base register for the tape control block.			R8	
	Calculate the CCW address in register 2 and store it in the tape control block.		TBCA(IPW$DTB)	R2	
	Set the SLI flag in the CCW on.		TBCH+4(IPW$DTB)		
	If the tape address is present, branch to..... >	OT40			
	Otherwise, branch to..... >	OT20			
OT12	Issue message 1Q56I INVALID TAPE ADDRESS/MODE SET.				IPW$WTO Chart AD
OT16	If a tape address is received, the ownership is released.				IPW$ULP Chart GC
OT20	Issue message 1Q55D SPECIFY TAPE ADDRESS FOR , and read reply.				IPW$WTR Chart AD
OT22	If the reply is blank:				
	• the tape control block is released.				IPW$RLW Chart AC
	• the queue word in the TCB is set to zero.		TCQW(IPW$DTC)		
	• the disposition indicator in the queue record is set to C'D'.		QRDP(IPQ$DQR)		
	• A branch is made to..... >	OT60			
OT24	Since segmentation is not valid for tape, set the number of records before split indicator in the queue record to zero.		QRBS(IPW$DQR)		
	If the cuu has been specified incorrectly, branch to..... >	OT12			

Labels	Chart GG02: IPW$$OT - Open Tape		Modified Data Fields	Reg. Usage	Calls
OT32	Set the default mode set in the TCB.		TCQW(IPW$DTC)		
	If no mode set has been specified, branch to..... >	OT40			
	Check the specified mode set, and if it is invalid branch to..... >	OT12			
	Set the specified mode set in the TCB.		TCQW(IPW$DTC)		
OT40	Set up the tape control block section descriptor.		TBSD(IPW$DTB)		
	Set up registers 0, 1, and 2 for getting a tape.			R0,R1 R2	
	Get a tape.				IPW$ULP Chart GC
	If no tape is received, branch to..... >	OT12			
	Set cuu in tape control block. Set up registers 0 and 3 for getting a LUB.		TBSD(IPW$DTB)	R0,R3	
	Get a LUB.				IPW$ULP Chart GC
	If no LUB is received, issue message '1R64I NO LUB AVAILABLE, DISP...'				IPW$WTO Chart AD
OT41	Release tape ownership.				IPW$ULP Chart GC
	Force disposition to "D"..... >	OT22			Chart AD
OT42	Set up the data disk word and the queue request word in the TCB as tape request word.		TCDW(IPW$DTC) TCQW(IPW$DTC)		
	Save mode in the TCB. Change mode in PUB with specified mode.		TBSM PUB (Byte 6)		
OT43	Issue message 1Q58A MOUNT TAPE ON .				IPW$WTR Chart AD
	Wait for the operator EOB.				
OT44	Set up register 1 for sense, and sense tape.			R1	IPW$CTT Chart AF
	If the tape has a file protect ring, branch to..... >	OT48			
	Rewind and unload the tape.				IPW$CTT Chart AF
	Issue message 1Q57A WRITE RING REQUIRED ON .				IPW$WTO Chart AD
	Wait for operator EOB, and branch to>	OT44			
OT48	If the tape is not ready, branch to >	OT43			IPW$CTT Chart AF
	If the tape is not positioned at load point, branch to..... >	OT58			IPW$CTT Chart AF

Labels	Chart GG03: IPW$$OT - Open Tape		Modified Data Fields	Reg. Usage	Calls
	Set up the data disk word to read first record.		TCDW (IPW$DTC) TBCH (IPW$DTB)		
	Read first record (length 80).				IPW$RDT
	If no volume label, branch to..... >	OT55			
	Write volume label to console (length 52).				IPW$WTO Chart AD
OT50	Issue message '1R35D VOLUME LABEL FOUND ... '.				IPW$WTR Chart AD
	Test operator response:				
	If '1', branch to..... >	OT55			
	If not 'D' or 'N', branch to..... >	OT50			
	Rewind and unload the tape.				IPW$CTT Chart AF
	If operator response was 'D' branch to..... >	OT41			
	otherwise go to..... >	OT43			
OT55	Reposition the tape.				IPW$CTT Chart AF
	Set mode.				IPW$CTT Chart AF
OT58	Set mode byte in TCB to X'00'.		TCQW(IPW$DTC)		
	Set CCB flags in the tape control block.		TBCM(IPW$DTB)		
	Reserve disk space for the queue record to be held in the queue space.				IPW$RQS Chart DA
	Initialize the data control fields in the TCB in preparation of the first PUT.		TCBC(IPW$DTC)		
	If the reflective spot on the tape has not been hit yet, branch to.... >	OT60			
	Backspace over the queue record and write a tapemark over it.				IPW$CTT Chart AF
OT56	Rewind and unload the tape.				IPW$CTT Chart AF
	Set the JSEP indicator to X'00' for subsequent tapes.		QRSP(IPW$DQR)		
	Branch to ask for a new tape..... >	OT16			
OT60	Release the work space for the second function register save area.				IPW$RLW Chart AC
	Clear tape information and return to caller.				IPW$RET

CHART GH: IPW$$AS - ASYNCHRONOUS SERVICE (6 PARTS)

Chart GH00: IPW$$AS - Asynchronous Service, General Flow and Macro Calls



Labels	Chart GH01: IPW$$AS - Asynchronous Service		Modified Data Fields	Reg. Usage	Calls
ASCS	The first 16 bytes constitute the section descriptor. 'ASCS V10M1'				
	On entry, the following registers' contents are relevant:				
	Reg. 0 Branch index to required function: 0 Service request 4 Detach DOS/VS subtask 8 Attach DOS/VS subtask			R0	
	Reg. 1 Address of service request block (SRB), if applicable			R1	
	Reg.10 POWER/VS permanent address area			RA	
	Reg.11 TCB address			RB	
	Reg.13 Task save area			RD	
	Reg.14 Caller's return address			RE	
	Reg.15 Function entry address			RF	
AS000	Save caller registers 14-9, inclusive.		IPW$DSV		IPW$SAV
	The function branch index, passed in register 0, and the address of the SRB are loaded in registers 4 and 8 respectively.			R4,R8	
	The entry point address, contained in register 15, is saved in register 9.			R9	
	Register 5 is loaded with the queue record address.			R5	
	If the asynchronous service anchor block already exists, branch to.... >	AS005			
	Otherwise, reserve storage for it.				IPW$RSW Chart: AC
	The virtual address of the anchor block is stored in the permanent address area and copied in register 7 to address the anchor block.		CAAB(IPW$DPA)	R7	
	The owner field of the buffer control word (BCW) associated with the anchor control block is addressed and updated to indicate that the anchor block belongs to the system and not to the task requesting it.			R1	
	The storage descriptor is built in the anchor block.		ABSD(IPW$DAB)		
AS005	Lock the asynchronous anchor block.				IPW$RSR Chart: AB
AS010	The branch index is used to branch to the appropriate function through the following branch table:				
	0 Service request..... >	AS300			
	4 Detach request..... >	AS200			
	8 Attach request..... >	AS100			

Labels	Chart GH02: IPW$$AS - Asynchronous Service		Modified Data Fields	Reg. Usage	Calls
	Attach DOS/VS subtask				
AS100	Increment the number of asynchronous service users by '1'.		ABUSCT (IPW$DAB)	R1	
	Set flag, indicating that the task uses asynchronous service.		TCCB(IPW$DTC)		
	Examine the use count. If not the first user, branch..... >	AS500			
	Load the function exit address in register 14.			RE	
	Clear out subtask ECB.		ABECB		
	Clear out communication ECB.		ABECB2 (IPW$DAB)		
	Attach the subtask.				ATTACH
	If the attach was successful, branch return to caller..... >	RE			
	Issue message '1QA0I NO SUBTASK AVAILABLE FOR...'				IPW$WTO
	The asynchronous service use flag is turned off.		TCCB(IPW$DTC)		Chart: AD
	The immediate stop flag is set in TCB.		TCTT(IPW$DTC)		IPW$$TR
	Branch to function exit..... >	AS500			
	Detach DOS/VS subtask				
AS200	The asynchronous service use count is decremented by '1'.		ABUSCT (IPW$DAB)	R1	
	The asynchronous service use flag in the TCB is turned off.		TCCB(IPW$DTC)		
	If the use count is not zero, branch>	AS500			
	If the subtask is already abnormally terminated, branch to..... >	AS500			
	Otherwise, detach the DOS/VS subtask				DETACH
	Set the subtask TIK field to zero.		ABTIK(IPW$DAB)		
	Exit..... >	AS500			
	Invoke service				
AS300	The subtask communication ECB is examined. If the subtask is still alive, branch to..... >	AS310			
	Branch to attach subtask again..... >	AS110		RE	

Labels	Chart GH03: IPW$$AS - Asynchronous Service		Modified Data Fields	Reg. Usage	Calls
AS310	The SRB (service request block) is chained as last entry in the service request queue.		ABNEXT,ABLAST (IPW$DAB)	R2	
	Unpost the event control block in the SRB.		SRBECB (IPW$DSR)		Chart: AC
AS330	The DOS/VS subtask is awakened.			R1	POST
	Release the asynchronous service anchor block.				IPW$RLR
	Wait for completion of the request.				IPW$WFC Chart: AA
	Lock the asynchronous anchor block again.				IPW$RSR Chart: AC
	Unchain the service request block from the queue:				
	If the SRB was not the first one in the queue, branch to..... >	AS334			
	Otherwise, the first in chain pointer is set to zero and a branch is made to analyze the return code..... >	AS400			
AS334	The SRB chain is scanned to locate the appropriate SRB. If found branch to..... >	AS338			
	If not found branch to..... >	AS350			
AS338	Unchain the SRB from the service request queue.		SRBNEXT SRBLAST (IPW$SRB)		
	The pointer to the last SRB is updated if applicable.		ABLAST (IPW$DAB)		
	Branch to continue..... >	AS400			
AS350	Set abend reason code '04'.			R0	
	Issue message:			R1	
	'1QA2I CATASTROPHIC LOGIC ERROR OCCURRED' and force POWER/VS to abend.				IPW$WTO Chart: AD
	Branch to..... >	PADS			

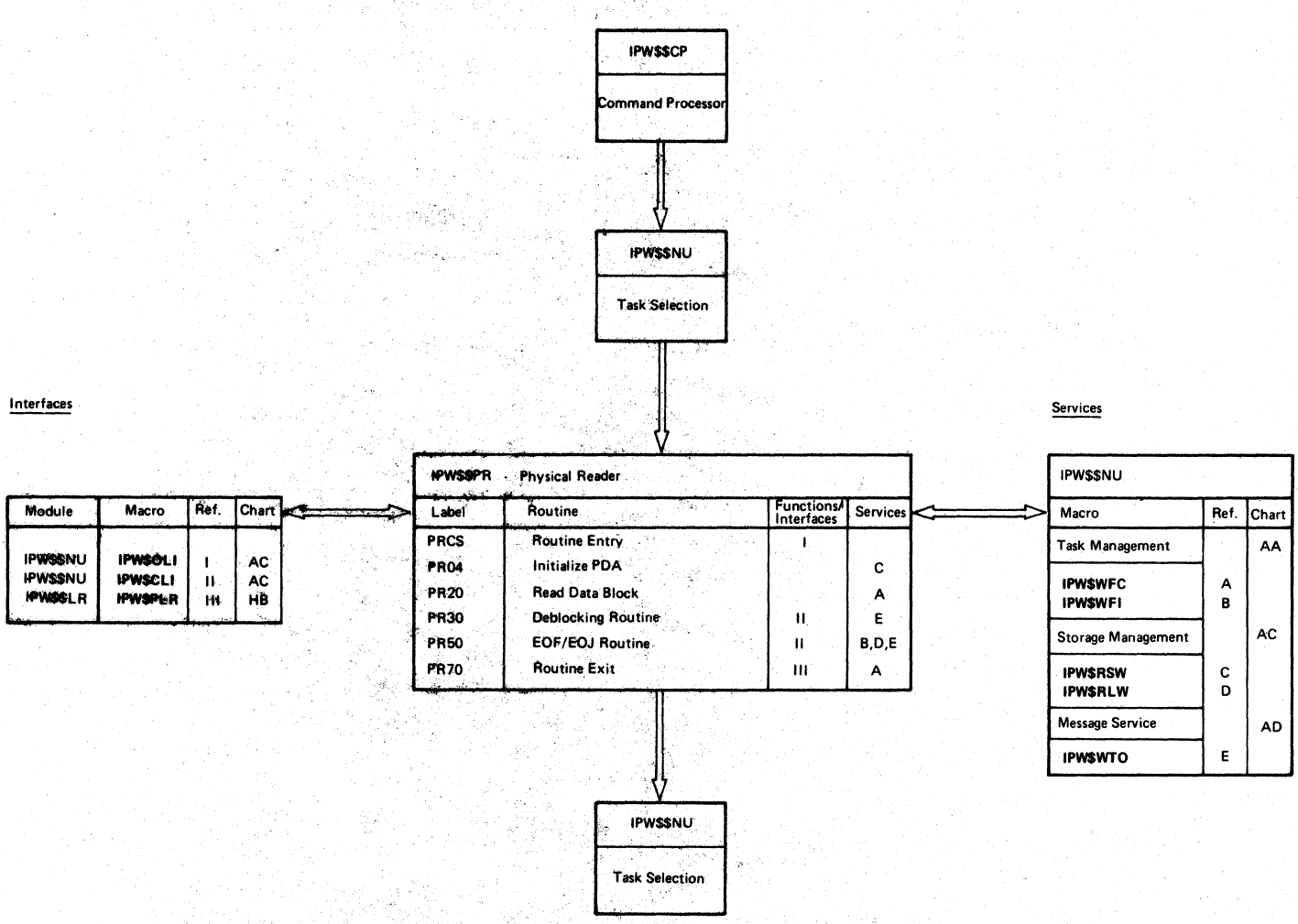
Labels	Chart GH04: IPW$$AS - Asynchronous Service		Modified Data Fields	Reg. Usage	Calls
	Analyze return code				
AS400	The return code byte of the SRB is now loaded in register 1 and used to branch to the appropriate routine handling the action to be taken.			R1	
	00 Successful completion..... >	AS500			
	04 Burst not installed..... >	AS500			
	08 Logic error occurred..... >	AS410			
	0C Invalid SETPRT request..... >	AS420			
	10 Phase not found in CIL..... >	AS430			
	14 Permanent I/O error..... >	AS440			
	18 Setup cancelled by operator... >	AS450			
	1C Reserved return-code..... >	AS410			
	20 Insufficient space CGS..... >	AS430			
	24 Invalid SETPRT request..... >	AS420			
	28 Not enough GETVIS storage..... >	AS470			
	2C Logic error occurred..... >	AS410			
	30 SETPRT routine not in SVA..... >	AS460			
	34 Internal macro call failed.... >	AS410			
	38 Invalid work area specified... > (cannot occur)	AS410			
	3C LFCB failure (cannot occur)... >	AS410			
AS410	Set task stop indicator.		TCTT(IPW$DTC)		
	Exit..... >	AS500			
AS420	Write message: '1QA3I SETPRT ERROR FOR...'				IPW$WTO Chart: AD
	Set task flush indicator.		TCTT(IPW$DTC)		
	Exit..... >	AS500			
AS430	Issue message: '1QA4I OUTPUT PROCESSING STOPPED...'				IPW$WTO Chart: AD
	Set task flush hold indicator, which causes the output to be kept in the queue with the disposition 'H'.		TCTT(IPW$DTC)		
	Exit..... >	AS500			
AS440	Set unrecoverable I/O error flag in TCB, C'u', and branch to function exit..... >	AS500	TCTT(IPW$DTC)		
AS450	Set task stop indicator.		TCTT(IPW$DTC)		
	Exit..... >	AS500			
AS460	Issue message: '1QA1I SETPRT ROUTINE NOT FOUND...'.				IPW$WTO Chart: AD
	Set task stop indicator.		TCTT(IPW$DTC)		
	Exit..... >	AS500			
AS470	Issue message: '1QA6I NO STORAGE AVAILABLE FOR...'.		TCTT(IPW$DTC)		IPW$WTO Chart: AD
	Set task stop indicator.				
	Exit..... >	AS500			
	Function exit				
AS500	Release asynchronous service anchor block.			R3	IPW$RLR
	Restore caller registers and return.			RE-R9	IPW$RET

Labels	Chart GH05: IPW$$AS - Asynchronous Service DOS/VS Subtask Processing		Modified Data Fields	Reg. Usage	Calls
	DOS/VS subtask				
	All service requests are handled on a first-in first-out basis.				
SUBTASK	The base register for the subtask is established by using register 9.			R9	
	The system communication region is addressed using register 2 and the subtask identification key (TIK) is moved into the anchor control block.		ABTIK(IPW$DAB)	R2	
ST005	Examine the first service request in the service queue.			R8	
ST010	If a service request is pending, branch to..... >	ST050		R8	
	Issue WAIT and wait to be activated again.			R1	WAIT
	Clear out ECB and branch to..... >	ST005			
ST050	If this is not a SETPRT request, branch to..... >	ST100			
	If the address of the SETPRT logic module in the SVA already exists in the anchor block, branch to..... >	ST070			
	Otherwise, issue LOAD with TXT=NO, to get the entry point address of the SETPRT routine in the SVA.			R0,R1 R3,R4	LOAD
	Test if the routine is in the SVA. If not, branch to..... >	ST100			
	Save the entry point address of the SETPRT logic module in the anchor control block and in register 2.		ABADR1 (IPW$DAB)	R2	
ST070	Issue SETPRT request.			R1,RE, RF	SETPRT
	Store return code in service request block (SRB).		SRBRTC (IPW$DSR)		
ST100	Post event control block in SRB.		SRBECB (IPW$DSR)		
	Activate POWER/VS (post master ECB).		PACB(IPW$DPA)	R1	POST
	Get next service request block to process. Branch to..... >	ST010			

READER ROUTINES

CHART HA: IPW$$PR - PHYSICAL READER (7 PARTS)

Chart HA00: IPW$$PR - Physical Reader, General Flow and Macro Calls



Labels	Chart HA01: IPW$$PR - Physical Reader		Modified Data Fields	Reg. Usage	Calls
PRSD	The first 16 bytes constitute the section descriptor:				
	'PRCS V7M0 '				
	On entry, the following register contents are relevant:				
	1: Byte 0 - number of buffers (01 or 02)				
	1: Byte 1 - device type card reader Bytes 2 and 3 - programmer logical unit card reader			R1	
	2: Byte 1 - device type 3540 reader Bytes·2 and 3 - programmer logical unit 3540 reader			R2	
	9: Physical reader base address			R9	
	10: Permanent area address		IPW$DPA	R10	
	11: TCB reader task		IPW$DTC	R11	
	13: Save area reader task.		IPW$DSV	R13	
	<u>Function Entry</u>				
	The entry parameters in register 1 are saved in register 6.			R6	
	The entry parameters in register 2 are saved in register 7.			R7	
PR02	The logical reader interface is opened through an IPW$OLI call.				IPW$OLI Chart AC
PR03	If the reader task has not been started with a connected 3540 diskette reader, branch to reserve a short physical work space.......... >	PR04			
	160 bytes of physical work space (PWS) for saving dependent data for a card reader device and a 3540 diskette device are obtained to maintain reenterability.				IPW$RSW Chart AC
	Addressability of PWS is set through register 8.			R8	
	Save the real address of the PWS in the PWS.		PERA(IPW$DPW)		
	Save the 3540 PSTART parameters in the PWS.		PEDI(IPW$DPW)		
	Indicate in the TCB that a JECL RDR statement is allowed in the input stream.		LWER(IPW$DTC)		
	Branch to continue................. >	PR05			
PR04	64 bytes of PWS (physical work space) for saving device dependent data are obtained to maintain reenterability.				IPW$RSW Chart AC
	Addressability of PWS is set through register 8.		IPW$DPW	R8	

Labels	Chart HA02: IPW$$PR - Physical Reader		Modified Data Fields	Reg. Usage	Calls
PR05	Device dependent data is saved in PWS.		PWDI(IPW$DPW)		
	The device type is translated into a displacement (in number of entries) in the DVB (Device Control Block Table) and multiplied by the entry length to get the proper byte displacement in register 7.		PWDT(IPW$DPW) DVDT(IPW$$PR)	R7	
	The record length for the specified card reader is then moved from the proper DVB to the PWS.		PWRL(IPW$DPW)		
	Register 6 is loaded with number of buffers.			R6	
	PDA (Physical Data Area) space is reserved for card reader CCB, related CCWs and data buffers.				IPW$RSW Chart AC
	The PDA size is loaded from the permanent area.				
	The virtual and real address of the CCB (first 16 bytes of PDA) is saved in PWS, and a branch is made to initialize the first buffer..... >	PR08	PBV1(IPW$DPW) PBR1(IPW$DPW)		
PR07	The second PDA is reserved for CCB, related CCWs, and data buffers.				IPW$RSW Chart AC
	The second PDA is loaded from the permanent area.				
	The virtual and real address of the CCB (first 16 bytes of PDA) is saved in PWS.		PVB2(IPW$DPW) PVR2(IPW$DPW)		
	Buffer Space Initialization				
PR08	The first 16 bytes of the buffer work space are now initialized as a card reader CCB:		IPW$DCB		
	• Wait for device end • Command retry option. • Programmer logical unit from PWS. • EXCP real. • Real address of first CCW is stored in CCB.		CBC1(IPW$DCB) CBC2(IPW$DCB) CBLC(IPW$DCB) CBLC(IPW$DCB) CBCA(IPW$DCB)		
	The number of CCWs to be generated is now calculated in register 5, taking into account remaining buffer space, record length, and CCW length.			R5	
	The number of CCWs to be generated is now multiplied with the CCW length to obtain the displacement of the last CCW from the beginning of the PDA.				
	This displacement is saved in PWS.		PWLC(IPW$DPW)		

Labels	Chart HA03: IPW$$PR - Physical Reader		Modified Data Fields	Reg. Usage	Calls
PR10	The CCW string is set up:				
	A CCW image is moved from the DVB entry to the PDA.		IPW$DCW	R3	
	The real data buffer address contained in register 6 is stored in the CCW.		CWDA(IPW$DCW)	R6	
	Register 4 is incremented with the data buffer length.			R4	
	Register 3 is incremented with the CCW length.			R3	
	The next CCW is set up.				
	The last CCW is unchained.		CWFL(IPW$DCW)		
	If double buffering was specified in PSTART command, branch to reserve and initialize the second PDA..... >	PR07		R6	
	The first buffer pointers are loaded in register 1 and register 2, an SVC 0 is issued, and a branch is made to wait for I/O completion.... >	PR22	PBV1(IPW$DPW) PBR1(IPW$DPW)	R1 R2	
	Read Data Block				
PR20	The first buffer pointers are loaded in registers 1 and 2.		PBV1(IPW$DPW) PBR1(IPW$DPW)	R1	
	If double buffering was specified, branch to test for active buffer... >	PR21	PWDB(IPW$DPW)		
	Otherwise, issue an SVC 0 for the first and only buffer, and branch to wait for I/O completion..... >	PR22			
PR21	If first buffer is not active, branch to make it active..... >	PR22	PWVE(IPW$DPW)	R1	
	Otherwise, registers 1 and 2 are loaded with second buffer pointers.		PBV2(IPW$DPW) PBR2(IPW$DPW)	R1 R2	
PR22	Make buffer pointed to by registers 1 and 2 active by saving registers 1 and 2 in PWS.		PWVE(IPW$DPW) PWRE(IPW$DPW)	R1 R2	
	Wait for I/O completion of active buffer.				
	Registers 0, 2, 3, and 6 are initialized for the deblock routine:				
	0: Virtual address first data buffer			R0	
	2: Real address first CCW			R2	
	3: Real address CCW last executed			R3	
	6: Real address last CCW in string			R6	
	If single buffering, branch to get record length..... >	PR28	PWDB(IPW$DPW)		

Lables	Chart HA04: IPW$$PR - Physical Reader		Modified Data Fields	Reg. Usage	Calls
	If unit exception occurred, branch to get record length..... >	PR28	CBSD(IPW$DCB)		
	Otherwise, issue an SVC 0 for the inactive buffer.				
PR28	Get record length in register 1, and enter the deblock routine (PR30) at PR40..... >	PR40	PWRL(IPW$DPW)	R1	
	Deblocking Routine				
PR30	Register 0 (virtual address data record) and register 1 (record length) are restored from task save area on return from logical reader.			R0 R1	
	Virtual address data record (register 0) and real address associated CCW (register 2) are incremented.			R0 R2	
PR40	(This entry point is taken whenever a new data block has been read in.)				
	Depending on various conditions, a record is passed to the logical reader or other action is taken.				
	If the record was not the last one read, branch to pass to the logical reader..... >	PR50			
	If last CCW executed was the last CCW in the string, branch to read next block..... >	PR20			

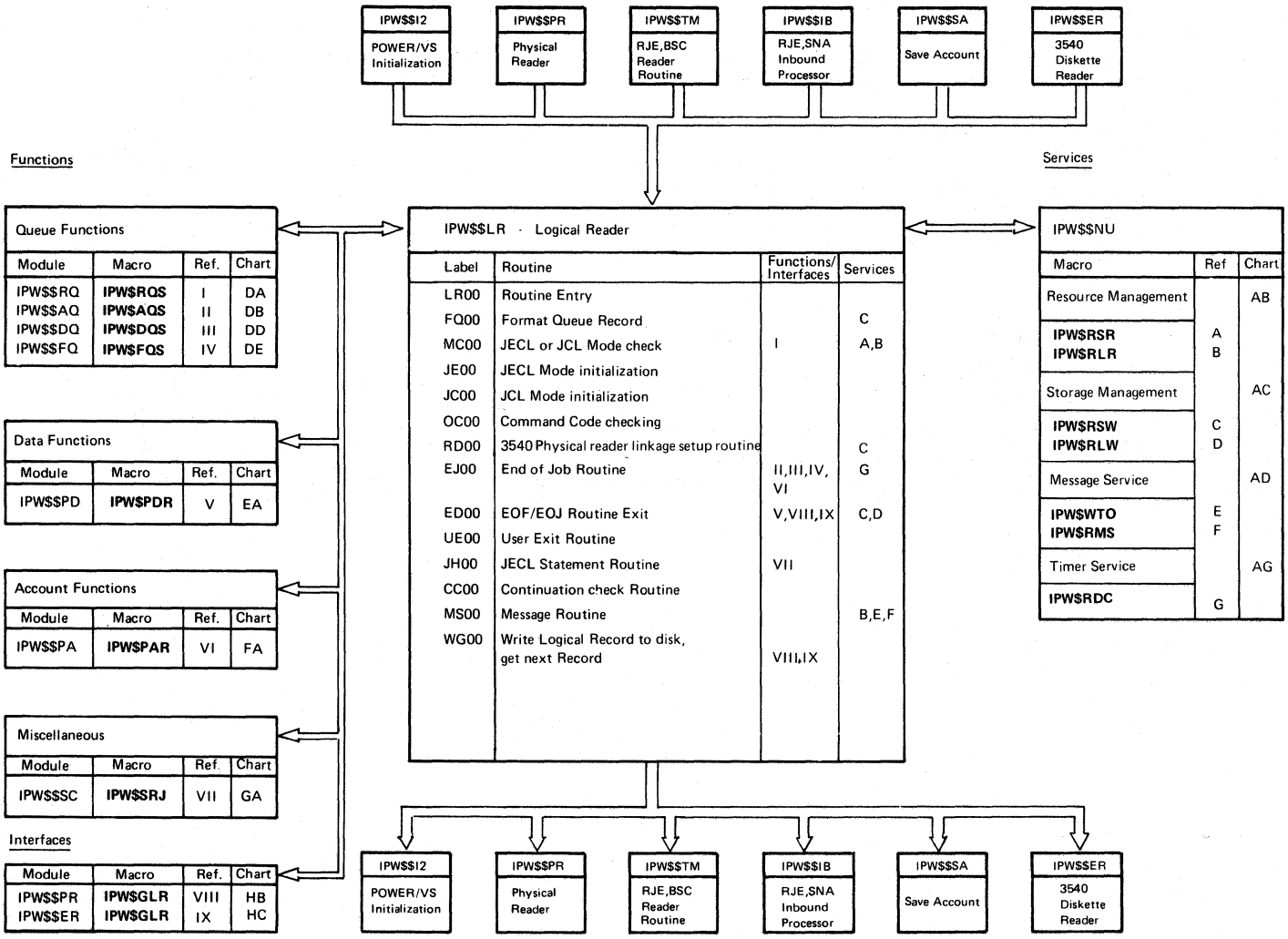
Labels	Chart HA05: IPW$$PR - Physical Reader		Modified Data Fields	Reg. Usage	Calls
	If unit exception is detected, branch to indicate unit exception..... >	PR42		R4	
	If ignored error, branch to read the next block..... >	PR20			
	Otherwise, pass the last record.... >	PR50			
PR42	If unit exception and device not 2560, branch to indicate unit exception to logical reader..... >	PR45			
	If device 2560, issue an IPW$PLR call to pass last record and issue 2 dummy reads to empty the card path.				IPW$PLR Chart JC
	Reset DEVICE END switch.		TCG2(IPW$DTC)		
PR45	(Entered if unit exception has been detected.)				
	Register 1 (record length) is set to 0 to signal unit exception to the logical reader.			R1	
PR50	Control is passed to the logical reader through an IPW$PLR call.				IPW$PLR Chart FC
	The return code posted in register 1 by the logical reader is tested, and on normal (nonzero) return the next record is passed to the logical reader..... >	PR30		R1	
	EOF/EOJ Exit				
	This routine is entered in case of:				
	unexpected unit exception, or unit exception on EOJ.				
PR60	Device dependent parameters in PWS (device type and programmer logical unit) are saved in register 6.			R6	
	Register 7 is set to binary zero to indicate that the reader task has been started without a connected 3540 diskette.			R7	
	If the task has not been started with a connected 3540 diskette, branch to release the work areas..... >	PR62			
	Otherwise, save the device dependent parameters (device type and programmer logical unit), which were saved in the PWS for the diskette device, in register 7.			R7	

Labels	Chart HA06: IPW$$PR - Physical Reader		Modified Data Fields	Reg. Usage	Calls
PR62	The PDA(s) is (are) released.				IPW$RLW Chart AC
	Register 1 is loaded with PWS address, and PWS is released.			R1	IPW$RLW Chart AC
	According to different conditions, branches are made to the appropriate routine.				
	If unit exception occurred, but not at EOJ, branch to indicate unexpected unit exception..... >	PR85			
	If device readied in the meantime, branch to..... >	PR70			
	If unit exception and EOJ, message 1Q34I is issued:				
	Message address is taken and stored in message request word.		TCMW(IPW$DTC)		
	Message is logged.				IPW$WTO Chart AD
PR70	The logical reader interface is closed.				IPW$CLI Chart AC
	If device readied in the meantime, branch to..... >	PR75			
	An IPW$WFI call is issued to wait for a possible card reader interrupt.				IPW$WFI Chart AA
	On card reader interrupt, indicate a new job stream in the card reader.				
PR75	If no STOP condition has been posted in the TCB, a branch is made to open the interface again..... >	PR02			
PR80	Using register 12 as a work register, the address of the task terminator is stored in the TCB, register 9 is loaded with the terminator base address, and a branch is made to terminate the task.		TCRC(IPW$DTC)	R12 R9	

Labels	Chart HA06.1: IPW$$PR - Physical Reader		Modified Data Fields	Reg. Usage	Calls
PR85	(This entry is taken in case of an unexpected unit exception.)				
	If device readied in the meantime, branch to..... >	PR88			
	Message 1Q35A is issued.				
	Message address is obtained and stored in message request word.		TCMW(IPW$DTC)		
	Message is logged.				IPW$WTO Chart AD
	If device readied in the meantime, branch to..... >	PR88			
	Wait for card reader interrupt.				IPW$WFI Chart AA
	On card reader interrupt, if a STOP condition has been posted in the TCB, a branch is made to detach the task..... >	PR80			
	Otherwise, branch to initialize work space again..... >	PR03			

CHART HB: IPW$$LR - LOGICAL READER (23 PARTS)

Chart HB00: IPW$$LR - Logical Reader, General Flow and Macro Calls



Labels	Chart HB01: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
LRSD	The first 16 bytes constitute the section descriptor: 'LRCS V7M0 ' On entry, the following register contents are relevant: 0: address of record to be handled 1: length of record to be handled 10: address of permanent area 11: address of reader TCB 13: address of reader task save area 14: address of entry point taken at first entry 15: address of logical reader.			R0 R1 R10 R11 R13 R14 R15	
	<u>Function Entry</u>				
LR00	This entry point is taken when a physical reader issues its first IPW$PLR call to the logical reader.				
	Register 9 is set up as logical reader base register.			R9	
LR10	Record address is saved in register 6.			R6	
	Record length is saved in register 7.			R7	
	If unit exception is detected (record length zero) a branch is made to return to the physical reader..... >	ED20			
	Otherwise, the forms switch is initialized.		LWFS(IPW$DTC)		
	LDA (Logical Data Area) space is reserved for the IPW$WTD function and the LDA is obtained from the permanent area.				IPW$RSW Chart AC
	Real and virtual address of buffer space are stored in the TCB.		TCDA(IPW$DTC)		
	If a user exit routine is available, a link is made..... >	UE00		R4	
	Branch to the mode check routine to analyze the first card..... >	MC00			
	<u>Format Queue Record</u>				
FQ00	Save register 1 in register 8 and indicate start of job in the TCB.		TCJB(IPW$DTC)	R8	
	If no queue space is available yet, branch to..... >	FQ02			
	Otherwise, clear the queue record body field.		QRBF(IPW$DQR)		

Labels	Chart HB02: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
FQ02	The first queue record is read in through an IPW$RQS call. The queue space is addressed through register 5.			R5	IPW$RQS Chart DA
	The data address is obtained from queue space and moved to the TCB.		TCDW(IPW$DTC)		
	The queue record is initialized.				
	Register 3 is set up to address the DMB.			R3	
	Standard start up date is moved from DMB to queue record.		QRDY(IPW$DQR)		
	User information field is blanked out.		QRUI(IPW$DQR)		
	Default job name is moved from DMB to queue record.		QRNM(IPW$DQR)		
	Reader queue record is indicated.		QRQI(IPW$DQR)		
	Default cancel code is moved from DMB to queue record.		QRCN(IPW$DQR)		
	Device type is moved from TCB to queue record.		QRDT(IPW$DQR)		
	Device unit number is moved from TCB to queue record.		QRCU(IPW$DQR)		
	Remote ID is moved from TCB to queue record.		TCRI(IPW$DQR)		
	Default class is moved from TCB to queue record.		QRCL(IPW$DQR)		
	Default priority is moved from DMB to queue record.		QRPY(IPW$DQR)		
	Number of copies is set to one.		QRNC(IPW$DQR)		
	Default forms number is set.		QRFI(IPW$DQR)		
	Default disposition is set (D).		QRDP(IPW$DQR)		
	Number of tracks is initialized at 1.		QRNT(IPW$DQR)		
	If the current job, allocated on the queue file, is not to be placed in the hold state, branch to..... >	FQ05			
	Otherwise, set the current job in the hold state.				

Labels	Chart HB03: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
FQ05	The DMB is reserved for the duration of the update of the job number in the master record.				IPW$RSR Chart AB
	The job number is updated:				
	The current job number is loaded in register 1 from the DMB.			R1	
	The current job number is stored in the queue record.		QRNO(IPW$DQR)		
	The job number is incremented by one and stored in the master record.		MRNO(IPW$DQC)	R1	
	The DMB is released again.				IPW$RLR Chart AB
	Register 1 is restored and a return is made to caller via register 4.			R1,R4	
	Mode Check Routine				
MC00	The incoming data record is examined and processed according to the conditions detected.				
	If the termination switch in the TCB indicates a stop condition, exit is made to task selection..... >	EJ45			
	A possible end of data record ID is reset.		TCGP(IPW$DTC)		
	A possible flush condition is reset.		TCTT(IPW$DTC)		
	The data record is now checked for various conditions, and if the record is not a JECL statement, branch to check for JC mode..... >	MC10			
	If no JECL operation code is found, branch to check for JC mode..... >	MC10			
	If the operation code is JOB, branch to handle a JOB statement..... >	MC20			
	If the operation code is CTL, branch to handle a CTL statement..... >	MC30			
	If the operation code is RDR, branch to handle a RDR statement..... >	MC40			
	A link is made to reserve queue space..... >	FQ00		R4	
	Branch to write the current record and to read the next card, after which JC mode processing is performed..... >	JC30			

Labels	Chart HB04: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
MC10	A link is made to reserve queue space..... >	FQ00		R4	
	Branch to check the statement in JC mode processing..... >	JC00			
MC20	The JOB operation code ID in the TCB is set.		LWOC(IPW$DTC)		
	The branch index used in the statement scan is set for job name, and a link is made to reserve queue space..... >	FQ00	LWBI(IPW$DTC)	R4	
	On return, a link is made to handle the JOB statement..... >	JH00		R8	
	Branch to write the current record and to read the next card, after which JECL mode processing is performed..... >	JE00			
MC30	The CTL operation code ID in the TCB is set.		LWOC(IPW$DTC)		
	The parameter form switch in the TCB is set to keyword form, and a link is made to handle the CTL statement... >	JH00	LWFS(IPW$DTC)	R8	
	A link is made to get a new record. >	WG30		R4	
	Branch to check what type of input mode is to be applied..... >	MC00			
MC40	The return address is set in register 4 which can be used by the next subroutine in case the RDR statement is not allowed and a mode check must be performed on the next card obtained from the physical reader.			R4	
	A link is made to handle the RDR statement..... >	JH00		R8	
	The return address to the mode check routine is again set in register 4 in case the linkage to the diskette reader cannot be set and a mode check must be performed on the next card obtained from the physical reader.			R4	
	If the processing mode specified in the RDR statement is not data mode, branch to set a linkage with the physical 3540 diskette reader..... >	RD00			
	Otherwise, reset the data mode switch in the TCB and branch to issue an error message and read the next card..... >	JH02	LWER(IPW$DTC)		

Labels	Chart HB05: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
	JECL Mode Routine				
JE00	A link is made to write the record to the data file and obtain the next record..... >	WG10		R4	
JE05	On return, a link is made to check the operation code..... >	OC00		R4	
	On return, the operation code routine branches into a branch table starting at the return point, and then branches to the appropriate routine, controlled by the return code, which is passed by the operation code routine.				
	If return code = 0, branch to handle data record..... >	JE00			
	If return code = 4, branch to handle '/&' condition..... >	JE20			
	If return code = 8, branch to handle '// JOB' condition..... >	JE20			
	If return code = 12, branch to handle JECL EOJ..... >	WG00			
	If return code = 16, branch to handle JECL JOB before JECL EOJ..... >	JE40			
	If return code = 20, branch to handle JECL CTL statement..... >	JE50			
	If the return code is 24, branch to handle a JECL RDR statement..... >	JE60			
JE20	If no flush condition is present, a branch is made to treat the current record as data..... >	JE00			
	If no DOS/VS flush condition present, branch..... >	JE00			
	If the current statement is a /& statement, a branch is made to reset the flush condition..... >	JE35			
	Record pointers register 6 and register 7 are saved, 'Inserted Record' is indicated, and register 6 and register 7 are set up to point to a '/&' record.		USCC(IPW$DTC)	R6,R7	
JE35	The flush condition is reset, and a branch is made back to treat the record as data..... >	JE00	TCTT(IPW$DTC)		

Labels	Chart HB06: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
JE40	This routine handles unexpected JECL JOB statement.				
	Register 0 and register 1 are set to point to a JECL EOJ record, and a branch is made to add the record to the data file..... >	WG45		R0,R1	
JE50	This routine initiates CTL statement processing. A link is made to handle the CTL statement..... >	JH00		R8	
	On return, a link is made to get the next record..... >	WG30		R4	
	A branch is made to process the new record..... >	JE05			
JE60	The return address is set in register 4, which can be used by the next subroutine in case the RDR statement is not allowed in the input stream and a link is made to handle the RDR statement..... >	JH00		R4	
	The return address is again set in register 4, and a branch is made to set a high level linkage with the physical diskette routine (IPW$$ER) >	RD00		R4	
	Job Control Mode Routine				
	This routine examines the first statement of the job being processed.				
JC00	If the current statement is not a // statement, branch to set default job name..... >	JC35			
	If no JCL operation code is found, branch to set default job name..... >	JC30			
	If the JCL operation code is not JOB, branch to set default job name..... >	JC35			
	Otherwise, if no job name is found, branch..... >	JC30			
	If job name is present, move job name to queue record..... >	JC10			
JC10	This routine is entered if the first statement is a // JOB statement. It moves the job name to the queue record.				
	The length of the job name is calculated in register 3. If the job name length exceeds 8 bytes, the number of bytes to be moved to the queue record is set to 8. The length of the job name is saved in register 4.			R3 R4	

Labels	Chart HB07: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
JC20	The job name field in the queue record is blanked out.		QRNM(IPW$DQR)		
	The job name is moved to the queue record using an execute instruction.		QRNM(IPW$DQR)		
	Sixteen bytes of user information is moved to the queue record space.				
JC30	A link is made to write the record to the data file and read the next one >	WG10		R4	
JC35	A link is made to scan the operation code of the next record..... >	OC00		R4	
	On return, the operation code scan routine provides a branch index into a branch table located at the return point, used to branch to the appropriate routine:				
	0: branch to handle data record.. >	JC30			
	4: branch to handle /& statement..... >	WG00			
	8: branch to handle unexpected // JOB statement..... >	JC40			
	12: branch to handle JECL EOJ statement..... >	JC50			
	16: branch to handle JECL JOB statement..... >	JC40			
	20: branch to handle JECL CTL statement..... >	JC60			
	24: branch to handle JECL RDR statement..... >	JC70			
JC40	Record pointer registers 0 and 1 are set to point to the EOJ record to be inserted. A branch is made to add the record to the data file..... >	WG45		R0,R1	
JC50	If reading from a 3540 diskette, branch to check for a forced EOJ statement..... >	JC53			
	If not reading from an SNA work station, branch to handle an unexpected JECL EOJ statement from the card reader..... >	JC55			
JC53	If a forced 3540 EOJ statement is passed, the record request word is set up with a dummy /& record and a branch is made to..... >	WG02	TCCC(IPW$DTC)	R0,R1	

Labels	Chart HB08: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
JC55	An unexpected JECL EOJ statement is invalidated by overwriting a $ sign with an asterisk. A branch is made to write the record to the data file..... >	JC30			
JC60	The CTL statement detected is handled through a link..... >	JH00		R8	
	On return, a link is made to get the next record..... >	WG30		R4	
	A branch is made to scan the new record..... >	JC35			
JC70	The return address is set in register 4, which can be used by the next subroutine in case the RDR statement is not allowed in the input stream, and a link is made to handle the RDR statement..... >	JH00		R4 R8	
	The return address is again set in register 4, and a branch is made to set a high level linkage with the physical diskette routine (IPW$$ER) >	RD00			
	Operation Code Check Routine				
OC00	This routine scans an input record and checks for JECL and JCL. A 'Normal Record' indication is made in the TCB. If the first character compares higher than '/', and can therefore not be the first character of a delimiter statement (JECL or JCL statement), branch back to caller.. >	(R4)			
	If the first character is a '/', branch to test for JCL..... >	OC10			
	Otherwise, if the record is a JECL statement, branch to check..... >	OC50			
	If neither JCL nor JECL statement, return to caller..... >	(R4)			

Labels	Chart HB09: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
OC10	If '/*' statement, branch to indicate data break..... >	OC20			
	If statement is a /& statement, branch to caller..... >	(R4)+4			
	Otherwise, if not a // statement, return to caller..... >	(R4)			
	If no JCL operation code is found, return to caller..... >	(R4)			
	If a // EXEC statement, indicate data break..... >	OC20			
	If a // JOB statement, return to caller..... >	(R4)+8			
	Otherwise, return to the caller.... >	(R4)			
OC20	A data break is indicated, and control is returned to the caller.. >	(R4)			
OC50	If no JECL operation code is found, return..... >	(R4)			
	If JECL EOJ statement, return..... >	(R4)+12			
	If statement is a JECL JOB statement before EOJ statement, return..... >	(R4)+16			
	If the current job has a flush condition, return..... >	(R4)			
	If the current statement is a CTL statement, branch to process it.... >	OC60			
	If the statement is a RDR statement, return to check it..... >	(R4)+24			
	Otherwise, return to write and get the next record..... >	R4			
OC60	Otherwise, the CTL operation code ID is set in the TCB, the form switch is set to 'Keyword', return to caller..... >	(R4)+20			
	3540 Physical Reader Linkage Setup Routine				
RD00	The interface to the physical diskette reader (IPW$$ER) is opened.				
	A physical save area is reserved for the physical diskette routine, which is obtained to save the registers used by the diskette routine.				IPW$RSW Chart AC

Labels	Chart HB10: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
	The new physical save area is initialized as follows:				
	• The address of the TCB is stored to define the owner.		0(R1)	R11	
	• The address of the logical save area is stored to define the high level interface between IPW$$LR and IPW$$ER.		4(R1)	R13	
	• The address of the physical diskette reader.		12(R1)	R15	
	• The base address of the physical diskette reader.		52(R1)	R15	
	• The entry address of the physical diskette reader.		8(R1)	R15	
	• Tne address of the physical work space.		48(R1)	R2	
	The address of the new physical save area is saved in the logical save area.		SVSV(IPW$DTC)		
	A branch is made to get the next record from the physical 3540 diskette device..... >	WG30			
	End of Job Routine				
EJ00	The termination condition code is copied from the TCB to the queue record.		QRCN(IPW$DQR)		
	Job termination time is obtained through an IPW$RDC call.				IPW$RDC Chart AG
	Job termination time is stored in the queue record. According to conditions detected, a branch is made to the appropriate routine to continue EOJ processing: If no flush condition, branch to add queue record to queue file..... >	EJ30	QRET(IPW$DQR)		
	Otherwise, the flush condition is reset. If no DOS/VS job flush condition was detected, a JECL flush condition must be present, so branch to write an account record..... >	EJ10	TCTT(IPW$DTC)		
	Otherwise, branch to add queue record to queue file..... >	EJ30			
EJ10	Write an account record.				IPW$PAR Chart FA
	The queue record is deleted from the queue set using an IPW$DQS call. The queue record is added to free set using an IPW$FQS call.				IPW$DQS Chart DD IPW$FQS Chart DE
	A branch is made to..... >	EJ40			

Labels	Chart HB11: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
EJ30	The queue record is added to the queue file using an IPW$AQS call.				IPW$AQS Chart DB
EJ40	The account record address is loaded into register 1.			R1	
	The account record length (58 bytes) is loaded in register 0.			R0	
	An IPW$PAR call is issued to add the account record to the account file.				IPW$PAR Chart FA
EJ42	The function trace indicator in the TCB is set to 'Complete EOJ'.		TCFT(IPW$DTC)		
	If 'Stop at EOJ' was not indicated, branch to continue..... >	EJ50			
	Otherwise, 'Detach' is indicated in the TCB.		TCTT(IPW$DTC)		
EJ45	Register 9 is set up as task terminator base register, and a branch is made to the task terminator..... >	(R9)+16		R9	
EJ50	If next record not available, branch to..... >	ED05		R7	
	A possible end of data record ID is reset.		TCGP(IPW$DTC)		
	If this is the start of an unexpected new job, reset unexpected job indicator and return to caller		LWUJ(IPW$DTC)	R4	
	Otherwise, check for user exit routine by branching to..... >	UE00			
	EOF/EOJ Exit This routine is entered in case of unit exception. On unexpected EOF (EOF before EOJ) the data block is written to the data file to make release of work space possible, work space is then released and control is passed back to the physical reader. On interrupt by the applicable reader device, the physical reader will pass control back and work space is reserved again.				
ED00	On unexpected EOF, EOB is indicated, and a data block is written.		TCGP(IPW$DTC)		IPW$PDR Chart EA
ED05	If the reader task has been started for the 3540 diskette only, there is no dormant linkage with a card reader, and a branch is made to release the logical data area..... >	ED10			

Labels	Chart HB12: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
	Register 1 is loaded with the address of the physical 3540 save area address.			R1	
	The physical 3540 save area is released.				IPW$RLW Chart AC
	The physical card reader save area is chained back to the logical save area.		SVSV(IPW$DTC)		
	A possible 3540 communication switch is reset.		LWER(IPW$DTC)		
	A possible end of data record ID is reset.		TCGP(IPW$DTC)		
	A branch is made to continue reading from the card reader..... >	WG42			
ED10	Register 3 is saved as it will be destroyed by the following release work space function.		SVR3(IPW$DSV)		
	Register 1 is loaded with the LDA address.			R1	
	The data LDA is released through an IPW$RLW call.				IPW$RLW Chart AC
	The LDA pointers in the TCB are set to zero.		TCDA(IPW$DTC)		
	Register 3 is restored.			R3	
	If an EOB record was indicated, implying EOF before EOJ, a branch is made to bypass release of queue space..... >	ED20			
	Otherwise, register 1 is loaded with queue space address, and queue space is released through an IPW$RLW call.			R1	IPW$RLW Chart AC
	Queue space pointers in the TCB are set to zero.		TCQA(IPW$DTC)		
ED20	End of input is indicated by setting register 1 to zero.			R1	
	Return to physical reader using an IPW$GLR call.				IPW$GLR Chart HB
	In case of EOF before EOJ, return is made to the logical reader at this point, after receiving the interrupt of the reader device.				

Labels	Chart HB13: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
	Record pointer registers 0 and 1 are saved in registers 6 and 7, respectively.			R6,R7	
	LDA space is reserved again.				IPW$RSW Chart AC
	Real and virtual addresses of the buffer space are stored.		TCDA(IPW$DTC)		
	Register 3, whose contents have been destroyed by the IPW$RSW call, is restored.			R3	
	A branch is made to check the new data record..... >	WG50			
	User Exit Routine				
UE00	The current record is examined and passed to the user exit routine according to the following conditions:				
	• If no user exit is available, return to the caller..... >	(R4)			
	• If flush condition, return to the caller..... >	(R4)			
	• If the current record has been user inserted, return..... >	(R4)			
	• If neither JECL nor JCL statement, return to the caller..... >	(R4)			
UE10	Registers are saved.		SVRE(IPW$DSV)		
	Record address and record length are saved in register 0 and register 1, respectively.			R0,R1	
	A link is made to the user exit routine.			R14	
	On return, user exit parameter registers 0 and 1 are stored in the TCB.		USCC(IPW$DTC)		
	The user exit return code is stored in the TCB.		USCC(IPW$DTC)		
	Reader registers are restored.				
	Register 1 is loaded with the user exit return code, which is used as a branch index into the following branch table:			R1	
UE20	0: Return to the calling routine..... >	(R4)			
	4: Branch to delete data record.. >	UE30			
	8: Branch to insert data record.. >	UE40		R8	
	12: Branch to flush DOS/VS job.... >	UE50			
	16: Branch to flush POWER/VS job.. >	UE60			

Labels	Chart HB14: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
UE30	The current record is deleted by bypassing the write to data file:				
	Delete condition is reset for next record.		USCC(IPW$DTC)		
	A branch is made to get the next record..... >	WG40			
UE40	Record pointer registers 0 and 1 are restored from the TCB.			R0,R1	
	Current record pointers registers 6 and 7 are saved in the TCB.		USCC(IPW$DTC)		
	Inserted record pointers are loaded into current record pointer registers 6 and 7.			R6,R7	
	Control is returned to the caller.. >	(R4)			
UE50	If at job boundary, branch to ignore flush..... >	UE80			
	Otherwise, set the POWER/VS cancel code in the queue record to X'0C' to indicate that only the records of the current DOS/VS job are to be flushed.		QRCN(IPW$DQR)		
	Branch to continue..... >	UE70			
UE60	If at job boundary, branch to ignore flush..... >	UE80			
	Otherwise, set the POWER/VS cancel code in the queue record to X'60' to indicate that all the records of the current POWER/VS job are to be flushed.		QRCN(IPW$DQR)		
UE70	Flush condition is indicated in the TCB.		TCTT(IPW$DTC)		
	Branch to get next record..... >	WG10			
UE80	Reset flush condition and issue message 1R57I.		USCC(IPW$DTC)		IPW$WTO
	Control is returned to the caller.			R4	
	JECL Statement Handler				
	This routine provides the interface with the parameter scan function. Parameter registers are set up prior to IPW$SRJ call.				
JH00	If the operation code is not RDR, branch to scan the statement directly..... >	JH08			
	If the RDR statement is read from a 3540 diskette device, branch to issue message 1Q90I..... >	JH02			

Labels	Chart HB15: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
	If the RDR statement is read from the card reader and the reader task has been started in combination with a 3540 diskette reader, branch to scan the statement..... >	JH04			
JH02	A link is made to the message subroutine to issue message 1Q90I * $$ RDR STATEMENT NOT ALLOWED, JOB FLUSHED..... >	MS05		R14	
	A flusn condition is set in the TCB.		TCTT(IPW$DTC)		
	Branch to flush the current POWER/VS job..... >	WG30			
JH04	Load the address of the physical work space in register 2.			R2	
	Blank the file name.		PFFI(IPW$DPW)		
	Load the address of the master record in register 3.			R3	
	Set following defaults into physical work space:				
	• Clear option byte (FEED)		PFOP(IPW$DPW)		
	If the option FEED in master record is not on, branch to..... >	JH05			
	• Turn option FEED on		PFOP(IPW$DPW)		
	• Number of diskettes to one		PFND(IPW$DPW)		
	• Volume sequence check option (to NO character blank)		PFSC(IPW$DPW)		
	• Verify option (to NO character blank)		PFVE(IPW$DPW)		
JH05	The RDR operation code ID in the TCB is set, and the branch index used in the statement scan is set for the device parameter.		LWOC(IPW$DTC) LWBI(IPW$DTC)		

Labels	Chart HB15.1: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
JH08	Register 1, on entry pointing to the JECL operation code, is set to point after the operation code.			R1	
	Register 3 is set up to contain the length of the remainder of the JECL statement minus one.			R3	
	Using registers 1 and 3 to control a translate and test instruction, a scan is made for the first parameter.			R1,R3	
	If found, a branch is made to process this parameter..... >	JH10			
	Otherwise, if no continuation punch is present, a branch is made to return to the calling routine..... >	JH80			
	If a continuation punch is present, a link is made to scan the continuation line..... >	CC00			
	On normal return, a branch is made to continue..... >	JH10			
	On error return, a branch is made to return to the caller..... >	JH80			

Labels	Chart HB16: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
JH10	If the parameters are not omitted, a branch is made to scan the parameter found..... >	JH20			
	Otherwise, the last parameter switch in the TCB is set ON, and a branch is made to scan possible continuation..... >	JH70	LWPI(IPW$DTC)		
JH20	Parameter registers 0 and 1 for the IPW$SRJ routine are set up:				
	Register 0 is loaded with the address of the statement end.			R0	
	Register 1 already points to the parameter to be scanned.			R1	
	An IPW$SRJ call is issued to scan the parameter pointed to by register 1.				IPW$SRJ Chart GA
	On return, if the parameter examined is valid (indicated by a positive value in pointer register 0), a branch is made to continue..... >	JH30			
	Otherwise, register 0 contents are converted to positive, a link is made to log message 1Q37I..... >	MS00		R0	
JH30	If the last parameter switch is on, indicating that statement scan has been completed, a branch is made to return to the calling routine..... >	JH70			
	Otherwise, if the current statement is not a CTL statement, a branch is made to continue..... >	JH40			
	If a CTL statement is being scanned, only one parameter is permitted, and the last parameter switch is therefore forced ON.		LWPI(IPW$DTC)		
	Error pointer register 0 is loaded with the operation code address and a link is made to issue message 1Q37I..... >	MS00		R0	
JH40	Register 1 is loaded with the address of a possible continuation punch.			R1	
	If the statement scan turns out to have stopped on that position, a branch is made to check for continuation..... >	JH50			

Labels	Chart HB17: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
	If the next parameter is indicated to be on a continuation line (comma followed by at least one blank), a branch is made as well..... >	JH50			
	Otherwise, register 1 is set to point one position after the delimiting comma.			R1	
	If the comma delimiting the parameter last checked was in column 71, a branch is made to check for continuation..... >	JH60			
	Otherwise, a branch is made back to scan the next parameter..... >	JH20			
JH50	If a continuation punch was given, a branch is made to check continuation..... >	JH60			
	Otherwise, error pointer register 0 is loaded with the address of the comma in error, a link is made to issue message 1Q37I..... >	MS00		R0 R14	
	A branch is made to return to the calling routine..... >	JH80			
JH60	A link is made..... >	CC00		R3	
	On normal return, branch to scan the next parameter..... >	JH20			
	On error return, a branch is made to return..... >	JH80			
JH70	If no continuation punch is present, a branch is made to bypass calling the continuation routine..... >	JH80			
	Otherwise, a link is made to scan continuation..... >	CC00		R3	
	On both normal and error return, a branch is made..... >	JH80			
JH80	Parameter switches in the TCB are reset, and control is passed back to the caller..... >	(R8)	LWPI(IPW$DTC)		

Labels	Chart HB18: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
	Continuation Check This routine is entered when a continuation punch is found in the current record. The current record is written to the data file and the next record is requested from the physical reader. This record, the continuation statement, is syntax checked and an error, if found, is flagged with error message 1Q37I. If the 'last parameter' switch is on, and the continuation statement itself contains a continuation punch, the continuation routine invokes itself. Otherwise, the first parameter on the continuation statement is searched. This parameter should start on one of columns 6 through 16 inclusively. A possible error return to the caller is made to an address four bytes past the normal return address.				
CC00	A normal record is indicated in the TCB. If current statement is not CTL, branch to get next record..... >	CC05	TCGP(IPW$DTC)		
	If the current statement is not a RDR statement, branch to write the current record and get the next one >	CC10			
CC05	Otherwise, a link is made to get next record..... >	WG30			
	A branch is made to handle the continuation line..... >	CC20			
CC10	A link is made to write record and read the next one..... >	WG10			
CC20	If the next record is not a JECL statement, branch to issue error message 1Q37I..... >	CC30			
	If the last parameter switch is ON, branch to check continuation column..... >	CC50			
	If column 5 is not blank, branch to issue error message 1Q37I..... >	CC30			
	If parameter starts on any of the columns 6 through 16, inclusively, return to caller..... >	(R3)			
	Otherwise, error pointer register 0 is made to point to column 16.			R0	
	Branch to issue error message 1Q37I..... >	CC40			

Labels	Chart HB19: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
CC30	Error pointer register 0 is made to point to column 1.			R0	
CC40	A link is made to message routine..... >	MS00			
	If no continuation punch is present, error return..... >	(R3)+4			
	Otherwise, branch to check continuation line..... >	CC00			
CC50	If no continuation punch, normal return to caller..... >	(R3)			
	Otherwise, branch to check next continuation line..... >	CC00			
	Message Routine				
	This routine logs error message 1Q37I, flagging invalid JECL information, or message 1Q90I, indicating RDR statement not allowed in the input stream. If message 1Q37I must be issued, the entry point is MS00. If message 1Q90I must be issued, the entry point is MS05. On entry, register 6 is supposed to contain the address of the statement in error, and register 0 contains the address of the column in error.				
MS00	Caller's registers are saved.				IPW$SAV
	If the operation code is CTL, branch to..... >	MS03			
	If no queue has been reserved yet, indicating that the RDR statement has been encountered at job boundary time, branch to..... >	MS02			
	Otherwise, set the current job in the hold state, and branch to issue message 1Q37I..... >	MS03	QRDP(IPW$DQR)		
MS02	Indicate the hold state for the next job to be allocated on the queue file.		LWER(IPW$DTC)		
MS03	Load the address of message 1Q37I into register 8 and branch to..... >	MS10		R8	
MS05	The caller's registers are saved.				IPW$SAV
	Load the address of message 1Q90I into register 8.			R8	

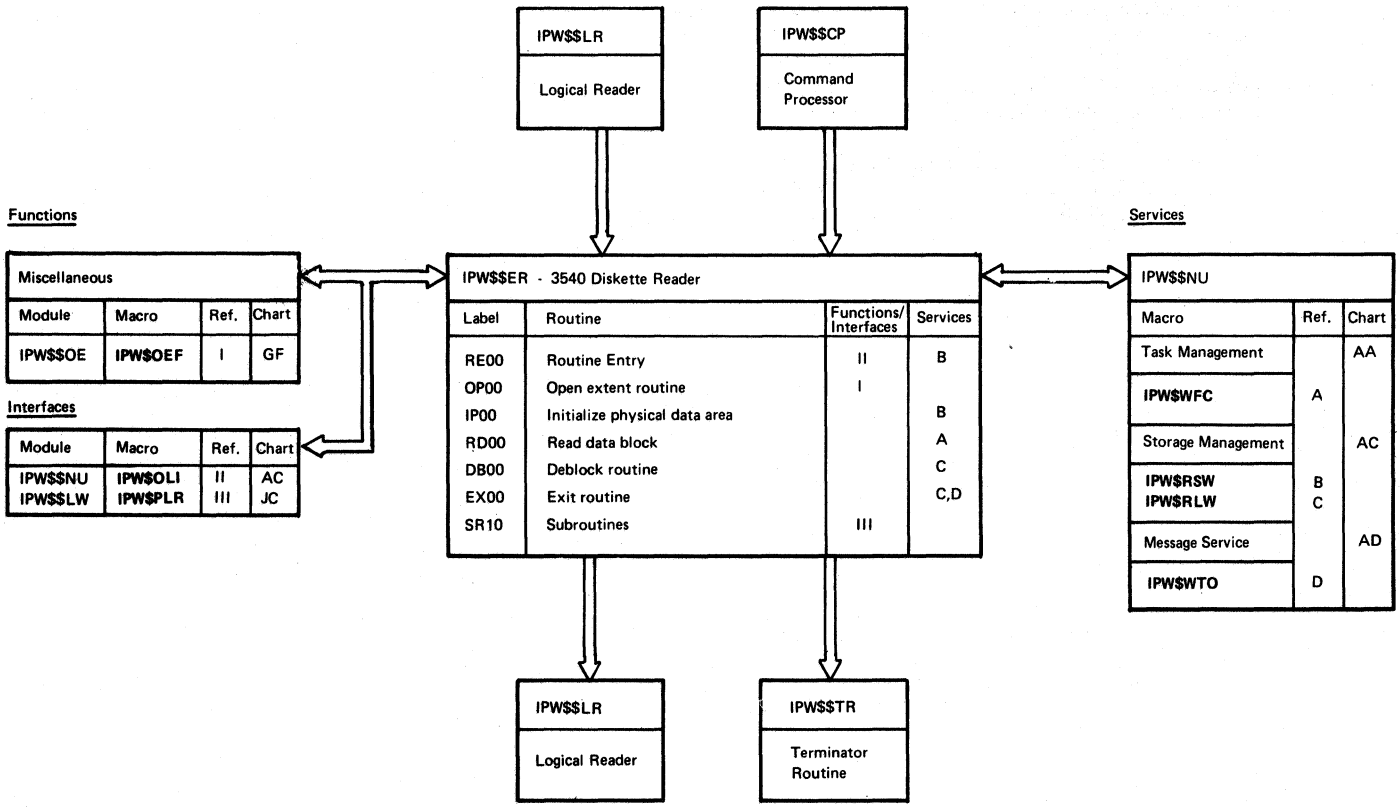
Labels	Chart HB20: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
MS10	The statement in error is prepared to be logged:				
	The byte previous to the record in error is set up as a message length indicator using register 6 as a work register.			R6	
	The column number of the column in error is calculated in register 4.			R4	
	The address of the statement in error is stored in the message request word.		TCMW(IPW$DTC)		
	The statement in error is now logged.				IPW$WTO Chart AD
	The message is printed.				IPW$WTO Chart AD
	If the current task is not an RJE task, a branch is made to return to caller..... >	MS20			
	Otherwise, the address of the statement in error is loaded in register 1.			R1	
	Remote ID and columns 79 and 80 of the statement in error are loaded in register 0 (low-order byte and 2 high-order bytes, respectively.			R0	
	An IPW$RMS call is issued to print the statement in error at the remote terminal.				IPW$RMS Chart AD
	Register 1 is loaded with the address of the message saved in register 8, the high-order bytes of register 0 are cleared, and an IPW$RMS call is issued to print the message.			R1 R0	IPW$RMS Chart AD
MS20	Control is returned to the caller.				IPW$RET
	Write Record, Get Next				
WG00	Set record information in record request word.		TCCC(IPW$DTC)	R6,R7	
WG02	Add record defined in record request word to the data file. Get record information from dummy JECL EOJ statement.				IPW$PDR Chart EA
WG05	Job boundary is indicated in the TCB. End of data record is indicated in the TCB.		TCJB(IPW$DTC) TCGP(IPW$DTC)		

Labels	Chart HB21: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
	Routine return register 4 is set to return to queue record format routine MC00. The flush condition test is skipped and a branch is made to.... >	WG20		R4	
WG10	If a flush condition is found, the record is not written to the data file and a branch is made to..... >	WG40			
WG20	Data record address is stored in the RRW in the TCB.		TCRV(IPW$DTC)		
	Data record length is stored in the RRW. Data record is added to data file through a IPW$PDR call.		TCRL(IPW$DTC)		IPW$PDR Chart EA
WG30	If no record has been inserted, a branch is made to obtain the next record..... >	WG40			
	Otherwise, the Record Inserted indication in the TCB is reset.		USCC(IPW$DTC)		
	The pointer registers are restored to point to the previous record.			R6,R7	
	The IPW$GLR call for the next record is skipped and a branch is made to test the previous record as if it had just been obtained..... >	WG50			
WG40	The record length is copied into parameter register 1.			R1	
WG42	The next data record is requested through an IPW$GLR call.				IPW$GLR Chart HB
	Record address is saved in register 6.				
	Record length is saved in register 7. Branch to..... >	WG50		R7	
WG45	Set up record request word.		TCCC(IPW$DTC)	R0,R1	
	Indicate:				
	job boundary, end of data record, unexpected job.		TCJB(IPW$DTC) TCGP(IPW$DTC) LWUJ(IPW$DTC)		
	Set up return register 4 to return to MC00 and add record to data file.				IPW$PDR Chart EA
WG50	If stop condition present, branch to detach..... >	EJ45			
	If an EOD record was indicated, branch to EOJ routine..... >	EJ00			
	Otherwise, if zero record length (unexpected EOF), branch to handle end of data..... >	ED00			

Labels	Chart HB22: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
	If reading from a 3540 data file, branch to write the record and get the next one..... >	WG10			
	If not EOF, and no user exit available, return to caller..... >	(R4)			
	If user exit available, but first character of current record compares higher than '/', and can therefore be neither JCL nor JECL statement to be processed by the user exit routine, a branch is made back to the caller.. >	(R4)			
	If the first character indicates a potential JCL or JECL statement, branch to the user exit routine.... >	UE00			

CHART HC: IPW$$ER - 3540 DISKETTE READER (11 PARTS)

Chart HC00: IPW$$ER - 3540 Diskette Reader, General Flow and Macro Calls Macro Calls



Labels	Chart HC01: IPW$$ER - 3540 Diskette Reader		Modified Data Fields	Reg. Usage	Calls
	This routine is entered when the Logical Reader (IPW$$LR) encounters a * $$ RDR statement, or when a PSTART command is issued for the 3540 Diskette Reader.				
ERCS	CSECT name.				
ERSD	The first 16 bytes constitute the section descriptor: 'ERCS V7M0'				
	General Register Usage				
	0 - **** - Service work register 1 - **** - Service work register 2 - **** - Service work register 3 - **** - Service work register 4 - **** - Work register 5 - **** - Work register 6 - **** - Pointer to last CCW in string 7 - **** - Record counter per track 8 - IPW$DPW - Physical work space 9 - PRCS - Physical reader base register 10 - IPW$DPA - POWER/VS nucleus 11 - IPW$DTC - Task control block 12 - **** - Reserved for nucleus use 13 - IPW$DSV - Task save area 14 - **** - Subroutine linkage register 15 - **** - Subroutine base register			R0 R1 R2 R3 R4 R5 R6 R7 R8 R9 R10 R11 R12 R13 R14 R15	
	Routine Entry				
RE00	Check if this routine is entered from the Logical Reader. If so, skip opening the interface to the Logical Reader and branch to..... >	RE05			
	Save the parameter values which were passed in register 1 in register 6.			R6	
	Open the interface to the Logical Reader routine.				IPW$OLI Chart AC
	Reserve physical work space to hold 3540 diskette information.				IPW$RSW Chart AC
	Set up register 8 as base to the physical work space area.			R8	
	Save the real address of the physical work space.		PERA(IPW$DPW)		

Labels	Chart HC02: IPW$$ER - 3540 Diskette Reader		Modified Data Fields	Reg. Usage	Calls
	Insert 3540 diskette information into the physical work space:				
	• File identification (from registers 4 and 5)		PEFI(IPW$DPW)		
	• Device type indication (from register 6)		PEDI(IPW$DPW)		
	Byte 1: Device type 3540 diskette reader Bytes 2 and 3: Programmer logical unit 3540 diskette reader				
	• PSTART parameters (from register 7)		PEPS(IPW$DPW)		
	Byte 1: Number of diskettes to be read Byte 2: Volume sequence check indicator Byte 3: Verify indicator				
	Store the device type code in the TCB.		TCDT(IPW$DTC)		
	Load the address of the master record register 3.			R3	
	Clear the option FEED.		PEOP(IPW$DPW)		
	If the option FEED in the master record is not on, then branch to... >	RE05			
	Turn on the option FEED in the physical work space.		PEOP(IPW$DPW)		
RE05	Set the 3540 communication byte in the TCB to X'02' to indicate reading from 3540.		LWER(IPW$DTC)		
	Set the multivolume identification to X'40' to indicate the first of a diskette sequence.		PEMI(IPW$DPW)		
	Set the open return code to X'40'.		PEOC(IPW$DPW)		
	The following fields have an initial value of binary zero set by the reserve work space service:				
	• Extent lower limit		PELO(IPW$DPW)		
	• Record length		PERL(IPW$DPW)		
	• Next sector address		PEED(IPW$DPW)		
	• Number of opened diskettes		PEOD(IPW$DPW)		
	• Volume sequence number		PESN(IPW$DPW)		

Labels	Chart HC02.1: IPW$$ER - 3540 Diskette Reader		Modified Data Fields	Reg. Usage	Calls
	Open Extent				
OP00	Open the first 3540 volume.				IPW$OEF Chart GF
	Check the open return code:				
	If open successful (PEOC=C'0'), branch to..... >	OP10			
	If forced end of volume (PEOC=C'E'), branch to..... >	FC00			
	Check the 3540 communication byte for a card reader with 3540. If not, branch to detach..... >	EX30			
	Set the termination byte in the TCB to C'F' to indicate flush, and branch to flush..... >	EX00	TCTT(IPW$DTC)		

Labels	Chart HC03: IPW$$ER - 3540 Diskette Reader		Modified Data Fields	Reg. Usage	Calls
OP10	Check whether the extent contains any records. If so, branch to process them..... >	OP20			
	Check for more diskettes:				
	If there are, branch to open the next extent..... >	OP00			
	If not, exit..... >	FD00			
OP20	Set the seek address to the begin extent.		PESK(IPW$DPW)		
	If the begin extent does not start on a track boundary, branch to..... >	OP30			
	Otherwise, decrease the track number by one.		PESK(IPW$DPW)		
OP30	Initialize the pre-SEEK address to seek for record 25 on the next track.		PESO(IPW$DPW)		
	Using register 1 as a work register, the record number of the next sector address indicator is decremented by one. If the record number is zero, the cylinder number is decremented by one, and the record number is set to 26.		PEED(IPW$DPW)	R1	
	Initialize Physical Data Area				
IP00	Using registers 2 and 3 as work registers, the amount of storage is calculated for the physical data area, which must be large enough to contain the CCB, 28 CCWs and the data buffers for 26 diskette records.			R2,R3	
	If the total space necessary is smaller than 2008 bytes, branch to reserve a single buffer space..... >	IP20			
	Otherwise, reserve the first part of 2008 bytes.				IPW$RSW Chart AC
	Save the CCB pointer in the physical work space.		PECU(IPW$DPW)		
	Using register 2 as a work register, calculate the displacement between the virtual and real addresses of the physical data area, and save it in the physical work space.		PECD(IPW$DPW)	R2	
	The size of the second physical data buffer to be reserved is calculated in register 5 and divided by the record length to test if it contains an integral number of data records.			R2,R4 R5	

Labels	Chart HC04: IPW$$ER - 3540 Diskette Reader		Modified Data Fields	Reg. Usage	Calls
	If the remainder in register 4 is zero, the data buffers to be allocated do not cross page boundaries, and a branch is made to >	IP10			
	Otherwise, increase the number in register 1 by one to allocate one buffer more in the second data area.			R1	
IP20	The number of data buffers to be allocated in the second data area is saved in the physical work space.		PENN(IPW$DPW)		
	Calculate the total space required for the second data area in register 5 (number of buffers multiplied by record length).			R4,R5	
IP20	The result is rounded off to the next multiple of 32 and copied into register 1.			R1,R5	
	The space requested in register 1 is reserved. (This space can be used as a second data area in case the total space is greater than 2008 bytes, or it can be used as a single data area in case the total space is smaller than 2008 bytes.)				IPW$RSW Chart AC
	If the space is to be used as a second data area, branch to..... >	IP30			
	Otherwise, save the virtual address of the space to be used as a single data area in the physical work space.		PECV(IPW$DPW)		
	Calculate the displacement between the virtual and real addresses and save it in the physical work space.		PECD(IPW$DPW)		
	Branch to..... >	IP40			
IP30	Save the real and virtual addresses of the second data area in the physical work space.		PERN(IPW$DPW) PEVN(IPW$DPW)		
IP40	Establish addressability for the CCB in register 1 to initialize the first 16 bytes of the physical data area as a CCB.			R1	
	Set the first communication byte to X'08' to indicate wait for channel end and return on data check.		CBC1(IPW$DCB)		
	Move the programmer logical unit to the CCB.		CBLC(IPW$DCB)		
	Set the EXCP real indication in the CCB.		CBLC(IPW$DCB)		

Labels	Chart HC05: IPW$$ER - 3540 Diskette Reader		Modified Data Fields	Reg. Usage	Calls
	Store the real address of the first CCW in the CCB.		CBCA(IPW$DCB)		
	Load the address of the first CCW into register 3 to establish CCW addressability.			R3	
	Move the SEEK CCW.		CWCC(IPW$DCW)		
	Store the seek address in the CCW.		CWDA(IPW$DCW)		
	Calculate the real address of the first data buffer in register 6, and the virtual address in register 4.			R4,R6	
	Save the real and virtual addresses of the first data buffer in the physical work space.		PEDA(IPW$DPW) PEDV(IPW$DPW)		
	Load the record length into register 7.			R7	
	Load the number of buffers to be allocated in a possible second data area in register 1.			R1	
	Calculate the number of buffers to be allocated in the first data area (or possible single data area) in register 5.			R5	
IP50	Move the read CCW image, the real buffer address and the record length for all CCW entries.		CWDS(IPW$DCW) CWDA(IPW$DCW) CWCT(IPW$DCW)		
IP60	Initialize the pre-SEEK CCW to perform seek overlap to record 25 on the next track as the last CCW in the string.		CWCC(IPW$DCW) CWDA(IPW$DCW) CWFL(IPW$DCW)		
	Load the real address of the pre-SEEK CCW into register 6.			R6	
	Set the record counter to the first record from the extent in register 7.			R7	
	Read Data Block				
RD00	Calculate the number of records read on the track in register 1.			R1	
	If the current track has not been completely read, or if the first track of the extent has to be read, branch to..... >	RD10			
	Otherwise, reset the record counter in register 7 and the record number in the physical work space to one, and branch to..... >	RD20	PESK(IPW$DPW)	R7	

Labels	Chart HC06: IPW$$ER - 3540 Diskette Reader		Modified Data Fields	Reg. Usage	Calls
RD10	Store the current record number in the physical work space.		PESK(IPW$DPW)		
	If a short string has to be generated to read the remaining record on the current track, branch to..... >	RD35			
RD20	Increment the track number by one to address the next track.		PESK(IPW$DPW)		
	If the pre-SEEK address equals the last track of the extent, branch to >	RD30			
	Otherwise, update the track number in the pre-SEEK address to perform seek overlap to the next track, and branch to..... >	RD40			
RD30	Set the pre-SEEK address equal to the current SEEK address.				
	If a complete track has to be read, branch to..... >	RD40			
RD35	Otherwise, generate a short CCW string to read the remaining records.				
RD40	Set up CCB addressability in register 1.			R1	
	Save the real address of the first CCW in register 2.			R2	
	Execute the channel program via SVC 0 and wait for completion.				IPW$WFC Chart AA
	Restore the address of the first CCW.		CBCA(IPW$DCB)		
	Load the address after the seek CCW into register 2, and the address of the last executed CCW into register 3.			R2,R3	
	Load the address of the first data buffer into register 0, and the record length into register 1. Calculate the number of buffers allocated in the first data area in register 4, and branch to pass the records to the logical reader..... >	DB20		R0,R1, R4	
	Deblock Routine				
DB00	Load the 3540 parameters into registers 0 and 1.			R0,R1	
	Update the record pointer in register 0.			R0	
	If the record belongs to the first or single data area, branch to..... >	DB10			

Labels	Chart HC07: IPW$$ER - 3540 Diskette Reader		Modified Data Fields	Reg. Usage	Calls
	Otherwise, reload register 0 with the address of the first record in the second data area.			R0	
DB10	Update the CCW pointer in register 2.			R2	
	Update the record counter in register 7 by one.			R7	
DB20	Check if the last record has already been deblocked. If not, branch to pass the record to the logical reader..... >	DB40			
	If I/O ends with the normal pre-SEEK CCW, branch to read a new block.... >	RD00			
	If the string is not broken at the inserted pre-SEEK CCW, branch to check for a special record..... >	DB30			
	Branch and link to reset the pre-SEEK CCW to a read CCW..... >	SR20		R14	
	Branch to read a new block..... >	RD00			
DB30	Update the record counter (register 7).			R7	
	Branch and link to reset a possible pre-SEEK CCW to a read CCW..... >	SR20			
	Check if the last cylinder is being read from. If not, branch to read the next block..... >	RD00			
	Check if the last record has been reached:				
	If so, branch to open the next extent..... >	DB60			
	Otherwise, read the next block..... >	RD00			
DB40	Check for data file processing. If so, branch to..... >	DB50			
	Check the record length (register 1), and if it is not 81, branch to..... >	DB55			
	Decrement the record length by one to make it 80.			R1	
	Increment the record pointer (register 0) by one.			R0	
	Branch to process in SYSIN mode.... >	DB55			
DB50	Indicate data mode in the TCB.		TCGP(IPW$DTC)		

Labels	Chart HC08: IPW$$ER - 3540 Diskette Reader		Modified Data Fields	Reg. Usage	Calls
DB55	Branch and link to pass the record to the logical reader ..... >	SR10		R5	
	If the last record on the last cylinder has not yet been passed to the logical reader, branch to continue processing..... >	DB00			
	Branch and link to reset a possible inserted pre-SEEK CCW to a read CCW >	SR20		R5	
DB60	Release the first or only part of physical data buffer space.				IPW$RLW Chart AC
	If a second data area has not been reserved, branch to clear the physical work space..... >	DB70			
	Release the second physical data area.				IPW$RLW Chart AC
DB70	Set the buffer number to zero.		PENN(IPW$DPW)		
	Check if there is a continuation diskette. If so, branch to open the next extent..... >	OP00			
	Exit Routine				
FD00	If the option FEED in the PWS is not on, branch to..... >	EX00			
	Branch and link to feed the next diskette..... >	SR50			
EX00	Check for data file processing, and if so, branch to handle end of data >	EX40			
	Check if the SYSIN file is linked to a card reader, and if so, branch to prepare for resetting the linkage.. >	EX60			
	Check for job boundary, and if so, do not insert the EOF record and branch to..... >	EX20			
	Load the address of the EOJ record into register 0 and the length into register 1.			R0,R1	
	Branch and link to pass the EOJ record to the logical reader..... >	SR10		R5	
	Set the disposition indicator in the queue record to C'H' to indicate job in HOLD state.		QRDP(IPW$DQR)		
	Issue message '1Q89I cuu EOJ ADDED jobname jobnumber'				IPW$WTO Chart AD
EX20	Branch and link to send the unit exception signal to the logical reader..... >	SR10		R5	

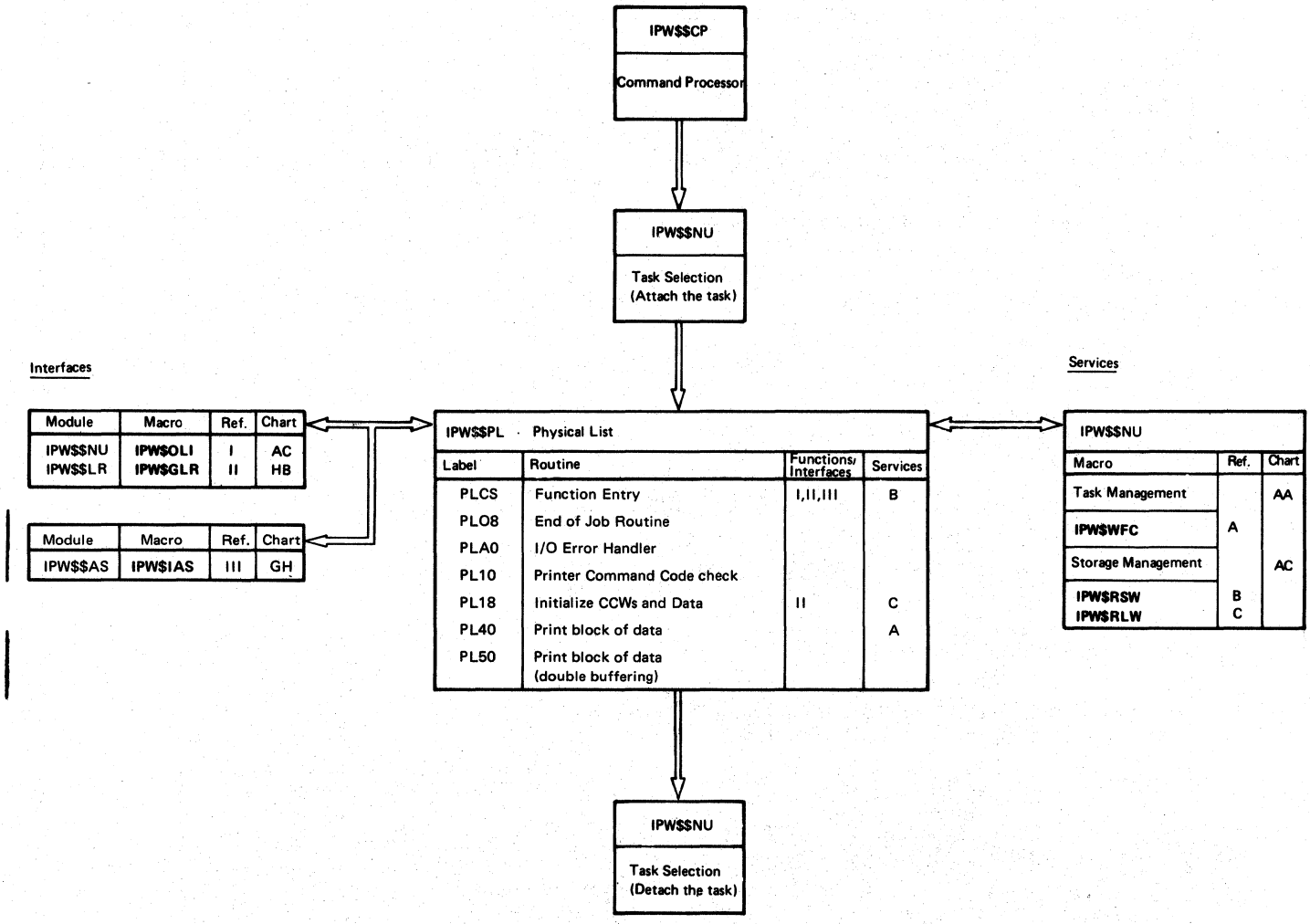
Labels	Chart HC09: IPW$$ER - 3540 Diskette Reader		Modified Data Fields	Reg. Usage	Calls
EX30	Set the stop condition in the TCB.		TCTT(IPW$DTC)		
	Exit to the terminator routine (IPW$$TR), which results in a detach of the current 3540 diskette task.				IPW$$TR Chart LC
EX40	Indicate last data record in the TCB.		TCGP(IPW$DTC)		
	Load the address of a dummy end of data record into register 0, and the length into register 1.			R0,R1	
	Branch and link to pass the end of data record to the logical reader.. >	SR10		R5	
EX60	Reset the variable fields of the 3540 part of the physical work space to binary zeros.		PEFI(IPW$DPW)		IPW$RLW Chart AC
	Branch and link to send the unit exception signal to the logical reader..... >	SR10		R5	
	The logical reader now resets the linkage to the card reader and resumes reading from the card reader.				
	Subroutines				
SR10	Pass the record to the logical reader.				IPW$PLR Chart JC
	Return to caller via link register 5.			R5	
SR20	Load the address of the forced pre-SEEK CCW into register 2.			R2	
	If there is no forced pre-SEEK CCW, branch to return..... >	SR25			
	Make the address virtual and use it as a base register.			R2	
	Move the read CCW saved in the physical work space back into the CCW string.		CWDS(IPW$DCW)		
	Clear the forced pre-SEEK CCW address in the physical work space.		PEBS(IPW$DPW)		
SR25	Return to caller via link register 4.			R4	

Labels	Chart HC10: IPW$$ER - 3540 Diskette Reader		Modified Data Fields	Reg. Usage	Calls
	Subroutine to insert a pre-SEEK CCW in the middle of the CCW string to read part of a track.				
SR40	If a short string has to be executed which is not on the last track, branch to..... >	SR45			
	Otherwise, change the pre-SEEK address to the current seek address, and calculate the number of remaining records to be read in register 1, depending on the last record of the extent.		PESO(IPW$DPW)	R1	
SR45	Calculate the address of the read CCW to be changed in register 3, and establish CCW addressability.			R3	
	Save the read CCW image in the physical work space.		PEDW(IPW$DPW)		
	Point register 2 to the address of the pre-SEEK CCW.			R2	
	Move the pre-SEEK CCW image over the read CCW to be changed.		CWCC(IPW$DPW)		
	Save the real address of the inserted pre-SEEK CCW in the physical work space.		PELS(IPW$DPW)		
	Return to caller via link register 14.			R14	
	Subroutine to build a CCB with CCW to FEED and SENSE				
SR50	Calculate length for the CCB and two CCW's in register one.			R1	
	Reserve work space address from work space in PWS.		PECV(IPW$DPW)		IPW$RSW
	Calculate the displacement (virtual address and real address).			R2	
	Build the CCB.				
	Build the CCW FEED.				
	Build the CCW SENSE.				
	Execute channel program.				SVC0
	Wait for operation to complete.				IPW$WFC
	Release work space.				IPW$RLW
	Return to caller.				

WRITER ROUTINES

CHART JA: IPW$$PL - PHYSICAL LIST (8 PARTS)

Chart JA00: IPW$$PL - Physical List, General Flow and Macro Calls



Module	Macro	Ref.	Chart
IPW$$NU	IPW$OLI	I	AC
IPW$$LR	IPW$GLR	II	HB

Module	Macro	Ref.	Chart
IPW$$AS	IPW$IAS	III	GH

Label	Routine	Functions/ Interfaces	Services
PLCS	Function Entry	I,II,III	B
PLO8	End of Job Routine		
PLA0	I/O Error Handler		
PL10	Printer Command Code check		
PL18	Initialize CCWs and Data	II	C
PL40	Print block of data		A
PL50	Print block of data (double buffering)		

Macro	Ref.	Chart
Task Management		AA
IPW$WFC	A	
Storage Management		AC
IPW$RSW	B	
IPW$RLW	C	

Labels	Chart JA01: IPW$$PL - Physical List		Modified Data Fields	Reg. Usage	Calls
PLDS	The first 16 bytes constitute the section descriptor:				
	'PLCS V10M1 '				
	On entry the following register contents are relevant:				
	9: Section base register			R9	
	10: address of permanent area		IPW$DPA	R10	
	11: task control block		IPW$DTC	R11	
	13: save area list task		IPW$DSV	R12	
	1: device information byte 0 - number of buffers '01' one print buffer '02' two print buffer 'F2' two print buffer and two data buffers byte 1 - device type byte 2-3 - programmer logical unit number			R1	
	<u>Function Entry</u>				
	The physical list routine is entered after the command processor has initialized a list task control block and printer device information (passed in parameter register 1) whenever a PSTART command is given for a list task.				
	The entry parameters in register 1 are saved in register 6.			R6	
	The interface with the logical writer is opened through an IPW$OLI call.				IPW$OLI Chart AC
	Storage for the 3800 TCB extension area is reserved when the printer is a 3800.				IPW$RSW Chart: AC
	The address is saved in the TCB.		TC3E(IPW$DTC)		
PL02	The first data record is requested from the logical writer through an IPW$GLR call...................... >	PL30			
	Register 7 is loaded with the record address. If this is zero, indicating that no record was available, the IPW$GLR call is reissued through a branch to........................ >	PL02		R7	
	Physical work space is reserved through an IPW$RSW call, to store the device-dependent printer data and, of printer CCB, CCWs, and data records to be printed by this list task. 64 bytes of work space are requested.				IPW$RSW Chart AC
	Physical work space is made addressable through register 8.		IPW$DPW	R8	

Labels	Chart JA02: IPW$$PL - Physical List		Modified Data Fields	Reg. Usage	Calls
	Device dependent information is stored.		PWDI(IPW$DPW)		
	Register 5 is loaded with the number of buffers.			R5	
	PDA (Physical Data Area) is now reserved through an IPW$RSW call, for use as CCB, CCWs and data buffer.				IPW$RSW Chart AC
	The appropriate PDA size is obtained from the permanent area.		BLBF(IPW$DPA)		
	Real address of PDA space is saved in register 2.			R2	
	The virtual and real addresses of the CCB (first 16 bytes of the PDA) are saved in the PWS. Branch to continue..... >	PL06	PBV1(IPW$DPW) PBR1(IPW$DPW)		
PL05	The second PDA is reserved for CCB, related CCWs and data buffer. The virtual and real addresses of the CCB (first 16 bytes of the PDA) are saved in the PWS.		PBV2(IPW$DPW) PBR2(IPW$DPW)	R1,R2	IPW$RSW Chart: AC
	Now the first 16 bytes of the buffer space are set up as a CCB, indicating:				
	• Wait for device end		CBC1(IPW$DCB)		
	• Accept unrecoverable I/O error		CBC1(IPW$DCB)		
	• Command chain retry		CBC2(IPW$DCB)		
	• Printer logical unit number		CBLC(IPW$DCB)		
	• EXCP real.		CBLC(IPW$DCB)		
	The real address of the first CCW, to follow immediately behind the CCB, is loaded in register 6 and stored in the CCB and in the PWS.		CBCA(IPW$DCB) PWCA(IPW$DPW)	R6	

Labels	Chart JA02.1: IPW$$PL - Physical List		Modified Data Fields	Reg. Usage	Calls
	Register 6, to be used to address the CCW when initializing the CCW chain, is loaded with the address of the first CCW.			R6	
	This first CCW is now initialized as a NOP CCW, and the CCW data address is initialized with the real address of the end of the PDA.		CWDA(IPW$DCW)		
	If double buffering was specified in the PSTART command, branch to reserve and initialize the second PDA..... >	PL05			
	Register 5 is now set up with the real address of the end of the buffer space, which is then stored in the PWS.		PWDA(IPW$DPW)	R5	
	The appropriate PDA size is obtained from the permanent area.		BLDB(IPW$DPA)		
	Register 5 is now set up with the virtual address of the end of the buffer space, which is then stored in the PWS.		PWDV(IPW$DPW)	R5	
	The address of the first record is saved in register 6.			R6	
	Register 7 is set up to contain the address of the proper printer command check table through a translate and test instruction executed on the device type byte, thus obtaining a displacement value in register 2. That value, multiplied with the table entry length, will yield the proper displacement in bytes in register 3. Using this value, register 7 is then loaded with the appropriate table address.			R2,R7 R3,R7	
	Now, the record address is reloaded in register 3.			R3	
	Register 1 is loaded with the address of the first PDA and register 6 is reestablished to point to the first CCW.			R1 R6	
	If this is not a 3800 printer, branch to..... >	PL10			
	Otherwise, a IPW$IAS TYPE=ATTACH request is issued to attach the DOS/VS subtask.				IPW$$AS Chart: GH01
	Branch to continue..... >	PL10			

Labels	Chart JA02.1: IPW$$PL - Physical List		Modified Data Fields	Reg. Usage	Calls
	End of job processing				
	At end of job, the work spaces are released and return is made to the logical writer to obtain the next job.				
PL08	Register 1 is loaded with the address of the first PDA and the PDA is released.			R1	IPW$RLW Chart: AC
	If a second PDA exists, it is released too.			R1	IPW$RLW
PL08A	If printer being used is a 3800, a request to detach the DOS/VS subtask is issued (IPW$IAS TYPE=DETACH).			R0	IPW$$AS Chart: GH01
PL09	The following device dependent information is saved in register 6: • number of buffers being used • device type • programmer logical unit number			R6	
	Register 1 is loaded with the PWS address and the PWS is released.			R1	IPW$RLW Chart: AC
	Indication is set; PWS released.			R8	
	Branch to continue..... >	PL02			
	Abnormal condition handler				
	This routine is entered whenever an unrecoverable I/O error has been detected by DOS/VS.				
PLA0	Prepare reply area. Issue message 1Q61D and wait for reply.		PWRA(IPW$DPW) PWML(IPW$DPW)	R5,R6	IPW$WTR Chart AD
	If reply is 'C', branch to..... >	PLA8			
	If reply is 'T' and single buffering is being used, branch to..... >	PL44			
	If reply is 'I' and double buffering is being used, branch to..... >	PL54			
PLA2	If reply is not 'R', branch to..... >	PLA0			
	If a backup count is specified, branch to..... >	PLB0			
	Branch to count skips to channel one not yet printed..... >	PLC0		R14	
	Set up for restart number of pages + 1.		TCRS(IPW$DTC)	R5	
PLA6	If single buffering is being used, branch to..... >	PL48			
	Clear print buffers. Branch to.... >	PL66		R1,R2, R14	

Labels	Chart JA02.1: IPW$$PL - Physical List		Modified Data Fields	Reg. Usage	Calls
	Indicate new start.		PWVE(IPW$DPW)		
	Branch to get next record..... >	PL26			
PLA8	Set termination byte to 'S' (Stop). Branch to termination routine.		TCTT(IPW$DTC)	R2	IPW$$TR Chart LC
PLB0	Test if reply is numeric (up to six digits). If not, branch to..... >	PLA0		R1,R2	
	Convert reply to binary and set up for restart number of pages + 1.		TCRS(IPW$DTC)	R1,R2, R5	
	Branch to count number of skip to channel ones not yet printed..... >	PLC0		R14	
	Add number of skip to channel ones not yet printed to number of pages to go back (specified by operator). Branch to..... >	PLA6	TCRS(IPW$DTC)	R5,R6	
	Page count subroutine This subroutine counts all skip to channel ones in the print buffer(s), which are not yet printed.				
PLC0	Calculate last executed CCW in the active print buffer.			R5,R6	
PLC2	(Note: NOP-CCW without chain-flag stops CCW-chain.) The remaining CCW-chain is scanned for skip to channel one operation code. If so, branch to..... >	PLC4			R5,R6
	Otherwise, address next CCW in the chain and loop through..... >	PLC2		R6	
PLC4	Add 1 to the skip to channel one count. Address next CCW in chain and branch to..... >	PLC2			
PLC6	If double buffering is not being used, return is made to the caller. >				R14
PLC8	The first CCW in the other buffer is addressed.				R6
PLD2	The CCW chain is scanned for skip to channel ones. If found, the page count is increased by 1 and the next CCW is addressed. Return to the caller..... >				R6 R5 R14

Labels	Chart JA03: IPW$$PL - Physical List		Modified Data Fields	Reg. Usage	Calls
	Command Code Check Routine				
PL10	The logical writer has passed the printer command code posted in the list task control block. This command code is examined using the translate and test table PLCT:				
	If command code invalid, branch to ignore command..... >	PL25			
	If command code is valid for any printer, branch to continue handling record..... >	PL18			
	If command code not only valid for 1 type of printer, branch to continue command code check..... >	PL12			
	If same device as during execution, branch to continue handling record..... >	PL18			
	If not valid for this particular printer, branch to ignore..... >	PL25			
PL12	Otherwise, the proper branch index is loaded from the device block (pointed to by register 7) into register 2, and branch into the following branch table.			R2	
PL14	0: Invalid, branch to ignore command..... >	PL26			
	4: Valid, branch to handle record..... >	PL18			
	8: Branch to empty buffer first.. >	PL16			
	c: Branch to perform FCB load.... >	PL15			
	10: Branch to perform printer setup..... >	PL32			
	A link is made to print PDA..... >	PL40			
PL15	If the FCB to be loaded is already loaded, branch to..... >	PL26			
	Save the new FCB phase name.		LWFB(IPW$DTC)		
	If double buffering, branch to wait for the completion of the last I/O. >	PL50	PWOT(IPW$DPW)		
	Convert logical unit number from CCB into LFCB macro expansion.				
	Wait for free LTA.				
	Call $$BATTF0 to load FCB buffer.				
	If load FCB successful, branch to.. >	PL26		R15	
	Otherwise, give LFCB error message.				IPW$WTO
	Indicate stop task, and branch to.. >	PL26	TCTT(IPW$DTC)		

Labels	Chart JA04: IPW$$PL - Physical List		Modified Data Fields	Reg. Usage	Calls
PL16	A link is made to print PDA..... >	PL40		R14	
	Branch to continue record handling..... >	PL18			
	Initialize CCWs and Buffer				
PL18	This routine calculates the remaining PDA space and checks if it can contain the data record and two accompanying CCWs. If so, the record and CCWs are moved into the PDA and an IPW$GLR call is issued to logical writer to get the next data record. The CCW string is started right behind the CCB at the beginning of the data buffer space and the data records are chained backwards in the PDA, starting at the end of the PDA. This procedure is followed to optimize the use of PDA space, since the variable length list records prevent any calculation of a data record start address if data records are chained forward normally in the PDA. If insufficient PDA is available, the data buffer is emptied first by writing the data records to the printer.				
	Register 1 is loaded with the address of the next CCW to be built.			R1	
	Register 2 is loaded with the address of the end of the available buffer space.			R2	
	The remaining PDA space is calculated by subtracting register 1 from register 2.			R1,R2	
	If remaining space is zero or more, a branch is made to attempt moving of CCW and data record..... >	PL20			
	Otherwise, a link is made to empty the PDA..... >	PL40			
	On return, a branch is made to build CCW and move data record to the PDA..... >	PL22			
PL20	If the record to be moved fits in the remaining PDA space, a branch is made to build CCW and move data record to the PDA..... >	PL22			
	Otherwise, a link is made to empty the PDA..... >	PL40			

Labels	Chart JA05: IPW$$PL - Physical List		Modified Data Fields	Reg. Usage	Calls
PL21	When the device is a 3800 and the CCW represents 'load forms control buffer', a branch is made to update the PUB2 area, signalling the fact that the FCB now loaded on the printer is unknown..... >	PL90		R14	
PL22	The command code, contained in the high-order byte of register 3, is stored in the new CCW.		CWCC(IPW$DCW)		
	The real data address, now pointing to the end of the available data buffer space, is loaded in register 2, the record length contained in register 4 is subtracted, and the updated real data address is stored back into the CCW.			R2	
			CWDA(IPW$DCW)		
	The CCW flags are initialized to indicate command chaining and suppress incorrect length indication.		CWFL(IPW$DCW)		
	Register 6 is set to point to the next CCW, which is now made the NOP CCW terminating string, while its data address is made to point to the end of the available data buffer space after moving the current data record.			R6	
			CWDA(IPW$DCW)		
	Register 5 is made to point to the target address of the current data record in the PDA, by subtracting the record length, as contained in register 4, from the address of the end of available buffer space.			R5	
	If the record length is not more than 256 bytes, a branch is made to move the record with a normal MVC instruction, subject of an EXECUTE instruction..... >	PL24			
	Otherwise, registers 2, 1, and 0 are set up as from-address, to-length and to-address, respectively, to be used as tne operands of the MVCL instruction that moves the data record.			R2,R1 R0	
	After moving the record to the PDA, a branch is made to obtain the next record from the logical writer..... >	PL2A			
PL24	Register 4 is decremented by one to contain the record length in machine format and the record is moved using an EXECUTE instruction.			R4	

Labels	Chart JA05.1: IPW$$PL - Physical List		Modified Data Fields	Reg. Usage	Calls
PL2A	If the CCW operation code was a clear printer X'87', a branch is made to empty the print buffer(s).				
	If double buffering, branch to wait for the completion of the last I/O. >	PL50	PWOT(IPW$DPW)	R14	
	(This is necessary because the double print buffer processing routine does not wait for the completion of the I/O.)				
PL25	Check if the last processed data record is the last record in the data buffer. If not so, or if double buffering is being used, or if the PDA is empty, branch to..... >	PL26			
	Load the CCB address into register 1, and execute the channel program via SVC 0.			R1	

Labels	Chart JA06: IPW$$PL - Physical List		Modified Data Fields	Reg. Usage	Calls
	Request the next record from the logical writer..... >	PL30			
	Wait for I/O completion..... >	PL41		R14	
	Branch to..... >	PL27			
PL26	The next data record is requested from the logical writer through an IPW$GLR call.				
PL27	If a record is available, a branch is made to process the new record..... >	PL10			
	If a zero address was passed, indicating that no record is available, a link is made to print the PDA..... >	PL40		R14	
	If single buffering is being used, branch to End of Job routine..... >	PL08			
	Set wait request, and branch to wait for the completion of the previous issued I/O..... >	PL50	PWOT(IPW$DPW)	R14	
	Branch to continue..... >	P108			
PL30	Request the next record from the logical writer.				IPW$GLR
	Save record address and length, respectively, in register 3 and register 4, and return to caller.			R2 R3,R4	
	SETPRT request handler This routine is entered whenever a SETPRT request (signalled by a dummy CCW code of 'FD') is encountered for a 3800 printer.				
PL32	A branch is made to empty the print buffer..... >	PL40		RE	
	If double buffering is being used, a wait request is issued by branching to..... >	PL50	PWOT(IPW$DPW)	RE	
	(This is necessary because the double print buffer processing routine does not wait for the completion of the I/O.)				
PL34	When the DUMP/TRACE option has been specified in the SETPRT parameter list, SYSLST is assigned to the same printer being used.				IPW$ULP Chart: GC

Labels	Chart JA07: IPW$$PL - Physical List		Modified Data Fields	Reg. Usage	Calls
	If SYSLST is already assigned, message '1QA6I' is written.				IPW$WTO
	The dump/trace flag is turned off.		SPLFLAG2 (IPW$DTE)		
PL36	Storage for the service request block (SRB) is acquired. The SRB is formatted:				IPW$RSW Chart: AC
	• SETPRT request		SRBREQ (IPW$DSR)		
	• address of SETPRT parameter list		SRBPARM		
	The 3800 TCB extension area is made addressable using register 2 as base.			R2	
	The proper programmer logical unit is inserted in the SETPRT parameter list.		SPPLUSYS		
	The actual copy group index is inserted in the parameter list.		SPPCINDX		
	The operator message suppress flag is set.		SPLFLAG1 (IPW$DTE)		
	If setup is requested, branch to... >	PL37			
	Otherwise, check if the new SETPRT request matches the previous request. If so, skip setup processing.				
	Branch to..... >	PL38			
	Copy the SETPRT parameter list into the TCB extension area.		SPLLIST (IPW$DTE)		
PL37	The service request block is passed to asynchronous service for processing by means of the IPW$IAS TYPE=SERVICE macro instruction.				IPW$$AS Chart ??
	Turn off flags for FCB verification, mark form, and offset stacking.		SPLFLAG2 SPLFLAG1 (IPW$$TE)		
PL38	The service request block is released and returned to the storage pool.				IPW$RLW
	When the dump/trace option was specified, SYSLST is unassigned by invoking the appropriate function.				IPW$ULP Chart: GC
	Branch to continue..... >	PL26			
	Print Routine				
	This routine handles the actual printing of list records, as well as all resulting normal and error I/O conditions.				

Labels	Chart JA07.1: IPW$$PL - Physical List		Modified Data Fields	Reg. Usage	Calls
PL40	On entry, register 5, supposed to point to the end of the free buffer space, is compared to the virtual end of the data space address in the PWS.			R5	
	If equal, indicating an empty block, immediate return is made to the caller.			R14	
	If double buffering is being used, branch to..... >	PL50			
	Otherwise, register 1 is loaded with the virtual CCB address as stored in the PWS, to serve as the parameter register for the EXCP (SVC 0), which is now issued.			R1	

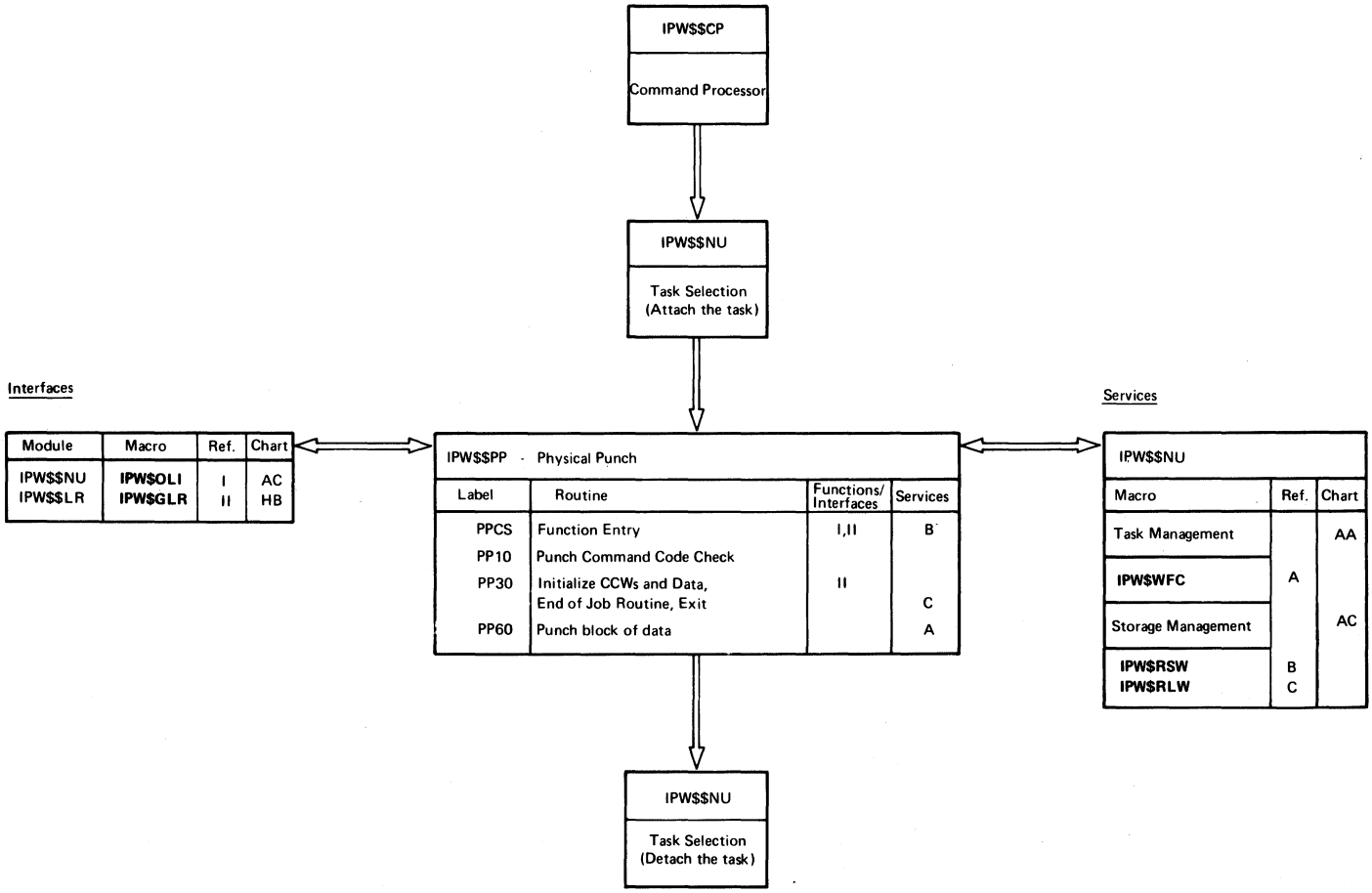
Labels	Chart JA07.2: IPW$$PL - Physical List		Modified Data Fields	Reg. Usage	Calls
PL41	On return, an IPW$WFC call is issued to have the list task wait for I/O completion.				IPW$WFC Chart AA
	The CCB is checked for unrecoverable I/O error. If so, branch to ask operator for proper action..... >	PLA0			
	The CCB is checked for unit exception (channel 12 overflow), and ignored errors.				
	If so, a branch is made to restart I/O to the printer at the point of CCW chain interruption..... >	PL44			
	If no channel 9 overflow is indicated in the CCB either, a branch is made to bypass I/O restart..... >	PL48			
PL44	Otherwise, the CCW chain interruption address is made CCW start address in the CCB, and a branch is made to the subroutine entry to restart the I/O at the point of interruption..... >	PL40	CBCA(IPW$DCB)		
PL48	The CCW start address in the CCB, which might have been changed on processing a channel 9 or 12 overflow, is now reset using the address saved in the PWS, and register 6 is set to point to the first CCW again.		CBCA(IPW$DCB)	R6	
	The data address in the first CCW is reset to the real address of the end of the data buffer space.		CWDA(IPW$DCW)		
	Register 5 is reset to the virtual address of the end of the data buffer space.			R5	
	A branch is now made back to the caller.			R14	
	Double Print Buffer routine				
PL50	Registers 1 and 2 are loaded with the virtual and real CCB addresses respectively.			R1,R2	
	If this is the first time through, branch to..... >	PL60			
	Register 1 is loaded with the CCB address of the active buffer.			R1	
	If the CCB is posted, branch to.... >	PL52			
PL51	Otherwise, issue IPW$WFC request and wait for completion of previous I/O.				IPW$WFC Chart: AA

Labels	Chart JA07.3: IPW$$PL - Physical List		Modified Data Fields	Reg. Usage	Calls
PL52	The CCB is checked for unrecoverable I/O error. If so, branch to ask operator for proper action..... >	PLA0			
	The CCB is checked for unit exception (channel 12 overflow), and ignored errors. If so, a branch is made to restart I/O to the printer at the point of CCW chain interruption..... >	PL54			
	If channel 9 overflow is indicated in the CCB, a branch is made to bypass I/O restart..... >	PL58			
PL54	Otherwise, the CCW chain interruption address is made CCW start address in the CCB.		CBCA(IPW$DCB)		
	An SVC 0 is issued. Branch to wait for I/O completion.. >	PL51			
PL58	Registers 1 and 2 are loaded with the virtual and real addresses of the other print buffers respectively.			R1,R2	
PL60	If this is a wait only request, branch to bypass the SVC 0..... >	PL62			
	Registers 1 and 2 are stored in the PWS to address the active print buffer.		PWVE(IPW$DPW)		
	Register 2 is loaded with the real address of the first CCW in the PDA, which is stored in the CCB and in the PWS.		CBDA(IPW$DCB) PWCA(IPW$DPW)	R2	
	Register 1 contains the virtual address of the CCB as stored in the active print buffer and serves as parameter register for the EXCP (SVC 0), which is now issued.				
PL62	The wait only request is turned off.		PWOT(IPW$DPW)		
	Registers 1 and 2 are loaded with the virtual and real addresses of the available print buffer.			R1,R2	
PL66	Register 5 is now set up with the real address of the end of the buffer space, which is that stored in the PWS.		PWDA(IPW$DPW)	R5	
	Register 5 is now set up with the virtual address of the end of the buffer space, which is that stored in the PWS.		PWDV(IPW$DPW)	R5	

Labels	Chart JA07.4: IPW$$PL - Physical List	Modified Data Fields	Reg. Usage	Calls
	Register 6 is set to point to the first CCW again. The first CCW is initialized as a NOP-CCW and the associated data area address is initialized with the real address of the end of the PDA.	CWDA(IPW$DCW)	R6	
	A branch is made back to the caller.		R14	
	Update PUB2 area:			
PL90	The TCB extension area is made addressable using register 2 as a base.			
	The high-order byte of the FCB field in the extension area is set to X'FF' to signal that the TCB image name is unknown on the printer.	SPLFCB (IPW$DTE)		
	The PUB2 is updated by issuing the MODCTB macro instruction.			
	First the MODCTB parameter list is built: • set ID flag	TE38GW (IPW$DTE)	R15	
	• set length of FCB field being updated.			
	• calculate displacement of FCB field in PUB2 area and store in parameter list.			
	• store pointer to programmer logical unit		R15	
	Address the parameter list using register 1 and clear the return register (15). Issue SVC 98.		R15,R1	
	Upon return, test the return register for successful completion. If so, branch back to caller via register 14.		R14	
	Otherwise, set task stop indicator and return to caller.	TCTT(IPW$DTC)	R14	

CHART JB: IPW$$PP - PHYSICAL PUNCH (7 PARTS)

Chart JB00: IPW$$PP - Physical Punch, General Flow and Macro Calls



Labels	Chart JB01: IPW$$PP - Physical Punch	Modified Data Fields	Reg. Usage	Calls
PPDS	The first 16 bytes constitute the section descriptor:			
	'PPCS V10M0			
	On entry the following register contents are relevant:			
	9: section base register		R9	
	10: address of permanent area	IPW$DPA	R10	
	11: Task control block	IPW$DTC	R11	
	13: save area punch task	IPW$DSV	R13	
	1: device type and logical unit number of punch.		R1	
	<u>Function Entry</u>			
	The physical punch routine is entered after the command processor has initialized a punch task control block and punch device information (passed in parameter register 1) whenever a PSTART command is given for a punch task.			
	The entry parameters in register 1 are saved in register 6.		R6	
PP02	The interface with the logical writer is opened through an IPW$OLI call.			IPW$OLI Chart AC
PP04	The first data record is requested from the logical writer through an IPW$GLR call.			IPW$GLR Chart HB
	On return from the logical writer, register 0 should contain the address of the first data record passed, and register 1 its length.		R6 R1	
	Register 4 is now loaded with the record length.		R4	
	Register 5 is loaded with the record address. If this is zero, indicating that no record was available, the IPW$GLR call is reissued through a branch............................ > PP04		R5	
	Physical work space is reserved through an IPW$RSW call, to store the device-dependent punch data and, possibly, the addresses of punch CCB, CCWs and data records to be punched by this punch task. 32 bytes of work space are requested.			IPW$RSW Chart AC
	Physical work space is made addressable through register 8.		R8	
	Device dependent information is stored.	PWDI(IPW$DPW)		

Labels	Chart JB02: IPW$$PP - Physical Punch	Modified Data Fields	Reg. Usage	Calls
	Register 7 is set up to contain the address of the proper printer command check table through a translate and test instruction executed on the device type byte. This gives a displacement value in register 2, which, multiplied by the table entry length, will yield the proper displacement in bytes in register 3. Using this value, register 7 is then loaded with the appropriate table address.		R7 R2 R3 R7	
	PDA (Physical Data Area) space is now reserved through an IPW$RSW call, for use as CCB, CCWs and data buffer. The appropriate PDA size is obtained from the permanent area.	BLBF(IPW$DPA)		IPW$RSW Chart AC
	Real address of PDA is saved in register 2.		R2	
	Virtual address of PDA is stored in PWS.	PBV1(IPW$DPW)		
	Record length is loaded in register 3.		R3	
	Real address of end of PDA is loaded in register 5, and stored in PWS.	PWDA(IPW$DPW)	R5	
	Now the first 16 bytes of the buffer space are set up as a CCB, indicating: • Wait for device end • Accept unrecoverable I/O error • Command chain retry • Punch logical unit number • EXCP real.	 CBC1(IPW$DCB) CBC1(IPW$DCB) CBC2(IPW$DCB) CBLC(IPW$DCB) CBLC(IPW$DCB)		
	The real address of the first CCW, to follow immediately behind the CCB, is loaded in register 2 and stored in the CCB, as well as in the PWS.	CBCA(IPW$DCB) PWCA(IPW$DPW)	R2	
	Register 6, to be used to address the CCW when initializing the CCW chain, is loaded with the address of the first CCW.		R6	
	This first CCW is now initialized as a NOP CCW, and the CCW data address is initialized with the real address of the end of the PDA.	CWDA(IPW$DCW)		
	Register 5 is now set up with the virtual address of the end of the PDA, which is then stored in the PSW.	PWDV(IPW$DPW)	R5	

Labels	Chart JB03: IPW$$PP - Physical Punch		Modified Data Fields	Reg. Usage	Calls
	Command Code Check Routine				
	The logical writer has passed the punch command code posted in the punch task control block. The command code is changed from X'09' to X'01'.		TCCC		
PP10	If command is X'X9' and device is 2450P, command code is changed to X'X1'.				
PP11	This command code is examined using a translate and test table.				
	If command code is invalid, branch to ignore command..... >	PP48			
	If not dummy command (X'00'), branch to continue..... >	PP12			
	If it is, pick up punch and feed command out of table, and branch... >	PP30		R7	
PP12	If this is the same device as during execution, branch to continue handling record..... >	PP20			
	Otherwise, load the proper branch index from device block (pointed to by register 7) into register 2, and branch into the following branch table			R2	
PP14	0: Invalid, branch to ignore command>	PP48			
	4: Valid, branch to handle record.. >	PP20			
	8: 2560 print, branch to indicate that the PDA has to be emptied.. >	PP16			
PP16	The switch to indicate that the PDA has to be emptied is set on in the TCB.		PPEB(IPW$DTC)		
	The record length is set to 1.				
	If separator cards are requested, the default stacker must be selected. The device type is checked and a branch made to the appropriate routine.				
PP20	Check if separator cards are requested. If not, branch to..... >	PP30			
	Check the device type:				
	If 2560, branch to..... >	PP26			
	If 1442N2, branch to..... >	PP24			
	If 5425, branch to..... >	PP28			
	Otherwise, it is 3525, 2520, or 2540. Turn off bits 0 and 1 of the command code.		TCCC(IPW$DTC)		
PP24	Turn off bit 1 of the command code.		TCCC(IPW$DTC)		
PP26	Check whether it is a load print buffer. If so, branch to..... >	PP30			

Labels	Chart JB04: IPW$$PP - Physical Punch		Modified Data Fields	Reg. Usage	Calls
PP28	Turn off bit 0 of the command code.		TCCC(IPW$DTC)		
	Check for stacker select command. If not, branch to..... >	PP30			
	Turn off bits 0, 1, and 2 of the command code.		TCCC(IPW$DTC)		
	Initialize CCWs and Buffer				
PP30	This routine calculates the remaining data buffer space and checks if it is enough to contain the data record and two accompanying CCWs. If this is true, record and CCW are moved into the PDA and an IPW$GLR call is issued to logical writer to get the next data record. The CCW string is started right behind the CCB at the beginning of the data buffer space, and the data records are chained backwards in the PDA, starting at the end of the PDA. This procedure is followed to optimize the use of PDA space, since the variable length punch records prevent any calculation of a data record start address if data records are chained forward normally in the PDA. If insufficient PDA space is available, the PDA is emptied first by writing the data records to the punch.				
	Register 1 is loaded with the address of the next CCW to be built.			R1	
	Register 2 is loaded with the address of the end of the available buffer space.			R2	
	The remaining buffer space is calculated by subtracting register 1 from register 2.			R2	
	If remaining space is zero or more, a branch is made to attempt moving of CCW and data record..... >	PP34			
	Otherwise, a link is made to empty the PDA..... >	PP60		R14	
	On return, a branch is made to build CCW and move data record to the PDA..... >	PP38			
PP34	If the record to be moved fits in the remaining PDA space a branch is made to build CCW and move data record to the PDA..... >	PP38			
	Otherwise, a link is made to empty the PDA..... >	PP60		R14	

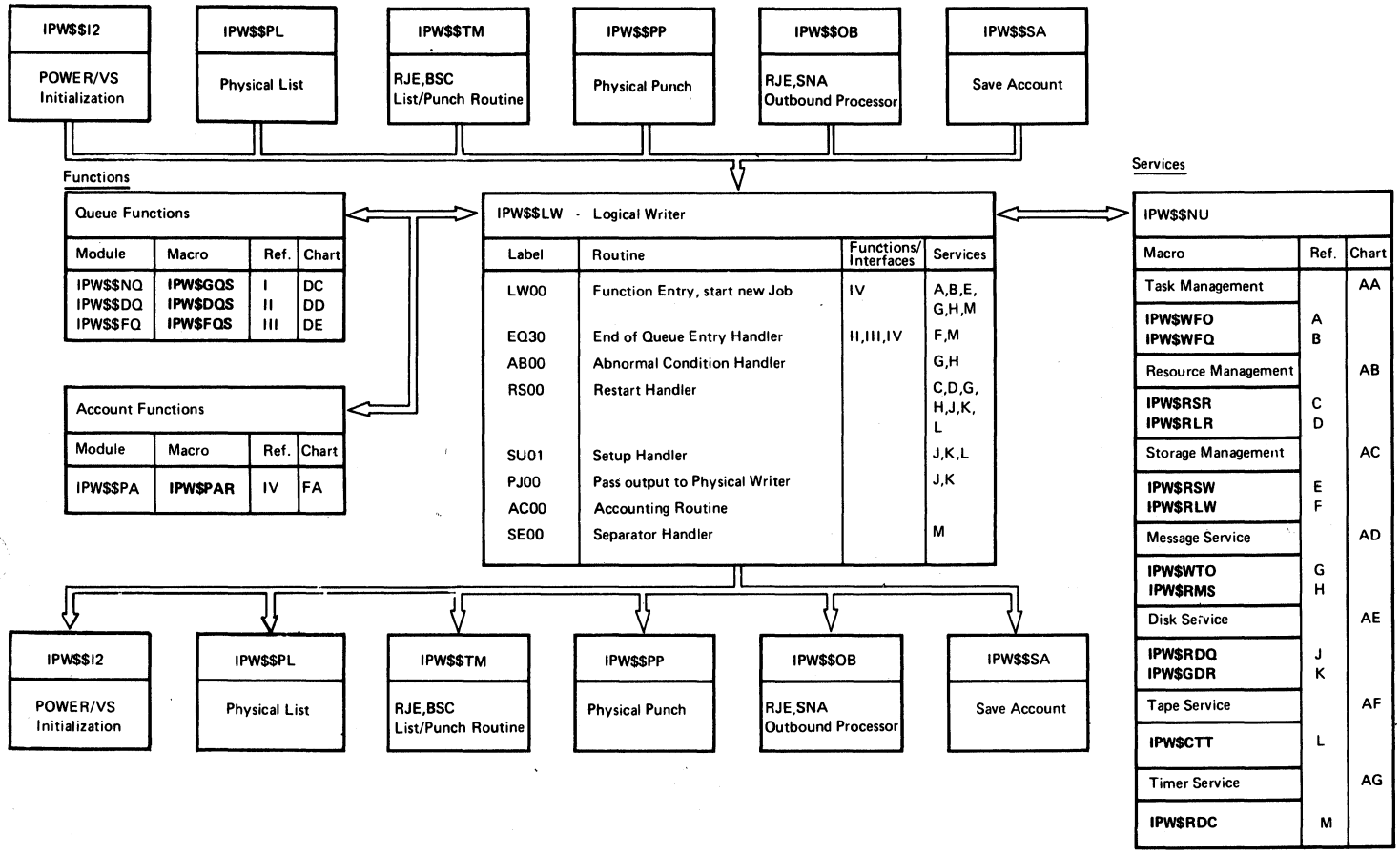
Labels	Chart JB05: IPW$$PP - Physical Punch		Modified Data Fields	Reg. Usage	Calls
PP38	The command code, contained in the high order byte of register 3, is stored in the new CCW.		CWCC(IPW$DCW)		
	The real data address, now pointing to the end of the available data buffer space, is loaded in register 2, the record length contained in register 4 is subtracted, and the updated real data address is stored back into the CCW.		CWDA(IPW$DCW)	R2	
	The CCW flags are initialized to indicate command chaining and suppress incorrect length indication.		CWFL(IPW$DCW)		
	The general purpose byte is saved in the CCW.		CWRE(IPW$DCW)		
	Register 6 is set to point to the next CCW, which is now made the NOP CCW terminating the CCW string, while its data address is made to point at the end of the available data buffer space after moving the current data record.			R6	
	Register 5 is made to point to the target address of the current data record in the PDA, by subtracting the record length, as contained in register 4, from the address of the end of available buffer space.			R5	
	If the record length is not more than 256 bytes, a branch is made to move the record with a normal MVC instruction, subject of an EXECUTE instruction..... >	PP44			
	Otherwise, registers 2, 1, and 0 are set up as from-address, to-length and to-address, respectively, to be used as the operands of the MVCL instruction that moves the data record.			R2,R1 R0	
	After moving the record to the PDA, a branch is made to obtain the next record from the logical writer..... >	PP46			
PP44	Register 4 is decremented by 1 to contain the record length in machine format and the record is moved using an EXECUTE instruction.			R4	
PP46	A test is made to see whether the PDA has to be emptied. If not, branch to get the next record..... >	PP48			
	Otherwise, the switch in the TCB is reset to X'00', and a link is made to empty the block..... >	PP60	PPEB(IPW$DTC)		

Labels	Chart JB06: IPW$$PP - Physical Punch		Modified Data Fields	Reg. Usage	Calls
PP48	The next data record is requested from the logical writer through an IPW$GLR call.				IPW$GLR Chart HB
	On return, the record length is saved again in register 4.			R4	
	The address of the record passed by logical writer is loaded in register 3. If a zero address was passed, indicating that no record available,			R3	
	a link is made to punch the buffer..... >	PP60		R14	
	On return, work space is released and return to logical writer. Otherwise, a branch is made to process the new record..... >	PP04			
	Punch Routine				
	This routine handles the actual punching of punch records.				
PP60	On entry, register 5, supposed to point to the end of the free buffer space, is compared to the virtual end of data space address in the PWS.				
	If equal, indicating an empty block, immediate return is made to the caller.			R14	
	Otherwise, register 1 is loaded with the virtual CCB address as stored in the PWS, tc serve as the parameter register for the EXCP (SVC 0), which is now issued.			R1	
	On return, an IPW$WFC call is issued to have the punch task wait for I/O completion.				IPW$WFC Chart AA
	Test for unrecoverable I/O error. If so, branch..... >	PP80			
	If not ignored errors, branch to... >	PP64			
	Otherwise, branch to restart from the broken CCW and restart I/O..... >	PP60			

Labels	Chart JB06.1: IPW$$PP - Physical Punch		Modified Data Fields	Reg. Usage	Calls
PP64	The CCW start address in the CCB is now reset using the address saved in the PWS, and register 6 is set to point to the first CCW again.		CBCA(IPW$DCB)	R6	
	The data address in the first CCW is reset to the real address of the end of the data buffer space.		CWDA(IPW$DCW)		
	Register 5 is reset to the virtual address of the end of the data buffer space.			R5	
	A branch is now made back to the caller.			R14	
PP80	Issue message 1Q61D and wait for reply. If reply is 'C', branch to..... > If reply is 'I', branch to..... > If reply is not 'R', branch from... >	 PP88 PP62 PP80	PWRA, PWML (IPW$DPW)		IPW$WTR Chart AD
	Get virtual address of failing CCW.			R5,R6	
PP82	Count data transfers not yet executed in CCW string.				
PP84	If not NOP-CCW bump CCW pointer, then branch to..... >	PP82			
	If end of data condition, then branch to..... >	PP86			
	If no card movement, then branch to >	PP86			
	Add one to counter.				
PP86	If not 2540 device, then branch to. >	PP87			
	If punch check add one to count, otherwise branch to..... >	PP87			
PP87	Set up for restart the number of cards not punched. Branch to..... >	PP64			
PP88	Set termination byte to 'S' (Stop). Branch to termination routine.				IPW$$TR Chart LC

CHART JC: IPW$$LW - LOGICAL WRITER (24 PARTS)

Chart JC00: IPW$$LW - Logical Writer, General Flow and Macro Calls



Labels	Chart JC01: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
LWSD	The first 16 bytes constitute the section descriptor:				
	'LWCS V10M1 '				
	On entry, the following register contents are relevant:				
	10: Permanent area address		IPW$DPA	R10	
	11: Address of writer TCB		IPW$DTC	R11	
	13: Address of writer task save area.		IPW$TSV	R13	
	Entry is made at this point whenever any writer task (physical list or physical punch) issues its first IPW$GLR call for a data record.				
	<u>Start New Job</u>				
LW00	Register 9 is loaded with the routine origin address to serve as the base register.			R9	
NJ00	The job boundary switch in the TCB is set.		TCJB(IPW$DTC)		
	The first time switch is set.		LWFT		
	To start processing a new job's output, a new queue record is obtained.				IPW$GQS Chart DC
	If a STOP condition is detected (implying end of tape with tape spooling), branch to detach........ >	EQ72			
	Register 5 is loaded with the queue space address from the TCB.			R5	
	If register 5 is non zero, indicating that a queue record has been obtained, a branch is made to continue......................... >	NJ30		R5	
	If the current task is either an RJE task or an active spool management (GETSPOOL) request then a branch is made to detach the task.............................. >	EQ85			
	If message 1Q34I has already been issued, branch to.................. >	NJ05			
	Otherwise, (logical writer, no queue record available) message 1Q34I is issued.				IPW$WTO Chart AD
	The selection field is set up to wait for queue record and a direct link is made to task management............ >	TM00	TCSF(IPW$DTC)		IPW$WFQ Chart AA
	On return a branch is made to retry retrieving a queue record......... >	NJ00			

Labels	Chart JC02: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
NJ30	LDA (Logical Data Area) space is reserved, and the LDA is obtained from the permanent area.				IPW$RSW Chart AC
	The real address of the LDA is stored in the TCB.		TCDA(IPW$DTC)		
	The virtual address of the LDA is used to initialize the previous record pointer in the TCB.		TCPR(IPW$DTC)		
	If tape spooling is active, a branch is made to bypass disk data address setting..... >	NJ35			
	The data address is moved from the queue record to the TCB.		TCDW(IPW$DTC)		
	If single buffering, branch to..... >	NJ35			
	The seek address of the first data block is moved from the queue record to the disk request word for the second data file buffer in the TCB.		TC2DW(IPW$DTC)		
	Storage for the second LDA (logical data area) is reserved.				IPW$RSW Chart: AC
	The virtual and real addresses of the LDA are stored in the appropriate disk request word.		TC2DA(IPW$DTC) TC2DV(IPW$DTC)		
NJ35	24 bytes of work space are reserved for accounting handling.				IPW$RSW Chart AC
	Register 4 is set up to address account work space.			R4	
	The work space address is stored in the TCB.		LWAW(IPW$DTC)		
	The task start time is saved for accounting.		LAST(LADS)		IPW$RDC Chart AG
	The number of copies is moved from the queue record to the TCB.		LWNC(IPW$DTC)		
	The actual copy count is saved for accounting.		LAWS(IPW$DTC)		
	The output device type as specified in the queue record is compared with the output device type for the previous job as saved in the TCB. If device types match, or if the current task is an RJE task, a branch is made to continue..... >	NJ38			
	If spool management is active (GETSPOOL) then branch to get data. >	PJ00			
	Otherwise, message 1Q41I is issued.				IPW$WTO Chart AB

Labels	Chart JC02.1: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
NJ38	Get first data record.				IPW$GDR Chart EB
	If this is not a list task..... >	NJ40			
	If this is not a FCB load..... >	NJ40			
	Otherwise load FCB..... >	PJ90			
	Get next data record.				IPW$GDR Chart EB
	If this is a 3800 printer, branch to..... >	NJ42			
	If RJE task, branch to..... >	NJ42			
	The form-number, flash identification, and paper thread request, as specified in the queue record, are compared with the appropriate values of the previous job. If one of them is different, branch to issue message..... >	NJ49			
	Otherwise, branch to..... >	NJ80			
NJ42	The forms ID as specified in the queue record is compared with the forms ID of the previous job, as saved in the TCB. If both IDs do not match, branch to issue message..... >	NJ44			
	If it is not a punch task, branch to>	NJ80			
	If the PAUSE option was not specified, branch to..... >	NJ80			
NJ44	Otherwise, if the current task is not an RJE task, a branch is made to issue forms change message..... >	NJ50			
	If RJE-BSC task, then branch to queue message..... >	NJ46	TCCU(IPW$DTC)		
	Establish addressability in register 1 for the SNA logical unit control block (LUCB).		TCB1(IPW$DTC)	R1	
	If session allows PDIR record branch to avoid queuing message.... >	NJ48	LUPD(IPW$DLU) LUPS(IPW$DLU)		
	Cancel addressability.			R1	
NJ46	The address of message 1Q40A is loaded into register 1, and the remote ID into register 0.			R1,R0	

Labels	Chart JC03: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
	Issue message 1Q40A ON ttt FORMS ffff NEEDED FOR jobname, jobnumber.				IPW$RMS Chart AD
NJ48	A link is made to the physical routine..... >	PJ88			
	A branch is made to initialize the new forms ID..... >	NJ70			
	If a clear print is automatically issued at end of job time, branch to..... >	NJ4A		R2	
	Otherwise, save current record control word in the TCB extension area.		TE38GW (IPW$DTE)		
	The I/O command code in the TCB is set to 'CLEAR PRINT' and the length is set to one.		TCCC(IPW$DTC) TCRL(IPW$DTC)		
	A link and branch is made to the physical routine..... >	PJ90		R8	
	Restore actual record control word.		TCRW(IPW$DTC)		
NJ4A	Issue message '1QA5A CUU SETUP REQ.'				IPW$WTO Chart: AB
	Branch to wait for operator response..... >	NJ54			
NJ50	Otherwise, message 1Q40A is issued:				
	Message address is obtained and stored in the message request word.		TCMW(IPW$DTC)		
	Message is logged.				IPW$WTO Chart AB
NJ54	An IPW$WFO call is issued to wait for the operator command in response to message 1Q40A.				IPW$WFO Chart AA
NJ70	If FLUSH was entered, branch to.... >	NJ80			
	If FLUSH HOLD was entered..... >	NJ80			
	The new forms ID in the TCB is set according to the queue record.		LWFI(IPW$DTC)		
	If RJE task, branch to..... >	NJ72			
	The new flash identifier and the paper thread request are saved in the TCB according to the queue record.		LWFH(IPW$DTC) LWPS(IPW$DTC)		
NJ72	Set the copy group index in the 3800 TCB extension area if present.		TE38CG1 (IPW$DTE)	R2	
NJ80	A check is made if a PSTOP restart has been done. If not, a branch is made to obtain data..... >	PJ10			

Labels	Chart JC03.1: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
	If PSETUP is given..... >	NJ85			
	Indicate restart, get numbers of copies left and get restart page/ card.		LWNC(IPW$DTC) TQRS(IPW$DTC)		
	If copies left equal zero..... > otherwise set up copies left.	NJ85	LWNC(IPW$DTC)		
NJ85	Indicate for accounting.		LASR(LADS)		
	Copy count for accounting.		LAWS(LADS)		
	If it is not a 5425..... > otherwise do a secondary feed. Insert count.	PJ10	TCCC(IPW$DTC) TCGP(IPW$DTC)		
	A branch is made to process restart >	PJ10			
	End of Queue Entry				
EQ30	If this is not a 3800 printer, branch to..... >	EQ34			
	Issue end of transmission..... >	PJ90	TCCC(IPW$DTC)	R8	
	Check the option byte in the master queue record if a clear print is wanted at end of job time. If so, issue clear print..... >	PJ90	TCCC(IPW$DTC)	R2 R8	
EQ34	A link is made to the physical routine..... >	PJ95			
	If restart is not requested, continue at..... >	EQ35			
	Otherwise: Set remaining copy counter to 1, indicate restart at EOJ, and continue with Restart Handler..... >	RS00	LWNC(IPW$DTC) LWEJ(IPW$DTC)		
EQ35	Get address of SYSCOM.			R2	ASYSCOM
	(VM Hand shaking change - CP close)				
	Test if VM=YES option has been specified for supervisor (bit in SYSCOM). If not, continue with normal processing..... >	EQ35A		R2	
	Branch and link to routine to close file when under VM..... >	EQVM		R8	
EQ35A	Release logical data area. Indicate job boundary.		TCJB(IPW$DTC)		IPW$RLW
	If a second logical data area exists, it is released too.				IPW$RLW Chart AC

Labels	Chart JC04: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
	The buffer pointers in the TCB (real and virtual LDA addresses) are set to zero.		TCDA(IPW$DTC)		
	A possible restart indication is set to zero.		QRRR(IPW$DQC)		
	If not PFLUSH branch to..... >	EQ36			
	If QRDP=K branch to..... >	EQ37			
EQ36	If no flush hold condition is present, branch to..... >	EQ40			
EQ37	If tape, branch to..... >	EQ39			
	Store number of remaining copies in the queue record.		QRCR(IPW$DQR)	R1	
	Indicate restart from the beginning of the job.		QRRR(IPW$DQR)		
	Save the current copy group index in the queue record if a 3800 TCB extension area is present.		QRCI(IPW$DQR)	R1	
EQ39	The queue record is set in HOLD state.		QRDI(IPW$DQR)		
	Set termination to flush.		TCTT(IPW$DTC)		

Labels	Chart JC04.1: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
EQ40	The queue record is deleted.				IPW$DQS Chart DD
	If no flush condition is present, branch to..... >	IA00			
	Otherwise, reset flush condition and set cancel code in QR to X'40'.		TCTT(IPW$DTC) QRCN(IPW$DQR)		
	Initialization of the account record is now entered.				
IA00	If accounting is not supported, branch to skip the accounting routine..... >	EQ58			
	If the current task is a punch task, a branch is made to bypass output page counting..... >	IA20			
	Otherwise, register 1 is loaded with the total page count from the account work space.			R1	
	The number of pages as kept in the queue record is subtracted.			R1	
	If the result is positive, a branch is made to continue page count initialization..... >	IA10			
	Otherwise, the extra page count in the queue record is set according to the extra page count in the account work space.		QRNE(IPW$DQR)		
	The total page count in the queue record is also set equal to the corresponding value in the account work space.		QRNP(IPW$DQR)		
	A branch is made to bypass page counting..... >	IA20			
IA10	The extra page count in the account work space is added to the page count in register 1.			R1	
	The result is stored back in the queue record.		QRNE(IPW$DQR)		

Labels	Chart JC05: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
IA20	The number of lines or cards in the queue record (QRLC) is subtracted from register 7 contents to determine the number of extra lines or cards.			R7	
	If the result is positive, a branch is made to continue..... >	IA30			
	Otherwise, the line or card count in register 7 is reset.			R7	
	The extra line or card count in the queue record is copied from the account work space.		QRNA(IPW$DQR)		
	The line count in register 7 is stored into the queue record, and a branch is made..... >	IA40	QRNR(IPW$DQR)		
IA30	The extra line or card count calculation is now completed by adding the extra line or card count in the account work space to register 7 contents.			R7	
	The result is stored in the queue record.		QRNA(IPW$DQR)		
	The line or card count in the queue record is copied from the QRLC field.		QRNR(IPW$DQR)		
IA40	The number of copies is loaded from LAWS in register 1 from the queue record.			R1	
	Register 0 is loaded with the number of copies remaining as kept in field LWNC in the TCB.			R0	
	The number of copies completed is calculated in register 1, and stored in the queue record.		QRNC(IPW$DQR)	R1	
	The work field LAWS in accounting record is cleared.		LAWS(IPW$DTC)		
	Line or card count register 7 is set to zero.			R7	
	The current date is stored in the account part of the queue record.		QRDY(IPW$DQP)		
	EOJ time is obtained, and stored in the queue record.		QRET(IPW$DQR)		IPW$RDC Chart AG

Labels	Chart JC06: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
	The saved start time is obtained and stored in the queue record.		QRST(IPW$DQR)		
	Register 1 is loaded with the account record address from the TCB.			R1	
	Register 0 is loaded with the account record length (72 bytes for list account record, 68 for punch).			R0	
	If a PFLUSH command was issued a branch is made to..... >	IA55			
	Otherwise if a PFLUSH command with hold was not issued a branch is made to..... >	IA56			
IA55	Set cancel code		QRNC(IPW$DQR)		
IA56	If it is a punch task a branch is made to..... >	EQ57			
	Otherwise get length of account record.			R0	
EQ57	The account record is written to the account file.				IPW$PAR Chart FA
EQ58	The queue record is added to the free set.				1PW$$FQS Chart DE
	Complete EOJ is indicated in the TCB.		TCFT(IPW$DTC)		
	The account work space is released.		LWAW(IPW$DTC)		IPW$RLW Chart AC
	Task conditions are now checked and branches are taken:				
	If stop at EOJ condition, branch to process detach..... >	EQ70			
	If non-RJE task, branch to get next queue record..... >	EQ60			
	If RJE,BSC or active spool management (GETSPOOL) task, branch to detach task..... >	EQ80			

Labels	Chart JC06.1: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
EQ60	If normal condition or RJE,SNA task, branch to get next queue record.... >	NJ00			
EQ70	If the current task is an initialization task, branch to return to physical routine..... >	EQ80			
	(VM Handshaking change - CP Close)				
EQ72	Test if VM=YES option has been specified for supervisor (bit in SYSCOM). If not, continue with normal processing..... >	EQ72A		R2	
	Branch and link to routine to close file when under VM..... >	EQVM		R8	
EQ72A	Register 9 is set up as task terminator base register, and a branch is made to the task terminator.			R9 R9	
EQ80	The queue record space is now released:				
	Queue record address is loaded in register 1.			R1	
	If nothing to release, branch to... >	EQ85			
	Queue record space is released.				IPW$RLW Chart AC
	The queue record pointers in the TCB are set to zero.		TCQV(IPW$DTC)		

Labels	Chart JC07: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
EQ85	Prior to return to the physical writer, register 1 is set to zero to indicate a detach condition.			R1	
	A branch is made to return to the physical writer..... >	PJ95			
	(VM Handshaking change - CP Close)				
EQVM	Use IPW$RSW macro to obtain 32 byte area from Storage Manager for parameter list.			R1	IPW$RSW (SM00)
	Move jobname and device address from other control blocks to parameter list for SVC56.				
	Issue SVC56 (CPCLOSE macro) to close VM file for current task output.			R1	SVC56
	Use IPW$RLW macro to release parameter list storage.			R1	IPW$RLW (SM50)
	Return to caller..... >			R8	
	Abnormal Condition Handler				
AB00	Dependent on the nature of the abnormal condition, a branch is made to the appropriate routine:				
	If stop at EOJ condition, branch to continue handling output..... >	PJ17			
	If flush condition, branch to issue message 1Q39I..... >	AB10			
	If flush (hold) condition, branch to issue message 1Q39I..... >	AB10			
	If PSTOP, restart..... >	AB40			
	If it is RJE task, branch to..... >	EQ70			
	If PSTOP, branch to..... >	AB40			
	Otherwise, a detach condition is present, and a branch is made to detach the task..... >	EQ70			

Labels	Chart JC07.1: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
AB10	If active spool management (GETSPOOL) task, then branch to..... >	PJ10			
	This routine is entered on a flush condition. Message 1Q39I is issued.				
	Possible PSETUP/PRESTART indicator is reset.		TCRS(IPW$DTC)		
	Flush condition is set in queue record.		QRCN(IPW$DQR)		
	Job boundary is set in the TCB.		TCJB(IPW$DTC)		
	Register 1 is loaded with the message address.			R1	
	If the task is not an RJE task, a branch is made to issue the message..... >	AB20			
	Otherwise, register 0 is loaded with the remote ID, and an IPW$RMS call is issued.			R0	IPW$RMS Chart AD
	On return, a branch is made to complete abnormal condition handling..... >	AB30			
AB20	For a local writer task, the message address is stored in the message request word, and the message is logged.		TCMW(IPW$DTC)		IPW$WTO Chart AD

Labels	Chart JC08: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
	If the current task is a punch task, a branch is made to continue..... >	PJ17			
	Otherwise, message 1Q39I is printed:				
	Using register 1 as a work register, the message and the job name are moved into the buffer and the message address and length are stored in the record request word, the command code is set to 'select translate table 0', and a link is made to output routine..... >	PJ90	TCCC(IPW$DTC) TCRL(IPW$DTC)	R1	
	On return the command code is set to 'skip to channel one' and a link is made to the physical routine..... >	PJ90	TCCC(IPW$DTC)	R8	
	On return, the command code is set to 'write', and a branch is made back. >	PJ17	TCCC(IPW$DTC)		
AB40	If this is a punch task, branch to do dummy secondary feed..... >	AB45			
	If this is not a punch task and stop immediate (PSTOP)..... >	AB50			
	If it is not immediate skip..... >	PJ17			
	otherwise..... >	AB50			
AB45	Is it not a 5425..... > otherwise do secondary feed.	AB50			
AB50	Indicate EOD.			R0	
	Link to physical routine to empty buffer..... >	PJ95		R8	
	Branch to terminate task..... >	EQ72			

Labels	Chart JC08.1: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
	Restart Handler				
RS00	If it is not a RJE restart branch to>	RS01			
	Get seek address and branch to..... >	PJ00	TCDW(IPW$DTC)		
RS01	If restart is already active branch to..... >	RS04			
	Otherwise, the restart current page count is initialized from the current page count in the TCB.		LARC+2(LADS)		
	If the current task is a spool management (GETSPOOL) task, then branch to..... >	RS03			
RS03	If the current task is a list task, a branch is made to bypass setting of restart current card count..... >	RS04			
	Otherwise, the restart active indicator is		TCG2(IPW$DTC)		
	Set and the restart current card count is set equal to the current card count.		LARC(LADS)		
RS04	Restart value as kept in field TCRS in the TCB is loaded in register 8.			R8	
	The restart sign code, as kept in the high-order byte of field TCRS, is loaded in register 2 and used as branch index into branch table RS08.				

Labels	Chart JC09: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
RS08	Branches are made from this branch table: 0: Should not occur (branch to itself)..... >	RS08			
	4: No sign given, branch to process absolete restart value..... >	RS12			
	8: Plus sign given, branch to calculate absolute restart value from forward restart..... >	RS16			
	12: Negative sign given, branch to calculate absolute restart value from backward restart..... >	RS20			
	16: Branch to process setup command..... >	SU01		R16	
	20: Branch to process start after PSTOP CUU, RESTART..... >	RS12			
RS12	A branch is made to restart printing from the point specified as restart value..... >	RS24			
RS16	The restart value in register 8 is added to the restart current count.			R8	
	A branch is made to restart printing from the calculated restart value..... >	RS24			
RS20	Negative sign; register 2 is loaded with the restart current count.			R2	
	Restart value in register 8 is subtracted.			R2	
	Calculated restart value is copied to register 8.			R8	
RS24	If the restart value in register 8 is not negative, a branch is made to process restart..... >	RS28			
	Otherwise, the restart value is set to zero first.			R8	
RS28	First, restart indication in the TCB is set to zero.		TCRS(IPW$DTC)		
	The restart current count is reset to zero.		LARC(LADS)		
	The restart active indicator is set.		TCG2(IPW$DTC)		
	The DMB is reserved for subsequent reading of the first queue record in set.				IPW$RSR Chart AB
	Register 6 is set up to address the reserved DMB.			R6	
	If the current queue record is not the first in set, a branch is made to read the first-in-set queue record, required to compute restart point.. >	RS32			

Labels	Chart JC10: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
	Otherwise, the queue record image is copied to the auxiliary area in the QCB, and a branch is made to bypass reading the first queue record..... >	RS36	QCQR(IPW$DQC)		
RS32	The disk address of the 'first-in-set' queue record is moved to the DMB		QCQW(IPW$DQC)		
	The first-in-set queue record is read in.				IPW$RDQ Chart AE
RS36	If the current task is a list task and not a spool management (GETSPOOL) task, a branch is made to continue. >	RS40			
	Otherwise, the restart card value in register 8 is compared to the maximum value and a branch is made around the compare for a list task..... >	RS44			
RS40	The restart page value in register 8 is compared to the maximum value for page restart.				
RS44	If higher, a branch is made to issue message 1Q42I..... >	RS48			
	Otherwise, the first queue record image is copied back to the queue record space from the DMB.		IPW$DQR		
	The DMB is released again.				IPW$RLR Chart AB
	A branch is made to continue..... >	RS56			
RS48	The DMB is first released.				IPW$RLR

Labels	Chart JC10.1: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
					Chart AB
RS50	The address of message 1Q42I is loaded in register 1.			R1	
	If it is a spool management (GETSPOOL) task, branch to..... >	RS54			
	If the current task is not an RJE task, a branch is made to issue the message..... >	RS52			
	Otherwise, the remote ID is loaded in register 0, and an IPW$RMS is issued to send the message.			R0	IPW$RMS Chart AD
	A branch is made to continue output processing..... >	PJ10			
	If the current task is a spool management (GETSPOOL) task, then get the address of the SPM parameter list, indicate EOF in SPL parameter list, and branch to continue output processing..... >	EQ30			
RS52	The address of message 1Q42I in register 1 is stored in the message request word, and the message is logged.		TCMW(IPW$DTC)		IPW$WTO Chart AD
RS54	Reset RESTART current counter		LARC(LADS)		
	Reset restart active indicator.		TCG2(IPW$DTC)		
	If restart is not issued at EOJ time, branch to continue output processing>	PJ10			
	Otherwise, reset RESTART/EOJ,counter, and continue with EOJ processing... >	EQ35	LWEJ(IPW$DTC)		

Labels	Chart JC11: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
RS56	If the current queue record is the last-in-set, a branch is made to locate the restart card/page..... >	RS66			
	If this is a 3800 printer, branch.. > (Note: Scanning for the restart page has been done from the beginning, because multiple SETPRT requests may occur in the output job. The last encountered SETPRT is reissued before processing is resumed with the restart page.)	RS66			
RS60	The disk address of the next-in-set queue record is moved to the TCB.		TCQW(IPW$DTC)		
	The next-in-set queue record is obtained.				IPW$RDQ Chart AE
	If the current task is a list task and not a spool management (GETSPOOL) task, a branch is made to continue processing page restart..... >	RS64			
RS62	Otherwise, the restart card value in register 8 is compared with the number of cards associated with this queue record, as found in the field QRLC.				
	If the restart value turns out to be within the range of this queue record, a branch is made to restart..... >	RS66			
	Otherwise, the restart active indicator is set and the restart current card count is set equal to the card count of this queue record, and a branch is made back to get the next queue record..... >	RS60	TCG2(IPW$DTC) LARC(LADS)		
RS64	The restart page value in register 8 is compared with the page count for this queue record.				
	If the restart value is within the range for this queue record, a branch is made to restart..... >	RS66			
	Otherwise, the restart active indicator is set and the restart current count is set equal to the page count for this queue record, and a branch is made back to get the next queue record..... >	RS60	TCG2(IPW$DTC) LARC+2(LADS)		
RS66	If the restart current count is not zero, a branch is made to continue..... >	RS68			

Labels	Chart JC12: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
	If the current queue record is the first queue record, a branch is made..... >	RS68			
	Otherwise, the disk address of the first queue record is copied into the TCB, and the first queue record is read in.		TCQW(IPW$DTC)		IPW$RDQ Chart AE
RS68	An empty block is indicated to the IPW$PDR function.		TCPR(IPW$DTC)		
RS70	If tape spooling is not active, a branch is made to bypass backspace of tape..... >	RS73			
	If not restart forward branch to... >	RS72			
	If end of data reached branch to... >	RS50			
	If restart at end of job branch to. >	RS50			
	Branch to process forward restart.. >	RS74			
RS72	An empty block is indicated to the IPW$PDR function.		TCPR(IPW$DTC)		
	Reset restart active indicator.		TCG2(IPW$DTC)		
	Otherwise, the tape is backspaced to the beginning of the file: • A backspace file call is issued • A forward space file call to skip tapemark • And a forward space record to position past first record.				IPW$CTT IPW$CTT Chart AF IPW$CTT Chart AF
	A branch is made to continue..... >	RS74			
RS73	The data pointer is moved from the queue record to the TCB.		TCDW(IPW$DTC)		
	When double data file buffering, the seek address of the first data block is moved from the queue record to the disk request word for the second data file buffer in the TCB. (This is done to indicate: first time through now.)		TC2DW(IPW$DTC)		
RS74	Register 6 is loaded with the restart current count.			R6	
RS76	Restart value in register 8 is compared to restart current count in register 6.			R6	
	If equal, a branch is made to exit and start printing..... >	RS90			

Labels	Chart JC12.1: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
RS80	Otherwise, the next data record is obtained. If the current task is a list task and not a spool management (GETSPOOL) task, a branch is made to test for a 'skip' operation..... >	RS84			IPW$GDR Chart EB
	Otherwise, the general purpose byte is tested if the punch operation causes a feed. If not, a branch is made back to get the next data record..... >	RS80			
	Otherwise, a branch is made to increment the restart current count..... >	RS86			
RS84	If the associated command code is a 'skip to channel 1 immediate' (start of a new page), branch to..... >	RS86			
	If the associated command code is a X'FD' (SETPRT REQUEST), the SETPRT parameter list is saved in the TCB extension area.		SPLLIST (IPW$DTE)	R1,R14	
	Branch to get next data record..... >	RS80			
RS86	The restart current page card count is incremented by one. A branch is made back to test if restart is complete..... >	RS76		R6	

Labels	Chart JC13: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
RS90	This routine is entered when the repositioning of the data file, required to accommodate the restart command has been completed, and restart condition can be deactivated.				
	The restart current value in register 6 is saved in account work space.		LARC(LADS)		
	Set restart active indicator.		TCG2(IPW$DIC)		
	Reset RESTART/EOJ indicator.		LWEJ(IPW$DTC)		
	If the current task is a list task and not a spool management (GETSPOOL) task, a branch is made to handle page restart exit..... >	RS94			
	Otherwise, if backward restart has been requested, the current card count is not changed and a branch is made back to resume output handling..... >	PJ10			
	On forward restart, the restart current count is copied into the current count.		LACR(LADS)		
	The total card count in register 7 is incremented by one, and a branch is made to reset restart condition.... >	RS98		R7	
	If no 3800 TCB extension area exists, branch to..... >	RS96		R2	
	If no setup is required, branch to. >	RS96			
	Save actual record control word.		TE38GW (IPW$DTE)		
	The address of the SETPRT parameter list and its length are stored in the record control word. The command code is set to X'FD'.		TCCC(IPW$DTC) TCRL(IPW$DTC)		
	A link is made to the physical routine..... >	PJ90		R3,R8	
	The actual record control word is restored.		TCRW(IPW$DTC)		
RS96	If restart is not forwarded branch to..... >	RS99			
	Otherwise, the restart current page count is copied into the current page count, and the total page count is incremented by one, using register 1 as a work register.		LACP(LADS) LATP(LADS)	R1	

Labels	Chart JC13.1: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
RS98	The restart condition is reset by setting the restart current counter to zero.		LARC(LADS)		
	Reset restart active indicator.		TCG2(IPW$DTC)		
	Exit is made..... >	PJ20			
RS99	If restart is not from zero a branch is made back to..... >	PJ10			
	Otherwise a get data record for the first record will be done after a skip to channel 1 is passed to the physical routine.				
	If not a FCB load record branch to. >	PJ10			
	Page Set-up Handler				
SU01	First, the I/O command code is checked if a buffer load or load UCS is requested.				
	If this is the case, a branch is made to execute this buffer load first.. >	PJ20			
	If this is not a SETPRT request, branch to..... >	SU05			
	Otherwise, clear out the copy group values in the just-obtained SETPRT request and execute the setup first..... >	PJ20	SPLCOPYG (IPW$DTE)	R2	
SU05	If the command code is not a 'skip immediate' (new page), a branch is made to translate nonblanks to C'X'..... >	SU10			
	Otherwise, the setup remaining page count in register 8 is decremented by one, stored in the TCB, and, if less than zero, a branch is made to exit setup handling..... >	SU40		R8	

Labels	Chart JC14: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
	Using register 2 as a work register, the total page number LAEP is incremented by one.		LAEP(LADS)	R2	
SU10	If the current list I/O command code will not cause a line to be printed (space), a branch is made to execute the I/O..... >	PJ20			
	Otherwise, the extra line counter in account work space is incremented by one, using register 1 as a work register.		LAER(LADS)	R1	
	The data record address is loaded in register 2 from the TCB, and its length in register 1.			R2 R1	
	If length is zero, a branch is made back..... >	PJ20			
	Otherwise, register 2 is set up to point to the last byte of the data record by adding the record length and subtracting one.			R2	
	In tne following loop, the line to be written is scanned backwards, starting with the last character. If the character scanned, pointed to by register 2, is non-blank, it is overwritten with X.				
SU20	If the current character is a blank, a branch is made to bypass overwrite..... >	SU30			
	Otherwise, it is replaced by X.				
SU30	If the line scan has not yet been completed, a branch is made back to check the previous character..... >	SU20		R2	
	Otherwise, a branch is made to pass the record to the physical writer..... >	PJ20			
SU40	Exit from setup handling. Restart current page count is reset.				
	If the current queue record is the first in set, a branch is made to bypass obtaining the first in set, required for restoration of page pointers..... >	SU50			
	Otherwise, the disk address of the first in set is moved to the TCB.		TCQW(IPW$DTC)		
	The first-in-set queue record is obtained.				IPW$RDQ Chart AE

Labels	Chart JC15: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
SU50	A link is made to the physical routine to empty the buffer..... >	PJ95			
	An empty block is indicated in the TCB.		TCPR(IPW$DTC)		
	If tape spooling is not active, a branch is made to bypass tape control..... >	SU60			
	Otherwise, the spooling tape is backspaced to the beginning of the file.				
	Backspace file call is issued.				IPW$CTT Chart AF
	Forward space file to skip tapemark.				IPW$CTT Chart AF
	Forward space record to position past first record.				IPW$CTT Chart AF
SU60	Restart/setup indicator is reset.		TCRS(IPW$DTC)		
	Restart current counter is reset.		LARC(LADS)		
	Reset restart active indicator.		TCG2(IPW$DTC)		
	If current task is a spool management (GETSPOOL) task, branch to..... >	SU62			
	Check for RJE. If not, branch to.. >	SU65			
SU62	Otherwise, branch to the physical routine..... >	PJ88			
	Upon return from the physical routine, branch to..... >	SU70			
SU65	An IPW$WFO macro is issued to wait for the next operator command.				IPW$WFO Chart AA
	If tape spooling, branch to..... >	SU75			
SU70	Otherwise the data pointer of the first-in-set queue record is copied to the TCB.		TCDW(IPW$DTC)		
	When double data file buffering, the seek address of the first data block is moved from the queue record to the disk request word for the second data file buffer in the TCB. (This is done to indicate: first time through now.)		TC2DW(IPW$DTC)		
SU75	The corresponding data record is obtained.				IPW$GDR Chart AE
	If the associated command code is a buffer load, branch to ignore it... >	SU75			
	Check for PSETUP or RESTART. If so, branch to..... >	PJ10			

Labels	Chart JC15.1: IPW$$LW - Logical Writer		Modified Data Field	Reg. Usage	Calls
	Indicate restart. Get number of copies left. Get restart card/page.		TCRS(IPW$DTC) LWNC(IPW$DTC) TCRS+1(IPW$DTC)		
	Get copy group index to be used		TE38CGI (IPW$DTE)	R2	
	Exit is taken..... >	PJ10			
	Pass Job Output				
PJ00	An IPW$GDR call is issued to obtain the next record				IPW$GDR Chart EB
PJ10	Reset restart page count		QRRR(IPW$DQR)		
	If no normal condition is posted in the TCB, a branch is made to..... >	AB00			
PJ17	Is first time for this copy? If no, branch to..... >	PJ15			
	Set first time indicator off.		LWFT(IPW$DTC)		
	A link is made to the separator routine..... >	SE00		R3,R15	
	If separator pages/cards are written, start over again; branch to..... >	PJ00			

Labels	Chart JC16: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
PJ15	If a restart or setup command has been given, a branch is made to process restart or setup..... >	RS00			
	Otherwise, a link is made to handle accounting..... >	AC00		R14	
PJ20	A link is made to pass the data record to the physical routine..... >	PJ90		R8	
	If end of data is not indicated, a branch is made to process the next data record..... >	PJ00			
	Otherwise, if page setup was active, a branch is made to close page setup..... >	SU40			
	If restart to be handled, branch to..... >	RS00			
	The option byte in the master queue record is examined to determine if the user wants separator pages. If yes,				
	Set first time indicator on. If this is not the last copy, then branch..... >	PJ25	LWFT(IPW$DTC)		
	A link is made to print end separator pages..... >	SE00		R3,R15	
PJ25	The general purpose byte is reset.		TCGP(IPW$DTC)		
	If the current queue record is the first in set, a branch is made to skip the following read of the first in set..... >	PJ30			
	Otherwise, the first-in-set pointer is moved from queue record to TCB.		TCQW(IPW$DTC)		
	An IPW$RDW call is issued to read the first-in-set queue record.				IPW$RDQ Chart AE
	If present task is an active spool management (GETSPOOL) task, branch to end-of-job processing..... >	EQ30			
PJ30	If the job is in boundary state branch to end-of-job processing.... >	EQ30			
	The number of copies left to be written is loaded in register 0.			R0	
	If all copies have been processed a branch is made to handle end-of-job condition..... >	EQ30			
PJ40	Otherwise, processing of the remaining copies is prepared:				

Labels	Chart JC17: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
	If on job boundary, branch to handle EOJ..... >	EQ30			
	An empty data block is indicated.		TCPR(IPW$DTC)		
	The data pointer is moved from queue record to TCB to reset the data pointer in the TCB to point to the start of the current job output.		TCDW(IPW$DTC)		
	The updated remaining copies count (having been decremented by one by the BCT instruction controlling the output copy loop) is stored back in the TCB.		LWNC(IPW$DTC)		
	The current page counter in the account work space is set to zero.		LACP(LADS)		
	The current line/card counter is set to zero.		LACR(LADS)		
	When double data file buffering, the seek address of the first data block is moved from the queue record to the disk request word for the second data file buffer in the TCB. (This is done to indicate: first time through now.)		TC2DW(IPW$DTC)		
PJ42	If this is not a 3800 printer, branch to..... >	PJ44			
	Otherwise, address 3800 TCB extension area and increase copy group index by one.		TE38CGI (IPW$DTE)	R1	
	The I/O command code is set to 'end of transmission' and the length is set to one.		TCCC(IPW$DTC) TCRL(IPW$DTC)		
	A link is made to the physical routine..... >	PJ90		R8	
PJ44	A skip to channel 1 is set and a branch is made to..... >	PJ90	TCCC(IPW$DTC)		
	Get first data record				IPW$GDR
	If it is not an FCB record, branch to..... >	PJ10			
	Otherwise, skip FCB load and branch to get next data record..... >	PJ00			
	A branch is made to handle current job output..... >	PJ00			
PJ88	Indicate set up to RJE.		TCRW(IPW$DTC)		

Labels	Chart JC17.1: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
PJ90	Parameter register 0 is set up with the data record address, register 1 is loaded with the record length, and			R0 R1	
PJ95	control is passed to the physical writer.				IPW$PLR Chart JC
	Is request for spool management (GETSPOOL)?				
	If no, return to caller..... >			R8	
	Request for open logical writer interface?				
	If yes, reset indicator and branch to reopen logical writer..... >	NJ00	TCSS(IPW$DTC)		
	Is request for close logical writer?				
	If yes, reset indicator and branch to close logical writer..... >	PJ25	TCSS(IPW$DTC)		
	Return to caller..... >			R8	

Labels	Chart JC17.1: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
	Account Handler This routine handles updating of counters relevant to accounting such as: if restart is active - Extra card line count (LAER) Extra page count (LAEP) Restart current card count (LARC) Restart current page count (LARC+2) if restart is not active - Current line card count (LACR) Total lines/cards from data file (R7) Total pages from data file (LATP)				
AC00	Register 0 is loaded with the increment value 1. If the current task is a punch task or active spool management request, a branch is made to handle punch accounting..... >	AC20		R0	
	Otherwise, if a 'space' is indicated in the general purpose byte, a branch is made to handle line accounting.. >	AC30			

Labels	Chart JC18: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
	If the list command code is not 'skip immediate', control is returned to the caller.			R14	
	The restart current line/card count (LARC) is loaded in register 1.			R1	
	If restart is active, a branch is made to update extra page count.... >	AC10			
	Otherwise, total page count (LATP) is incremented by one, using register 1 as a work register.		LATP(LADS)	R1	
	Current page count is incremented by one, using register 1 as a work register.		LACP(LADS)	R1	
	Control is returned to the caller.			R14	
AC10	The restart current line/card count is incremented by one and stored back in account work space.		LARC(LADS)	R1	
	The extra page count is incremented by one, using register 1 as a work register.		LAEP(LADS)	R1	
	If the restart current page count has become equal to the current page count before restart, a branch is made to reset the restart condition..... >	AC50			
	Otherwise, control is passed to the caller.			R14	
AC20	If no indication is found in the general purpose byte that the current I/O operation will cause a feed on the punch, control will be returned to the caller immediately.			R14	
AC30	(This routine is common to line and card account handling.)				
	If restart is active, a branch is made to update the extra line/card count..... >	AC40			
	Otherwise, the total line/card count from data file as kept in register 7 is incremented by one.			R7	

Labels	Chart JC19: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
	The current line/card count is incremented by one using register 1 as a work register, and control is returned to the caller.		LACR(LADS)	R1 R14	
AC40	The extra line card count is incremented by one using register 1 as a work register. If the current task is a list task and not a spool management (GETSPOOL) task, control is returned to the caller.		LAER(LADS)	R1 R14	
	Otherwise, the restart current card count is incremented by one using register 1 as a work register, and if the restart current card count has not yet reached the current card count, and control is returned to the caller.		LARC(LADS)	R1 R14	
AC50	Otherwise, the restart condition is reset by setting the restart current line card count to zero, reset restart active indicator, and control is returned to the caller		LARC(LADS) TCG2(IPW$DTC)	R14	
	Separator Handler				
SE00	This routine prints a number of separator pages or punches a number of separator cards as specified in the field QRSP of the queue record.				
	If tape spooling is not active, branch to..... >	SE01			
	If end separator pages to be printed, branch to..... >	SET0			
	Otherwise, read in trailing queue record.			R6,R14 R15	IPW$DQS Chart:
	Branch to..... >	SE01			
SET0	Read in header queue record: • backspace record			R3,R6	IPW$CTT Chart: AF
SE01	Register 6 is loaded with the number of separator pages requested.			R6	
	If this number is zero or a spool management (GETSPOOL) task, control is passed to the caller.			R3	
	The logical data area is used as work space to build the separators.				
	The restart information is saved. The request word is saved.		LWTC(IPW$DTC) LWSR(IPW$DTC)		
	Branch to set up the printer if applicable..... >	SS01		R8	

Labels	Chart JC20: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
SE02	Clear out restart information.		TCRS(IPW$DTC)		
	If the current task is a punch task, branch to..... >	SCT0			
	The page size (depth) is obtained from the queue record. If zero, the system default value (SYSCOM) is taken.		QRER(IPW$DQR)	R1,R2	
SCT0	Set up the record address and the command code in the TCB.		TCRV(IPW$DTC) TCCC(IPW$DTC)		
	Check for end of job. If so, branch to punch the end separator cards... >	SCEJ			
	Check for a 5425. If so, branch to >	SCT5			
	Load a record length of 80 into register 1 and store it in the TCB.		TCRL(IPW$DTC)	R1	
	Fill the buffer with the attention character X'78'.		SCDS		
	Check the number of required separator cards in register 6, and if necessary, set it to a minimum of 3.			R6	
SCT1	Branch and link to punch the separator cards..... >	SCT8		R2	
	Load the buffer address into register 8.			R8	
	Save the job name.		SCNM(SCDS)		
	Load the character count into register 1, and the starting address into register 2.			R1,R2	
	Fill the buffer with separator card blanks (X'38').		SCDS		
SCT3	Using registers 3 and 6 as work registers, the job name is translated.			R3,R6	
	Set the number of separator cards to be punched to 1 and branch and link to punch the card..... >	SCT8		R6,R2	
	Branch to..... >	SE99		R3	
SCT5	Set up for punching the separator cards for the 5425, which are punched in all positions, and the job name printed 12 times on each card.			R1	
SCT7	Increment the card count by one.		LAER(LADS)	R1	
	Set the command code to punch primary (X'05') and branch and link to the physical routine..... >	PJ90	TCCC(IPW$DTC)	R8	
	Check for a stop condition. If so, branch to detach the task..... >	EQ72			

Labels	Chart JC20.1: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
	Set the command code to print and feed (X'41'), and branch and link to the physical routine..... >	PJ90	TCCC(IPW$DTC)	R8	
	When all separator cards have been punched, branch to..... >	SE99		R3	
SCEJ	Set up for punching two blank end separator cards and branch and link to the physical routine..... >	SCT8		R1,R6, R2	
	Branch to..... >	SE99		R3	
SCT8	Increment the end count by one.		LAER(LADS)	R1	
SCT9	Check for a stop condition. If so, branch to detach the task..... >	EQ72			
	Branch and link to the physical routine..... >	PJ90		R8	

Labels	Chart JC21: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
	When all separator cards have been punched, return to caller via link register 2.			R2	
	The buffer address is loaded in register 8, which serves buffer addressability.			R8	
SE05	If the last CCW issued was a skip to channel 1, branch to..... >	SE06			
	Perform skip to channel 1.	PJ90		R8	
SE06	Perform 3 "space 3" operations.	PJ90		R8	
	If task is in stop state, branch to >	EQ72			
	If restart to be done, branch to... > (start over again)	SE02			
	If page size is less than 40 lines, skip writing of block letters. Branch to..... >	SE0G			
	Translate jobname into table displacement.		LWSN(LWDS)		
SE07	Get centering information.				
SE0A	Set up request word.		TCRW(IPW$DTC)		
SE0B	Get start addresses.				
SE0C	Get layer of character until all characters are finished.				
	Print line..... >	PJ90		R8	
	Loop until all 12 lines printed.... >	SE0C			
	Perform "space 3"..... >	PJ90		R8	
	If through with 2nd line, branch to >	SE0F			
	Get Class and Priority and Number		LWJN(LWDS)		
	Translate to displacement.				
	Branch to..... >	SE0A			
SE0F	Add to extra records.		LAER	R1	
SE06	Add to extra pages.		LAEP	R1	
	If this is last page, branch..... >	SE85			
	Restore request word. The 120-byte buffer is filled with blanks.		TCRW(IPW$DTC)		
	19 asterisks are moved to beginning and end of the buffer.		LWS0(LWSP) LWS8(LWSP)		

Labels	Chart JC21.1: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
	If an EOJ condition exists, a branch is made around moving START to the separator line..... >	SE10			
	Otherwise, 'START' is moved to the separator line to indicate a start separator.		LWS1(LWSP)		
	A branch is made to continue separator line setup..... >	SE20			
SE10	END is moved to the separator line.		LWS1(LWSP)		
SE20	Job name is moved from queue record to separator line.		LWS2(LWSP)		
	Job number is moved to separator line:				
	Job number is loaded in register 1 from queue record, converted to packed decimal in work field, and unpacked to separator line.		LWCD(LWSP) LWS3(LWSP)	R1	
	Job suffix number is moved to separator line:				
SE22	Job suffix number is loaded in register 1 from queue record, converted to packed decimal in work field, and unpacked to separator line.		LWCD(LWSP) LWS4(LWSP)	R1	
SE23	User information is moved from queue record to separator line.		LWS5(LWSP)		
	The date is now moved to the separator line:				
	A COMRG macro is issued to obtain system date.				
	If system date is in mm/dd/yy format, a branch is made to handle date insertion in separator line..... >	SE25			
	Otherwise, the day of month is moved to the separator line.		LWS6(LWSP)		

Labels	Chart JC22: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
	The second number in system date, representing the month number, is packed into a work field, and a branch is made to complete date insertion..... >	SE26	LWCD(LWSP)		
SE25	The second number field of system date is now used as day of month and moved to the separator line.		LWS6(LWSP)		
	The first number field, representing the month number now, is packed into the work field.		LWCD(LWSP)		
SE26	The month number is loaded in register 2 in binary, multiplied by 4 to serve as index to address correct month name in table.			R2	
	LLMT, then loaded with the correct month name address.				
	Month name addressed by register 2 is moved to separator line.		LWS6(LWSP)		
	Year number (2 digits) is moved to separator line.		LWS6(LWSP)		
	The time is obtained using IPW$RDC call, stored in work field, and edited into separator line.		LWCD(LWSP) LWS7(LWSP)		IPW$RDC Chart AG
	Separator record information is stored in the TCB.		TCRV(IPW$DTC)		
	Separator line length (80) is loaded in register 1 and stored in the TCB.		TCGP(IPW$DTC)	R1	
SE40	The number of "space 3" to be performed is calculated.				

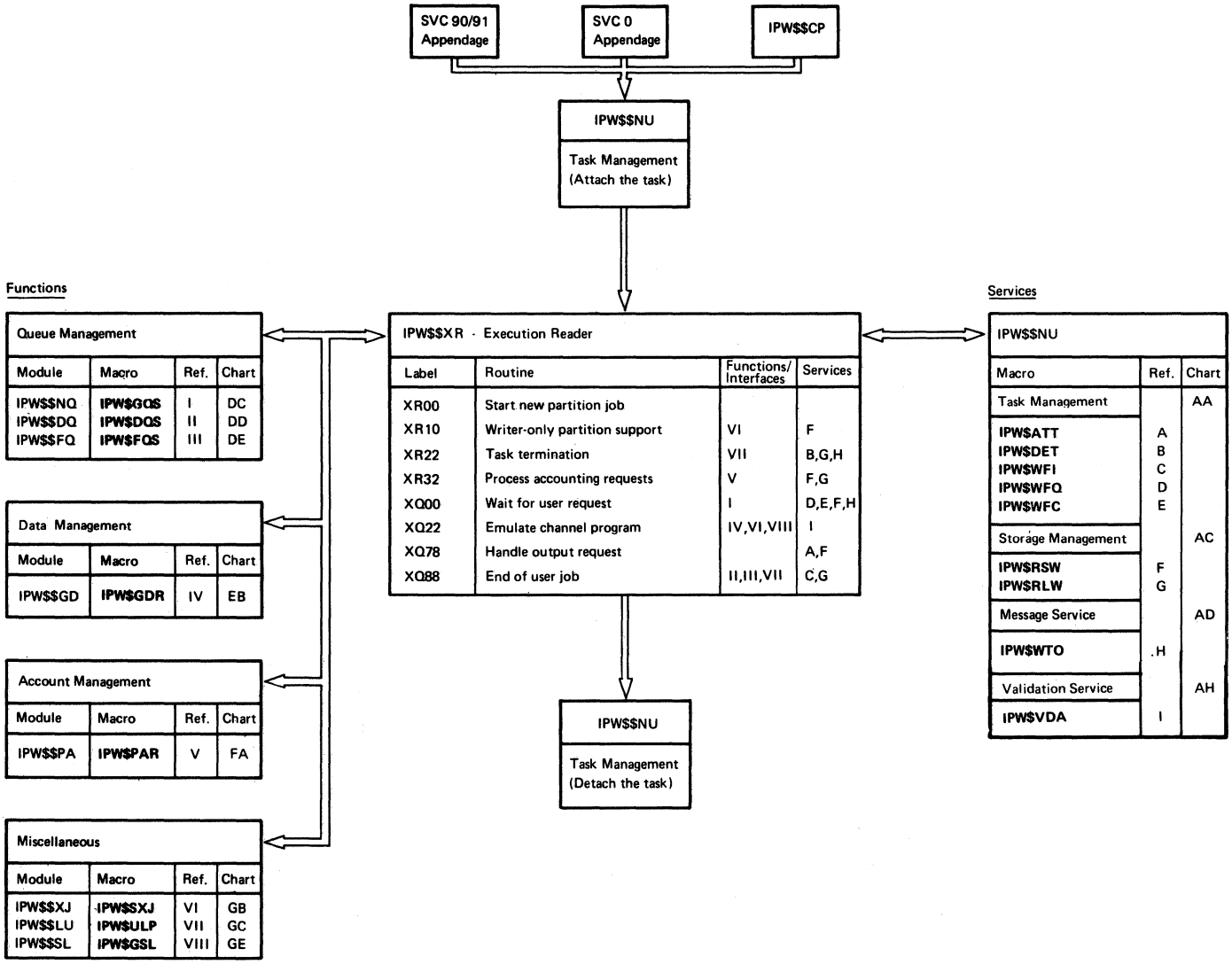
Labels	Chart JC23: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
SE60	The I/O command code in the TCB is set to 'space three lines'.		TCCC(IPW$DTC)		
SE70	Check for a stop condition. If so, branch to detach the task..... >	EQ72			
	A link is made to the physical routine..... >	PJ90			
	On return, branch back until all lines are spaced..... >	SE60			
	Set the I/O command code in the TCB to 'print line'.		TCCC(IPW$DTC)		
	Register 2 is loaded with 8 to signal 8 writes of separator lines.			R2	
SE80	Using register 1 as a work register, the line length is updated to 120.		TCGP(IPW$DTC)		
	The data pointer is restored.		TCRV(IPW$DTC)		
	The separator line is updated.		LWSP(LADS)		
	Check for a stop condition. If so, branch to detach the task..... >	EQ72			
	A link is made to the physical routine..... >	PJ90			
	On return, branch back until 8 lines are spaced..... >	SE80		R2	
	If not a 3800 printer, branch to... >	SE82			
	If end separators, branch to..... >	SE82			
	If mark form is wanted, branch to.. >	SE84			
SE82	If a separator page has been completed, a branch controlled by register 6 is made back to initiate printing of the next separator page..... >	SE05			
	Branch to complete last page..... >	SE05			
	The I/O command code in the TCB is set to 'mark form'. The length is set to one.		TCCC(IPW$DTC) TCRL(IPW$DTC)		
	A link is made to the physical routine..... >	PJ90		R8	
SE85	If START separators, branch..... >	SE99			
	The I/O command code in the TCB is set to 'skip to channel one'		TCCC(IPW$DTC)		
	Check for stop condition. If so, branch to detach task..... >	EQ72			

Labels	Chart JC23.1: IPW$$LM - Logical Writer		Modified Data Fields	Reg. Usage	Calls
	A link is made to the physical routine..... >	PJ90		R8	
	If restart to be done, branch to start over again..... >	SE02			
	Return to caller via register 3.				
SE99	Load buffer address in register 8.			R8	
	Restore restart information.		TCRS(IPW$DTC)		
	An empty block is indicated in the TCB.		TCPR(IPW$DTC)		
	If tape spooling, return to caller via register 3.				
	The DTA pointer of the first-in-set queue record is copied to the TCB.		TCDW(IPW$DTC)		
	When double data file buffering, the seek address of the first data block is moved from the queue record to the disk request word for the second data file buffer in the TCB. (This is done to indicate: first time through now.)		TC2DW(IPW$DTC)		
	Return to caller via register 3.			R3	
	Setup Printer Subroutine				
	The subroutine sets up the printer (non-impact) with the new or last-used printer setup but without copy group values, no flush, and no copy modification. The routine is only invoked by the separator-page handler in order to avoid multiple copies of each page when copy grouping is in use.				
SS01	If not a 3800 printer, return to caller..... >	R2		R2	
	If output was not destined for a 3800 printer, return to caller..... >	R2			
	If end separators, branch to..... >	SS10			
	If start separators and record is not a SETPRT record, return to caller.. >	R2		R2	
	Address SETPRT parameter list and branch to continue..... >	SS20		R8	
SS10	Use logical data area as a work area and copy the current SETPRT parameter list into it.				

Labels	Chart JC23.2: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
	Set the command code and the length in the TCB.		TCCC(IPW$DTC) TCRL(IPW$DTC)		
SS20	Set initialize printer flag. Clear out copy group values. Set no flushing. Set no copy modification.		SPPFLAG1 SPPCOPYG SPPFLASH SPPCPMOD		
	A link is made to the physical routine..... >	PJ90		R8	
	Return to caller..... >	R2		R2	

CHART KA: IPW$$XR - EXECUTION READER (16 PARTS)

Chart KA00: IPW$$XR - Execution Reader, General Flow and Macro Calls



Labels	Chart KA01: IPW$$XR - Execution Reader		Modified Data Fields	Reg. Usage	Calls
	This routine is entered when the PSTART command of the Command Processor attaches an Execution Read Task.				
XRCS	CSECT name				
XRSD	The first 16 bytes constitute the section descriptor: 'XRCS V7M0 '				
	Register usage: 0: **** - Service work register 1: **** - Service work register 2: **** - Service work register 3: **** - Service work register 4: IPW$DDE - Entry in PDB 5: IPW$DQR - Queue record space 6: IPW$DPD - PDB 7: IPW$DCB - User command control block 8: IPW$DCW - User channel command word 9: XRCS - Base register 10: IPW$DPA - POWER/VS nucleus 11: IPW$DTC - Task Control Block 12: **** - Reserved for nucleus use 13: IPW$DSV - Task save area 14: **** - Function linkage register 15: **CS - Function Base register			R0 R1 R2 R3 R4 R5 R6 R7 R8 R9 R10 R11 R12 R13 R14 R15	
	Start New Partition Job				
XR00	Save reader PUB address in packed form.		TCDE(IPW$DTC)	R2	
	Save device type.		TCDT(IPW$DTC)		
	Initialize the storage descriptor with the reader device address in hexadecimal form.		TCCU(IPW$DTC)		
	Set the job boundary switch to X'80' to indicate that no read request has been received yet.		TCJB(IPW$DTC)		
	Set the device type code to C'R' to indicate reader partition.		PDDT+1(IPW$DPD)		
	If the device is a normal console branch to..... >	XR02			
	If the device is not a display console branch to..... >	XR04			
XR02	Set the device type code to C'C' to indicate writer-only partition.		PDCL(IPW$DPD)		
XR04	Set up registers 0 and 4 for scanning the entries in the PDB.			R0,R4	

Labels	Chart KA02: IPW$$XR - Execution Reader		Modified Data Fields	Reg. Usage	Calls
XR06	Set the device type code to C'L' to assume a printer device.		TLCL(IPW$DTL)		
	If the device is not a printer, set the device type code to C'P' to indicate punch device.		TLCL(IPW$DTL)		
	If the device type code (PDCL) does not indicate a writer-only partition (C'C') branch to wait for a user request..... >	XQ00			
	Writer-only Partition Support				
	Initialization				
	Reserve queue space to form a dummy queue record to allow for the initialization of output queue records.				IPW$RSW Chart AB
	Save the real address of the queue space.		TCQA(IPW$DTC)	R0	
	Save the virtual address of the queue space.		TCQV(IPW$DTC)	R1	
XR10	Insert default values in the queue space:				
	• Move date of job		QRDY(IPW$DQR)		
	• Move default name		QRNM(IPW$DQR)		
	• Store job number		QRNO(IPW$DQR)		
	• Set default priority		QRPY(IPW$DQR)		
	• Set queue record identifier to C'C' to indicate console record		QRQI(IPW$DQR)		
	• Set class identifier to C'A'		QRCL(IPW$DQR)		
	• Set default output class		QRCL(IPW$DQR)		
	• Set forms identifier to blanks		QRFI(IPW$DQR)		
	• Set flash identifier to blanks		QRFL(IPW$DQR)		
	• Set disposition to C'D'		QRDP(IPW$DQR)		
	• Set number of copies to X'01'		QRNC(IPW$DQR)		
	• Set cuu address		QRCU(IPW$DQR)		
	• Set user information		QRUI(IPW$DQR)		
	• Set POWER/VS cancel code		QRCN(IPW$DQR)		
	Wait for User Request				
XR11	Unpost the event control block in the TCB.		TCEB+2(IPW$DTC)		
	Check the reader entry in the PDB for a user request.				
	If an accounting request was issued branch to..... >	XR32			
	If a read request was issued branch to..... >	XR18			
	Check termination switch for STOP or PEND posted. If so, and job is at job boundary, branch to..... >	XQ82			

Labels	Chart KA03: IPW$$XR - Execution Reader		Modified Data Fields	Reg. Usage	Calls
	Set up registers 0 and 4 for examining the remaining entries.			R0,R4	
XR13	Check the remaining entries in the PDB. If a request was issued branch to handle the output request..... >	XR42			
	Exit to task selection to wait for a user request.				IPW$WFC Chart AA
	On return from task selection branch to..... >	XR12			
	Writer-only Partition Handler				
XR18	Move the channel command word to the TCB.		TCRW(IPW$DTC)		
	Load the record address into register 1.			R1	
	Link to the JECL statement analysis routine to handle the incoming statement.				IPW$SXJ Chart GB
XR20	Store the updated CCW address in the CCW.		CBCS(IPW$DCB)		
	Set the residual count to zero.		CBCT(IPW$DCB)		
	Set the necessary flags in the first communication byte.		CBC1(IPW$DCB)		
	Calculate the user partition task PIB in register 7 and set it to dispatchable.			R7	
	Check for a job in progress, and if so branch to..... >	XR12			
	Branch to reinitialize the queue record..... >	XR10			
	Task Termination				
XR22	If this partition was started as "MT"-multi tasking, then set indication off and branch to..... >	XQ84	PDMT(IPW$DPD)		
	Load the address of the partition communication region into register 2.			R2	
	Reset the POWER/VS control flags in the communication region to zero.				
	The reader device and all unit record partition assignments which relate to the devices in the PDB are unassigned.				IPW$ULP Chart GC
	If a read request is still outstanding, the necessary flags are set in the CCB.		CBC1(IPW$DCB)		

Labels	Chart KA04: IPW$$XR - Execution Reader		Modified Data Fields	Reg. Usage	Calls
XR28	The user partition PIB is set to dispatchable.		PDPB(IPW$DPD)		
	If there is any queue space, it is returned to the storage pool.				IPW$RLW Chart AC
	Issue message 1Q33I to the operator.				IPW$WTO Chart AD
	Release the PDB.				IPW$RLW Chart AC
	Detach the Execution Read Task.				IPW$DET Chart AA
	• The task is removed from the task selection list. • The TCB storage is released.				
	Process Accounting Requests				
XR32	If there is no queue space available yet, the request is ignored..... >	XR38			
	The total length of the execution account record (EAR) is calculated in register 1.			R1	
	If the request was initiated by an SVC 91 branch to..... >	XR34			
	SVC 90 processing				
	Add user account size plus 8 bytes for the block descriptor to the size of the EAR.			R1	
	If the EAR size exceeds the maximum, ignore the request and branch to... >	XR38			
	Reserve space for the EAR.				IPW$RSW Chart AC
	Move the user account information to the EAR.				
	Branch to..... >	XR38			
	SVC 91 processing				
XR34	If an SVC 90 has been issued before the SVC 91 was issued branch to.... >	XR36			
	Calculate the size of the EAR in register 1			R1	
	Reserve space for the EAR.				IPW$RSW Chart AC
	Set up register 7 as a base register for the EAR.			R7	

Labels	Chart KA05: IPW$$XR - Execution Reader		Modified Data Fields	Reg. Usage	Calls
XR36	Fill in the execution account record:				
	• Move the date from the partition JA table.		AEDY(AEDS)		
	• Move the start time from the partition JA table.		AEST(AEDS)		
	• Move the stop time from the partition JA table.		AEET(AEDS)		
	• Move the user information from the queue record.		AEUI(AEDS)		
	• Move the job name from the queue record.		AENM(AEDS)		
	• Move the job number from the queue record.		AENO(AEDS)		
	• Set the record identifier to C'E'.		AERI(AEDS)		
	• Move the cancel code from the queue record.		AECN(AEDS)		
	• Move the 'from' terminal identifier from the queue record.		AEFJ(AEDS)		
	• Move the 'to' terminal identifier from the queue record.		AETJ(AEDS)		
	• Move the class identification from the queue record.		AECL(AEDS)		
	• Move the priority from the queue record.		AEPY(AEDS)		
	• Move the spool statistics from the partition JA table.		AE#L(AEDS)		
	• Move the length of the SIO table from the partition JA table.		AESL(AEDS)		
	• Move the DOS/VS account record from the partition JA table.		AEJN(AEDS)		
	Write the execution account record.				IPW$PAR Chart FA
	Release the work space for the EAR.				IPW$RLW Chart AC
	Reset the SVC 90 request in the TCB.		TCGW+8(IPW$DTC)		
XR38	Post the user event control block.		PDCB(IPW$DPD)		
	Set the user partition PIB to dispatchable.		PDPB		
	Reset the request entry in the PDB.		PDCB(IPW$DPS)		
	If this is a writer-only partition branch to..... >	XR12			
	Otherwise, branch to..... >	XQ00			
	Handle Output Request				
XR42	Check the job boundary switch to determine whether any input request has yet been received. If so, branch to create a new writer task to handle the output request..... >	XR44			

Labels	Chart KA06: IPW$$XR - Execution Reader		Modified Data Fields	Reg. Usage	Calls
	Since no input request has been received yet, there is no job identification for the output records. The output request is therefore ignored and the user command control block is posted as though the output operation were completed:				
	• Reset device entry in the TCB.		TLCB(IPW$DTC)		
	• Set the residual count to zero.		CBCT(IPW$DCB)		
	• Set channel and device end.		CBSD(IPW$DCB)		
	• Post the traffic bit.		CBC1(IPW$DCB)		
	Set the user partition PIB to dispatchable.				
	If this is a writer-only partition. >	XR12			
	Otherwise, branch to..... >	XQ00			
XR44	Reserve storage for the TCB to be associated with the new task.				IPW$RSW Chart AC
	Set up register 8 as a base register for the new TCB.			R8	
	Initialize the storage descriptor.		TNSD(IPW$DTC)		
	Move the device address in hexadecimal to the storage descriptor.		TNCU(IPW$DTC)		
	Reserve storage for the queue records to be generated by the task.				IPW$RSW Chart AC
	Set up register 1 as a base register for the new queue record.			R1	
	Initialize the new queue record:				
	• Copy values from the reader queue record.		QNBF(IPW$DQR)		
	• Set class identification to C'A'.		QNCL(IPW$DQR)		
	• Set forms identifier to blanks.		QNFI(IPW$DQR)		
	• Set flash identifier to blanks.		QNFL(IPW$DQR)		
	• Set disposition to C'D'.		QNDP(IPW$DQR)		
	• Set number of copies to X'01'.		QNNC(IPW$DQR)		
	• Set cuu address.		QNCU(IPW$DQR)		
	• Set device type.		QNDT(IPW$DQR)		
	• Set counts to zero.		QNNA(IPW$DQR)		
	• Set more counts to zero.		QNNR(IPW$DQR)		
	If the additional count value is zero, it is set to the maximum.		QNBN(IPW$DQR)		

Labels	Chart KA07: IPW$$XR - Execution Reader		Modified Data Fields	Reg. Usage	Calls
XR46	Set up register 2 as a base register for the disk management block.			R2	
	Set the queue record identifier to C'P' to indicate punch record.		QNQI(IPW$DQR)		
	Set punch values.		QNSP(IPW$DQR)		
	Check the device type in the PDB for printer device. If not, branch to >	XR47			
	Set the queue record identifier to C'L' to indicate list record.		QNQI(IPW$DQR)		
	Set list values.		QNSP(IPW$DQR)		
	Move the line table from the queue control block to the TCB.		TNGW(IPW$DTC)		
XR47	Set up the RJE identifiers if necessary.		QNTJ(IPW$DQR)		
XR50	Initialize the new task registers by storing them in the save area of the new TCB:				
	2: save area address 4: IPW$DTL address 6: IPW$DPD address			R2 R4 R6	
	Reserve storage for the TCB extension area of the new task.				IPW$RSW Chart: AC
	Initialize the 3800 printer setup portion of the extension area with system defaults: • Set initialize printer flag • Set forms identifier to blanks • Set DEBUG=NORM flag • Set flash identifier to blanks • Set number of copies to be flashed to 255 • Set TRC=no flag • Set first copy group value to one • Set copy group index to one • Set correct length of SETPRT parameter list. • Set default required flag		SPLFLAG1 SPLFORMS SPLFLAG2 SPLFLASH SPLFLSHC SPLFLAG1 SPLCOPYG SPLCINDX SPLLNGTH (IPW$DTE) TE38RQB		
	Note: The defaults are taken of the PUB2 area related to the device at a later time.				
XR58	Attach the new writer task into the system.				IPW$ATT Chart AA
	The owner address in the PDB is reset, so that control will be passed to the execution list task when a user request is received.		TLTC(IPW$DDE)	R1	
	Reset task entry.		PDCB(IPW$DPD)		

Labels	Chart KA07.1: IPW$$XR - Execution Reader		Modified Data Fields	Reg. Usage	Calls
	If this is a writer-only partition. >	XR12			
	Otherwise, branch to..... >	XQ00			
	Handle LST/PUN from SLI				
XR60	Save request word.			R2,R3	
	If length of record <71, branch to. >	XR62			
	Set length to 71.				
XR62	A call is made to IPW$$XJ to scan JECL record.				IPW$SXJ Chart GB
	If not a JECL record branch to..... >	XR64			
	Restore request word			R2,R3	
	Branch to get next record..... >	XQ55			

Labels	Chart KA07.1: IPW$$XR - Execution Reader		Modified Data Fields	Reg. Usage	Calls
XR64	Restore request word.			R2,R3	
	Branch to..... >	XQ62			
	Check for 3540				
XR70	If in the current queue entry the physical unit for 3540 data spooling is specified, an extra device list entry is initialized in the PDB. All requests to the specified 3540 will be trapped and spooled as long as the current queue entry is in effect.				
	Using register 3 as a base register, the PUB table is scanned for a matching entry.			R3	
	If no matching entry is found, branch to..... >	XR76			
XR72	Check if matching entry is a 3540. If so, branch to..... >	XR78			
XR76	Issue message 1Q88I INVALID 3540 UNIT				IPW$WTO Chart AD
	Set cancel code and branch to..... >	XQ06			
XR78	Initialize the 3540 spool entry:				
	• Update the first entry address		PDPA(IPW$DPD)		
	• Store the PUB address		TLPU(IPW$DDE)		
	• Store the TCB address		TLTC(IPW$DDE)		
	• Set the device type		TLDT(IPW$DDE)		
	• Set the device class		TLCL(IPW$DDE)		
	• Update the number of entries		PDNE(IPW$DPD)		
	Process new request. Branch to.... >	XQ00			
	Emulate Invalid Operation				
XR82	Set the channel status byte to X'20' to indicate channel program check.		CBSC(IPW$DCB)		
	Set the first communication byte to X'20' to indicate unrecoverable I/O error.		CBC1(IPW$DCB)		

Labels	Chart KA07.2: IPW$$XR - Execution Reader		Modified Data Fields	Reg. Usage	Calls
	If the user accepts unrecoverable errors, branch to get the next command..... >	XO72			
	Update the CCW address and store it in the CCB.		CBCS(IPW$DCB)		
	Reset the residual count to zero.		CBCT(IPW$DCB)		
	Post channel and device end.		CBC1(IPW$DCB)		
	Calculate the task PIB address in register 7.			R7	
	Set the PIB to X'1A' to indicate cancel due to I/O error.				
	Issue message IR30I.				
XR84	Indicate fetch for EOJ and cancel in progress in the PIB.				
	Branch to get the next command..... >	XQ74			
	Emulate Transfer in Channel				
XR86	Load the address of the next CCW into register 8 and branch back..... >	XQ28		R8	
	Emulate Sense Operation				
XR89	If an invalid sense command was issued branch back..... >	XQ32			
	Get the address of the data field in register 1 and the length in register 2 and move the sense information, with all bits set to binary zero.			R1,R2	
	Wait for User Requests				
XQ00	Unpost the event control block in the TCB.		TCEB+2(IPW$DTC)		
	Check the reader entry in the PDB for a user request.				
	If an accounting request was issued, branch to..... >	XR32			
	If a read request was issued, branch to..... >	XQ12			
	Set up register 4 for examining the remaining entries.			R4	

Labels	Chart KA08: IPW$$XR - Execution Reader		Modified Data Fields	Reg. Usage	Calls
XQ02	Check for end of list. If so, branch to..... >	XQ04			
	Check the remaining entries in the PDB. If a request was issued branch to handle the output request..... >	XR42			
XQ04	Check for a 3540 request. If so, branch to..... >	XQ12			
	Check whether the operator has entered a PFLUSH command. If not, branch to..... >	XQ10			
XQ05	Check whether the user partition is to be canceled due to a program request or because the operator has entered a PFLUSH command. If so, branch to..... >	XQ06			
	If a PFLUSH command with the HOLD operand has been entered, the disposition of the queue record is saved, to preserve the input queue for later processing.		QRDI(IPW$DQR)		
	Otherwise, branch to..... >	XQ10			
XQ06	Using register 7 as a base register to the partition PIB, the cancel flags in the PIB are set.			R7	
	The dump options are set off for the partition.				
XQ07	If SLI work space is present, it is released.				
XQ08	The termination byte in the TCB is reset.		TCTT(IPW$DTC)		
	Set the POWER/VS cancel code in the queue record.		QRCN(IPW$DQR)		
	Indicate normal record in the TCB.		TCGP(IPW$DTC)		
	If a buffer is available branch to. >	XQ09			
	Reserve buffer space.				IPW$RSW Chart AC
	Store the real and virtual buffer addresses in the TCB.		TCDA(IPW$DTC)		
	Point register 7 to the data buffer.			R7	
XQ09	Add 2 to the buffer address and store it as the previous record in the TCB. Move a /& into the buffer.		TCPR(IPW$DTC)		
XQ10	Exit to task selection to wait for a user request.				IPW$WFC Chart AA
	On return from task selection branch to..... >	XQ00			

Labels	Chart KA09: IPW$$XR - Execution Reader		Modified Data Fields	Reg. Usage	Calls
	Handle Read Request				
XQ12	Check the job boundary switch in the TCB:				
	If a job is in progress, branch to emulate the channel program..... >	XQ26			
	If a job has just completed execution, branch to the end of job routine..... >	XQ82			
XQ14	Get a new queue record.				IPW$GQS Chart DC
	If a record is obtained branch to.. >	XQ18			
	Issue message 1Q34I to the operator.				IPW$WTO Chart AD
XQ15	Indicate Q state.		POWFLQ1		
	If the task is in S or E state, branch ..... >	XR22			
XQ16	Exit to task selection to wait for a new entry to be added to the queue.				IPW$WFQ Chart AC
	Check the termination indicator. If a PSTOP or PEND command has been issued branch to the task termination routine..... >	XR22			
	Get a new queue record.				IPW$GQS Chart DC
	If no record is obtained, branch to >	XQ16			
	Reset Q state indicator.		POWFLQ1		
	Set up register 5 as a base register for the queue record.			R5	
XQ18	Ready the task to obtain data records and to react to requests from the user program being executed in the related DOS/VS partition:				
	• Move data seek address into disk request word.		TCDW(IPW$DTC)		
	• Set job boundary switch to X'FF' to indicate job in progress.		TCJB(IPW$DTC)		
	• Clear the general purpose byte.		TCGP(IPW$DTC)		
	• Move the virtual work space address into blocking control word in preparation of the first read.		TCPR(IPW$DTC)		
	• Move 'from' RJE ID in printable format to TCB.		TCFL(IPW$DTC)		
XQ19	If in the current queue entry the physical unit for 3540 data spooling is specified branch to..... >	XR70			
	Otherwise, branch to..... >	XQ00			

Labels	Chart KA10: IPW$$XR - Execution Reader		Modified Data Fields	Reg. Usage	Calls
	Emulate Channel Program				
XQ26	Set up register 8 as a base register for the channel command word. For EXCP real, branch to handle invalid operation..... >	XQ32		R8	
XQ28:	Validate data area addresses.				IPW$VDA Chart AH
	If valid, branch to..... >	XQ29			
	If invalid, branch to..... >	XQ2A			
XQ29	If not control command..... >	XQ32			
	Classify Command				
XQ2A	If the IDAL flag or data chain flag is present in the channel command word, which feature is not supported by POWER/VS, branch to..... >	XQ32			
	Check the low-order four bits of the command code to determine which handling routine to branch to for handling the command:				
	• Invalid operation..... >	XR82			
	• Transfer in channel..... >	XR86			
	• Sense operation..... >	XR88			
	• Write operation..... >	XR82			
XQ30	• Control command..... >	XR89			
	• Read operation..... >	XQ46			

Labels	Chart KA11: IPW$$XR - Execution Reader		Modified Data Fields	Reg. Usage	Calls
XQ39	Set residual count to zero and branch to get next command..... >	XQ70			
	Emulate Read Operations				
XQ46	Check whether the user program attempts to read past a /& statement. If not, branch to..... >	XQ48			
	Using register 7 as a base register to the PIB, the PIB is set to indicate cancel due to reading past /& and a branch is made to cancel the task..... >	XQ34		R7	
XQ48	Check the pointer to the SLI work area in the PDB to determine whether the task is in SLI mode of operation. If not, branch to..... >	XQ52			
	Check whether the SLI routines require an additional record from the data file. If not, branch to..... >	XQ50			
	Get the next data record from the data file.				IPW$GDR Chart EB

Labels	Chart KA12: IPW$$XR - Execution Reader		Modified Data Fields	Reg. Usage	Calls
XQ50	Get the next record from the source statement library.				IPW$GSL Chart GE
	If the record is a data record, branch to..... >	XQ54			
	If SLI record begins with '* $' branch to..... >	XR60			
	If not, branch to..... >	XQ62			
XQ52	Check the general purpose byte for end of data. If so branch to the end of job routine..... >	XQ82			
	If reader device has read only function go to..... >	XQ5D			
XQ53	If read only switch is on..... >	XQ5C			
XQ5A	Get the next data record from the data file.				IPW$GDR Chart EB
XQ5B	Check if user CCB address changed. If not, then branch to..... >	XQ00			
XQ54	Check the new record to see whether it is a JECL statement. If not, branch to..... >	XQ56			
	Link to the JECL statement analysis routine (IPW$$XJ).				IPW$SXJ Chart GB
	If the statement is not JECL, branch to..... >	XQ56			
	Branch to get the next record..... >	XQ46			
XQ5D	If command code is not read only... >	XQ53			
	If read only switch is on, then branch to bypass read to..... >	WQ5B			
	If not on, then turn switch on and read card..... >	XQ5A	TCG2		
XQ5C	Turn read only switch off, bypass read, and give the same card buffer to register..... >	XQ5B	TCG2		
XQ56	Load the record address in register 0 Load the record length in register 1.			R0 R1	
	Check for a 3540 request. If not, branch to..... >	XQ60			
	Check for a 3540 data record. If so, branch to..... >	XQ58			
	Issue message 1Q89I PROGRAM OUT OF SEQUENCE.				IPW$WTO Chart AD
	Branch to cancel the job..... >	XQ32			

Labels	Chart KA12.1: IPW$$XR - Execution Reader		Modified Data Fields	Reg. Usage	Calls
XQ58	Check for 3540 end of file. If not, branch to..... >	XQ62			
	Set unit exception.		CBSD(IPW$DCB)		
	Set special EOF.		CBC2(IPW$DCB)		
	Branch to..... >	XQ72			
XQ60	Check for 3540 data record. If so, branch to get the next record..... >	XQ46			
XQ62	Check the record for end of file (/*) or end of job (/&), and if so, set the appropriate flags in the PIB.				

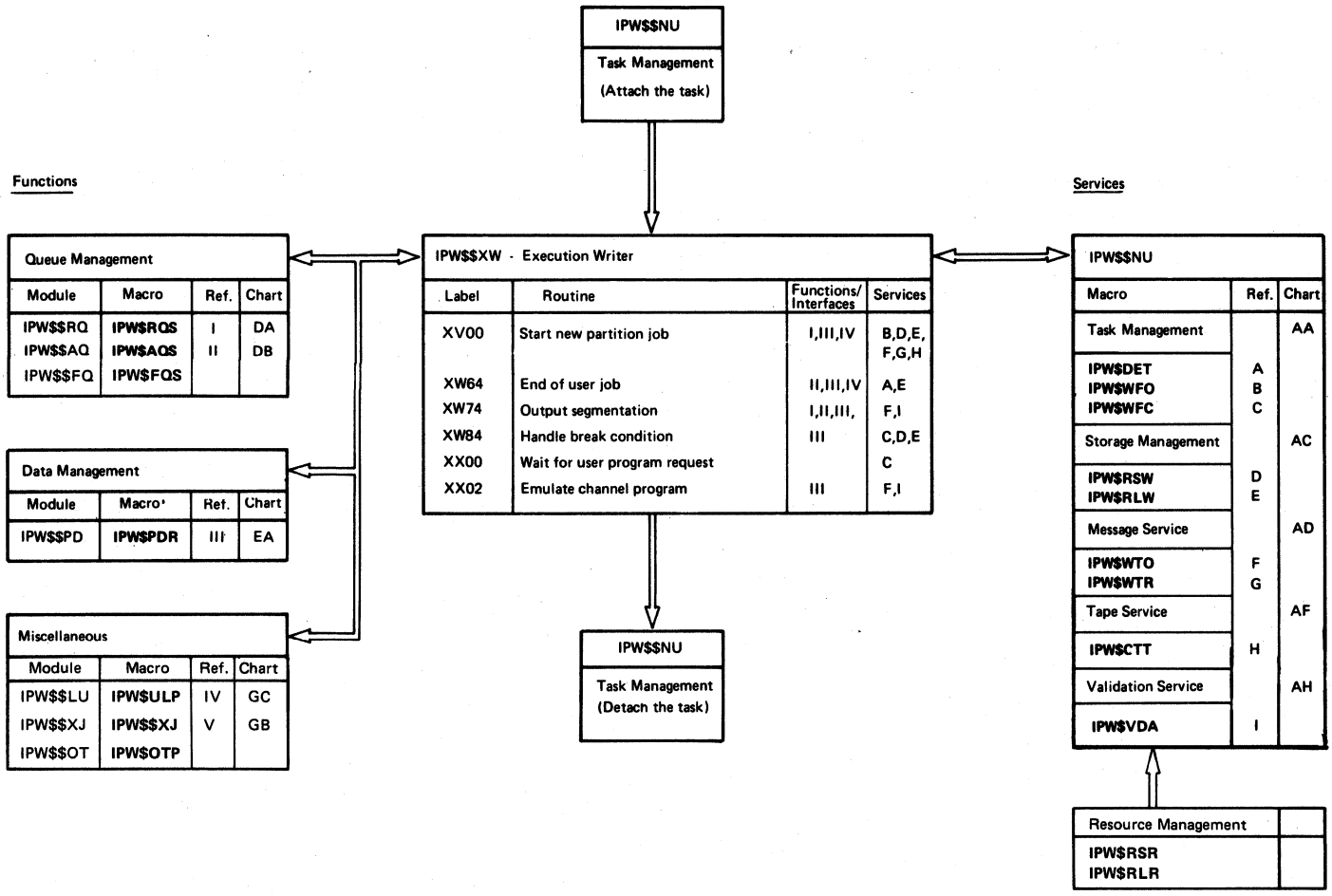
Labels	Chart KA13: IPW$$XR - Execution Reader		Modified Data Fields	Reg. Usage	Calls
XQ64	Set EOF on reader.		CBC1(IPW$DCB)		
	Set unit exception.		CBSD(IPW$DCB)		
XQ66	Load the user record length into register 3.			R3	
	Load register 2 with maximum record length: 96 when reading from 5425, 128 when reading from 3540, otherwise 80.			R2	
XQ67	Set residual count to zero.		CBCT(IPW$DCB)		
	If CCW count is the same as maximum record length, branch to..... >	XQ69			
	If CCW count is lower than maximum record length, branch to..... >	XQ68			
	Otherwise, calculate residual count and store it in CCB.		CBCT(IPW$DCB)		
	Calculate number of bytes to move.				
XQ68	If SLI bit is not on, indicate wrong length error.		CBSC(IPW$DCB)		
XQ69	If wrong length is posted, break I/O>	XQ72	CBSC(IPW$DCB)		
	Get Next Command				
XQ70	If the command chaining bit is on, load the address of the next CCW into register 8 and branch to handle the next command..... >	XQ28		R8	
XQ72	Set the necessary flags in the first communication byte, and update the CSW CCW address in the CCB.		CBC1(IPW$DCB) CBCS(IPW$DCB)		
	If the emulator interface is active, post the emulator ECB if required.				
	Set the user partition PIB to dispatchable.		PDPB		
XQ74	Reset the task entry in the PDB.		TLCB(IPW$DDE)	R7	
	Check the general purpose byte for a data break condition. If not, branch to..... >	XQ80			
XQ76	Scan the entries in the PDB for relating writer tasks, and if found, indicate the break condition in the writer TCB.		TNTT(IPW$DTC)		
XQ78	Release the data buffer. Clear the addresses in the TCB.		TCDA(IPW$DTC)		IPW$RLW Chart AC

Labels	Chart KA14: IPW$$XR - Execution Reader		Modified Data Fields	Reg. Usage	Calls
XQ80	Check the general purpose byte for the end of data. If not, branch to wait for the next user request..... >	XQ00			
	Set the job boundary switch to X'00'.		TCJB(IPW$DTC)		
	Branch to wait for the next user request..... >	XQ00			
	End of User Job				
	If writer only partition, branch to Release data buffer. >	XQ84			IPW$RLW Chart AC
XQ84	Set up register 8 for shutting down any subordinate writer tasks. (R8 = address of first entry in PDB.)			R8	
	If partition starts as 'MT' (multi-tasking) branch to..... >	XQ93			
XQ86	Scan the entries in the PDB.				
	If DISP is not N, branch to..... >	XQ90			
	If the device entry contains a class code of 'N', the device class is reset to C'L' (for list) or C'P' (for punch).		TLCL(IPW$DDE)		
XQ88	Release the ownership of the device, and branch to..... >	XQ92			IPW$ULP Chart GC
XQ90	If a task has been started: • Reset the ownership in the TCB. • Set terminator type to C'S'. • Post event control block.		TLTC(IPW$DDE) TNTT(IPW$DTC) TNEB+2(IPW$DTC)		
	Wait for signal from the list task.				IPW$WFC Chart AA
XQ92	Load the address of the next entry into register 8.			R8	
	Check for the end of entries. If not, branch to..... >	XQ86			
	Check if this is a writer-only partition. If so, branch to..... >	XQ96			
	Check for a 3540 unit specification. If not, branch to..... >	XQ94			
	Release the 3540 device list entry: • Zero the 3540 entry • Update the first entry address • Update the number of entries		PDER(IPW$DPD) PDPA(IPW$DPD) PDNE(IPW$DPD)		
XQ94	Delete the queue record from the queue.				IPW$DQS Chart DD
	Release the queue record.				IPW$FQS Chart DD

Labels	Chart KA15: IPW$$XR - Execution Reader		Modified Data Fields	Reg. Usage	Calls
XQ96	Check the termination indicator in the TCB. If it is blank branch to get the next from queue..... >	XQ14			
	Otherwise, branch to the reader termination routine..... >	XR22			

CHART KB: IPW$$XW - EXECUTION WRITER (17 PARTS)

Chart KB00: IPW$$XW - Execution Writer, General Flow and Macro Calls



Labels	Chart KB01: IPW$$XW - Execution Writer	Modified Data Fields	Reg. Usage	Calls
	This routine is initiated by the Execution Read Processor, which has acquired a queue space and formed the queue record for this task.			
XWCS	CSECT name			
XWSD	The first 16 bytes constitute the section descriptor: 'XWCS V10M1			
	Register usage			
	0: **** - Service work register		R0	
	1: **** - Service work register		R1	
	2: **** - Service work register		R2	
	3: **** - Service work register		R3	
	4: IPW$DDE - Entry in PDB		R4	
	5: IPW$DQR Queue record space		R5	
	6: IPW$DPD - PDB		R6	
	7: IPW$DCB - User CCB		R7	
	8: IPW$DCW - User CCW		R8	
	9: XWCS - Base register		R9	
	10: IPW$DPA - POWER/VS nucleus		R10	
	11: IPW$DTC - Task control block		R11	
	12: **** - Reserved for nucleus use		R12	
	13: IPW$DSV - Task save area		R13	
	14: **** - Function linkage register		R14	
	15: **CS - Function base register/ 2nd base register			
	Start New Partition Job			
	Set up second base register, using register 15.		R15	
XV00	Reserve space for the data buffer in which the output records are to be formed.			IPW$RSW Chart AC
	Store the data buffer addresses in the TCB.	TCDA(IPW$DTC)		
	The buffer control word (BCW) of the queue record and of the TCB extension area, which have been acquired by the execution reader task, are updated, so that the ownership references the proper execution writer task. (Note: This is necessary so that the storage can be released properly if abnormal termination of the task should occur.)		R1	
	The jobnumber obtained from the reader queue entry is saved for accounting purposes.	QRJ#(IPW$DQR)	R1,R14	
XV06	Update job number so that all output may be easily manipulated by operator.			

Labels	Chart KB01.1: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
	Reserve queue block.			R3	IPW$RSR Chart AB
	Get next number for this queue entry.		QRNO(IPW$DQR)	R1	
	If number is >32767, set back to 1.		MRNO(IPW$DQC)	R1	
XV08	Release queue block.				IPW$RLR Chart AB
	Set the tape flag in the TCB to X'00' to indicate 'no tape'.		TCQW(IPW$DTC)		
	If disposition not ='T', branch to. >	XV14			
XV12	Call IPW$OTP to open tape.			R2,R0	IPW$OTP Chart GG
	If disposition has been changed by open function, branch to..... >	XV14			
	If this is first open, branch to... >	XV16		R0	
	Otherwise branch to..... >	XX38			
XV14	Reserve a queue record.			R2	IPW$RQS Chart AC
	Set up record pointer.		TCBC(IPW$DTC)		
	Set up seek address.		TCDW(IPW$DTC)		
	If disposition changed for tape, branch to..... >	XX38			
XV12	If not a 3800 printer, branch to... >	XV20		R2	
	Branch to get the system defaults for the particular printer device.. >	XVA0		R14	
	If no modification character arrangement table is specified, assume the first character arrangement table as such.		SPLCMCHR (IPW$DTE)		

Labels	Chart KB02: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
	Forms Control Buffer Processing				
XV20	If FCB name specified, branch..... >	XV20		R3	
	If 3800 printer, branch to..... >	XV30			
	Link to get default FCB name..... >	XV70	TCCT(IPW$DTC)	R2	
	Branch if not a FCB type printer... >	XV40			
XV22	Set up for loading the FCB by moving the model GENL and the phase name into the work space and branch and link to the phase load subroutine.. >	XV74		R3 R14	
	If an error has occurred branch to..... >	XV26			
	Set up register 0 for converting the FCB to line table and branch and link to the line table subroutine..... >	XV80		R0	
	If an error has occurred branch to..... >	XV28			
	Check the device type code. If the device is a 3211 or 3203-4 (X'43'), a 3203 (X'4A'), a 5203 (X'4C'), or a 5203U (X'4D') branch to..... >	XV24			
	If a 3800 printer, branch to..... >	XV56			
	Otherwise, indicate no FCB printer and branch to check for UCS buffer loading..... >	XV40	TCCT(IPW$DTC)		
XV24	Branch and link to set up a special record containing LFCB macro expansion..... >	XV76		R2	
	Pass record to data file.			R2	IPW$PDR
	The total record counter is decremented by one because the FCB record is not a user data record.		QRNR(IPW$DQR)	R1	
	If there is no UCS name specified, branch to..... >	XV56			
	Otherwise, indicate end of block and branch to..... >	XW76	TCGP(IPW$DTC)	R2	
	Branch to process UCS load..... >	XV40		R2	IPW$PDR
XV26	Copy the default line table from the queue control block into the TCB... >	XV90		R14	
XV28	Issue message IQ54I FCB/UCB ERROR FOR .				IPW$WTO Chart: AD
	Indicate that FCB is wrong.		TCCT(IPW$DTC)		

Labels	Chart KB03: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
	If 3800 printer, branch to..... >	XV56			
	Otherwise, branch to process UCS buffer load..... >	XV40			
XV30	Copy the default line table from the master queue record into the TCB... >	XV90		R14	
	Branch to process SETPRT record.... >	XV56			

Labels	Chart KB03.1: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
	UCS Buffer Loading				
XV40	Check whether UCS buffer loading is needed. If not, branch to..... >	XV56			
	Check the device type code. If the device is not a 1403U (X'42'), a 3211 or 3203-4 (X'43'), a 3203 (X'4A'), or a 5203U (X'4D'), branch to..... >	XV56			
	The UCS buffer length is set to 512 bytes for a 3211 or 3203-4 printer or to 240 bytes for all other supported printers.		TCRW(IPW$DTC)	R3	
XV42	Set up for loading the UCS buffer by moving the model GENL and the phase name into the work space and branch and link to the phase load routine..... >	XV74		R3 R14	
	If an error has occurred, branch to..... >	XV50			
	If the UCS buffer length is incorrect, branch to..... >	XV50			
	Set up the record control word in the TCB by storing the data address and the fold operation code and pass the record to the data file.		TCRW(IPW$DTC)	R1	IPW$PDR Chart EA
	Branch and link to the update count subroutine..... >	XV72		R14	
	Set up the record control word in the TCB by setting the block code and pass the record to the data file.		TCRW(IPW$DTC)	R1	IPW$PDR Chart EA
	Branch and link to the update count subroutine..... >	XV72		R14	
XV48	Set up the record control word in the TCB by storing the text address and the UCB command and pass the record to the data file.		TCRW(IPW$DTC)	R1	IPW$PDR Chart EA
	Branch and link to the update count subroutine..... >	XV72		R14	
	Branch to clear the UCS name in the TCB..... >	XV56			
XV50	Issue message 1Q54I FCB/UCS ERROR FOR to the operator.				IPW$WTO Chart AD
XV56	Clear the UCS name in the TCB.		TCCT(IPW$DTC)		
	Clear the restart information in the TCB.		TCRS(IPW$DTC)		

Labels	Chart KB04: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
	3800 Setup processing				
	If not a 3800, branch to..... >	XV58			
	Address TCB extension area using register 2.			R2	
	Set up record control word in the TCB by storing the SETPRT parameter list address and the X'FD' operation code.		TCRW(IPW$DTC) TCCC(IPW$DTC)	R3	
	Pass the SETPRT record to the data file.			R1	IPW$PDR Chart: EA
	If no copy grouping is specified, use the copy value.		QRNC(IPW$DQR)		
XV58	If the number of copies indicator is zero, it is set to one.		QRNC(IPW$DQR)		
	Branch to wait for a user request..... >	XX00			

Labels	Chart KB05: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
	End of User Job				
XV60	Check the record count in the queue record to see if any output has been generated. If so, branch to..... >	XV62			
	If tape spooling, branch to..... >	XV62			
	Set disposition to 'D' and free queue set.		QRDP(IPW$DQR)	R1	IPW$FQS Chart: DE
	Branch to..... >	XV66			
XV62	Set end of data and branch and link to the NOP creation subroutine..... >	XW76	TCGP(IPW$DTC)	R2	
	Indicate end of segment in the queue record.		QRSN(IPW$DQR)		
	Check the disposition indicator to see whether the record has to be added to the reader queue. If not, branch to..... >	XV64			
	For reader entry, set the suffix number to zero.		QRSN(IPW$DQR)		
	Set the queue record identifier to C'R'.		QRQI(IPW$DQR)		
	Set the disposition to C'D'.		QRDP(IPW$DQR)		
XV64	Add the current queue set to the class chain.			R1	IPW$AQS Chart: DB
XV66	Return the data buffer space to the storage pool.			R1	IPW$RLW Chart: AC
	Check for tape processing, and if so:				
	• Release the tape assignment. Restore original mode in PUB. • Return the tape control space to the storage pool.		PUB(Byte 6)	R0,R1	IPW$ULP Chart GC IPW$RLW Chart AC
XV68	Return the queue space, if present, to the storage pool.				IPW$RLW Chart AC
	Reset the space addresses in the TCB.		TCQA(IPW$DTC)	R0,R1	
	Return the TCB extension area, if present, to the storage pool.			R1	IPW$RLW Chart AC
	Set the Execution read task to dispatchable.		TCSF(IPW$DTC)		
XV69	Detach the Execution writer task: • The task is removed from the task selection list. • The TCB storage is released.				IPW$DET Chart AA

Labels	Chart KB05.1: IPW$$XW- Execution Writer	Modified Data Fields	Reg. Usage	Calls
	Set up Default FCB Name			
XV70	Set up name as: $$BFCB for 3211 or 3203-4 $$BFCB3 for 3203 $$BFCB5 for 5203	TCCT(IPW$DTC)		
	Return 0 (R2) if not FCB printer.		R2	
	Return 4 (R2) if FCB printer.		R2	
	Update Line Count Subroutine			
XV72	Increment the line/card count in the queue record by one.	QRLC(IPW$DQR)	R1	
	Increment the number of lines spooled in the PDB.	PD#L(IPW$DPD)	R1	
	Return to caller via link register 14.		R14	
	Phase Load Subroutine			
XV74	Fill in the parameter list for the FETCH routine: • Store the phase name address. • Store the load list pointer. • Store the end of list indicator. • Set the option switch to X'01' to indicate no text loading.		R1,R2	
	Load the address of the parameter list into register 1.		R1	
	Ensure that the load may indeed take place by loading the entry point into register 0 and issuing an SVC 4.		R0	
	If the phase is not present in the core image library, or if its size is invalid, return to caller via link register 14.		R14	
	Store the buffer length in the TCB.	TCRS(IPW$DTC)		
	Set the option switch in the parameter list to X'00' to indicate loading of text.			
	Load the address of the parameter list into register 1.		R1	
	Load the phase by loading the entry point into register 0 and issuing an SVC 4.		R0	
	Load the load point address of the phase in register 3 and the directory list pointer in register 2.		R3 R2	

Labels	Chart KB05.2: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
	If not a 3800 printer, branch back to caller.			R14	
	If the phase just loaded is not an FCB, take error exit.			R2	
	Get the FCB length out of the header and save it in the TCB.		TCRS(IPW$DTC)	R2	
	Bump with register 3 over the FCB header.			R3	
	Return to caller via link register 14.			R14	
	Build FCB-record				
XV76	If no FCB name is specified in the TCB, return to caller..... >	R2		R2	
	Initialize LFCB macro expansion in data buffer and set up record control word in the TCB with an X'FF' command code.		TCRW(IPW$DTC) TCCC(IPW$DTC)	R3	
	Return to caller via register 2.... >	R2		R2	
	Process FCB to line table				
	The carriage control line table in the TCB is set according to the specified FCB image. Register on entry: R0 - lengths of FCB image R3 - points to 1st byte of FCB				
	Clear out line table in TCB.		XWLC(IPW$DTC)		
	Add the FCB start address to register 0, so that register 0 points to the end of the FCB image.			R0	
	If not a 3800 printer, branch to... >	XVC0			
	For a 3211 or 3203-4 printer, the index byte is ignored, if present.			R3	
XV83	Each FCB buffer position is checked for a channel specification. If the channel position is valid, it is stored in the TCB at its proper location.		XWLC(IPW$DTC)	R0,R1 R2,R3	
	If an invalid channel position is specified, branch to..... >	XV90			
	If end of FCB image, branch to..... >	XV94			
XV90	The carriage control line table in the TCB is set to the default line table from the master queue record.		XWLC(IPW$DTC)	R2	

Labels	Chart KB05.3: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
	The default page size is saved in the queue record.		QRER(IPW$DQR)		
	If not a 3800 printer, return to caller via register 14.....	R14		R14	
	Otherwise, subtract 6 from the default page size (the top and bottom 1/2 inch of the page are unprintable for the 3800 printer).		QRER(IPW$DQR)		
	Return to caller via register 14... >	R14		R14	
XV94	All channel positions are now converted to positions relative to channel one.		XWLC(IPW$DTC)	R0,R1 R2,R3	
	Return to caller via register 14 with a displacement of 4..... >	4(R14)			
	Get system defaults for 3800 Printer				
	The system default information is obtained from the PUB2 area and moved in the SETPRT parameter list which is part of the TCB extension area.				
XVA0	Address the TCB extension area using register 2 as a base.			R2	
	If no system defaults are requested, return to caller..... >	R14		R14	
	The logical data area (LDA) is used as a work space to carry a copy of the PUB2 area.				
	The content of the PUB2 area is obtained by issuing the EXTRACT macro instruction. Construct the EXTRACT parameter list:				
	• set length • set data area address • set ID flag (extract function) • set physical unit supplied flag • set physical unit address		SVC98LN SVC98AR SVC98ID SVC98FL SVC98SE		
	Clear out return code and issue SVC98.			R1,R15	
	On return, test for successful completion. If not, force program check by branching to..... >	XWCS			
	Reestablish the second base register, which has been destroyed by the EXTRACT supervisor function.			R15	

Labels	Chart KB05.4: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
	Now each field in the SETPRT parameter list is checked if specified. If not, the appropriate system default is inserted into the SETPRT parameter list, which is part of the TCB extension area.				
	If a FCB name is specified or no FCB default name is defined, branch to..... >	XVA2			
	Save FCB name in TCB extension area and also in the TCB.		SPLFCB TCCT(IPW$DTC)		
XVA2	If a character arrangement table is specified, branch to..... >	XVA4			
	Otherwise, set the default character arrangement table in the TCB extension area.		SPLCHAR1 (IPW$DTE)		
XVA4	If a copy modification module is already specified, branch to..... >	XVA6			
	Otherwise, set default copy modification name in TCB extension area.		SPLCMMOD (IPW$DTE)		
XVA6	If a flash id is already specified, branch to..... >	XVA8			
	Otherwise, copy the default flash id into the TCB extension area and the queue record.		SPLFLASH (IPW$DTE) QRFL(IPW$DQR)		
XVA8	If a forms id is already specified, branch to..... >	XVB0			
	Otherwise, copy the default forms name into the TCB extension area and the queue record.		SPLFORMS (IPW$DTE) QRFI(IPW$DQR)		
XVB0	If BURST=Y|N is specified, branch to..... >	XVB4			
	Otherwise, take the default burst specification and save it also in the queue record.		SPLFLAG1 (IPW$DTE) QRPS(IPW$DQR)		
XVB4	Clear out the portion of the logical data area which has been used as work area, obtaining the default settings out of the PUB2.			R2	
	Return to caller..... >	R14		R14	
	Validate 3800 FCB Image				
XVC0	Since the subroutine uses more registers than are available, contents of registers 4 - 7 are saved in the own save area addressed by register 13.		SVR4(IPW$DSV)	R4-R7	

Labels	Chart KB05.5: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
	The first 1/2 inch of the page (top margin) is not printable; therefore, counting the page size starts with the first printable line on the page. The FCB image is scanned to locate the first printable line. The total number of printable lines of the FCB image is calculated by taking the length of the FCB image and subtracting the number of bytes that represent the first and last half inch of the paper. This value is used as page size.				
XVC2	Locate the first printable line of the page.			R1,R3 R4,R5 R6	
	Clear out work registers 1 and 2.			R1,R2	
XVC4	Each FCB buffer position is checked for a channel specification. If the channel position is valid, it is stored in the TCB at its proper location.		XWLC(IPW$DTC)	R0,R1, R2,R3	
	If an invalid channel position is specified, branch to..... >	XVD4			
	If end of FCB image, subtract from page size the number of lines which represent the last half inch of the page.			R1,R2 R4,R5 R6	
	Restore registers 4 - 7. and branch to..... >	XV94		R4-R7	
XVD4	Restore registers 4 - 7. Branch to..... >	XV90		R4-R7	
	<u>Merge the current printer setup</u> This subroutine merges the current printer setup contained in the TCB extension area with the just-obtained SETPRT-request. Register on entry: R1 - SETPRT parameter list address R3 - TCB extension area address R14 - return address Each field of the SETPRT parameter list is checked if specified. If so, the appropriate field in the TCB extension area is updated with the new value. In addition, if the new printer setup requires operator intervention, the segmentation-required flag is set.				

Labels	Chart KB05.6: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
XVE0	If INIT=Y is specified in the SETPRT parameter list, set the appropriate flag in the TCB extension area.		SPLFLAG1 (IPW$DTE)		
XVE2	If SEP=M is specified in the SETPRT parameter list, set the appropriate flag in the TCB extension area.		SPLFLAG1 (IPW$DTE)		
XVE4	If SEP=0 is specified in the SETPRT parameter list, set the appropriate flag in the TCB extension area.		SPLFLAG1 (IPW$DTE)		
XVE6	If FCB verification is specified, set the appropriate flag in the TCB extension area.		SPLFLAG2 (IPW$DTE)		
XVE8	If DCHK=U (unblock data check) is specified in the SETPRT parameter list, set the appropriate flag in the TCB extension area.		SPLFLAG2 (IPW$DTE)		
XVF0	If the DEBUG keyword is specified in the SETPRT parameter list, set the appropriate flag in the TCB extension area.		SPLFLAG2 (IPW$DTE)		
XVF2	If BURST=Y|N is specified in the SETPRT parameter list, set the appropriate flag in the TCB extension area.		SPLFLAG1 (IPW$DTE)		
XVF4	If TRC=Y|N is specified in the SETPRT parameter list, set the appropriate flag in the TCB extension area.		SPLFLAG1 (IPW$DTE)		
	If TRC is not specified but INIT=Y, the TRC=N flag is set in the TCB extension area.		SPLFLAG1 (IPW$DTE)		
XVG0	If an FCB name is specified in the SETPRT parameter list, save FCB name in TCB extension area.		SPLFCB (IPW$DTE)		
XVG2	If the same copy groupings are specified, branch to..... >	XVG6			
	If no copy groupings are specified, branch to..... >	XVG5			
	Set the new copy group value in the TCB extension area and set the segmentation-required flag.		SPLCOPYG (IPW$DTE) TE38RQB (IPW$DTE)		
	Calculate the number of transmissions needed.		QRTC(IPW$DQR)	R0,R2 R3	
	Branch to continue..... >	XVG6			

Labels	Chart KB05.7: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
XVG5	If INIT=Y was specified, set the first copy group value and the transmission count to one. Set segmentation-required flag.		SPLCOPYG (IPW$DTE) QRTC(IPW$DTE) TE38RQB (IPW$DTE)		
XVG6	If a different CINDX value is not specified, branch to..... >	XVH0			
	If CINDX is not specified in the SETPRT request, branch to..... >	XVG7			
	If CINDX value is > 1, update the TCB extension area. Branch to..... >	XVG8	SPLCINDX (IPW$DTE)		
XVG7	If INIT=Y was not specified, branch to..... >	XVH0			
	Set CINDX value to one.		SPLCINDX (IPW$DTE)		
XVG8	Set the segmentation-required flag.		TE38RQB (IPW$DTE)		
XVH0	If a forms value different from the previous one is specified, update the TCB extension area and set the segmentation-required flag.		SPLFORMS (IPW$DTE) TE38RQB (IPW$DTE)		
XVH4	If FLASH was specified, update flash id and flash count in the TCB extension area.		SPLFLASH SPLFLSHC (IPW$DTE)		
	If INIT=Y was specified but no flash id, reset flash id and count in the TCB extension area.		SPLFLASH SPLFLSH (IPW$DTE)		
XVI0	If CHARS has been specified in the SETPRT request, the CHARS values are updated in the TCB extension area.		SPLCHARS (IPW$DTE)		
XV12	If MODIFY was specified, update appropriate fields in the TCB extension area.		SPLCPMOD SPLCHAR1 (IPW$DTE)		
	Return to caller via register 14... >	R14			
	Handle SETPRT request				
XW00	Check if the device is a 3800 printer. If not, branch to..... >	XW88			
	Address the SETPRT parameter list using register 1.			R1	
	The TCB extension area containing the current printer setup is addressed by register 3.			R3	
	Check each field of the SETPRT parameter list. If specified, the appropriate field in the TCB extension area is updated with the new value by branching to..... >	XVE0		R14	

Labels	Chart KB05.8: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
XW06	If no segmentation has been forced, branch to..... >	XW18			
	(Note: Segmentation is done when the new SETPRT request requires a different printer setup for BURST, FORMS, FLASH, COPYG, and CINDX.)				
	Turn off segmentation-required flag.		TE38RQB (IPW$DTE)		
	Check the record count to see if any records have been trapped. If not, branch to..... >	XW18			
	Set end of data, branch, and link to the NOP creation routine..... >	XW76	TCGP(IPW$DTC)	R2	
	Add the current queue set to the class chain.			R0	IPW$PDR Chart DB
	Reserve a new queue record.				IPW$RQS Chart DA
	Reset record counts in the queue record to zero.		QRNA(IPW$DQR) QRNR (IPW$DQR)		
	Set the number of tracks used for spooling the output to one.		QRNT (IPW$DQR)	R1	
	If not tape spooling, update the request word in the TCB.		TCDW(IPW$DTC)		
XW08	Set the suffix number in the queue record to zero.		QRSN(IPW$DQR)		
	Update job number so that all output may be easily manipulated by the operator:				
	Reserve a disk management block (contains master queue record).			R3	IPW$RSR Chart AB
	Get next number for this queue entry.		QRNO(IPW$DQR)	R1	
	Increase next job number by one; if number is > 32767, set back to one.		MRNO(IPW$DQC)	R1	
XW10	Release the disk management block.				IPW$RLR Chart AB
XW12	Update the queue record with the values obtained from the SETPRT request:				
	• forms-id • flash-id • copy groupings • paper thread request		QRFI(IPW$DQR) QRFL(IPW$DQR) ORCG(IPW$DQR) QRPS(IPW$DQR)		
XW14	Check if the CINDX value is > 1. If so, the copy count is set to one, assuming that the user will manage the transmission himself.		QRNC(IPW$DQR) QRCI(IPW$DQR)		

Labels	Chart KB05.9: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
XW16	Check if the transmission count is zero. If not, set number of copies to the same value.		QRNC(IPW$DQR)		
XW18	If no FCB name, or the same FCB, has been specified in the SETPRT request, branch to..... >	XW32			
	Copy the default line table from the master record in the TCB..... >	XV90		R14	
	If the logical data area (LDA) is not empty, reserve temporary storage to carry the FCB image to be loaded.			R3	IPW$RSW Chart AC
XW24	Save the temporary work space address.		TE38GW (IPW$DTE)	R2	
	Set up for loading the FCB by moving the model GENL and the phase name into the workspace.			R3	
	Branch to load the FCB phase..... >	XV74		R14	
	If an error has occurred, branch to..... >	XW26			
	Set up register 0 for converting the FCB to line table and branch and link to the line table subroutine..... >	XV80		R0 R14	
	If an error has occurred, branch to..... >	XW26			
	Save the new FCB phase name (including prefix) in the TCB.		TCCT(IPW$DTC)	R2	
	Branch to continue..... >	XW30			
XW26	Issue message: '1Q54I FCB/UCS ERROR.'				IPW$WTO Chart AD
	Clear out FCB name in the TCB extension area (SETPRT parameter list).		SPLFCB (IPW$DTE)	R2	
XW30	When temporary storage has been used, it is released and returned to the storage pool.			R1	IPW$RLW Chart AC
XW32	Set up record control word in the TCB with the SETPRT parameter list address and with the X'FD' command code.		TCRW(IPW$DTC) TCCC(IPW$DTC)		
	Place record in data file.			R1	IPW$PDR Chart EA
	The total record count is decremented by one because the SETPRT record is not a user data record.		QRNR(IPW$DQR)	R1	

Labels	Chart KB05.10: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
	Some printer setup flags are turned off: • FCB verification • offset stacking • mark form		 SPLFLAG1 SPLFLAG2 SPLFLAG2 (IPW$DTE)		
	Branch to..... >	XX76			
	Handle Q-SETPRT request				
XW36	Move the proper system/programmer logical unit number out of the CCB into the SETPRT parameter list.		SPLLUSYS (IPW$DTE)		
	Get the address of the data field in register 1 and the length in register 2. Move the current SETPRT parameter list into that area.			R1 R2	
	Branch to get next operand..... >	XX76			
	Output Segmentation				
	Count-driven Segmentation				
XW58	Save current request word.			R0,R1	
XW76	Set end of data and branch and link to the NOP creation subroutine..... >	XW76	TCGP (IPW$DTC)	R2	

Labels	Chart KB06: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
	Restore request word.		TCRW(IPW$DTC)	R0,R1	
	Add the current queue set to the class chain.			R2	IPW$AQS Chart DB
	If intermediate storage is on magnetic tape, branch to the tape segmentation routine..... >	XV12		R0	
XW59	Reserve a new queue record.			R2	IPW$RQS Chart DA
	Update the request word in the TCB.		TCDW(IPW$DTC)		
	Increment the suffix number in the queue record by one.		QRSN(IPW$DQR)		
	Reset the record counts in the queue record to zero.		QRNA(IPW$DQR) QRNR(IPW$DQR)		
	Set the track number (number of used tracks) to one.		QRNT(IPW$DQR)	R1	
	Issue message 1Q53I OUTPUT SEGMENTED FOR.				IPW$WTO
	If not a 3800 printer, branch to... >	XV60			
	Otherwise, save request word.			R0,R1	
	Set up record control word in the TCB by storing the SETPRT parameter list address and an X'FD' as command code.		TCRW(IPW$DTC) TCCC(IPW$DTC)		
	Branch to..... >	XW61			
XW60	If FCB load is not needed, branch to>	XX38			
	Link to get LFCB record..... >	XV76		R2	
XW61	Pass FCB/SETPRT record.				IPW$PDR Chart EA
	Subtract one from the total record count in the queue record, because the FCB SETPRT record is not a user data record.		QRNR(IPW$DQR)	R1	
	Branch to..... >	XX38			

Labels	Chart KB06.1: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
	Program-driven Segmentation				
XW62	Check the record count to see if any records have been trapped. If not, branch to..... >	XW66			
	Set end of data and branch and link to the NOP creation subroutine..... >	XW76	TCGP(IPW$DTC)	R2	
	Add the current queue set to the class chain.			R2	IPW$AQS Chart DB
	If spooling to tape..... >	XW63			
	Reserve a new queue record, if spooling to disk.			R2	IPW$RQS Chart DA
XW63	Reset the record counts in the queue record to zero.		QRNA(IPW$DQR) QRNR(IPW$DQR)		
	Set the track number (number of used tracks) to one.		QRNT(IPW$DQR)	R1	
	If tape spooling, branch to..... >	XW64			
	Update the request word in the TCB.		TCDW(IPW$DTC)		
XW64	Set the suffix number in the queue record to zero.		QRSN(IPW$DQR)		
XW66	Set the forms identifier to blanks.		QRFI(IPW$DQR)		
	Validate data area address. If the address is invalid branch to..... >	XW88		R2	IPW$VDA Chart AH
	Save the new FCB name in the TCB.		TCCT(IPW$DTC)	R1	
	If a forms ID is present the new forms ID is moved to the queue record. If not, branch to..... >	XW67	QRFI(IPW$DQR)		
XW67	Is it tape spooling? If not, branch to..... >	XX76			
	Reserve Queue record.			R1	IPW$RQS Chart DA
	Branch to get the next command..... >	XX76			

Labels	Chart KB07: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
	Handle Break Condition				
XW68	Force end of block and branch and link to the NOP creation subroutine >	XW76	TCGP(IPW$DTC)	R2	
	Release the data buffer space.				IPW$RLW Chart AC
	If task stopped, branch to..... >	XW74			
XW70	Reset the data break condition in the TCB.		TCTT(IPW$DTC)		
XW72	Check for a user request. If so, branch to..... >	XW74			
	Unpost the event control block and exit to task selection.		TCEB+2(IPW$DTC)		IPW$WFC Chart AA
	Check for a data break condition. If so, ignore it and branch to..... >	XW70			
	Check for a stop condition. If not so, branch to..... >	XW72			
	Recover from Break Condition				
XW74	Reserve new buffer space.				IPW$RSW Chart AC
	Store the new buffer addresses in the TCB.		TCDA(IPW$DTC)		
	Check for stop condition. If so, branch to..... >	XV60			
	Branch to process the next user request..... >	XX02			
	NOP Creation Subroutine				
XW76	This subroutine creates a NOP record to pass the end of data or end of block condition for the queue record.				
	• Set data address to blanks • Set length to one • Set NOP command code		TCCC(IPW$DTC) TCGP(IPW$DTC) TCCC(IPW$DTC)		
	Pass the record to the data file.			R0	IPW$PDR Chart EA
	Return to caller via link register 2.			R2	
	Handle special printer commands.				
XW78	Set up register 2 for checking the command code against the entries in the printer command table.			R2	

Labels	Chart KB08: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
XW80	Check the command code against the entries in the printer command table.			R2	
	If the command code matches with one of the printer command tables branch to..... >	XW82			
	If the command code is not in the table, it represents an immediate command, branch to..... >	XX56			
XW82	If a correct printer has been specified, branch to..... >	XW85		R3	
	If the command is invalid branch to >	XW88			
XW85	If the command should be ignored, branch to..... >	XX76			
	If not a 3800 printer, branch to... >	XW86			
	If unblock data check command code, set the unblock data check flag in TCB extension area and branch to... >	XW86	SPLFLAG1 (IPW$DTE)		
XW85A	If block data check command code, turn off the unblock data check flag in the TCB extension area and branch to..... >	XW86	SPLFLAG1 (IPW$DTE)		
XW85B	If not, initialize printer command code, branch to..... >	XW85C			
	Clear out current printer setup in TCB extension area.		SPLLIST (IPW$DTE)		
	Set initialize printer flag.		SPLFLAG1		
	Branch to get default line table... >	XV90		R14	
	Branch to..... >	XW32			
XW85C	If a load FCB command code, set the FCB unknown flag in the TCB extension area.		SPLFCB (IPW$DTE)		
XW86	If the command does not imply a data transfer, branch to treat it as a control command..... >	XX55			
	Validate data area address. If the address is invalid, branch to..... >	XW88		R2	IPW$VDA Chart AH
	If the command does not represent an FCB load, branch to treat it as a write operation..... >	XX21			
	Set up registers 0 and 3 with the data address and length for processing the FCB load.			R0,R3	
	Branch and link to the FCB line table subroutine..... >	XV80		R14	
	If an error has occurred, branch to >	XW88			

Labels	Chart KB08.1: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
	Branch to the write routine..... >	XX21			
	Emulate Invalid Operation				
XW88	Set the channel status byte to X'20' to indicate channel program check.		CBSC(IPW$DCB)		
	Set the first communication byte to X'20' to indicate unrecoverable I/O error.		CBC1(IPW$DCB)		
	Bump to the next CCW in chain.			R8	
	If the user accepts unrecoverable errors, branch to get the next command..... >	XX78			
	Issue message IR30I.				IPW$WTO
	Set up the cancel code X'1A' in register 0 and branch to..... >	XX80		R0	
	Emulate Transfer in Channel				
	Validate data area address. If the address is invalid, branch to..... >	XW88		R2	IPW$VDA Chart: AH
	If the address of the next CCW is not on doubleword boundary, branch to.. >	XW88		R1	
	If more than 255 TIC-commands have been specified in a single channel program, branch to..... >	XW88	TLCB(IPW$DDE)	R1	
	Load the address of the next CCW into register 8 and branch back..... >	XX04		R8	
	If the user will accept unrecoverable I/O errors, branch to..... >	XW88			
	Otherwise, ignore error and branch to..... >	XX76			
	Wait for User Program Request				
XX00	Check the termination indicator in the TCB for a data break condition. If so branch to handle the break condition..... >	XW68			
	Check the termination indicator for a stop condition. If so, branch to.. >	XV60			

Labels	Chart KB09: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
	Check the entries in the PDB to see if the user program has issued a write request. If so branch to the emulation routine..... >	XX02			
	Unpost the event control block in the TCB.		TCEB+2(IPW$DTC)		
	Exit to task selection.				IPW$WFC Chart AA
	Branch to check for a user request..... >	XX00			
	Emulate Channel Program				
XX02	Set up register 8 as a base register for the channel command word.			R8	
	Check for EXCP real, which POWER/VS does not support when issued from a virtual partition.				
	If so, branch to..... >	XW88			
	Classify Command				
	The CCW address in the CCB is checked to determine whether it is in the user partition or in the LTA (logical transient area). If the CCW is not a valid address, branch to..... >	XW88			
XX03	If the IDAL flag or data chain flag is present in the channel command word, which feature is not supported by POWER/VS, branch to..... >	XW88			
	Check the low-order four bits of the command code to determine which handling routine to branch to for handling the command:				
	• Invalid operation..... >	XW88			
	• Tranfer in channel..... >	XW90			
	• Sense operation..... >	XX12			
	• Read operation..... >	XX16			
	• Write operation..... >	XX18			
	• Control command..... >	XX42			

Labels	Chart KB10: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
	Emulate Sense Operation				
XX12	If an invalid sense command was issued branch to..... >	XW88			
XX13	Validate data area address. If the address is invalid, branch to..... >	XW88			IPW$VDA Chart AH
	Get the address of the data field in register 1 and the length in register 2 and move the sense information with all bits set to binary zero.			R1,R2	
	If the command code is a sense I/O (X'E4') - only valid for a 3800 - the type and model of the 3800 printer are moved into the data area.				
	Branch to get the next command..... >	XX76			
	Emulate Read Operation				
XX16	Since any read operation addressed to an output device is assumed to be a dummy, it is ignored.				
	If this is not a punch task, branch to get the next command..... >	XX76			
	Validate data area address. If the address is invalid, branch to..... >	XW88		R2	IPW$VDA Chart AH
	Branch to continue..... >	XX56			
	Emulate Write Operation				
XX18	Validate data area address. If the address is invalid, branch to..... >	XW88			IPW$VDA Chart AH
	If the command code represents a SETPRT request, branch to..... >	XW00			
	If a punch command is being processed, branch to the punch write routine..... >	XX24			
	Branch and link to the line count subroutine for updating the line counter..... >	XX88		R14	
	If the command is a write and skip to channel one command, branch to..... >	XX22			
	If not a 3800 printer, branch to... >	XX21			
	If the command code is a load graphic modification module or copy modification module, branch to..... >	XW88			

Labels	Chart KB11: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
XX21	Form the record control word in the TCB:				
	• Move the user CCW.		TCRW(IPW$DTC)		
	• Set general purpose byte to X'01' to indicate data record if necessary.		TCGP(IPW$DTC)		
	• Clear the rest of the general purpose byte.		TCGP+1(IPW$DTC)		
	Branch to..... >	XX38			
XX22	Collect the record text from the user area:				
	• Move the user CCW.		TCRW(IPW$DTC)		
	• Set command code to X'01' to indicate write with no space.		TCCC(IPW$DTC)		
	• Set general purpose byte to X'01' to indicate data record.		TCGP(IPW$DTC)		
	• Clear the rest of the general purpose byte.		TCGP+1(IPW$DTC)		
	Collect the user record.			R2	IPW$PDR Chart EA
	Branch and link to the line count subroutine for updating the line counter ..... >	XV72		R14	
	If the CCB address has changed, branch to..... >	XX00			
	Set command code to X'8B' to indicate skip to channel one.		TCCC(IPW$DTC)		
	Set general purpose byte to indicate a length of one.		TCGP(IPW$DTC)		
	Branch to..... >	XX32			
	Emulate Punch Write Operation				
XX24	Form the record control word in the TCB:				
	• Move the user CCW.		TCRW(IPW$DTC)		
	• If the command code does not indicate potential print with no feed, set general purpose byte to X'01' to indicate data record.		TCGP(IPW$DTC)		
	• Clear the rest of the general purpose byte.		TCGP+1(IPW$DTC)		
	If the output has disposition 'I', and it is a 3525, 2560, or 2560 print record, branch to..... >	XX64			
	If the output disposition is I, and it is a punch record, the record is to be added to the input queue and the command code is therefore set to zero. Branch to..... >	XX38			

Labels	Chart KB12: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
	Test Output Segmentation				
XX32	Check if segmentation has been requested. If not, branch to..... >	XX38		R3	
	Check for punch record. If not, branch to..... >	XX34			
	For a punch record, check for a normal data record. If not, ignore segmentation and branch to..... >	XX38			
	Load current record count and branch to..... >	XX36		R1	
XX34	Check for skip-to-channel-one command. If not, ignore segmentation and branch to..... >	XX38			
	Load current page count.			R1	
XX36	If segmentation limit has been reached, branch to segmentation routine..... >	XW58			
XX38	Pass record to data file.			R0	IPW$PDR Chart EA
	If X'63' command then indicate end of block and write NO-OP..... >	XX64	TCGP(IPW$DTC)	R2	
XX39	If the CCB address has changed, branch to..... >	XX00			
	If the disposition indicates tape processing, and the reflective spot on the tape has been hit, force segmentation for next request.		QRBS(IPW$DQR)		
XX40	If the previous command was a control command, do not update the data transfer command and branch to..... >	XX68			
	Branch to count routine..... >	XX64			

Labels	Chart KB12.1: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
	Emulate Control Command				
XX42	If the command is not a NOP, then branch to.. ..... >	XX43			
	Validate data address.			R2	IPW$VDA Chart AH
	If invalid, ignore command..... >	XX76			
	Is 1st 4 characters of data area '* $$'?				
	If not, branch to..... >	XX76			
	If so, set up request word.		TCRW(IPW$DTC)		
XX4A	Scan JECL record.			R2	IPW$SXJ Chart GB
	If not a JECL record, branch to.... >	XW94			
	If it was LST/PUN card for our device, we do not handle output anymore, post to Stop.		TCTT(IPW$DTC)		
	Branch to..... >	XX76			
XX43	If the command is a printer command, branch to the printer control routine..... >	XX54			

Labels	Chart KB13: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
XX44	If the output disposition is I, the record has to be added to the input queue. However, control operations cannot be emulated for input. Therefore, ignore the command and branch to get the next user CCW.... >	XX76			
	If the device is not a 2560P, branch to..... >	XX46			
	If the command is a punch/SS command, branch to treat it as a write command..... >	XX18			
XX46	Form the record control word in the TCB:				
	• Move the user CCW.		TCRW(IPW$DTC)		
	• If the command is a card motion command and the device is a 3525P, 3525RP, or a 2540P, set the general purpose byte to X'40' to indicate data record.		TCGP(IPW$DTC)		
XX50	• Otherwise, set the general purpose byte to X'00'.		TCGP(IPW$DTC)		
XX52	Branch to handle punch control command..... >	XX30		R1	
	Emulate Printer Control Operations				
XX54	If the command code represents an LFCB operation code branch to..... >	XW62			
	Branch and link to the line count subroutine to update the line counter..... >	XX88		R14	
	If the command code is not a skip or space branch to..... >	XW78			

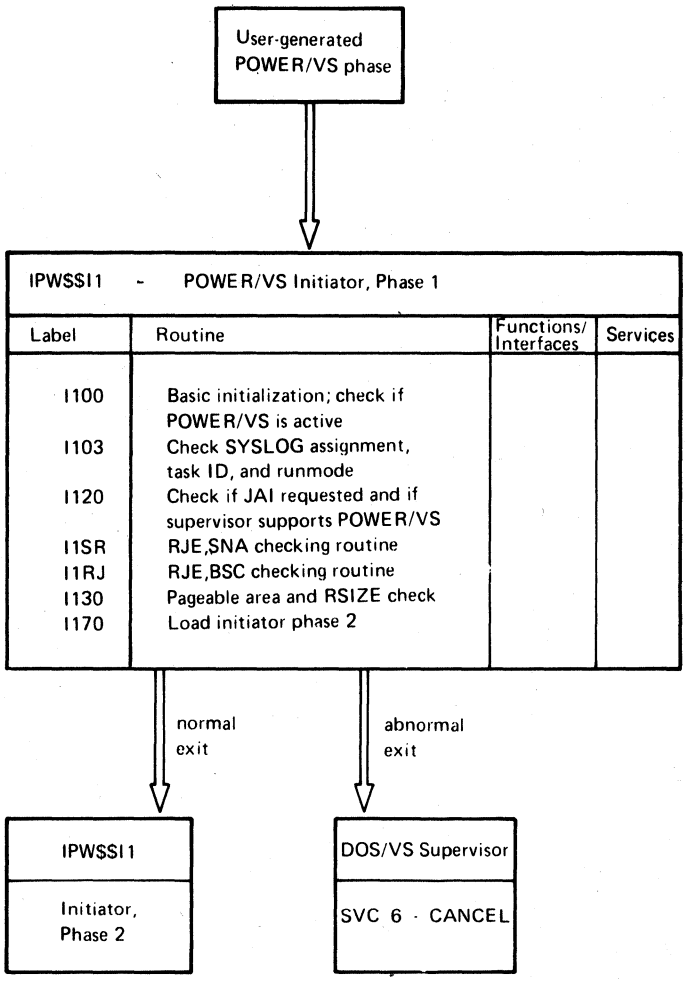
Labels	Chart KB14: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
XX56	Form the record control word in the TCB:				
	• Move the command code.		TCRW(IPW$DTC)		
	If the command is a 'Skip to Channel one' or not an immediate command, branch to..... >	XX57			
	Otherwise, check if the previous command was a 'Print No Space.' If yes, branch to..... >	XX58			
XX57	Load the address of a dummy I/O area into register 1 and store it in the TCB.		TCRW(IPW$DTC)	R1	
	Set a record length of one.		TCGP(IPW$DTC)		
	Combine the previous Print No Space command with the current immediate command. Branch to..... >	XX76			
	Branch to..... >	XX32			
	Increment and Test Counts				
XX64	If user data has been added to the file, the line/card counter in the queue record is incremented by one.		QRLC(IPW$DQR)	R1	
	Increment the list (PD#L) or punch (PD#C) value in the PDB by one.		PD#L(IPW$DPD) PD#C(IPW$DPD)	R1 R1	
XX68	Check whether the output limit has been exceeded. If so, the limit value is increased by the standard value and message 1Q52I OUTPUT LIMIT EXCEEDED FOR is issued to the operator.		QRBM(IPW$DQR)	R1,R2	IPW$WTO Chart AD
	If the CCB address has changed, branch to..... >	XX00			
XX70	Check the record type for a punch record. If so, bypass the page counting routine and branch to..... >	XX76			
	Check the control character in the record just written. If it does not represent a skip to channel one, branch to get the next user CCW.... >	XX76			
	Increment the number of pages spooled in the PDB.		PD#P(IPW$DPD)	R2	
	Increment the number of pages in the queue record.		QRNP(IPW$DQR)	R1	

Labels	Chart KB15: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
	Get Next Command				
XX76	Load the address of the next channel command into register 8.			R8	
	Check the command for command chaining. If not, branch to..... >	XX78			
	If channel 9 was posted, branch to break the chain..... >	XX78			
	If channel 12 was not posted, branch to process the new command..... >	XX04			
XX78	Indicate no cancel by setting register 0 to zero.			R0	
XX80	Indicate completion of the channel program to the user:				
	• Store the updated CCW address.		CBCS(IPW$DCB)	R8	
	• Set the residual count to zero.		CBCT(IPW$DCB)		
	• Set the necessary flags in the communication byte.		CBC1(IPW$DCB)		
	If the emulator interface is active, post the emulator ECB if required.			R1,R2 R3	
XX84	Get the address of the task PIB in register 8.			R8	
	Set the PIB to dispatchable.				
	If a cancel condition has occurred (register 0 = nonzero), indicate cancel due to I/O error and signal fetch for EOJ and cancel in progress in the PIB.				
XX86	If the CCB address has changed, branch to..... >	XX00			
	Reset the task entry in the TCB.		TLCB(IPW$DTC)	R7,R8	
	Branch to wait for the next user request..... >	XX00			

Labels	Chart KB16: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
	Carriage Control Emulation				
XX88	Get the command code, which represents either a skip or a space code in register 1.			R1	
	If the command code represents a Print No Space, return to caller via register 14 with a displacement of 4.			R14	
	If the command code is not a Skip or Space code, return to caller via register 14.			R1 R14	
XX90	If the command code represents a skip code branch to..... >	XX94		R1,R2	
	Calculate the new value of the line count in register 2.			R2	
	Check for carriage channel 9 processing. If not branch to check for carriage channel 12 processing..... >	XX91			
	If necessary post channel 9 overflow.		CBC2(IPW$DCB)		
	Set unit check.		CBSD(IPW$DCB)		
	Branch to check if the page is filled..... >	XX92			
XX91	Set master record addressibility.			R3	
	If channel 12 is not in the LTAB specified, branch to..... >	XX92			
	If the current line count is lower than LTAB channel 12, then branch to>	XX92			
	Is option MULT12 in the master record is on, branch to..... >	XX91A			
	If channel 12 already passed, branch to..... >	XX92			
XX91A	Post channel 12 overflow.		CBSD(IPW$DCB)		
XX92	If the page is filled, the page size is decremented.		XWPC(IPW$DTC)		
	Branch to store the new count..... >	XX96			
XX94	Calculate the new skip value in register 2.			R2	
XX96	Store the new line/skip value in the TCB.		XWLC(IPW$DTC)	R2	
	Return to caller via register 14 with displacement of 4.			R14	

CHART LA: IPW$$I1 - INITIATOR, PHASE 1 (7 PARTS)

Chart LA00: IPW$$I1 - Initiator, Phase 1, General Flow and Macro Calls



Labels	Chart LA01: IPW$$I1 - Initiator, Phase 1		Modified Data Fields	Reg. Usage	Calls
	Entry to this routine is from the user-generated POWER/VS phase.				
I1SD	The first 16 bytes constitute the section descriptor: 'I1CS V10M0'				
	Register usage				
	0: **** - Work register			R0	
	1: **** - Service work register			R1	
	2: **** - Service work register			R2	
	3: **** - Service work register			R3	
	4: **** - Service work register			R4	
	5: **** - Service work register			R5	
	6: IPW$DGN - Address of generation table			R6	
	7: COMREG - BG communications region address			R7	
	8: SYSCOM - SYSCOM address			R8	
	9: **** - Base register			R9	
	10: **** - Start address of POWER/VS nucleus			R10	
	11: **** - Line table address			R11	
	12: **** - Service work register			R12	
	13: **** - Pointer to modules			R13	
	14: **** - Linkage register			R14	
	15: **** - Service work register.			R15	
I100	Set up addressability of SYSCOM and BG COMREG in registers 8 and 7.			R8,R7	
	If POWER/VS is already active (IJBFLG03 = POWA), branch to..... > where message 1Q22I POWER/VS ALREADY ACTIVE is printed and the partition is canceled.	I1CN			
I103	Using SYSIR, check if SYSLOG is assigned to a 1052, a 3277, or a CRT.		SYSLOGT		
	If this is so, branch to..... >	I105			
	If not, branch to..... > where error message 1Q06I SYSLOG NOT ASSIGNED TO CONSOLE is printed and the job is canceled.	I1CN			
I105	If POWER/VS is the maintask, continue at..... >	I115			
	If not, go to..... > where error message 1Q02I POWER/VS CANNOT RUN AS A SUBTASK is printed and the job is canceled.	I1CN			
I115	Next the POWER/VS runmode is tested. If this is virtual (register 1 is zero), processing continues at..... >	I120		R1	
	If not, go to the..... > routine where message 1Q01I POWER/VS CANNOT RUN IN REAL MODE is printed and the job is cancelled.	I1CN			

Labels	Chart LA02: IPW$$I1 - Initiator, Phase 1		Modified Data Fields	Reg. Usage	Calls
I120	If POWER/VS accounting support is not requested (GNJA ≠ C'A'), or if accounting support is requested and also supported by the supervisor (JCSW1 = X'80'), control is passed to..... >	I125			
	if not, branch and link to..... >	I1MS			
	where error message 1Q10I SUPERVISOR WITHOUT ACCOUNTING SUPPORT is printed. Then the account flag in the generation table is reset.		GNJA(IPW$DGN)		
I125	If the supervisor supports POWER/VS (IJBFLG03 = X'08'), control is passed to..... >	I1SR			
	If not, branch to the..... >	I1CN			
	routine where error message 1Q08I SUPERVISOR WITHOUT POWER/VS SUPPORT is printed and the job is canceled.				
I1SR	RJE,SNA CHECKING ROUTINE If no RJE,SNA processing is needed (GNSR = X'00'), control is passed to>	I1RJ			
	to check for RJE,BSC processing.				
	If no number of logical units is specified (GNSU = 0), or if the number of logical units specified is higher than the number of SNA remotes (GNSU > GNSR), the number of logical units is set equal to the number of SNA remotes.		GNSU(IPW$DGN)		
RS02	Addressability of the SNA remote control block is set up in register 2, and the pointer to the first block is saved. Then the number of SNA remotes is set in register 15 for scanning purposes.		RMPT(IPW$DRM)	R2 R15	
RS05	The punch and list routing IDs are checked for validity, and if they are invalid, a link is made to..... >	RS35			
	where message 1Q16I INVALID PUN (or LST) ROUTING FOR remid is printed, and the routing IDs are reset to X'00'.		RMPR(IPW$DRM) RMLR(IPW$DRM)		
RS25	If the terminal characteristics of this SNA remote control block are described in another remote control block (REF=mmm in the PRMT macro), the information is moved using registers 4 and 5 as work registers.		RMRI(IPW$DRM)	R4,R5	
RS30	The address of the next SNA remote control block is loaded into register 2 and a branch is made back to..... >	RS05			
	When all SNA remote control blocks have been checked, processing is continued at..... >	RS40			

Labels	Chart LA03: IPW$$I1 - Initiator, Phase 1		Modified Data Fields	Reg. Usage	Calls
RS35	The current remote ID is calculated in register 4 and a branch made to. > to print the error message.	RJ58			
RS40	A GETVIS macro is issued to obtain virtual storage for the SNA remote entries, the LU table, and the logon storage using registers 0, 1, and 4 as work registers. If no virtual storage can be obtained (register 15 not zero), a branch is made to..... > where error message 1Q26I GETVIS AREA TOO SMALL is printed, and the job is terminated.	I1CN		R0,R1, R4	
RS45	The address of the SNA remote control block is saved. The space of the SNA remote control block (RMCB) in the GETVIS area is cleared and all SNA remote entries are moved into it. All LU names specified in the PRMT macro(s) are copied into the LU table. The RMCB space is rounded off upwards to a 2K boundary. Then 2K is added to this address in register 2, which will be the minimumbegin address of the SUCB/LUCB space. Then the SNA remote control block is initialized with the following fields from the POWER/VS generation table: • ACB password length • ACB password • Number of SNA remote entries • First SNA remote ID • Last SNA remote ID • Storage descriptor • Translate table (EBCDIC to ASCII) • Translate table (ASCII to EBCDIC)		GNRM(IPW$DGN) RMAL(IPW$DRM) RMAP(IPW$DRM) RMSR(IPW$DRM) RMFR(IPW$DRM) RMHR(IPW$DRM) RMSD(IPW$DRM) RMEA(IPW$DRM) RMAE(IPW$DRM)	R2	

Labels	Chart LA03.1: IPW$$I1 - Initiator, Phase 1		Modified Data Fields	Reg. Usage	Calls
	For each SNA remid a pointer is set to the associated LU name string in the LU table. If the pointer is zero then no LU name was specified. Using registers 4 and 5 as work registers, the begin address of the SUCB/LUCB pool is calculated and saved in the generation table. To calculate the SUCB/LUCB space requirements, the highest SESSLIM value specified for all remids is used. Then the space requirements for WACBs and compation tables are calculated, plus 4K for JECL specified compaction tables. If the GETVIS area available does not satisfy the required space needs for RJE,SNA then branch to..... >	I1CN	GNSS(IPW$DGN)	R4,R5	
I1RJ	RJE,BSC CHECKING ROUTINE If no RJE is specified (no PLINE macro included, thus GNNL = X'00'), control is passed to..... >	I130		R2 R11 R15	

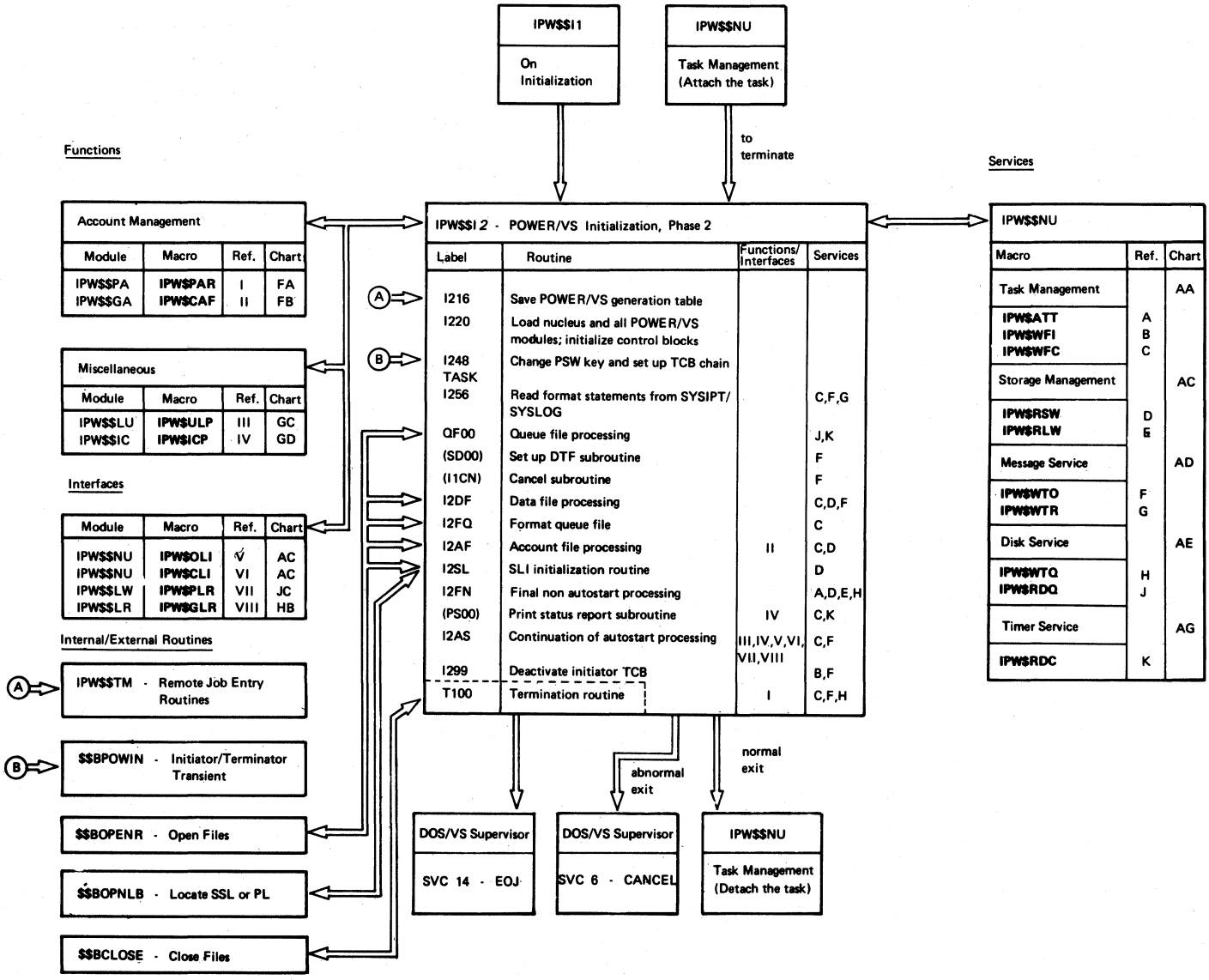
Labels	Chart LA04: IPW$$I1 - Initiator, Phase 1		Modified Data Fields	Reg. Usage	Calls
RJ00	Point register 4 to the PUB table and check if the line blocks in the generation table have corresponding line addresses in the PUB.			R4	
RJ10	If this is not the case, or if the corresponding device type is not 2701 or 2703, the line block is removed from the table, the number of valid line blocks is decreased by one, and message 1Q14I NO MATCHING PUB FOR CUU is printed in routine..... >	I1MS	GNNL(IPW$DGN)		
RJ15	All line blocks are processed. If GNNL or GNNR is zero, no remote blocks are available and processing continues at..... >	I130			
RJ25	Validated line blocks are written into the line table; this is repeated until all line blocks have been processed.		LNEBLK		
RJ30	The length of the remote table is calculated and saved in register 3. Then the remote table is moved after the line table and the start address of the remote table is placed in register 2.		GNNR(IPW$DGN)	R3 R2	
RJ33	The pointer to the first remote block is saved and a check is made to see if punch and list routing are valid; if not, branch and link to..... > where message 1Q16I INVALID PUN/LST ROUTING FOR ID is printed. Then the routing is defaulted to central punch/list routing (X'00').	RJ55	RMPT PUNROU(RMTBLK) LSTROU(RMTBLK)	R2 R15	
RJ45	If no remote references are to be resolved, branch to..... > Otherwise (RMTREF = X'FF'), the last six bytes of the remote block specified by REF= are copied into the remote block that is being checked (this allows the copying of all information except routing from one remote block to another). Check if trace is required and include the appropriate trace information in the block.	I130	REFNUM(RMTBLK) REFINF(RMTBLK) TRACE RMTBLK+4	R4 R5	

Labels	Chart LA05: IPW$$I1 - Initiator, Phase 1		Modified Data Fields	Reg. Usage	Calls
I130	This routine validates the partition size values. The address of the first module is loaded into register 13, the number of modules into register 12. The following steps are taken to calculate the required pageable area size in register 2: • Set aside 2K for the permanent command processor work area plus 2K for module alignment. • Load the CI directory entries of all POWER/VS phases using the Local Directory List, and calculate their combined phase length. This is done in the routine..... >	LOAD	LDSL	R13 R12 R2	
	• Optionally, add the length of the following phases: - IPW$$TM if RJE,BSC is supported (GNNL ≠ X'00'). - IPW$$GA, IPW$$PA, and IPW$$SA if job accounting is included (GNJA = C'A'). - IPW$$SL if SLI support is requested (GNSL ≠ C' '). - IPW$$SM if spool management support is requested. (GNSP=C'S'). - IPW$$SN, IPW$$LN, IPW$$LF, IPW$$MP, IPW$$MD, IPW$$IB, and IPW$$OB if RJE,SNA is supported (GNSR ≠ X'00'). - Reader exit phase if RDREXIT is specified (GNRE ≠ 8X'40').				
I155	Next the real partition size is calculated in register 4. • Determine from the boundary box (register 3) the real partition size. • If 'realsize'(in register 4) is more than 128K it is set to 128K.			R4 R3 R4	
I160	• If 'realsize' is less than 10K, go to..... > where error message 1Q03I INSUFFICIENT REAL STORAGE is printed and the partition is canceled.	I1CN			
I165	Calculate the size of the actual pageable area in register 4 by subtracting the 'realsize' value (or, the value of the end of IPW$$I1 minus the start of the partition, whichever is greater) from the virtual partition size. If this size is less than the required size calculated and stored in register 2, go to..... > where message 1Q05I PAGEABLE AREA TOO SMALL is printed and the partition is canceled.	I1CN		R4 R2	
I170	Set the load address for the initialization phase 2 (IPW$$I2) to the start of the pageable area + 2K (for CP work area) and load phase 2. (Control is passed via register 1.)			R4 R1	IPW$$I2 Chart LB

Labels	Chart LA06: IPW$$I1 - Initiator, Phase 1		Modified Data Fields	Reg. Usage	Calls
	Subroutines				
	Print Remote ID Routine - Prepare for Printing REMID				
RJ55 RJ58	Get the correct remote ID into register 4 and place it in the REMID field in printable format. Strip this field of leading zeros, if present.		RMID	R4	
RJ60	Get the address of message 1Q16I and branch and link to..... > Then return to the calling routine.	I1MS			
	Cancel Routine - Print Message and Cancel				
I1CN	Branch and link to..... > to get the proper message printed and then issue the CANCEL macro to cancel the job or partition.	I1MS			
	Print Message Routine - Print an Error Message				
I1MS	Load the address of the console CCW into register 3 and create a CCW string. Get the console CCB address in register 1 and write the message to the console. Then return to the calling routine.			R3 R1	
	Load Routine - Load POWER/VS Phase Headers				
	Register usage for this routine: 3: **** - phase name address 4: LDDS - local directory list address 5: **** - length of phase loaded 14: **** - return register.			R3 R4 R5 R14	
LOAD	The phasename address is loaded into register 3.			R3	
	A LOAD macro is issued and a test is made to see if the phase is found (LDSW ≠ X'04'). If the phase is not found, control is passed to..... > where message 1Q15I PHASE xxxxxxxx NOT FOUND is printed and the partition is canceled.	I1CN			
I1FN	Otherwise, the number of text blocks is loaded into register 5. This value is multiplied by the block length and the length of the last block is then added. The local directory list is reset and control is returned to the calling routine.		GENL	R5	

CHART LB: IPW$$I2 - INITIATOR/TERMINATOR (25 PARTS)

Chart LB00: IPW$$I2 - Initiator/Terminator, General Flow and Macro Calls



Labels	Chart LB01: IPW$$I2 - Initiator/Terminator Initiator, Phase 2		Modified Data Fields	Reg. Usage	Calls
I2BD	The first 16 bytes constitute the section descriptor: 'I2CS V10M0 ' Entry to this routine is from IPW$$I1. Register usage 0: **** - Parameter and work register 1: **** - Parameter and work register 2: **** - Work register 3: **** - Work register 4: PHDS - DTFPH address 5: IPW$DQC - DMB address 6: IPW$DMC - Address of POWER/VS generation table (on entry); next MCB address 7: **** - Work register 8: GNCB - Base register 1 9: **** - Base register 2 10: IPW$DPA - Virtual start of POWER/VS partition (passed from IPW$$I1) 11: IPW$DTC - Task control address register 12: IPW$DTC - Asynchronous address register 13: IPW$DSV - Save area register 14: **** - Link register 15: **** - Work register.			R0 R1 R2 R3 R4 R5 R6 R7 R8 R9 R10 R11 R12 R13 R14 R15	
I216	Move non-RJE part of the generation table (pointed to be register 6) into IPW$$I2. If RJE is supported (GNNL = X'00'), branch and link to..... > where IPW$$TM is loaded. IPW$$TM is given control at label INIT; it moves the RJE part of the generation table into its storage area. Control is then returned to IPW$$I2 where the address of tne next phase is saved from register 2 and, if it was loaded, the address of IPW$$TM.	LOAD	GNCB(IPW$DGN) NEXT RTAM	R6 R2	IPW$$TM Chart MA

Labels	Chart LB02: IPW$$I2 - Initiator/Terminator Initiator, Phase 2		Modified Data Fields	Reg. Usage	Calls
I220	Load the POWER/VS nucleus (IPW$$NU) into real storage via a branch and link to..... >	LOAD			
	Save the end address of the POWER/VS nucleus (from register 2) as the start address of the fixable (real) area. Save the start address of the virtual POWER/VS partition. From these values calculate and save the number of fixable pages (maximum = current) and save the begin address to be fixed and the length of the fixed area. Then PFIX all required pages (addresses in register 2).		PAFA PAPA NRPM,NRPC PFIX	 R2	
I224	With the use of the relocation factor in register 5 (from IPW$$NU - CARL), the constants (their total number is in register 4) in the CAT (Control Address Table) are now relocated (note that the Relocating Loader is not a POWER/VS prerequisite).			R5 R4	
	Next the following fields are initiated:				
	• The page real storage address (from register 0)		PCRA	R0	
	• The page virtual storage address (from register 3)		PCVA	R3	
	• The page control byte address (from register 2)		PCPC	R2	
	• The first buffer address (from register 2)		PCFA	R2	
	• The address of the first fixed page (from register 3)		SCFP(IPW$DSC)	R3	
	• The address of the last permanent page (from register 3).		SCLP(IPW$DSC)	R3	
	With the use of the boundary box and the BG communications region the following values are set up:				
	• Start address of the LTA in the CAT (from register 2)		PALS	R2	
	• End address of the LTA+1 in the CAT (from register 2)		PALE	R2	
	• Address of POWER/VS PIB (from register 3)		CAPB	R3	
	• End address of POWER/VS partition+1 in the CAT (from register 4)		PAEN	 R4	
	• Number of real pages in the statistical information area (from register 4)		NRPG	 R4	
	• Start address of the pageable area in the CAT (from register 4).		PAVA	R4	

Labels	Chart LB03: IPW$$I2 - Initiator/Terminator Initiator, Phase 2		Modified Data Fields	Reg. Usage	Calls
I229	The storage control block is initialized by setting all available fixable pages to zero and the last page indicator in the storage assignment table to (X'40'), as well as the 'first page fixed' indicator (X'80').		SCBT		
I230	The values in the message control block (MMCB) are initialized.		MMRD,MMWT,MMCA (all IPW$DMM)	R2	
I232	All POWER/VS modules, whose names are moved to the Local Directory List pointed to by register 4, are loaded in routine..... >	LOAD	LDSL(LDDS)	R4	
	then their names are replaced by the load addresses found.		CAFM		
	For RJE,BSC, the highest remote ID and number of lines are saved as well as the load address.		NRRE NRLI CATM		
	For the other, optional, POWER/VS modules load addresses are supplied as required. Actual loading is performed by..... >	LOAD	CAPA,CAGA, CASA,CASL CARE		
I244	If spool management not specified (GNSP≠C'S'), branch to..... >	I248			
	Define POWER/VS cross-partition XECB's ICXP and SMXP (SVC92)		ICTA SMTA	R0,R1, R14, R15	
I245	If definition failed, set GNSP-C'M', branch to..... >	I248			
I246	Load spool management phase IPW$$SM >	LOAD	CASF		

Labels	Chart LB03.1: Initiator/Terminator Initiator, Phase 2	Modified Data Fields	Reg. Usage	Calls
I248	$$BPOWIN is called to set the PSW key to zero (to allow updates to supervisor control blocks) and POWER/VS active is indicated in SYSCOM (IJBFLG03 = X'04'). Next the POWER/VS partition indicator is set in the COMREG (POWFLG1 = X'20').	IJBFLG03 POWFLG1		$$BPOWIN Chart LD
	Set up the initial Task Control Block (TCB, address in register 11) chain, which includes: • The wait TCB (set to W=always wait state). • The initiator/terminator TCB (set to D=dispatchable) • The command processor TCB (set to I=awaiting posting).	R11 TCSF(IPW$DTC) TCTN(IPW$DTC) TCTP(IPW$DTC)		
	Then the base register is set up for the second part of IPW$$I2 (which will run as a subtask) and saved, as well as the address of the first instruction of that part. Last the current = maximum number of tasks (2) is saved in the applicable areas and control is passed to task selection in the nucleus	TCR8(IPW$DTC) TCRC(IPW$DTC) NRTC,NRTH	R2	IPW$$NU Chart AA

Labels	Chart LB04: IPW$$I2 - Initiator/Terminator Initiator, Phase 2		Modified Data Fields	Reg. Usage	Calls
	Note: Task selection will find the initiator/terminator to be the only dispatchabel task. Thus control is given back to the initiator/ terminator at label TASK and initialization continues from there on as a POWER/VS subtask.				
TASK I252	Addressability of the save area (in register 13) and the second base register (register 9) are set up; all addresses of subroutines are relocated with the relocation factor calculated in register 5.			R13 R9 R5	
I256	If SYSIPT is not assigned (LUBXXXP get X'FE'), a branch is made to.... >	I2L1		R2	
	Otherwise, the input file, pointed to by register 4, is opened and the first input card is read. If this card is not the FORMAT= statement, error message 1Q13I ERRONEOUS AUTOSTART CARD(S) READ is issued and processing continues at label..... >	I2L1		R4	IPW$WFC Chart AA
	If a FORMAT statement was read, it is moved to a save area and the next record is read.		LGIN		
I2L1	If no FORMAT statement was supplied or if SYSIPT was not assigned, message 1Q11D FORMAT QUEUES= is printed to which the operator must reply NO or END/EOB, A, D, or Q. Processing continues with the values thus obtained.		LGIN		
I2FC	A check is made to see what FORMAT specifications were given. Action taken is: A - account file is flagged for output Q - queue file is flagged for output D - data and queue file are flagged for output NO or no data - no action. A combination of these actions may occur. If an invalid character is found or if commas are missing, message 1Q11D is reissued and the output flags are reset.		 PHOP(IJAFILE) PHOP(IJQFILE) PHOP(IJDFILE) PHOP(IJQFILE)		
I260	The basic CCW chain is set up and initialized (using registers 2 and 3). This chain is common to data, queue, and account file handling during initialization.		CWDA(IJW$DCW)	R2,R3	

Labels	Chart LB05: IPW$$I2 - Initiator/Terminator Initiator, Phase 2		Modified Data Fields	Reg. Usage	Calls
QF00	Start of Queue File Processing				
	Register 4 contains the address of the queue file DTF.			R4	
	Register 5 contains the Disk Management Block (DMB) address.			R5	
	Branch and link to the..... > subroutine to set up the queue file DTFPH. Upon return from this routine register 2 points to the device table entry of the related DASD, and the DTF device type code is supplied.	SD00		R2	
	The queue file is opened. If RPS is supported (DTFPH switch byte set to X'40'), a SET SECTOR CCW is included in the chain and the queue file sector table values are set up.				
QF08	All device-dependent values are set in the disk management block (DMB) using register 3.		QC#R,QC#T,QCMA, QCMV,QCQW,QCQA, QCQV (all IPW$DQC)	R3	
	If this is a warm start (PHOP = X'80'), branch to..... >	QF12			
	Otherwise, a master skeleton record is built; then branch to..... >	QF16	MRQF(IPW$DQC)		
QF12	Read a master record and test its version/modification level. In the case of a mismatch, branch to..... > where message 1Q04I QUEUE FILE MISMATCH is printed and the partition is canceled.	I2CN			
QF16	The master record common part is now set up. Values initiated are:				
	• date		MRDY(IPW$DQC)		
	• time		MRST(IPW$DQC)		
	• sublibrary		MRSL(IPW$DQC)		
	• account option		MRJA(IPW$DQC)		
	• default priority		MRPY(IPW$DQC)		
	• job log option		MRLG(IPW$DQC)		
	• master list values		MRLV(IPW$DQC)		
	• master punch values.		MRPV(IPW$DQC)		
I2WQ	The Following Routines are for Warm Start Only				
	In the case of cold start, continue at label..... >	I2DF			
	If this is a normal warm start (the POWER/VS session before this one was ended by a valid PEND command, thus the POWER/VS termination status field was set to normal, MRTT = C'N').				

Labels	Chart LB06: IPW$$I2 - Initiator/Terminator Initiator, Phase 2		Modified Data Fields	Reg. Usage	Calls
WQ00	Set master record seek address in the queue record seek address and update the record counter via a branch and link to..... >	UPDT			
	All records in the queue file are read, the master class list in the permanent area is built by..... >	I1CL			
	and the counts of free queue records, records in use, and number of tracks in use are initialized using register 2. First and last-in-class queue records are identified.		NRTM,NRQR NRQF,NRQM	R2	
I2AB	This is the routine for abnormal warm start (previous POWER/VS session not ended by PEND command, MRTT = C'A').				
	The queue record addresses are updated via a branch and link to... >	UPDT			
	and then the queue file is read. If the record read is the last queue record (queue record identifier QCQI = C'D'), go to..... >	AB52			IPW$RDQ Cnart AE
	Queue entries that were not added to the file are written back.				IPW$WTQ Cnart AE
AB16	If these unadded records are moreover incomplete, they are deleted from the file, and a free queue record is indicated (QCQI = C'F').		QCQI(IPW$DQC)		
AB20	Forward and backward pointers are checked and, if incorrect, set to the proper values.		QCQP(IPW$DQC) QCQN(IPW$DQC)		
AB52	The queue file is scanned sequentially for free records. When these are found, the free chain is built. The file is then processed again at label..... >	I2WQ			
I2DF	Start of Data File Processing				
	Register 4 contains the address of the data file DTFPH.			R4	
	Register 5 contains the address of the DMB.			R5	
	Set up the data file DTFPH in the.. > subroutine. Upon normal return from this routine register 2 points to the device address table of the related DASD; the device characteristics of the DTF are saved. The number of tracks per cylinder in the data file is saved in the DMB and the next module address in register 3 is set.	SD00	DFCB DC#T(IPW$DQC) MOCU	R2 R3	

Labels	Chart LB07: IPW$$I2 - Initiator/Terminator Initiator, Phase 2		Modified Data Fields	Reg. Usage	Calls
DF00	Calculate in and store from register 1 DBLK values in master record and block length table. Also store (from register 3) DBLK+8 (or, if not a multiple of 32, the next higher multiple) in the block length table. Calculate and store the number of records per track.		MRDB(IPW$DQC) BLDB BLBF DC#R(IPW$DQC)	R1 R3	
	The data file (DTFPH) pointed to by register 4, is opened and a check is made for RPS (DTFPH switch by PHRP is X'80').			R4	
DF13	The created MCBs are checked if more than one extent on a volume is defined. The symbolic units must be the same. If this is not true, message 1Q19I is given and initialization is canceled.				
DF16	Otherwise, if RPS is supported, a SET SECTOR CCW is included in the chain and the data file sector table values (from register 3) are set up (record number in register 7).		DCSC(IPW$DQC)	R3 R7	
DF20	If this is a warm start data file, branch to..... >	DF48			
	Otherwise, the number of tracks is zeroed and register 7 is pointed to the first MCB.		NRTM	R7	
DF24	Format Routine for Cold Start Data Files				
	A channel program is built which writes a track of records, each track (DBLK) bytes long. This channel program is issued for each track on all extents of the data file.		PHCB(PHDS)		IPW$WFC Chart AA
DF48	Get the track group size; if this is cold start, branch to..... >	DF52			
	For warm start, calculate the number of track groups per cylinder and save this value; then branch to..... >	DF76	NRGC		
DF52	The cold start track group size is loaded into register 1. The number of track groups/cylinder calculated in such a way that a maximum spooling advantage i s obtained (1 to 1 ratio queue and data file if possible). This value is checked against the tracks/cylinder value of the device and the TRACKGP= specification. Depending on the values calculated, one or two messages may be printed using message management service: • 1Q17I QUEUE FILE TOO SMALL and/or • 1Q09I TRACK GROUP CHANGED TO nn		NRTG	R1	IPW$WTO Chart AD

Labels	Chart LB08: IPW$$I2 - Initiator/Terminator Initiator, Phase 2		Modified Data Fields	Reg. Usage	Calls
DF76	Build the track group control table, one byte per track (using register 3). The byte contains (head address+1) of the corresponding track or, for last-in-track group or non-available tracks, X'00'.		DCTR(IPW$DQC)	R3	
I2FQ	Queue File Formatting Routine Register 4 contains the address of the queue file DTFPH.			R4	
	On cold start the queue file is formatted. In the case of warm start (PHOP=X'80'), branch to..... >	FQ36			
	A CCW chain is set up and the master record, containing a pointer to the first record in the 'free' queue, is written. Then the CCW is updated and the queue record count field is incremented by 1.		SCCHH,SR,SECV QRCT		IPW$WFC Chart AA
FQ04	Process all track groups and set last to X'00'. Then check if total number of queue groups in register 2 (NRQR) is acceptable; if not, set it to the value of track groups (NTRG).		TRCT	R2	
FQ12	Store the queue group value (from register 2) in all applicable fields. Update the queue file disk address through the..... > subroutine and deduct one from the number of records to be written. When this value is zero, the forward pointer is reset.	UPDT	NRQR,NRQF,NRQW CUQN	R2	
FQ20	All track groups are subsequently inspected. The order in which this is done is by taking track groups in turn from each of the data file extents, if more than one extent is available. The cylinder number is updated and the addresses of both unfilled and filled modules are saved (via register 2). When a module is filled, it is flagged with X'FF' in the first byte of its disk address.		QCDF(IPW$DQC) MOCU MONT(MODS)	R2	

Labels	Chart LB09: IPW$$I2 - Initiator/Terminator Initiator, Phase 2		Modified Data Fields	Reg. Usage	Calls
FQ32	The forward pointer is inserted in the 'next record in set' field. Then the queue record is written, the queue file disk address is updated in the..... >	UPDT	QCNS(IPW$DQC)		IPW$WFC Chart AA
	subroutine, and the record number in the count field is updated. When all queue records are written, a dummy last record (QCQI = C'D') is written.		QRCT+4 QCQI		IPW$WFC Chart AA
I2AF	Account File Processing				
	Register 4 contains the address of the account file DTFPH.			R4	
	Register 5 contains the ACB address.			R5	
	The ACB address in the file control block and the MCB address in the account file DTFPH are set to 0.		CAAC CAA0		
	If no accounting support is specified (account switch in the generation table not set to C'A'), control is passed to label..... >	I2SL			
	The account file DTFPH is set up by the..... > subroutine. The device characteristics are saved. Then storage is obtained for the ACB; if not enough real storage is available, exit to..... >	SD00 NOROOM	AFCB		IPW$RSW Chart AC
	The ACB address is stored in the Control Address Table (CAT) and a pointer (in register 7) is set up to the device characteristics table. Then the account file is opened and the ACB is formatted via a branch and link to..... >	AC00		R7	

Labels	Chart LB10: IPW$$I2 - Initiator/Terminator Initiator, Phase 2		Modified Data Fields	Reg. Usage	Calls
	If this is a cold start account file (PHOP byte in DTFPH set to X'80'), the account file is erased and an EOF record is written as the first record on each track. The 'accept I/O errors' bit in the CCB is set on (ACCM = X'10') and control is passed to..... >	I2SL	ACCM(IPW$DAC)		
AF04	This routine is for warm start only. If RPS is supported (PHRP = X'40'), the SET SECTOR CCW and the READ SECTOR CCW are relocated.				
AF08	In the case of abnormal warm start (MRTT = C'A'), a branch is made to..... >	AF12			
	Otherwise, the disk address of the last record is copied to the ACB and the seek bucket. In register 7 the number of full tracks used is calculated; then branch to..... >	AF36	ACSA(IPW$DAC) SADR	R7	
AF12	The CCW chain is initialized, the track counter (register 7) and the sector value bucket are cleared and reads without data transfer are performed. These dummy reads are used to find EOF records on the tracks. After each track has been scanned, the track counter is updated. The CCB is tested for unit exception (CCBSTA1 = X'01') after each read. If an EOF record is found, branch to..... >	AF24	ACSE(IPW$DAC) SCCHH	R7	IPW$WFC Chart AA
	If no EOF record is found and the upper limit is reached, branch to..... >	AF36			
AF24	If the file is empty (SCCHH = ACL0), branch to..... >	AF52			
	Otherwise, the head address is repositioned one track backwards in the seek/search bucket.		SHH		
AF36	Use register 6 to calculate the new residual file capacity, except the last track, and store this value in the ACB.		ACAC(IPW$DAC)	R6	
AF40	The count field address is loaded into register 2, the CCWs are initialized, and the count field of record 1 is read.			R2	IPW$WFC Chart AA
AF44	If the data length of this record is larger than the maximum record size in the master record, then update the master record.		MRAZ(IPW$DQC)		

Labels	Chart LB11: IPW$$I2 - Initiator/Terminator Initiator, Phase 2		Modified Data Fields	Reg. Usage	Calls
	Using registers 3, 6, and 7, the new residual track capacity is calculated and stored; next a read count and data is issued. If record 1 is hit again (no EOF is found), branch to..... >	AF56	ACLC(IPW$DAC)	R3,R6 R7	
	If not, reading continues looping via AF44 until the EOF record (data length of 0) is hit. When hit, it is erased. This allows the file to be updated again.				
AF52	The count field in the ACB and the seek addresses in the ACB and DMB are cleared and reset to the begin address (lower limit).		ACCF(IPW$DAC) ACSA(IPW$DAC)		
	The maximum record size in the master record, which was defaulted to 2008, is reset to 1.		MRAZ(IPW$DQC)		
AF56	If required, the pointer to the current last record is saved into the ACB.		ACSA(IPW$DAC)		
AF60	Update the master record with the current account file address and post the CCB to accept I/O errors.		MRAS(IPW$DQC) ACCM(IPW$DAC)		
I2SL	SLI Initialization Routine				
	Register 4 contains the address of the library file DTFPH Register 5 contains the address of the DMB.			R4 R5	
	Clear the addresses of the MCB SSLs. If no SLI support is required (MRSL = X'40'), branch to..... >	I2FN	CAL7,CAL8		
	Otherwise, the private SSL DTFPH is set up in routine..... >	SD00			
	If SYSSLB is assigned (R2 ≠ 0), the library open switches are set, the library is opened, and the disk address of the private SSL is moved to bucket 1.		BUK1	R2	

Labels	Chart LB11.1: IPW$$I2 - Initiator/Terminator Initiator, Phase 2		Modified Data Fields	Reg. Usage	Calls
SL04	The parameter list address is loaded into register 2 and the address of $$BOPNLB into register 1; then control is passed to $$BOPNLB to open the system SSL. On return from $$BOPNLB the disk address of the system SSL is in bucket 2.		BUK2	R2 R1	$$BOPNLB
	If the system SSL does not exist then branch to..... >	SL05			
	Storage is reserved for the system SSL. If not enough space is available (R0=0), branch to..... >	NOROOM		R0	IPW$RSW Chart AC
	Otherwise, the MCB address is saved in register 1 and copied to register 6. Then the IJSSLB MCB is formatted in routine..... >	FS00		R1,R6	

Labels	Chart LB12: IPW$$I2 - Initiator/Terminator Initiator, Phase 2		Modified Data Fields	Reg. Usage	Calls
SL05	Set the extent lower limit, load LUB-INDEX of SYSRES in R4 and point register 2 to the device table. Then format the SSL fields for the DMB in routine..... >	SQ00	MCLO(IPW$DMC)	R4 R2	
	If the private SSL is not open (LBSW=X'01'), branch to..... >	I2FN			
	Otherwise go through the same steps as for the system SSL.				
I2FN	Final Non-Autostart Processing				
	Real work space is reserved for two buffers and two TCBs; the required buffer length was calculated at DF00. If no work space can be found (register 0 = 0 on return from IPW$RSW), an error message is printed and the job is canceled by..... >	NOROOM			IPW$RSW Chart AC
	If work space can be obtained for all these areas, the work space is released again.				IPW$RLW Chart AC
	The queue file DTFPH address is loaded into register 4 and, if this is a warm start queue file (PHOP = X'80'), the status report is printed by..... >	PS00		R4	
FN00	The address of the page fault appendage exit is established via register 1, the terminator status indicator is set to abnormal (MRTT = C'A'), a master record is written and the 'return open failure' bit is set in the COMREG (JCSW1 = X'40').		MRTT JCSW1	R1	IPW$WTQ Chart AE
	Last, the task selection field in the task control block is set to indicate an inactive task.		TCDF(IPW$DTC)		
	The following addresses are set to zero: • Line manager TCB • Message control block • SNA control block • General purpose work area		TALM(IPW$DPA) CARM(IPW$DPA) CASM(IPW$DPA) CAGP(IPW$DPA)		
	If RJE,BSC is not supported (GNNL = X'00'), branch to..... >	I270			
	Otherwise, the appendage indicator is set in the POWER/VS PIB (PIBPUBAS = X'40'). Next, the length of the line manager TCB is obtained in register 1 and space is reserved for the line manager. If no space is available, branch to..... >	NOROOM	PIBPUBAS	R1	IPW$RSW Chart AC

Labels	Chart LB13: IPW$$I2 - Initiator/Terminator Initiator, Phase 2		Modified Data Fields	Reg. Usage	Calls
	Otherwise, the address of the line manager TCB is saved, the task ID is set, and the address of the line manager save area (located in the LMF part of the TCB) is placed in the task register save area via register 3. Then the entry point of the line manager (address of IPW$$TM+16) is loaded into register 3 and the line manager is attached.		TALM(IPW$DPA) TCTI(IPW$DTC) TCRD(IPW$DTC)	R3 R3	IPW$ATT Chart AA
I270	If RJE,SNA is not supported (GNSR = X'00'), branch to..... >	I275			
	Otherwise, the length of the SNA control block is obtained in register 1, and space is reserved for the SNA control block. If no space is available, branch to..... >	NOROOM		R1	IPW$RSW Chart AC
	Otherwise, the address is saved.		CASM(IPW$DPA)		
	Next the SNA control block is initialized with the following fields: • Storage descriptor • Maximum number of SNA unit control blocks • Address of SNA remote control block • Address of SNA unit control block space • ACB • The address in the ACB which points to the APPLID is relocated to point to the user specified APPLID in the GNCB. • Address of logon space • Address of compaction table pool		 SNSD(IPW$DSN) SNSU(IPW$DSN) SNRM(IPW$DSN) SNSP(IPW$DSN) SNAC(IPW$DSN)		
	The proper address of the general purpose work area (lockword) is obtained in register 1 and saved.		CAGP(IPW$DPA)	R1	
	Processing is continued at..... >	I276			
I275	If RJE,BSC is not supported (GNNL = X'00'), branch to..... >	I290			
I276	The length of the message control block is obtained in register 1 and space is reserved for the message control block. If no space is available, branch to ..... >	NOROOM		R1	IPW$RSW Chart AC
	Otherwise, the address is saved. Addressability of the message control block is set up in register 1, and the message control block is initialized with the storage descriptor.		CARM(IPW$DPA) MSSD(IPW$DMS)	 R1	

Labels	Chart LB13.1: IPW$$I2 - Initiator/Terminator Initiator, Phase 2		Modified Data Fields	Reg. Usage	Calls
I290	If no spool management support (GNSP-C' '), branch to..... >	I2AS			
SP00	If definition of cross-partition XECB's failed (GNSP-C'M'), message IQ27I UNABLE TO INITIALIZE SPOOL MANAGEMENT, is printed and branch to>	I2AS		R0,R14, R15	IPW$WTO Chart AD
	Storage is obtained for spool manager master TCB: if not enough real storage is available, exit to..... >	NOROOM		R1	IPW$RSW Chart AC
	TCB is initialized: • Descriptor. • Task save area. • TCB address. • Address of cross-partition XECB's in wait list.		TCSD(IPW$DTC) TCRD(IPW$DTC) TCSV(IPW$DTC) TASP(IPW$DPA) ICWL(IPW$DPA) SPWL(IPW$DPA)		
	Load entry point to spool management phase into register 3.			R3	
	Attach the task.				IPW$ATT Chart AA
	Control is returned from IPW$$SM with master spool management TCB in 'Q' state.				

Labels	Chart LB14: IPW$$I2 - Initiator/Terminator Initiator, Phase 2		Modified Data Fields	Reg. Usage	Calls
I2AS	AUTOSTART Code				
	If the card input area is empty, no (more) autostart processing is required and control is passed to.. >	AS12			
	Otherwise, message 1Q20I AUTOSTART IN PROGRESS is printed using message management service; register 4 is pointed to the SYSIPT CCB and			R4	
	register 3 is used to update the EOF address in the DTFDI. The interface			R3	
	with the logical reader is set up and		CINP+65		IPW$OLI
	an * $$ JOB statement is written onto				Chart AC
	disk using registers 0 and 1,			R0,R1	IPW$PLR
	followed by all autostart statements				Chart JC
	read.				IPW$WFC
					Chart AA
AS08	At EOF on the reader, an * $$ EOJ statement is written, followed by a				IPW$PLR
	null-record (to indicate end-of-physical-read-function) the input				Chart JC
	file is closed and the logical reader interface is closed.				IPW$CLI
					Chart AC
AS12	The system logical units are unassigned (number of units to be				IPW$ULP
	unassigned in register 2, POWER/VS pib address in register 1, address of			R2 R1	Chart GC
	logical unit in register 3). If no autostart was processed (CDIN is			R3	
	empty), branch to..... >	I299			
	Otherwise, a dummy class is indicated in the task class list (TCCT =				
	X'5CxxxxFF') and the interface with		TCCT(IPW$DTC)		
	the logical list is set up.				IPW$OLI
					Chart AC
AS20	All autostart statements are now read back. If a PSTART or S command is				IPW$GLR
	found, it is processed by passing it to the command processor. If the				Chart HB
	autostart statement read results in a request for activation of the				IPW$ICP
	command processor (register 7 ≠ 0),				Chart GD
	branch to..... >	AS32			
	All other statements are read until end-of-data is found (CDIN blank),				
	which causes a branch to..... >	AS48			

Labels	Chart LB15: IPW$$I2 - Initiator/Terminator Initiator, Phase 2		Modified Data Fields	Reg. Usage	Calls
AS32	Command Processor Interface				
	Read a data record. If any of the statements READER=, PRINTERS=, or PUNCHES= is found and they contain valid specifications, these statements are moved into the command processor control block (CPB). In all other cases a dummy statement (null record) is put in this area. After each update of the CPB the command processor TCB is activated (TCEB+2 = X'80').		CPOP(IPW$DTC) TCEB(IPW$DTC)		IPW$GLR Chart HB
AS48	The termination type in the initiator/terminator TCB is set to 'immediate stop' (TCTT = C'S'). The end-of-job statement is read and the interface with the logical list is closed. After successful completion, the terminator type is reset to 'normal' (TCTT = C' ').		TCTT(IPW$DTC) TCTT(IPW$DTC)		IPW$GLR Chart HB IPW$CLI Chart AC
I299	Message 1Q12I POWER/VS INITIALIZATION COMPLETE is printed using message management service. A wait multiple list is built in the task work area, pointing to the CCB's of all data file extents.		TCGW(IPW$DTC)		IPW$WTO Chart AD IPW$WFI Chart AA
I29C	If the termination byte is 'E', branch to..... >	T100			
	A wait multiple is issued to scan for double buffering posting.				IPW$WFQ Chart AA
	Double buffering is scanned by..... >	DB00			
	Branch to..... >	I29C			
	Termination Routine				
T100	If a POWER/VS supported partition is in DOS/VS 'stopped' state, issue message 1Q25I.				IPW$WTO Chart AD
TC0A	If all tasks are detached (NRTC=2), branch to..... >	T104			
	Clear out ECB at TCGW.		TCGW(IPW$DTC)		
	Wait multiple on double buffering or termination posting.				IPW$WFQ Chart AA
	Double buffering is scanned by..... >	DB00			
	Branch to..... >	TC0,			
T104	The status report is printed by.... >	PS00			
	POWER/VS accounting record is written by..... >	PX00			

Labels	Chart LB15.1: IPW$$I2 - Initiator/Terminator Initiator, Phase 2		Modified Data Fields	Reg. Usage	Calls
	A null record is written to indicate EOF to accounting. Termination status is set to normal (NRTT='N'). The master record is written and POWER/VS files are closed.				
T108	Termination message 1Q21I POWER/VS HAS BEEN TERMINATED is written using message management service and the job is ended by an SVC 14.				IPW$WTO Chart AD
	Double Buffering Scan Routine				
DB00	Point to ECB list minus 4.				
DB02	Point to next in list.				
	If end of list, then return.			RE	
	Is CCB posted complete? If no, branch..... >DB02				
	Otherwise is MCB still owned?				
	If no, branch..... >	DB02			
	Is TCB posted with C'2'? If no, branch..... DB02				
	Is WLR or unrecoverable I/O error set? If so, set task to be dispatchable to termination routine.				
	Otherwise, unpost CCB,		MCCM(IPW$DMC)		
	Set TCDB off.		TCDB(IPW$DTC)		
	Post TCED.		TCED(IPW$DTC)		
	Free MCB.		MCLK(IPW$DMC)		
	If operation was not write, branch to..... >	DB02			
	Free data space. Branch to..... >	DB02			PW$RLW Chart AC
	POWER/VS Accounting Record Routine				
PX00	Gather necessary accounting information for POWER/VS partition accounting record from partition job accounting record				IPW$PAR Chart FA
	Close account file.				IPW$PAR Chart FA
	Return.			RE	

Labels	Chart LB16: IPW$$I2 - Initiator/Terminator Subroutines	Modified Data Fields	Reg. Usage	Calls
	Format Account Control Block Routine			
AC00	Registers used by this routine are:			
	1: work register		R1	
	2: address of DMB		R2	
	3: work register		R3	
	4: address of DTFPH		R4	
	5: ACB virtual address		R5	
	6: ACB real address		R6	
	7: pointer to device characteristics table		R7	
	14: return register		R14	
	15: base register for this routine.		R15	
	The ACB storage descriptor is initialized and next pertinent values are set in the ACB control field:	ACSD(IPW$DAC)		
	• Device address			
	• Maximum track capacity	ACMT(IPW$DAC)		
	• Residual track capacity	ACLC(IPW$DAC)		
	• Number of tracks per cylinder	AC#T(IPW$DAC)		
	• Tolerance	ACTL(IPW$DAC)		
	• Overhead	ACOH(IPW$DAC)		
	• PUB device type code	ACPB(IPW$DAC)		
	• DTFPH device type code.	ACDT(IPW$DAC)		
	The write/read channel program in the ACB is set up; for this purpose the CCW addresses are relocated.	ACSD(IPW$DAC) ACCH(IPW$DAC)	R3	
	If RPS is supported (DTFPH switch byte set to X'40'), the channel program is updated to include a SET SECTOR and READ SECTOR CCW.			
AC04	In register 3 the maximum file capacity is calculated. This value is stored as both maximum and residual capacity in the ACB. The 20% empty limit of the available account file capacity is then calculated from this value and stored in the ACB.		R3	
		ACMC(IPW$DAC) ACAC(IPW$DAC)		
		ACEC(IPW$DAC)		
	Format Module Control Block Routine			
FM00	Registers used by this routine are:			
	1: COMREG pointer		R1	
	2: PUB pointer		R2	
	3: LUB pointer		R3	
	4: DTFPH pointer		R4	
	5: work register		R5	
	6: MCB address		R6	
	7: pointer to extent information		R7	
	9: base register 2		R9	
	14: return register		R14	
	15: base register for this routine.		R15	

Labels	Chart LB17: IPW$$I2 - Initiator/Terminator Subroutines		Modified Data Fields	Reg. Usage	Calls
	The MCB is initialized and the partition COMREG is obtained. The SYSIR macro is issued to obtain PUB and LUB addresses. The DMB pointer is restored in register 5 and EXCP REAL is indicated in the MCB (MCLU = X'80'). Then the real MCB address is obtained (REALAD macro). The seek and TIC addresses are set to 'real' values, and the SET SECTOR CCW is also set to 'real' if RPS is supported (PHRP = X'40').		MCLO(IPW$DMC) MCLU(IPW$DMC) MCSK(IPW$DMC) MCSS(IPW$DMC)	R5	
FM04	In the same way the TIC, search, and CCW addresses are set to 'real' values. Control is then returned to the calling routine via return register 14.		MCSH,MCTI,MCCA (all IPW$DMC)	R14	
	Format MCB Source Statement Library File Routine				
FS00	Registers used by this routine are: 1: COMREG pointer 2: PUB pointer 3: LUB pointer 4: pointer to logical unit 5: DMB address 6: MCB address 14: return register 15: base register for this routine.			R1 R2 R3 R4 R5 R6 R14 R15	
	Indicate SSL file in the MCB (MCSD+4 = C'L') and get the partition COMREG. Issue SYSIR to obtain PUB and LUB addresses. Next the device address table in placed in the MCB and EXCP REAL is indicated (MCLU = X'80').		MCSD(IPW$DMC) MCSD,MCLU (BOTH IPW$DMC)		
	The real address of the MCB is obtained through the REALAD macro. This address, in register 0, is used to set the real seek and TIC addresses. If the device does not have RPS (SYSXXXJ = X'04'), branch to..... >	FS04	MCSK,MCSS (both IPW$DMC)	R0	
	Otherwise, the BG COMREG is obtained and the supervisor is tested for RPS support; if no RPS support is included (RMSROPEN ≠ X'01'), branch to..... >	FS04			
	The CCW for RPS support is relocated and the SLB sector table address is stored in the MCB.		MCSX(IPW$DMC)		

Labels	Chart LB18: IPW$$I2 - Initiator/Terminator Subroutines	Modified Data Fields	Reg. Usage	Calls
FS04	The search and TIC addresses are restored to their real values using registers 3 and 0. The SSL blocklength is stored in the read CCW and its address is loaded into the CCB. Then control is returned to the calling routine via register 14.	MCSH,MCTI (both IPW$DMC) MCRW+6(IPW$DMC) MCCA(IPW$DMC)	 R3,R0 R14	
	Account File Extent Exit Routine			
I2AX	Registers used by this routine are: 1: extent information pointer 5: ACB pointer The high and low order extent limits and the symbolic unit are moved into the ACB. Then the EXCP REAL indicator is set on in the ACB (ACLU = X'80') and control is returned via a LBRET 1 macro.	 ACLO,ACHI,ACLU (all IPW$DAC)	 R1 R5 	
	Construct Class List Routine			
I2CL	Registers used by this routine are: 2: work register 3: relative track and record number 6: module control block address 7: class table address 8: base register 1 9: base register 2. Move the seek address of the current entry, set up register 6 and calculate the relative head number of the current entry in register 3. Next in this register the relative record number is placed. The class table address pointer is set up in register 7. Depending on whether a read or list entry or neither is specified this pointer is incremented by no bytes (read), 128 bytes (list), or 256 bytes (punch entry) to contain the address of the proper table.	 SADR	 R2 R3 R6 R7 R8 R9 R6 R3 R7	
CL00	The class of the current queue entry is now converted to a displacement in the master class list. This displacement is added to the address in register 7. If the entry is first-in-class, its relative record number is stored in the respective data field. If the entry is last-in-class, its record is number stored in the last-in-class field and the 'live' indicator is set on (CTQL = X'80'). Control is then returned to the calling routine via register 14.	 CLSV CTQF(IPW$DCT) CTQL(IPW$DCT)	 R7 R14	

Labels	Chart LB19: IPW$$I2 - Initiator/Terminator Subroutines		Modified Data Fields	Reg. Usage	Calls
	Data File extent Exit Routine				
I2DX	Registers used by this routine are:				
	0: read address of work space			R0	
	1: virtual address of work space			R1	
	2: message address and work register			R2	
	3: work register			R3	
	4: DTFPH data file address			R4	
	6: module control block address			R6	
	7: extent information			R7	
	8: base register 1			R8	
	9: base register 2			R9	
	14: branch register			R14	
	15: work register.			R15	
	Registers 6 and 7 are initialized.			R6,R7	
	The current module number is stored in register 3. If this is the first extent, branch to..... >	DX04		R3	
	If more than 5 extents are present, the address of an error message is placed in register 2 and control is passed to..... >	I2CN		R2	
	Otherwise, space is reserved. If not enough space is available, branch to..... >	NOROOM			IPW$RSW Chart AC
	The address of the next MCB is stored, the skeleton data MCB is moved to the virtual work space and its addressability is established in register 6.		CAQ1	R6	
DX04	If the lower head limit is not zero (XTLL ≠ X'00'), or if the extent is not on a cylinder boundary, a message address is loaded into register 2; then branch to..... >	I2CN		R2	
DX12	Calculate the total number of cylinders and store this value from register 2 into the NCYL field.		NCYL	R2	
	Calculate the number of tracks and store this value. Update the module number in register 3 and store this value in the MNUM field and in the MCB. Then this value is converted and the MCB is formatted via a branch and link to..... >	FM00	NRTR MNUM MCSA(IPW$DMC)	R3	
	The count is saved in the read/write CCW.		MCRW(IPW$DMC)		
	Last the module is initialized and a pointer to the next module is saved. Control is then passed to OPEN via a LBRET 2 macro instruction.		MOCU		
	Messages from this routine:				
	• 1Q18I TOO MANY EXTENTS IJDFILE • 1Q19I INVALID EXTENT IJDFILE				

Labels	Chart LB20: IPW$$I2 - Initiator/Terminator Subroutines		Modified Data Fields	Reg. Usage	Calls
	Queue File Extent Exit Routine				
I2QX	Registers used by this routine are:				
	1: extent information pointer			R1	
	4: DTFPH queue file address			R4	
	5: disk management block address			R5	
	6: module control block address			R6	
	7: extent information pointer			R7	
	8: base register 1			R8	
	9: base register 2			R9	
	14: return register			R14	
	15: work register.			R15	
	Registers 6 and 7 are initialized to address the MCB and the extent information field, respectively. Then the MCB is formatted via a branch and link to..... >	FM00		R6,R7	
	The 'next queue entry pointer' is initialized to contain the extent lower limit and initial values are set in the master record seek address.		QCMW(IPW$DQC)		
	Now relative lower and upper limit are calculated in registers 2 and 3 and from these values the total number of queue entries is calculated in register 2 and then stored into the 'number of queue records' field. Control is returned to the calling routine via a LBRET 1 macro instruction.		NRQR	R2,R3 R2	
	Load Routine (Load phase specified in GENL local directory list)				
LOAD	Registers used by this routine are:				
	2: load address			R2	
	3: phase name address			R3	
	4: local directory list address			R4	
	5: length of loaded phase			R5	
	8: base register 1			R8	
	9: base register 2			R9	
	14: return register.			R14	
	Register 4 is based on the local directory list (LDDS) and the phasename address (LDPN) is loaded into register 3. Now the POWER/VS phase is loaded by means of the LOAD macro. The number of text blocks is loaded into register 5 (from field LDNT). Then the total phase length in bytes is calculated and stored in register 5. The LDL is reset to its initial value and control is returned to the calling routine.		GENL	R4 R3 R5 R5	

Labels	Chart LB21: IPW$$I2 - Initiator/Terminator Subroutines	Modified Data Fields	Reg. Usage	Calls
	Insufficient Real Storage and Cancel Routine			
NOROOM	Registers used by this routine are:			
	2: message address		R2	
	6: work register		R6	
	8: base register 1		R8	
	9: base register 2.		R9	
	The real size is loaded into register 6, converted into a multiple of 1,024, and unpacked into the message field. The address of message 1Q03I INSUFFICIENT REAL STORAGE ALLOCATED is loaded into register 2.	M03B	R6 R2	
I2CN	The message address is stored in the message request word and a message is issued on the console. Then the job is canceled.	TCMW(IPW$DTC)		IPW$WTO Chart AD
	Print Status Report Routine			
PS00	Registers used by this routine are:			
	1: communications region pointer		R1	
	2: PUB pointer		R2	
	3: LUB pointer		R3	
	4: work register		R4	
	7: pointer to SYSLST logical unit		R7	
	8: base register 1		R8	
	9: base register 2		R9	
	14: return register		R14	
	15: base register for this routine.		R15	
	Save the user registers and set up link to communications region. The SYSIR macro is issued to obtain the applicable PUB and LUB addresses. If SYSLST is not assigned, control is immediately returned to the calling routine. If SYSLST is assigned (LUBXXXP ≠ X'FE') and if it is a printer, the report can be printed.			IPW$SAV

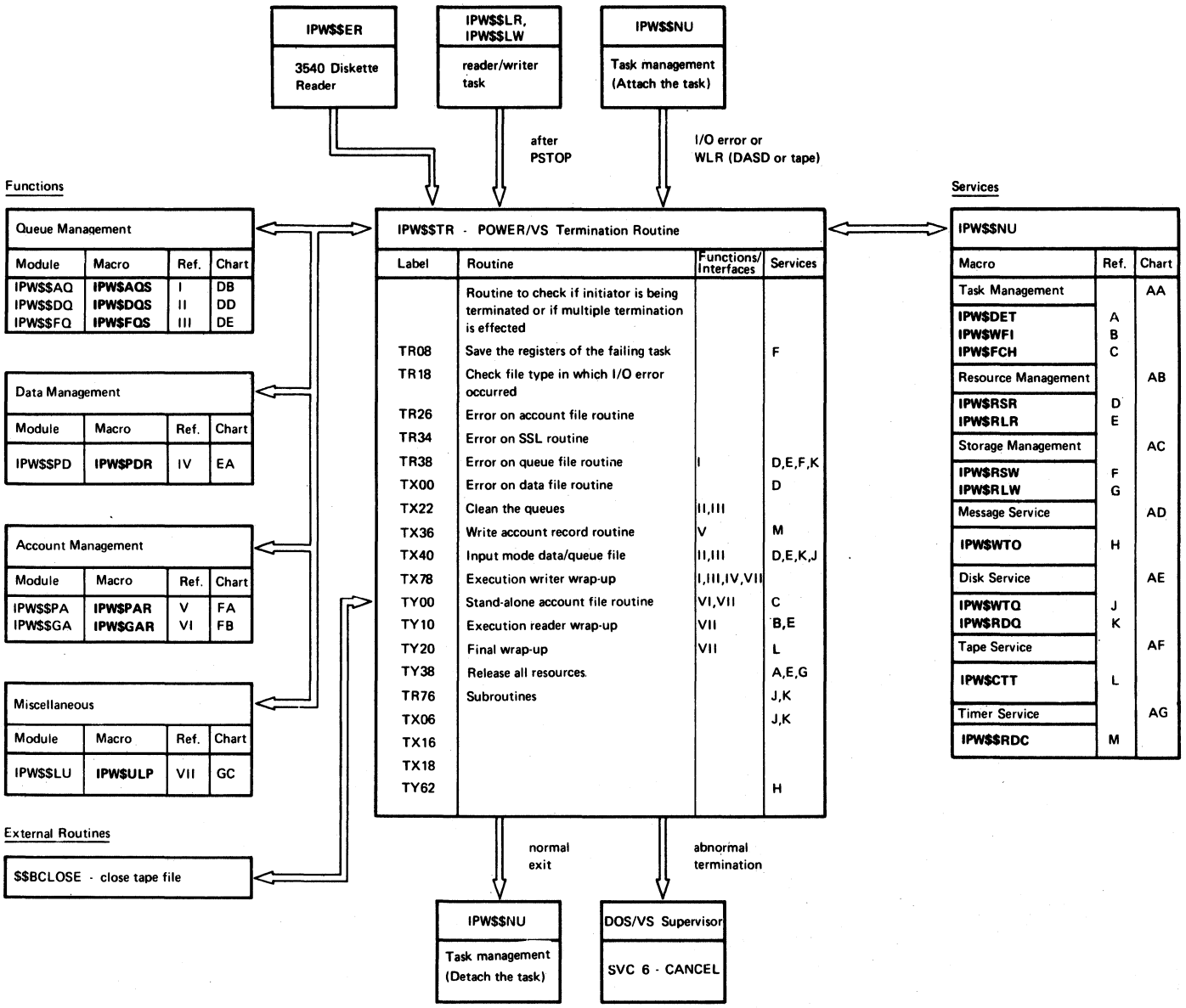
Labels	Chart LB22: IPW$$I2 - Initiator/Terminator Subroutines		Modified Data Fields	Reg. Usage	Calls
	The device address is unpacked in a work area. Register 4 points to the print CCB and the printer is skipped to a new page. The report header, containing the date and time, is printed. Next the following fields are printed: • IJQFILE header • Number of queue records • Number of free queue records • Maximum number of queue records used • IJDFILE header • Total number of tracks for the data file • Track group size • Data block size. If no account support is included, a line stating that is printed and control is passed to..... >	PS70	WORK FLD4 FLD7	R4	
PS50	If account support is included (MRJA = C'A'), the following information is now printed: • IJAFILE header • Total number of tracks for account file • Percentage of account file filled.				
PS70	Finally these fields are printed: • Real storage allocated to POWER/VS partition • Number of times tasks were waiting for real storage • Maximum number of real pages fixed • Maximum number of tasks active at one time. (Issue display command to let CP invoke the Print Status Task.) The save area address is restored and control is returned to the calling routine.		TCEB(IPW$DTC)		IPW$ICP Chart GD IPW$WFC Chart AA

Labels	Chart LB23: IPW$$I2 - Initiator/Terminator Subroutines		Modified Data Fields	Reg. Usage	Calls
	Set Up DTFPH Routine				
SD00	Registers used by this routine are:				
	2: device table address			R2	
	3: loop control register			R3	
	4: DTFPH address			R4	
	6: PUB pointer			R6	
	7: LUB pointer			R7	
	8: base register 1			R8	
	9: base register 2			R9	
	14: return register			R14	
	15: base register for this routine.			R15	
	The partition communications region is obtained via the COMREG macro. Then the SYSIR macro is issued to obtain the applicable PUB and LUB addresses. If the unit found is unassigned (LUBXXXP = X'FE') and if the assignment is not to SYSSLB, branch to..... >	SD12			
	If the assignment is to SYSSLB, register 2 is set to zero and control is returned via register 14.			R2 R14	
SD04	The device table address is saved in register 2 and the number of entries is loaded into register 3.			R2 R3	
SD08	The DTF device type code (DVPB) is checked against SYSXXXT until a match is found; the remaining number of entries is still in register 3. The DTF type code is moved into the DTFPH and control is returned via register 14.			R3 R14	
SD12	The DTFPH filename is moved into the message and 1Q07I INVALID LOGICAL UNIT XXXXXXXX is printed on the console using message management service; then the job is canceled.		M07B		IPW$WTO Chart AD
	Set Up DMB Fields for SSL Routine				
SQ00	Registers used by this routine are:				
	1: COMREG pointer			R1	
	2: device table address			R2	
	3: LUB pointer			R3	
	4: pointer to logical unit			R4	
	5: DMB address			R5	
	6: MCB address			R6	
	7: PUB pointer			R7	
	14: return register			R14	
	15: base register for this routine.			R15	
	Get the partition COMREG and the LUB and PUB addresses using the SYSIR macro.				

Labels	Chart LB24: IPW$$I2 - Initiator/Terminator Subroutines		Modified Data Fields	Reg. Usage	Calls
SQ04	Scan through the device table until the device with the proper device type code is found (DVPB = SYSXXXT).				
SQ08	The device characteristics are now moved into the SSL file device table and next the records/track and tracks/cylinder values are moved into the DMB. If the device is not an RPS device (MCSS ≠ X'23'), control is returned to the calling routine via register 14.		LFCB MCNR(IPW$DQC) MCLT(IPW$DQC)	R14	
SQ10	The sector value is set to zero and, using registers 1, 2, 3, and 7, the sector value is then calculated and stored in the DMB. This process continues until all records are handled and then control is passed to the calling routine via register 14.		SLSC(IPW$DQC) LF#MS	R1,R2 R3,R7 R14	
	Update CCHHR of Queue File Routine				
UPDT	Registers used by this routine are: 1: work register 7: address of CCHHR 8: base register 1 9: base register 2 14: return register.			R1 R7 R8 R9 R14	
	The current record number in register 2 is checked to see if it is the last on the track; if not, branch to.... >	UPD1		R2	
	The record number in the count field is set to 1 and, if the upper head is not reached, branch to..... >	UPD2	CTRN		
	Record, head, and cylinder number are updated and the cylinder number+1 is stored in the count field. Control is returned via register 14.		CTCC	R14	
UPD1	The record number if updated and stored in the count field. Control is returned via register 14.		CTRN	R14	
UPD2	The head number is updated and stored in the count field. Control is returned via register 14.		CTHH	R14	

CHART LC: IPW$$TR - TASK TERMINATOR (21 PARTS)

Chart LC00: IPW$$TR - Task Terminator, General Flow and Macro Calls



Labels	Chart LC01: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
	Entry to this routine is from any failing POWER/VS routine.				
TRSD	The first 16 bytes constitute the section descriptor:				
	'TRCS V10M0				
	Register usage:				
	0: **** - work register			R0	
	1: **** - work register			R1	
	2: **** - work register			R2	
	3: **** - work register			R3	
	4: IPW$DDE - device list entry pointer			R4	
	5: IPW$DQR - queue record pointer			R5	
	6: IPW$DQC - DMB pointer			R6	
	6: IPW$DPD - partition control block pointer				
	7: IPW$DCT - class table pointer			R7	
	8: **** - work register; link register			R8	
	9: **** - base register			R9	
	10: IPW$DPA - pointer to permanent area			R10	
	11: IPW$DTC - address of TCB			R11	
	12: **** - link register for task selection			R12	
	13: IPW$DSV - address of register save area			R13	
	14: **** - link register			R14	
	15: **** - function routine base register.			R15	
	Set up permanent area and TCB addressability in registers 10 and 11.			R10,R11	
TR04	If the failing task is the initiator task (IPW$$I2), the job is canceled (termination to be accomplished by $$BPOWIN). If the failing task is the command processor task, recovery is not possible, message 1Q82I I/O ERROR DURING TTT, POWER/VS TERMINATED is printed in the..... > subroutine, and the job is canceled.	TY70			
TR06	If this routine has been entered before (meaning that more than one unrecoverable I/O error has occurred), message 1Q75I MULTIPLE TERMINATION OF TASK, POWER/VS TERMINATED is printed in the..... > subroutine, and the job is canceled.	TY70			

Labels	Chart LC02: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
TR08	Addressability is established in register 9. The addressability of the save area is set up in register 2 and the failing task's registers 0 and 1 in SVR0 and SVR1.			R9 R2	
			SVR0(IPW$DSV) SVR1(IPW$DSV)		
	If the TCB is not owned by IPW$$SA (TCTI ≠ C' ACT'), branch to..... >	TR12			
	Otherwise, the account trace indicator in the failing task's TCB (label TCAT) is checked to see if any account function was active. No such function is active if the main routine had only just started, was in the 'active' state, or had just been ended at the time the failure occurred. In any of these cases, branch to..... >	TR10			
	Otherwise, (failure during open, get, or close function), branch to..... >	TR16			
TR10	If the task had output to spool files (TCGW+8 = 4C' '), check spool function activity at..... >	TR12			
	Otherwise, (the task's main routine active), branch to ..... >	TR14			
TR12	Now a check is made to see which spool function was being performed. If the function trace indicator in the TCB (label TCFT) indicates that the task has just been initiated after the logical end of the spool function or a 'ready for input' state during the spool function (TCFT = X'00', X'40', C'O', C'E', or C'I'), branch to..... >	TR14			
	Otherwise, (spool macro being executed and the task's registers have already been saved by the function), branch to ..... >	TR16			
TR14	The pertinent registers of the failing task (register 4 - register 8) are saved in SVR4.		SVR4(IPQ$SDV)		
TR16	Now space is reserved for the register save area and the TR save area and the address is saved in register 13.			R13	IPW$RSW Cnart AC

Labels	Chart LC03: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
	Addressability of the queue record and the disk management block is set up in registers 5 and 6, respectively. If this routine was not entered due to an unrecoverable I/O error (TCTT ≠ C'U'), clean up the queues at ..... >	TX20		R5,R6	
	Otnerwise, the POWER/VS cancel code is set to 'canceled due to unrecoverable I/O error' (QRCN = X'70'). Addressability of the synchronous save area is set up in register 1, register 2 is set up as MCB counter (set to 8), and register 3 is pointed to the first MCB (A(CAQ1)).		QRCN(IPW$DQR)	R1,R2 R3	
TR18	Now a check is made to see if the MCB of the failing task indicates a system MCB. If so, branch to..... >	TR22			
	If the failing function is in the account file, branch to..... >	TR26			
	Otherwise (failing device is not a system module), branch to ..... >	TX22			
TR22	The 'failing device found' switch (X'FF') is set in the work space (address in register 1) and a branch table is used to branch to the proper routine. The failure may have occurred in:			R1	
	• queue file; branch to..... >	TR38			
	• any data file; branch to..... >	TX00			
	• private or system SSL; branch to >	TR34			
TR26	Account File Routine				
	Message 1Q61I IRRECOVERABLE ERROR ON AFILE N CUU is printed in the..... > subroutine.	TY70			
TR28	If no task is waiting for an account function, branch to..... >	TR32			
	If a task was waiting for a put account function, branch to ..... >	TR30			
	If a task was waiting for a get account function, the termination type in the TCB is set to 'immediate stop' (TNTT = C'S').		TNTT(IPW$DTC)		
TR30	For a task waiting for either a get or put account function, completion of that function is simulated by restoring the task's registers and setting the return address as if the account function has been completed successfully and the task is set dispatchable.		TNRE(IPW$DTC) TNRC(IPW$DTC) TNSF(IPW$DTC)		
	Branch to check the next task..... >	TR28			

Labels	Chart LC04: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
TR32	Job accounting support is suppressed (MRJA = C' ') as well as put account record support (CAPA = 4X'00'). Then message 1Q74I ACCOUNT SUPPORT FUNCTIONS TERMINATED is printed in the..... >	TY70	MRJA(IPW$DQC) CAPA		
	subroutine. If the account function was save account, branch to..... >	TX22			
	Otherwise, (put account), the account function is finished at..... >	TY00			
TR34	System or Private SSL Routine Message 1Q61I IRRECOVERABLE I/O ERROR ON SYSTEM/PVT SSL is printed in the..... >	TY70			
	subroutine, and the queues are cleaned up at ..... >	TX22			
TR38	Queue Record Routine The CCW real address is obtained from the CCB addressed by register 1 and placed in register 1. Using the VIRTAD macro, the corresponding virtual address is obtained and from the seek CCW, of the virtual address of the BBCCHH field is obtained in register 0. Then the address of the complete MBBCCHH field is obtained in register 8. If this address is not that of the master queue record, branch to..... >	TR42		R1 R0 R8	
	Otherwise, the error occurred in a master queue record and the partition must be canceled. Therefore, the DMB is reserved and message 1Q63I IRRECOVERABLE I/O ERROR IN QUEUE MASTER REC - CUU is printed in the..... >	TY70			IPW$RSR Chart AB
	subroutine.				
TR40	Message 1Q76I POWER/VS CANNOT CONTINUE is printed in the..... >	TY70			
	subroutine, and the partition is canceled.				
TR42	Message 1Q61I IRRECOVERABLE I/O ERROR ON QFILE N CUU is printed in the... >	TY70			
	subroutine. If the error occurred during a 'reserve queue set' function (TCFT = C'R'), the free set is inaccessible. In that case, branch to ..... >	TR46			
	If the error occurred neither during a reserve nor during a release function, branch to..... >	TR48			

Labels	Chart LC05: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
	If the record was not the new first in free set, it is ignored because it does not belong on any class chain; in that case, branch to..... >	TR44			
	If the error occurred during a 'release queue set' function (TCFT = C'F') and if the record in error is the new first record in the free set, the pointer to the first record in the free set is reset to its original		MRQF(IPW$DQC)		
TR44	value and the function trace indicator is set to 'release function completed' (TCFT = C'E'); then branch to..... >	TX22	TCFT(IPW$DTC)		
TR46	Message 1Q67I FREE SET NOT ACCESSIBLE is printed in the ..... >	TY70			
	subroutine, and the partition is canceled via a branch to ..... >	TR40			
TR48	If the error occurred during a 'get next queue record' function (TCFT = C'N'), no queue record is in the queue space as yet; then the class chain is obtained at..... >	TR56			
	Otherwise (add queue or delete queue function), the DMB is reserved. If the failing record is the one in the queue buffer, it is deleted. If not, the class must be scanned at..... >	TR66			IPW$RSR Chart AB
TR50	Delete Specific Queue Set from System Files				
	The address of the first-in-set is read.		QCQW(IPW$DQC)		IPW$RDQ
	Subsequently the addresses on the previous-in-set and the next-in-set in the queue sets preceding and following the set in error are made to point around the erroneous set (point to each other), the queue set in error is deleted..... >	TX06	QCQW(IPW$DQC) TCGW(IPW$DTC)		Chart AE
	The DMB address is obtained in register 3. Then the DMB is released.			R3	IPW$RLR Chart AB
	Message 1Q64I JOB jobname RDR/PUN/LST SET DELETED is printed in the..... >	TY70			
	subroutine, and branch to account routine at..... >	TX36			
TR56	If the error occurred during an IPW$GQS function, the class chain in error is not known and must now be found. To do this the DMB is				IPW$RSR
	reserved using register 3 and the address of the bad record is			R3	Chart AB
	saved in TCGW. Then queue space is reserved and the address of the task class list is loaded in register 4.		TCGW(IPW$DTC)	R4	IPW$RSW Chart AC

Labels	Chart LC06: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
TR58	With the address of the class entry in register 7, a search is made for a live entry. If no such entry is found, the task is detached at .... >	TX30		R7	
TR60	When the queue set in error is found, it is deleted in the ..... >	TR76			
	subroutine and message 1Q65I UNKNOWN RDR/LST/PUN SET DELETED is printed in the..... >	TY70			
	subroutine. Then accounting is performed in..... >	TX36			
TR66	For add to queue and delete from queue functions, the known class chain is now scanned for the record in error. To do this, first the address of the failing record is saved in TCGW and the message space, and the queue record information is saved in the message space. Then queue space addressability is set up in register 5. Then the class type is determined from the queue record identifier (QRQI) and the address of the class table is loaded into register 7.		TCGW(IPW$DTC) TCMW(IPW$DTC)	R7	
TR68	Through the class ID and the index to the class entry (both found in register 8), the address of the class entry in error is loaded into register 7. Then new queue space is obtained and the real and virtual addresses of this space are saved in TCQA.		TCQA(IPW$DTC)	R8 R7	IPW$RSW Chart AC
	Then the record is deleted in the . >	TR76			
	subroutine, and a message is set up.				
	Message 1Q65I UNKNOWN RDR/LST/PUN SET DELETED is printed in the..... >	TY70			
	subroutine. Then the DMB is released and, if the current (not the failing) record is the first-in-set, branch to..... >	TR74			IPW$RLR Chart AB
	Otherwise, if the failing record was the first-in-set, no more queue file clean up is required, so branch to..... >	TX36			
TR74	If the function trace indicator shows that an add to queue function was being processed at the time of the failure (TCFT = C'A'), the request is executed and branch is to..... >	TX36			IPW$AQS Chart DB
	Otherwise, queue file clean up is finished at ..... >	TX22			
	<u>Delete Bad Record in Chain Routine</u>				
	Registers used by this routine are:				

Labels	Chart LC07: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
	1: relative record address			R1	
	2: address of field containing absolute disk address			R2	
	5: address of queue space			R5	
	8: return register			R8	
	14: link register.			R14	
TR76	The relative address of the first record in the chain is loaded in register 1. Register 2 contains the address of the field in which the absolute address will be returned.			R1,R2	
	Branch and link to..... >	TX18			
	to get the address converted to absolute. If the failing record is				
	found, branch to..... >	TR80			
TR78	Read a queue record. When the bad				IPW$RDQ
	record is found, branch to..... >	TR82			Chart AE
	If the end of the chain is found before a bad record is hit, the task				
	is detached at..... >	TR94			
TR80	Set the queue record seek address to zero.		QCQW(IPW$DQC)		
TR82	Set up queue space addressability in register 5. Get the relative			R5	
	last-in-queue pointer in register 1			R1	
	and go to..... >	TX18			
	to have the address converted (absolute address in TCQW). If the		TCQW(IPW$DTC)		
	bad record is the last record, branch				
	to..... >	TR86			
TR84	Otherwise, read a new record and, if				IPW$RDQ
	the previous record was the bad one,				Chart AE
	branch to..... >	TR88			
	If the beginning of the chain is reached (reading was backward) before				
	a bad record is found, the task is				
	detached at..... >	TR94			
TR86	Zero tne last, and bad, record. If		TCQW(IPW$DTC)		
	the queue set constitutes the entire class, the entire class is deleted				
	and control is returned to the calling routine via register 8.			R8	
	Otherwise, branch to..... >	TR92			
TR88	The reverse pointer (to the bad		QRQP(IPW$DQC)		
	record) is updated to point to the queue set before the bad record and				
	then written back. If the bad record				IPW$WTQ
	was the first in chain, the new address is converted from absolute to				Chart AE
	relative in routine..... >	TX16			
	and updated in the chain.				
TR90	The next in queue pointer is updated		QCQN(IPW$DQC)		
	and the record written back. Then control is returned to the calling				IPW$WTQ
	routine via register 8.			R8	Chart AE

Labels	Chart LC08: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
TR92	The forward pointer is cleared and written back. The new last record address is converted from absolute to relative in routine..... >	TX16	QCQN(IPW$DQC)		IPW$WTQ Chart AE
	and the last-in-class pointer is set, as well as the active indicator. Then control is returned to the calling routine via register 8.			R8	
TX00	Data File Routine				
	If the error occurred on a data file, message 1Q61I UNRECOVERABLE I/O ERROR ON XFILE N CUU is printed in the... >	TY70			
	subroutine, and, if the error occurred on a get data record function (function trace indicator TCFT = C'G'), branch to..... >	TX04			
	If the function was a put data record function (function trace indicator TCFT = C'P'); message 1Q64I JOB jobname RDR/LST/PUN SET DELETED is printed in..... >	TY70			
	the function is ignored because the queue set has not been added to any class chain, and accounting is performed at..... >	TX36			
TX04	With the use of register 3, the DMB is reserved and the queue set in error is deleted at..... >	TR50		R3	IPW$RSR Chart AB
	Delete Specific Queue Set in Class Chain Routine				
	Registers used by this routine are:				
	1: relative disk address			R1	
	2: absolute disk address pointer			R2	
	3: work register			R3	
	7: address of class table			R7	
	8: return register			R8	
	14: link register.			R14	
TX06	The address of the applicable (reader, list, or punch) class table is loaded into register 7.			R7	
TX08	The index to the class table entry is calculated in register 3; register 7 is then pointed to the relevant entry in the class table. If this is not the first set in the queue, the previous set's pointers must be updated; this is done via a branch to..... >	TX12		R3,R7	
	Otherwise, if this entry is also the last in the queue, the entire chain is deleted and the calling routine is returned to via link register 8.		CTQF(IPW$DTC)	R8	

Labels	Chart LC09: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
TX10	If the entry is just the first in the chain, the absolute address of the next in chain is obtained in register 2 and converted to relative in..... >	TX16		R2	
	This relative address is returned in register 1 and saved as the new first in set address, the active indicator is set (CTQL = X'80'), and branch to..... >	TX14	CTQF(IPW$DTC) CTQL(IPW$DTC)	R1	
TX12	Read the previous set's first record, update the next queue set in class pointer to point around the bad queue set, and write the record back.		QCQN(IPW$DQC)		IPW$RDQ Chart AE IPW$WTQ Chart AE
	If this is not the last in set, branch to..... >	TX14			
	Otherwise, the absolute address of the queue set previous to the bad queue set in chain is obtained in register 2 and converted to relative in..... >	TX16		R2	
	This relative address is returned in register 1 and saved as the new last in set address, the active indicator is set (CTQL = X'80'), and return is made to the calling routine via link register 8.		CTQL(IPW$DTC)	R1 R8	
TX14	The address of the previous set is saved, the next set in the chain is pointed to and obtained, the previous set pointer in the next set just obtained is reset to point around the bad queue set, and the next set's first record is written back. Then return is made to the caller via link register 8.		TCGW(IPW$DTC) QCQW(IPW$DQC) QCQN(IPW$DQC)	R8	IPW$RDQ Chart AE IPW$WTQ Chart AE
	Disk Address Conversion Routine (absolute to relative) Registers used by this routine are: 0: work register 1: relative record class 2: absolute record address pointer 3: module control block address 14: return register.			R0 R1 R2 R3 R14	
TX16	The cylinder number is obtained from the field addressed by register 2 and placed in register 1. The starting cylinder number of the queue set is subtracted and this value is multiplied by the number of tracks per cylinder and the number of records per track to obtain the relative record number in register 1. Then return via register 14.			R2 R1 R14	

Labels	Chart LC10: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
	Disk Address Conversion Routine (relative to absolute)				
	Registers used by this routine are:				
	0: work register			R0	
	1: relative record address			R1	
	2: pointer to field containing absolute address			R2	
	3: module control block address			R3	
	14: return register.			R14	
TX18	The high order bits are stripped of the absolute address, the DMB address is set in register 3. Then the			R3	
	relative track and relative cylinder number are calculated in register 0;			R0	
	the remainders of these values are the relative record and track, respectively. Together with the relative cylinder number these values are stored in the relative disk address field pointed to by register 2. Then control is returned to the calling routine via register 14.		QCQW(IPW$DQC)	R2 R14	
TX20	Clean the Queues				
	This routine is entered at TX22 in the case of unrecoverable I/O error. This entry point (TX20) is for PSTOP. Therefore, 'canceled due to PSTOP' is set in the POWER/VS cancel code (QRCN = X'30').		QRCN(IPW$DQR)		
TX22	Entry Point for Irrecoverable I/O Error				
	If the function trace indicator (TCFT) is set to X'00' or C' ' (main routine just initialized or finished meaning no queue function has yet been invoked or logical end of spool functions successfully reached), no clean up is required; in that case branch to..... >	TX30			
	Otherwise, if writing of an account record was pending (TCFT = C'E'), or if the error occurred during account record writing, (TCFT = C'L'), retry on the account function is effected at..... >	TX36			
	If the function was writing to spool (output, TCFT = C'R', C'O', C'P', or C'A'), branch to..... >	TX24			
	Otherwise (reading from spool), check the input function at..... >	TX40			

Labels	Chart LC11: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
TX24	If this is an execution processor task (TCTI = C'E '), it must be the execution writer, so branch to..... >	TX78			
	The error occurred on a non-execution processor task. The address of the queue space is obtained in register 1, and, if there is no queue space branch to..... >	TX30		R1	
	If tnis is not a reader task, branch to..... >	TX26			
	Otherwise, load the address of the message page into register 3 and the address of the message area into register 1.			R1,R3	
	The text of message 1Q64I is moved from the message page to the message area.		TRMS(TRWS)		
	A link is made to the message routine to log message '1Q64I JOB jobname RDR SET DELETED'..... >	TY70		R14	
	If tne current task is a reader task, started for a card device or 3540 diskette device, branch to..... >	TX36			
TX30	General Exit from Queue/Data File Clean Up				
	If the task was not an execution processor, branch to ..... >	TX32			
	Otherwise, (TCTI = C'E '), set up addressability of the synchronous save area in register 1, reload the failing task's registers, and, if the task was an execution reader task, branch to..... >	TY10		R1 R4-R8	
	Otherwise (execution writer task), branch to..... >	TX78			
TX32	If the queue set was not deleted and returned to the free set before, it is done now.				IPW$DQS Chart DD IPW$FQS Chart DE
	If the task was not an RJE list, orpunch task, branch to..... >	TX34			
	If RJE task, branch to..... >	TY38			
	Otherwise, if the device used by the task is a tape (TCQW = X'80'), branch to..... >	TY20			
	If not (unit record device), branch to..... >	TY30			
TX34	If it was a save account function, branch to..... >	TY00			
	Otherwise, if the task was a reader task, branch to..... >	TY30			

Labels	Chart LC12: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
	If not, the task is an unknown task; in that case branch to..... >	TY38			
	Otherwise, (TCTI = C'STAT'), branch to..... >	TY26			
TX36	Write Account Record Routine				
	If the task was owned by an execution processor, the accounting function is ignored; branch to..... >	TX30			
	The current time value is obtained via an IPW$RDC call, and then stored in the queue record.		QRET(IPW$DQR)		IPW$RDC Chart AG
	The virtual queue file address is loaded into register 1, and the length of the account record which depends on the class type (RDR,LST, or PUN) is determined and placed in register 0. If it is none of these class types, branch to..... >	TX34		R1	
	In case of stopped list or punch task, counters for pages, lines/cards, and number-of-copies are updated and moved into the queue record.		QRNE(IPW$DQR) QRNP(IPW$DQR) QRNA(IPW$DQR) QRNR(IPW$DQR) QRNC(IPW$DQR)	R7	
	If the start time is zero, bypass writing the account record and branch to..... >	TX32			
	The account record that could not be written before is now written. Then branch to ..... >	TX32			IPW$PAR Chart FA
TX40	Input Mode Data/Queue File				
	If TCTI = C'STAT', no clean up is necessary, so branch to..... >	TY26			
	Otherwise, if the function is neither punch nor list, branch to..... >	TX52			
	If the device used is tape, branch to ..... >	TX36			
	If a PSTOP command without RESTART was issued, branch to..... >	TX51			
	Otherwise (TCTT = C'R'), get the address of the logical list save area in register 1 and point register 4 to the account counter DSECT (LADS).			R1,R4	
	Read the first queue record.				IPW$RDQ Chart AE
	The remaining number of copies is saved and the increment is set in register 0. If restart is active, update the restart page count in register 0, and branch to..... >	TX50		R0 R0	
	If restart is not active, update the current page or card, depending on the class type.		LACP LACR		

Labels	Chart LC13: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
TX50	Save the restart page count in the queue record.		QRRR(IPW$DQR)		
TX51	If this is not a physical writer task, branch to..... >	TX52			
	If the device being serviced is not a 3800 printer, branch to..... >	TX52			
	If the output is not for a 3800, branch to..... >	TX52			
	Using register 1 as a base address for the TCB extension area, the current copy group index is saved in the queue record.		QRCI(IPW$DQR)	R1	
TX52	If the function during which the error occurred was 'get next queue record', 'get next data record', or at a stage in between these two (TCFT = C'N', C'G', or C'I'), branch to..... >	TX68			
	Otherwise, based on the values in the indicator, the queue set may be deleted or, if it had already been deleted, it may be freed. If freeing is not required, branch to..... >	TX54			IPW$DQS Chart DD IPW$FQS
	Otherwise, after freeing, branch to..... >	TX36			Chart DE
TX54	If at this stage a 'release queue set' function is not indicated (TCFT = C'F'), branch to ..... >	TX30			
	Otherwise, a test is made to see if the queue set has been released successfully. To do this, the DMB is once again reserved, the first in set queue record is read and the queue record identifier is inspected to see if the queue set was freed (QCQI = C'F') or reused for another queue set. Then the DMB is released again and, if the queue set was not freed before, it is freed now.				IPW$RSR Chart AB IPW$RDQ Chart AE IPW$RLR Chart AB IPW$FQS Chart DE
	At the successful completion of this routine, branch to..... >	TX36			
TX68	Routine for Non-Tape Device and Non-Deleted Queue Set				
	With the proper address in register 3, tne DMB is reserved. The first record in the queue set is now read.			R3	IPW$RSR Chart AB IPW$RDQ Chart AE

Labels	Chart LC13.1: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
	The execution switch is reset and, if the task was ended by a PSTOP command, the cancel code in the queue record is set to 'canceled due to PSTOP'. The 'copy' counter and 'remaining copy' counter are updated if required. If this task does not belong to an execution reader, branch to..... >	TX76	QRXS(IPW$DQR) QRCN(IPW$DQR) QRNC(IPW$DQR) QRCR(IPW$DQR)		
	Otherwise, the disposition is reset from either 'K' to 'L' or from 'D' to 'H' (to prevent the job from being dispatched immediately).		QRDP(IPW$DQR)		

Labels	Chart LC14: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
TX76	The queue record is written.				IPW$WTQ Chart AE
	If RJE task, then branch to..... >	TX64			
	If reader queue record then branch t>	TX64			
	If list or punch queue record then the line bit in the master class table is set on..... >	TX64	CTQL(IPW$DQC)		
TX78	Execution Writer Wrap-up				
	Register 1 is pointed to the synchronous register save area, and the failing task's registers 4-8 are reloaded. Register 6 contains the address of the PDB and register 4 that of the device list entry. If this is an execution reader, an error has occurred (this is the output mode handler); therefore, branch to..... >	TY10		R1 R4-R8	
	Otherwise, get the queue space address. If no queue space is available, branch to..... >	TX86			
	If the execution writer uses tape, branch to..... >	TX84			
	Otherwise, if the current record or the first record in the current queue set is a bad record, bypass segmentation and branch to..... >	TX86			
TX80	If no output was generated, release the queue set, and branch to..... >	TX86			IPW$FQS Chart DE
	Otherwise (bad record somewhere in the queue set), segmentation of the queue set is effected. This is done by forcing end-of-data (blank data field, length of one, NOP CCW) and setting an 'I' disposition to 'D', while at the same time setting the queue record identifier to 'reader'.		TCCC TCGP QRDP(IPW$DQR) QRQI(IPW$DQR)		IPW$PDR Chart EA
	Now the queue set is added to the class chain; branch to..... >	TX86			IPW$AQS Chart DB
TX84	Message 1Q61I IRRECOVERABLE I/O ERROR ON CUU is printed in the..... > subroutine.	TY70			
TX86	Control of the device in the user partition being spooled is passed back to the execution reader by setting the address of the controlling TCB (that is in the device list entry IPW$DDE) in the partition control block (IPW$DPD) to point to the execution reader TCB. The execution reader task will initiate an execution writer task to continue processing. The execution reader is set dispatchable, as well		TLTC(IPW$DTL) TCLB(IPW$DTC)		

Labels	Chart LC15: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
	as the partition. Then messages 1Q68I SEGMENTATION FORCED FOR jobname nnnnn ttt CUU and 1Q69I DEFAULT OPTIONS TAKEN FOR jobname nnnnn ttt CUU are printed in the..... >	TY70			
	subroutine.				
	If the device is tape (TCQW = X'80'), unassign the device using registers 0, 1, and 3. Then branch to..... >	TY38		R3,R1	IPW$ULP Chart GC
TY00	Save Account (PACCOUNT) Task Wrap-up Routine				
	If the 'failing device found' switch is set, the failing device was not an output device for put account, so branch to..... >	TY02			
	Otherwise, message 1Q61I UNRECOVERABLE I/O ERROR ON PACCOUNT OUTPUT DEVICE is printed in the.... >	TY70			
	subroutine, and branch is to..... >	TY04			
TY02	If the account file was in the process of being erased or if the operator requested deletion of the account file, branch to..... >	TY05			
TY04	The ACB is reset to 'put' mode and its original values are restored.				IPW$CAF Chart FB
TY05	Message 1Q66I ACCOUNT FILE KEPT and message 1Q72I PACCOUNT TERMINATED are printed in the..... >	TY7C			
	subroutine.				
	If the activity was on disk, or if output spooling was active, branch to..... >	TY38			
	Otherwise, the CCB pointer is obtained in register 3; if no CCB is initialized, branch to..... >	TY08		R3	
	If the activity was not on tape, branch to..... >	TY06			
	Otherwise, the CCB pointer for the tappe is stored in register 3 and set to a dummy SVC value. Then a fetch is simulated to close the SA tape file. This is done via an SVC 2 to $$BCLOSE.			R3	IPW$FCH Chart AA transient $$BCLOSE
TY06	Unassign the device and branch to..... >	TY38			IPW$ULP Chart GC
TY08	Set up a dummy CCB and point register 3 to it; then unassign the device and branch to..... >	TY06		R3	IPW$ULP Chart GC

Labels	Chart LC16: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
TY10	Execution Reader Wrap-up				
	Establish addressability of the PCB in register 6.			R6	
	Release DMB to prevent execution writer waiting on locked resource.				IPW$RLR Chart AB
	Get the number of entries in register 0 and the first entry address in register 4.			R4	
	Scan through the reader table and, for those tasks that have been started, reset the ownership, set the task termination code to stop (TNTT = C'S'), and post the live indicator in the event control block. Then wait for the writer task to sign stop completion.		TLCT(IPW$DTL) TNTT(IPW$DTC) TNEB(IPW$DTC)		IPW$WFC Chart AA
TY14	The POWER/VS control flags in theuser partition are reset, as well as the partition control block.		POWFLG1,2 POWPCB		
	Then unassign SYSRDR. If the partition is writer-only, branch to..... >	TY18			IPW$ULP Chart GC
TY16	Unassign the device specified in the entry.		TCGW(IPW$DTC)		IPW$ULP Chart GC
TY18	Point register 4 to the next entry in the PDB and continue to unassign, via branches to TY16, until all units have been handled.			R4	
	Message 1Q70I TASK FAILURE, STOPPED TTT is printed in the..... >	TY70			
	subroutine and the following fields are updated: • job control switches (turn on bits 4 and 5) • the PIB cancel code (set to X'FF') • the PIB fetch EOJ monitor flag (X'08') • the PIB flag byte is set dispatchable (X'01').		JCSW1(CMRG) PIBCNCL PIBFLG2 PIBFLG		
	Then branch to..... >	TY38			
TY20	Physical Writer with Tape - Wrap-up				
	Register 7 is based on the tape control block. If no unrecoverable I/O error was encountered (IPW$$TR entered via a PSTOP command), or if the error found is not a tape error, branch to..... >	TY22		R7	
	Otherwise, message 1Q61I IRRECOVERABLE I/O ERROR ON CUU is printed in the..... >	TY70			
	subroutine. Then branch to..... >	TY24			

Labels	Chart LC17: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
TY22	The command is issued to rewind and unload tne tape.				IPW$CTT Chart AF
TY24	The tape unit is unassigned; then branch to..... >	TY30			IPW$ULP Chart GC
TY26	Validate Unit Record CCB and Unassign the Device				
	If an unrecoverable I/O error was encountered (TCTT = C'U'), message 1Q73I STATUS DISPLAY TERMINATED is printed in the..... >	TY70			
	subroutine.				
TY28	If the unit record device that failed is the console or a spool management task, it need not be unassigned, so branch to..... >	TY38			
	Otherwise, unassign the device at.. >	TY36			
TY30	Unassign the Unit Record LUB				
	If the entry condition found was not an unrecoverable I/O error, branch to..... >	TY34			
	If the 'failing device found' switch is set, the failing device was not a unit record device, so branch to... >	TY32			
	Otherwise, message 1Q61I UNRECOVERABLE I/O ERROR ON CUU is printed in the..... >	TY70			
	subroutine.				
TY32	Message 1Q71I TTT,CUU TERMINATED is printed in the..... >	TY70			
	subroutine; then branch to..... >	TY36			
TY34	Message 1Q33I STOPPED TTT,CUU is printed in the ..... >	TY70			
	subroutine.				

Labels	Chart LC17.1: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
TY36	In case the current task is a spool management task, then bypass this routine and branch to release resource to..... >	TY38			
	If task is not a reader or list task, branch to..... >	TY36D			
	Load register 8 from register save area.			R2,R8	
	If zero, work space is already released, branch to.....>	TY36D			
	Let register 1 point to CCB(1 buffer).			R1	
	If no double buffering, branch to.. >	TY36B			
	If it is not a writer task, branch to..... >	TY36D			
	If this buffer is active, then let register 1 point to the other CCB.			R1	
	If no double buffering, branch to.. >	TY36B			
TY36C	Check if CCB is already posted. This could be the CCB having the I/O error and the reason to be here.				
	If not posted, wait on I/O completion.				IPW$WFC
TY36D	Assignments made for this task are now to be unassigned.				
	Load register 2 with the cuu field from the TCB.			R2	
	Load the branch index to request for a generic unassign into register 0.			R0	
	Load the address of the POWER/VS PIB into register 1.			R1	
	The current unit record device is unassigned.				IPW$ULP Chart GC
	If the task is not a reader task, branch to..... >	TY38			
	If the reader task has not been activated in combination with a 3540 diskette reader, branch to..... >	TY38			

Labels	Chart LC18: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
	Load the address of the register 7 field in the physical save area of the TCB, which can contain the programmer unit number, as a CCB like field into register 3.			R3	
	If the physical work space of the reader task has already been released, branch to..... >	TY37			
	Otherwise, reload register 3 with the address of PELU - 6, to use this part of the physical work space as a CCB like field.			R3	
TY37	Point register 1 to the PIB of the POWER/VS partition.			R1	
	Load register 0 with the branch index to request to unassign the programmer logical unit in the CCB like field addressed by register 3.			R0	
	Tne programmer logical unit assigned to the 3540 diskette device is unassigned.				IPW$ULP Chart GC
TY38	A new base register domain is established.			R9	
	If the task uses the asynchronous service subtask, a IPW$IAS TYPE=DETACH request is issued to detach the subtask.				IPW$IAS Chart GH
TY38F	Release Resources				
	Register 7 contains the address of the first resource, and register 8 contains the number of resources.			R7,R8	
	A scan through all control blocks is made. If the control block is for a function or a device that is supported by the present system, locked, and owned by the task, it is now released.				IPW$RLR Chart AB
	Release Work Space				
	The address of the first page is loaded into register 8.			R8	
TY44	The address of tne first buffer control word (BCW) on this page is loaded into register 7.			R7	
TY46	The buffer length is obtained in register 2. If the length is zero (end of page), branch to..... >	TY56		R2	
	If the length n is positive (buffer not in use), branch to..... >	TY54			
	Otherwise, load the positive buffer length into register 3.			R3	

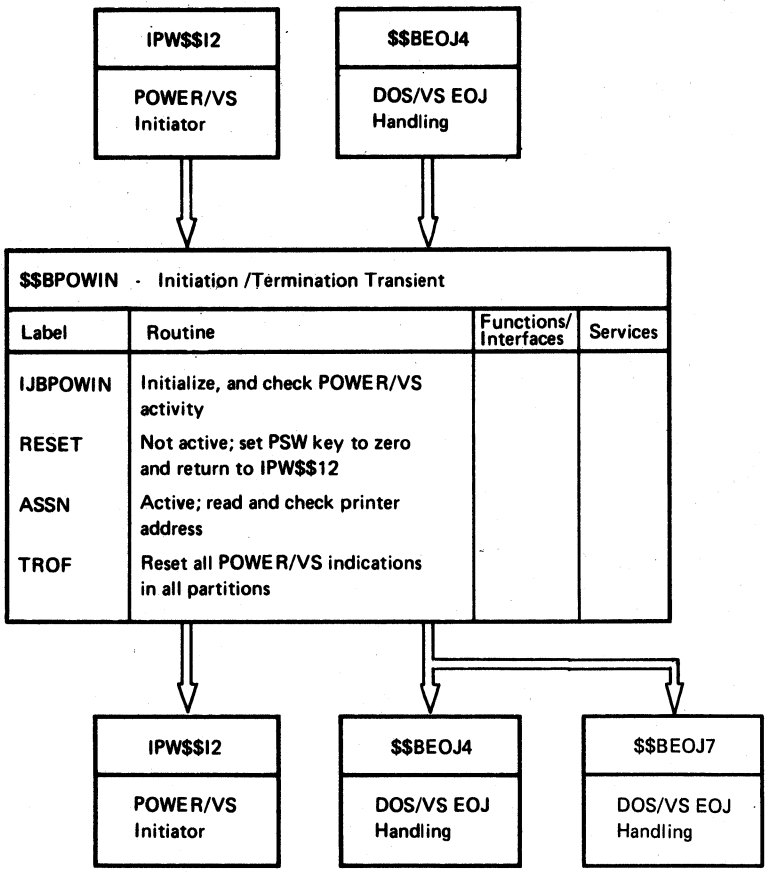
Labels	Chart LC19: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
TY48	The length of the next buffer is loaded into register 4. If that buffer is not active or if end-of-page is reached, the buffer after that is obtained. This process is continued until a second active buffer is found or end-of-page is reached.			R4	
TY50	Two active buffers have now been found, or one active buffer has been found and end-of-page was reached. The address of the second buffer is saved in register 6 and, if the owner of the first buffer is the present task, the first buffer is released. Otherwise, the first buffer is ignored, the address of the second buffer is updated to make it the first buffer (address now in register 7), and a search for another active buffer is started at..... >	TY46		R7	IPW$RLW Chart AC
	This process is continued until all work space is released.				
TY54	The BCW pointer is updated and the next buffer checked at..... >	TY46		R7	
TY56	If end-of-page is reached, switch to the next page (address in register 8); if there is another page, branch to..... >	TY44		R8	
	Otherwise, all active work space has been scanned. If this is not an RJE task (TCTI < C'0'), branch to..... >	TY67			

Labels	Chart LC19.1: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
	If this is not an SNA reader or writer, branch to..... >	TY66			
TY58	If this is an SNA inbound processor task, the address of the inbound processor is loaded into register 2. Otherwise the address of the SNA outbound processor task is loaded into register 2.			R2	
TY64	Addressability of the SNA work area (WACB) is set up in register 3.			R3	
	The address of the RJE,SNA error routine (WAER) is loaded into register 1.				
	The processing switch in the SNA work area is set to X'FE' to indicate that the logical interface is closed.		WASW(IPW$DWA)		
	The RJE,SNA error routine is branched to via register 1.			R1	
TY66	The address of the RJE,BSC routine (IPW$$TM) is loaded into register 1 and branched to.			R1	Phase IPW$$TM
TY67	If not a spool management request (TCTI≠'J'), branch to..... >	TY68			
	The address of the error routine is set up in register 15 and the base register 9 is established. Direct branch to termination routine in IPW$$SM.		TCXA(IPW$DTC) CASF(IPW$DPA)	R15 R9	

Labels	Chart LC20: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
TY68	The task is detached.				IPW$DET Chart AA
	Write Message to Console Routine				
	Register used by this routine is:				
	14: return register.			R14	
TY70	Write a message to console using message service and return to the calling routine via return register 14.			R14	IPW$WTO Chart AD

CHART LD: $$BPOWIN - INITIATOR/TERMINATOR TRANSIENT (4 PARTS)

Chart LD00: $$BPOWIN - Initiator/Terminator Transient, General Flow and Macro Calls



Labels	Chart LD01: $$BPOWIN - Initiator/Terminator Transient		Modified Data Fields	Reg. Usage	Calls
	Register usage				
	0: **** - Work register			R0	
	1: **** - Work register			R1	
	2: **** - Work register			R2	
	3: **** - Work register			R3	
	4: SCOM - Address of SYSCOM			R4	
	5: **** - Work register			R5	
	6: **** - Work register			R6	
	7: **** - Work register			R7	
	8: **** - Work register			R8	
	9: **** - Work register			R9	
	10: **** - Work register			R10	
	11: **** - Work register			R11	
	12: CMRG - Address of communications region			R12	
	13: **** - Work register			R13	
	14: **** - Linkage register			R14	
	15: **** - Base register.			R15	
IJBPOWIN	The address of the communications region is placed in register 12 and the addressability of the AR PIB is set up in register 5. Then the SYSCOM address is set up in register 4 and the POWER/VS flag is tested. If the flag is on (POWER/VS active), branch to..... >	PW02		R12 R5 R4	
	Otherwise, the PSW key is set to zero and an exit is made from $$BPOWIN via an SVC 11.			R3	
	Check for DOS/VS Subtask Cancel				
PW02	If DOS/VS multitasking (AP=YES) is supported, and the maintask has cancelled, branch to..... >	PW03		R1 R1	
	If a subtask which is not the asynchronous service subtask, has been cancelled, branch to..... >	PW03		R2,R10	
	The task waiting for completion of the asynchronous service request is made dispatchable.		TCSF(IPW$DTC)		
	Post stop state.		TCTT(IPW$DTC)		
	Return is made to $$BEOJ7 via SVC 2.				
PW03	The following checks are made: • If a normal EOJ, then branch to turn off POWER/VS switches..... >	PW36			
	• Otherwise, if a dump was issued, also branch to turn off POWER/VS switches..... >	PW36			
	• Otherwise, if a program request for a cancel was made, then force a dump and branch to..... >	PW08	PIBCNCL		

Labels	Chart LD01.1: $$BPOWIN - Initiator/Terminator Transient		Modified Data Fields	Reg. Usage	Calls
	Otherwise, copy the cancel code of the subtask in the maintask PIB and set flags to indicate cancel, fetch EOJ, and post maintask PIB.		PIBCNCL PIBFLG PIBFLG2	R1	
	Issue Termination Message and Assign SYSLST for Dump				
PW08	The address of SYSLST for this partition is calculated in register 6.			R1,R2 R3,R6, R11,R14	GLUB Chart LD
	With register 2 as a work register, the CCW chain required for message writing is set up.			R2 R1	
	The address of the CCW chain is stored in the SYSLOG CCB.		CCBCCW	R1	
PW12	Message 1Q30D ABNORMAL POWER/VS TERMINATION. PRINTER= is printed on SYSLOG. Then the operator reply is analyzed. If the CANCEL key was hit (CCBSTA1 = X'01'), the message is reissued..... >	PW12			

Labels	Chart LD02: $$BPOWIN - Initiator/Terminator Transient		Modified Data Fields	Reg. Usage	Calls
PW16	If the operator typed 'no' or END/EOB (blank input area), branch to..... >	PW36	INPT	R0	
PW20	The operator replied with a printer address. If the address is invalid, message 1Q30D INVALID PRINTER TYPE, RE-ENTER is loaded into the message buffer and printed via a branch back to..... >	PW12		R2,R3	
PW24	A valid address was given and is now translated to a hexadecimal device address.		DVAD,INPT		
PW28	The device address is now checked against the PUB table; if the end-of-table is found, if the device is down, or if the device type is invalid (PUBDEVTY not between X'40' and X'50'), retype the invalid printer message at..... >	PW12		R7	
PW32	The assignment of the valid printer is now made.		LUBXXXP	R0,R7	
PW36	Delete all POWER/VS Indications				
	The POWER/VS active bit in the SYSCOM is reset and the address of the POWER/VS table is deleted. The number of partitions is loaded into register 0 and the PIB2 pointer in register 2.		IJBLG03 IJBPWR	R0,R5 R3 R2	
PW40	The addressability of the partition COMREG is established in register 12. If the partition pointed to is the POWER/VS partition branch to..... >	PW48		R2,R5 R12	
	If this partition is supported by POWER/VS, branch to..... >	PW64			
PW44	Otherwise, reset the POWER/VS flags. Then zero the PCB address.		POWFLG1,POWFLG2 PCWPCB		
	The number of partitions in register 0 is decremented, and if not zero, branch back to..... >	PW40		R0	

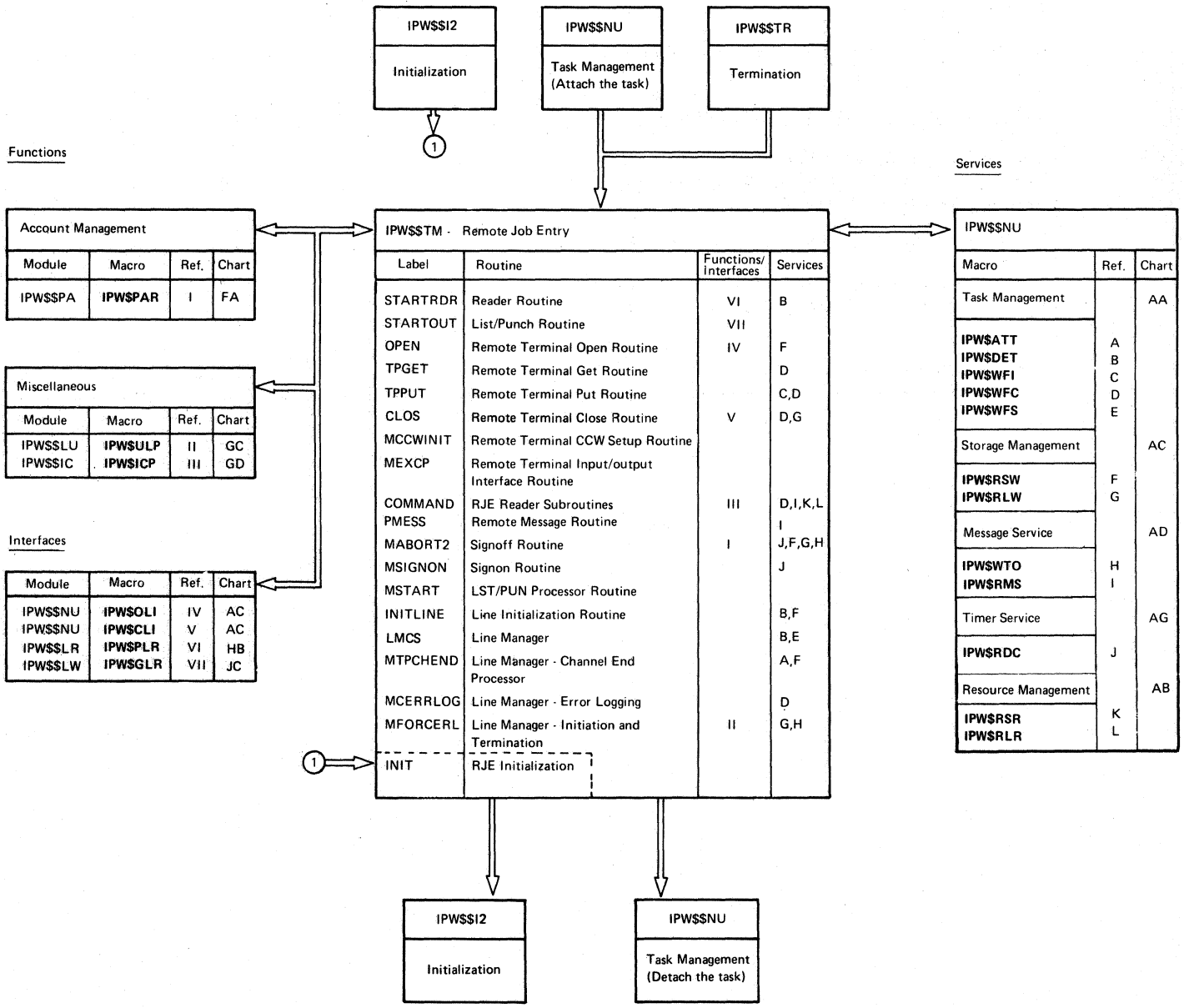
Labels	Chart LD03: $$BPOWIN - Initiator/Terminator Transient		Modified Data Fields	Reg. Usage	Calls
	POWER/VS partition				
PW48	Reset the job control switch for the partition COMREG.		JCSW1		
	A branch is made to get the LUB parameters..... >	GLUB		R1,R2 R3,R5 RC,RE	
	The following loop checks all non-system LUBs:				
PW52	If the LUB is unassigned, branch to..... >	PW60			
	The PUB address assigned to the LUB is calculated and the device type is examined.			R7	
	If the device is a DASD device, branch to..... >	PW60			
	If the device is a 3800 printer, a SETPRT parameter list is built:			R1	
	• save registers 0 and 15			R0,R2,R3	
	• set proper LUB address in SETPRT parameter list		SPPLUNIT (SPLIST)	RF	
	• indicate LUB address		SPPFLAG1		
	The SETPRT request is issued, which sets up the printer with hardware/system defaults.				
	Registers 0 and 15 are restored.			R0,R2 R3,RF	
PW56	The LUB is unassigned, and the PUB ownership is cleared.		LUPXXXP	R7	
PW60	The next LUB entry is addressed, and a branch is made to loop through... >	PW52		R6	
	If no more LUBs are to be checked, branch to..... >	PW44			
PW64	If the partition was not stopped, then the partition is set dispatchable (bit 7 in PIBFLG of the PIB is set on), and further, if there is a QTAM wait, then bit 4 of PIBFLG is turned off. Also in the PIB, the flags for EOJ and cancel are set, and the cancel code is stored in the PIB.		PIBFLG PIBFLG2 PIBCNCL	R1,R10	
PW76	Then the devices described in the partition PIB are checked, except for the console, and any LUB's which are assigned to the devices are unassigned, except for devices which are device type 'do no intercept' (C'N').		LUB	R1,R2, R3,R5 R6,R7, R8,R9, R13,R14	
	Branch back to..... >	PW44			

Labels	Chart LD03.1: $$BPOWIN - Initiator/Terminator Transient		Modified Data Fields	Reg. Usage	Calls
	Delete POWER/VS Spool Management XECB's				
	If spool management was not requested, then branch to..... >	PW47		R1,R5, R10,R12	
	The macro XECBTAB is used to first delete the internal reader XECB and next the spool manager XECB. Then the XECBTAB macro is used to check if the internal reader is active.			R1,R2, R15	
	If not active, then branch to..... >	PW46		R2,R5 R15	
	Then the XPOST macro is used to notify the user that POWER/VS is terminating.				

Labels	Chart LD04: $$BPOWIN - Initiator/Terminator		Modified Data Fields	Reg. Usage	Calls
PW46	A check is made of the spool manager interface using the XECBTAB macro. If not active, branch to..... >	PW47			
	Then the XPOST macro is used to notify the user that POWER/VS is terminating.				
PW47	A return is made back to $$BEOJ4 via SVC 2.				
	Subroutine to Get LUB Parameters				
GLUB	Using the PIB address in R5 and the COMREG address in R12, the following calculations are made:			R5,R11, R12	
	• Partition FICL multiplied by 2. (R1)			R1,R2,	
	• The number of system units. (R2)			R3	
	• The number of programmer units. (R3)				
	Return to caller			R14	

CHART MA: IPW$$TM - BSC REMOTE JOB ENTRY ROUTINES (49 PARTS)

Chart MA00: IPW$$TM - BSC Remote Job Entry Routines, General Flow and Macro Calls



Labels	Chart MA01: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
	RJE Physical Reader Routine				
	This routine is entered after it has been attached by the line manager when it receives an enquiry from the terminal indicating that the reader is ready.				
	Relevant register contents passed by the line manager:				
	4: BCA pointer			R4	
	7: base register			R7	
	8: base register			R8	
STARTRDR	Register 9 is set up to point to the LCB, and a link is made to open the reader..... >	OPEN (MA05)		R9 R14	
GETLOOP	If a buffer was saved, it is released now. Register 3 is set up to point to the record buffer and a link is made to obtain an input record..... >	TPGET (MA07)		R3 R14	IPW$RLW Chart AC
	If signoff is in progress, branch to>	GETLOOP (MA01)			
	If the current job stream is not on a job boundary, branch to bypass RJE command processing..... >	NOTCMD (MA01)			
	Otherwise, a check is made for presence of the RJE command identifier, '* ..', and if found, a branch is made to process the RJE command.... >	COMMAND (MA22)			
	If no user is currently signed on, branch to process invalid signon... >	NOSIGNON (MA23)			
	If no RJE command, and the record contains only blanks, (indicating auto-turnaround on 2780), the record is ignored and a branch is taken to process EOT..... >	GETLOOP (MA01)			
NOTCMD	Otherwise, data pointer registers 0 (input record address) and 1 (input record length) are set up to point to the record just read.			R0,R1	
PUTL	The record is passed to the logical reader.				IPW$PLR Chart JC
	If not EOF (record length in register 1 not zero), branch to get next record..... >	GETLOOP (MA01)			
	If EOF not on job boundary, branch to issue diagnostic message indicating unexpected EOF..... >	EOFRDR (MA23)			
CLOSRDR	If EOF on job boundary, the reader is closed by branching and linking to. >	CLOS (MA19)			
	The reader TCB in the LCB is zeroed. The input class pointer of the reader TCB is saved in the LCB.		LCBTCBAD(IPW$DLC) LCBWORK(IPW$DLC)		
PDET	The line manager is posted. The task is detached.		TCEB	R1,R6	IPW$DET Chart AA

Labels	Chart MA02: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
	RJE Physical Writer Routine				
	This routine, which handles remote list/punch output, is entered after it has been attached by the line manager when it receives an ACK from the terminal indicating that the printer or punch is ready.				
	Relevant register contents passed by the line manager:				
	4: BCA pointer			R4	
	7: base register			R7	
	8: base register			R8	
STARTOUT	Register 9 is set up to address the LCB.			R9	
	If starting up a message writer, branch to issue the messages..... >	MESSWTR (MA03)			
	A link is made to open the list/ punch..... >	OPEN (MA05)		R14	
PUTLOOP	If necessary, reset message switch for list.		LCBOUT(IPW$DLC)		
	A data record is obtained from the logical writer.				IPW$GLR Chart HB
	If return after end of job, branch to..... >	CLOSOUT3 (MA04)			
	If record length pointer register 1 is zero, indicating end of data, a branch is made to reset output switch and close the list/punch..... >	CLOSEX3 (MA04)			
	If end of job (data record address in register 0 zero), branch to close the list/punch..... >	CLOSOUT (MA02)			
	Otherwise, link to the PUT routine. >	IPPUT (MA10)		R14	
	Branch to get next data record..... >	PUTLOOP (MA02)			
CLOSOUT	A link is first made to flush the output buffer..... >	MSGCLOSE (MA19)			
	Branch to..... >	PUTLOOP (MA02)			

Labels	Chart MA03: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
	Message Routine				
MESSWTR	This routine lists all outstanding messages for the current remote ID.				
	If not invalid SIGNON, branch..... >	MESSWR1			
	Force minimum buffersize.		LCBBFSIZ	R1	
MESSWR1	A link is first made to open the			R14	
	If not invalid SIGNON, branch to... >	MESSWR1			
	Force minimum buffersize.		LCBBFSIZ	R1	
	list..... >	MSGOPEN (MA05)			
	On return, the command code in the list TCB is set to skip to the next page.				
	If the last printed output was a message, branch to..... >	NEXTMSG (MA03)			
	Otherwise, set the command code to eject.		TCCC(IPW$DTC)		
	A link is made to the PUT routine..... >	TPPUT (MA10)		R14	
	Set the message indicator.		LCBOUT(IPW$DLC)		
	If the task has been stopped immediately, branch to..... >	MSGSTOP		R11	
NEXTMSG	Get the next message from the remote message queue, via a branch and link to..... >	GETMSG (MA04)		R6	
	If there are no messages left, branch to..... >	MSGEXIT (MA03)			
	Set up data pointer and length field in the TCB.		TCRL(IPW$DTC) TCCC(IPW$DTC)		
	The write command code is set in the TCB.		TCCC(IPW$DTC)		
	A link is made to the PUT routine..... >	TPPUT (MA10)		R14	
	Retrieve the data pointer from the TCB.			R1	
	Delete the messages from the subchain via a branch and link to..... >	DELMSG (MA04)			
	Test for immediate stop. If yes, branch to..... >	MSGSTOP			
	Otherwise branch to process next message..... >	NEXTMSG			

Labels	Chart MA03.1: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
MSGEXIT	If the list task was interrupted, the command code in the TCB is set to space 3 lines.		TCCC(IPW$DTC)		
	A link is made to the PUT routine 4 times..... >	TPPUT (MA10)		R14	
MSGDONE	Link to close the list..... >	MSGCLOSE (MA19)		R14	
MSGDONE2	If the user is not signing off, branch..... >	CLOSMSG (MA04)			
	Set restart flag and branch to detach the writer task..... >	CLOSMSG (MA04)	LCBDONE(IPW$DLC)		
MSGSTOP	Release output buffer using IPW$RLM.			R1	
	Branch to close message writer..... >	MSGDONE2			

Labels	Chart MA04: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
KILLMSG GETMSG	If there are no messages left, return to caller via register 6.			R6	
	The function indicator is set to X'10' to indicate that a BSC message is to be obtained, and a link is made to the remote message service in the POWER/VS nucleus. If called by the message writer, return via register 6.		TCMW(IPW$DTC)	R6	IPW$$RMS Chart AD
DELMSG	The function indicator is set to X'18' to indicate that a BSC message is to be deleted, and a link is made to the remote message service in the POWER/VS. nucleus. If called by the message writer, return via register 6.		TCMW(IPW$DTC)	R6	IPW$RMS Chart AD
	Otherwise, branch to try again..... >	KILLMSG (MA04)			
	Close Routine LCB				
CLOSEXIT	Registers 7 and 8 are set up as base registers.			R7,R8	
CLOSEX1	Register 9 is set up as LCB pointer.			R9	
	If end of chain, branch to find correct LCB, if not found yet..... >	CLOSEX1 (MA04)			
	Register 4 is set up as BCA pointer.			R4	
	If the current task is a reader task, branch to close the reader task.... >	CLOSRDR (MA01)			
CLOSEX3	Clear the output record count to prevent an attempt to flush the output buffer and branch for punch task..... >	CLOSEX2 (MA04)	IPERECNT		
	Reset hot list switch. Branch to close the list..... >	CLOSOUT3 (MA04)	LCBOUT(IPW$DIC)		
CLOSEX2	Reset hot punch switch.		LCEOUT(IPW$DLC)		
CLOSOUT3	Test for MSGWRTR If not go to..... >	CLOSOUT4			
	Clear register 13.			RD	
	BAL to..... > (Continue after CLOSOUT4)	MSGCLOSE			
CLOSOUT4	BAL to..... >	CLOS (MA19)		RE	IPW$CLI
	The current list/punch forms ID in the TCB general work area is saved.				
CLOSMSG	Detach the task..... >	PDET (MA01)			

Labels	Chart MA05: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
	Open Logical Reader/Writer Interface				
OPEN	An IPW$OLI call is issued to open the logical reader/writer interface.				IPW$OLI Chart AC
	Make the line manager the owner of the work space just obtained to prevent release by IPW$$TR.				
MSGOPEN	(Alternative entry point to bypass opening the logical interface)				
	The appropriate code table is moved to the BCA, and the appropriate buffer size is loaded in register 1.		BCACODE(IPW$DBC)	R1	
	If the current task is a reader task, 64 bytes are added to the buffer space count in register 1 to provide enough space for an input record (80 bytes) in the same buffer.			R1	
	The buffer space is reserved.				IPW$RSW Chart AC
	Real and virtual buffer space addresses, as passed in register 0 and 1, respectively, are saved.		RLBUFAD(IPW$DBC)		
	If the current task is not a reader task, branch around reader CCW string preparation..... >	MBPPOPEN (MA05)			
	Otherwise, the remote acknowledgement character in the BCA is set to odd acknowledgement.		LCBACK(IPW$DBC)		
	A link is made to initiate CCW chain for read operations..... >	MCCWINIT (MA20)		R6	
	Link to initiate read of first block of data..... >	MEXCP (MA22)		R6	
	Branch back to caller.			R14	
MBPPOPEN	The acknowledgement is saved.		LCBACK(IPW$DBC)		
	A link is made to initiate CCWs for write operations..... >	MCCWINIT (MA20)		R6	
	The maximum record count in the BCA is set to 2 (1 for the 3741).		TPBMXREC (IPW$DBC)		
	If the terminal has the multiple record feature, the maximum record count is set to 7.		TPBMXREC (IPW$DBC)		
	The data buffer address is loaded in register 3.			R3	
	If the terminal is a 2780 type, branch..... >	MBN7OPN1 (MA06)			

Labels	Chart MA06: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
	Otherwise, data buffer address in register 0 is decremented by one, so that register 0 now points to the byte immediately preceding the data buffer. The maximum record count is set to 255. If terminal type is 3780, branch to >	MBN7OPN1 (MA06)		R0	
	Reset component selection and wait for completion.				IPW$WFC Chart AA
	If task has been stopped immediately, branch to..... >	CLOSEX3			
GWJDJO	If a buffer was saved, it is released now. Branch to..... >	FORMCLOS (MA18)			IPW$RLW Chart AC
MBN7OPN1	The data buffer address in register 1 is now stored in the carriage control address and the remote data address of the BCA. If the device to be opened is not a punch, branch..... >	MBPROPEN (MA07)	TPBLCCAD (IPW$DBC) TPBFDATA (IPW$DBC)		
	Otherwise, component select in the data buffer is set to device 2 (punch device). If the transparency feature is not present, branch back to caller to bypass punch select. Otherwise, if 2780, branch..... >	MBN7OPN2 (MA06)		R14	
	If not, initialize maximum record count at 6. If 2770 buffer size is 512 (which accommodates 6 records) branch back to caller. Otherwise, maximum record count is reset to 3, and, if 2770 buffer size is 256 (3 records), branch back. Otherwise, maximum record count is set to 1, and branch back to caller.		TPBMXREC (IPW$DBC) TPBMXREC (IPW$DBC) TPBMXREC (IPW$DBC)	R14 R14 R14	
MBN7OPN2	If maximum record count higher than 4, it is set to 4. 2780 punch select is set up in the data buffer. The length of the punch select record just set up (3) is moved to the appropriate CCW. Current remote output record count is set to 4 (maximum record count) to force actual execution of the punch select.		TPBMXREC (IPW$DBC) IOBCCW4 (IPW$DBC) TPBRECNT (IPW$DBC)		

Labels	Chart MA07: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
	If no buffer was saved, return to caller.			R14	
	Write component select, and wait for completion.				IPW$WFC Chart AA
	If an error occurred, branch to.... >	GWJDJO (MA06)			
	Otherwise, return to caller.			R14	
MBPROPEN	Carriage control consolidation is prohibited by invalidating the last remote command code entry. If no horizontal format control is supported on the terminal printer, branch back to caller to bypass tab setting.			R14	
	Otherwise, a tab initialization record is built: • Horizontal tab record ID is moved in. • Every 10 columns a tab character (X'05') is inserted. • Using register 15 as a work register, the data buffer address in the BCA is incremented to point past the tab initialization record. • The current remote output record count is set to 1. • Immediately following the non-data (tab initialization) part of the record (the location TPBFDATA points to currently), a 'new line' character (X'15') is inserted. • The block length of the tab initialization record is calculated in register 15 and stored in the appropriate CCW.		TPBFDATA(IPW$DBC) TPBRECNT(IPW$DBC) IOBCCW4(IPW$DBC)	R15 R15	
	Branch back to caller.			R14	
	Remote Terminal Read Routine				
TPGET	CCB address (= BCA address) is loaded in register 1, and an IPW$WFC call is issued to wait for I/O completion if I/O was started on the terminal.			R1	IPW$WFC Chart AA
	On I/O completion, the address of the next (logical) record is loaded in register 15.			R15	
	If real end of transmission has been received, branch..... >	PUTL (MA01)			
	If terminal has the EBCDIC feature, branch..... >	MBHGNASC (MA08)			
	If the input data has already been translated, branch to..... >	MBHGNASC (MA08)			

Labels	Chart MA08: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
	Otherwise, the input data is translated to EBCDIC:				
	• Data start address is loaded in register 1.			R1	
	• Data length is loaded in register 2.			R2	
	• The maximum string length to be translated by one translate instruction (256) is loaded in register 0.			R0	
LOOPTR	• If the length of the string remaining to be translated has become less than 256, the remainder is translated outside this loop, by branching to..... >	LASTTR (MA08)			
TRTEX	• Otherwise, 256 bytes are translated.				
	• String pointers register 1 (address) and register 2 (remaining length) are updated, and branch back to translate next string..... >	LOOPTR (MA08)		R1 R2	
LASTTR	Translate last block.				
MBHGNASC	If the first data character is STX (start of text), indicating non-transparent text, branch..... >	MBHGNTRS (MA08)			
	The length of the first buffer is checked.				
	If terminal type 2780, branch..... >	MBHGTRSP (MA08)			
	If the data pointer in the BCA does not point to the start of the data block, it is decremented by 3.			R15	
MBHGTRSP	80 data bytes are moved to the caller's buffer, pointed to by register 3.				
	Register 15 is incremented by 83 to point to the next data record.			R15	
	Branch to the GET exit routine..... >	MGSOTEST (MA10)			
MBHGNTRS	For non-transparent input, data pointer register 15 is decremented by one to point to the DLE character.			R15	
	If register 15 still does not point to the STX character it is decremented once more by one.			R15	
MB2780G	A translate and test table is initialized to scan for a termination character (end-of-medium or inter-record separator).				
	If none found, branch..... >	MGHGTRSP (MA08)			

Labels	Chart MA09: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
	If a termination character is detected, the 80-bytes buffer of the caller is blanked out, and column pointer register 6 is initialized with the start address of the caller's buffer.			R6	
MBHGNFLD	Using register 5 as a work register, the length of the non-blank field of the input record is calculated in register 2, and this non-blank field, if any, is moved to the caller's buffer.			R5 R2	
	If the termination character is not an IGS character, implying that no compressed blanks are present, branch..... >	MBHGNREC (MA10)			
	Otherwise, register 15 and 6 are incremented to point immediately past the current input record, and immediately past the non-blank area in the caller's buffer, respectively.			R15,R6	
	If the terminal code type is ASCII, the byte containing the compressed blank count is translated to EBCDIC, and the compressed blank count (6 low order bits of this byte), is loaded in register 2.			R2	
	Register 6 is incremented to allow for this number of compressed blanks.			R6	
	Register 2 is set up to point to the last available byte of the caller's buffer.			R2	
	The address of the last byte of the uncompressed record as calculated in register 6 is subtracted.			R6	
	If this results in a negative value, implying that the uncompressed record will not fit in the caller's buffer, branch..... >	MBHGIREC (MA10)			
	Otherwise, a string of the input record, starting past the termination character that stopped the last scan, and its accompanying compressed blank count, and as long as will still fit in the blank part of the caller's buffer, is scanned for the next termination character.				
	If found, branch back..... >	MBHGNFLD (MA09)			
	Otherwise, move the last field to the caller's buffer.				

Labels	Chart MA10: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
MBHGIREC	If the uncompressed record will fit in the caller's buffer, input data pointer register 15 is incremented with the residual byte count still in register 2.			R15	
MBHGNREC	Register 15 is incremented with the number of characters to be skipped when scanning for termination character.			R15	
MSGOTEST	The updated data pointer is stored back in the BCA.		TPBFDATA (IPW$DBC)		
	The translate and test table used is restored.				
	If the input buffer has not yet been exhausted, return.			R14	
	Otherwise, link to initiate read of next block..... >	MEXCP (MA22)			
	Return to caller.			R14	
	Remote Terminal Put Routine				
TPPUT	Clear work registers 3 and 6.			R3,R6	
	Get command code to be analyzed in register 6, and branch to the proper routine via the following branch table.				R6
KTFORK	R6= 0 : NOP or ignored command.... >	(R14)			
	R6= 4 : Forms change..... >	MOUNTFRM (MA18)			
	R6= 8 : Data handling CCW..... >	MPNIMED (MA11)			
	R6=12 : Printer space immediate... >	MPNIMM (MA10)			
	R6=16 : Printer skip immediate.... >	MPNIMM (MA10)			
	R6=20 : Punch only commands..... >	MPUNCH (MA10)			
MPUNCH	Check if this is a punch task. If so, branch to handle data..... >	MPNIMED (MA11)			
	Otherwise, return to caller..... >	(R14)			
MPNIMM	Check if this is a punch task. If so, return to caller..... >	(R14)			
	Check if the command can be combined. If so, branch to..... >	MPSETCC (MA17)			

Labels	Chart MA11: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
	Check if this is a space command. If so, branch to..... >	MPDUMMY (MA11)			
	Set immediate command.		TPBLCCC		
	Check if it is the same channel. If so, ignore redundant skip, and return to caller..... >	(R14)			
MPDUMMY	Indicate a length of 1 in register 3, and the address of blank data in register 1, and branch to write a dummy line for control..... >	MPUT3 (MA11)		R3,R1	
MPNIMED	Insert the data address in register 1 and the data length in register 3.			R1,R3	
MPUT	Indicate the maximum punch length of 80 in register 0.			R0	
	Check if this is a punch task. If so, branch to..... >	MPUT1 (MA11)			
	Indicate the maximum print length (from LCBPRLEN) in register 0.			R0	
	branch to..... >	MPUT2 (MA11)			
MPUT1	Set special command code.		TCCC(IPW$DTC)		
MPUT2	Check if the data length is correct. If so, branch to..... >	MPUT3 (MA11)			
	Set the data length to the maximum (from register 0).				
MPUT3	If the current output record count has already reached its maximum value, branch to empty the buffer first..... >	MPFLUSH (MA18)			
	Otherwise, the address of the first available byte in the buffer is loaded in register 15.			R15	
	If the remote output device is not a punch device, or if the terminal has not got the transparency feature, branch to process non-transparent output..... >	MBHPRINT (MA13)			
	If the buffer is currently empty, branch..... >	MBHPUN1 (MA12)			
	If the terminal is a 2780 type, branch..... >	MBHPUNEW (MA12)			
	Otherwise, first available buffer space pointer register 15 is decremented by 6.			R15	

Labels	Chart MA12: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
	Using registers 5 and 2 as work registers, the byte count of the appropriate CCW is incremented by 80 (record length of the punch record to be added to the buffer), and branch to move the card to the buffer..... >	MBHPUMOV (MA12)		R5,R2	
MBHPUN1	A link is made to initialize CCW chain for write operations..... >	CCWINIT2 (MA20)		R6	
	Record address is reloaded in register 1.			R1	
	Register 15 is set up to point to the beginning of data: • For non 2780 terminal, 12 bytes past buffer start. • For 2780 terminal, 36 bytes past buffer start.			R15	
MBHPUNEW	The first 6 characters of the data area are set up as punch control characters.				
MBHPUMOV	The next 79 characters (to become columns 2-80 of the card to be punched) are blanked out and the record is moved into the buffer.				
	If the card just moved is the first in this buffer, branch..... >	MBHPU1ST (MA13)			
	Otherwise, the record just moved is added logically to buffer contents:				
	• Register 0 is initialized with the normal punch record length (86: 6 bytes punch control + 80 bytes data). • If terminal is not a 2780, branch..... >	MBHPUPDT (MA13)		R0	
MLWA	• Using registers 2 and 5 as work registers, the real data address is moved to the appropriate CCW, • The CCW minus data length field is moved to the 'last remote carriage control' CCW field, and data length is stored in this CCW from register 0.		IOBCCW4(IPW$DBC)	R2,R5	
	• Using registers 1 and 2 as work registers, the next CCW is set up as a TIC CCW transferring to the write ETB CCW. • The address of this CCW is stored in the BCA.		TPBLCCAD (IPW$DBC)	R1,R2	

Labels	Chart MA13: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
MBHPUPDT	• Data pointer register 15 is incremented to point to the first available position in the buffer. • This updated data pointer is stored back in the BCA. • Using register 1 as a work register, the current remote put count in the BCA is incremented by one.		TPBFDATA (IPW$DBC) TPBRECNT (IPW$DBC)	R15 R1	
	Return to the caller.			R14	
MBHPU1ST	Length of first card in buffer (82: 2 bytes of control information plus 80 bytes of data), is loaded in register 0.			R0	
	Data pointer register 15 is set to point 4 bytes past the beginning of the buffer.			R15	
	Branch to process..... >	MLWA (MA12)			
MBHPRINT	Check for second time through. If so, skip compression and branch to. >	MBHPRNC (MA16)			
	Using register 2 as a work register, all characters from X'01' through X'3F' are translated to binary zero.			R2	
	If this is a 3741 printer task, branch to..... >	MBH3741		R9	
	Translate illegal characters and branch..... >	MBHRNM			
BMH3741	Do not translate 3741 control characters.				
MBHRNM	Parameter registers 2 and 6 for the subsequent compression routine are set up:				
	• Register 2 is loaded with the address of the uncompressed record.			R2	
	• Register 6 is loaded with the length of the uncompressed record.			R6	
	If punch device, branch past compression routine..... >	MBHPRNHT (MA15)			
	If horizontal format control not supported, branch..... >	MBHPRNHT (MA15)			
	Otherwise, register 2 is decremented by one.			R2	
	In this loop all trailing blanks in one 10 column tab interval will be replaced by one tab control character (X'05').				

Labels	Chart MA14: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
MBHPRTGP	The remaining character count in register 6 is decremented by 10.			R6	
	If no characters left to be scanned, branch out..... >	MBHPRNC (MA16)			
	Otherwise, increment data pointer register 2 by 10 to point to the end of the next tab interval.			R2	
MBHPRCON	In this inner loop, the tab interval is scanned backwards for the last non-blank in this tab interval:				
	• Count register 5 is initialized at 10.			R5	
MBHPRTBL	• If the current character is non-blank, branch..... >	MBHPRNBL (MA14)			
	• Otherwise, register 2 is decremented by one to point to the previous character, and, if the current tab interval has not been exhausted yet, branch back to continue scan..... >	MBHPRTBL (MA14)		R2 R5	
MBHPRNBL	The number of trailing blanks in this tab interval to be compressed is calculated in register 5 (negative for the sake of simplicity of calculation).			R5	
	If no blanks to be compressed, branch to scan next tab interval..... >	MBHPRTGP (MA14)			
	Otherwise, register 2, currently pointing to the last non-blank character, is incremented by one to point to the first trailing blank, which will be replaced by a TAB character.			R2	
	If only one trailing blank, branch to scan next tab interval..... >	MBHPRTGP (MA14)			
	Otherwise, the record length in register 3 is corrected to allow for the number of blanks to be compressed, and the TAB character is inserted.			R3	
	Using register 5 as a work register, the line is compressed by moving the part of the line not yet scanned back to immediately behind the TAB character just inserted, causing any blanks to be overlaid.			R5	
	Branch back to scan next tab interval..... >	MBHPRTGP (MA14)			

Labels	Chart MA15: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
MBHPRNHT	If the compress feature is not present, branch.....	> MBHPRNC (MA16)			
MBHCLOOK	The remainder of the line is now compressed. If data pointer register 2 does not point to a blank, branch.....	> MBHCINCR (MA15)			
	Otherwise, the maximum blank field length (63) is loaded in register 3, and, the address of the first blank is loaded in register 5.			R3,R5	
MBHCSCNB	If the character currently examined is not blank, branch.....	> MBHCENDB (MA15)			
	Otherwise, register 5 is incremented by one to point to the next character, and, if the line is exhausted, register 3 is set to end of line plus one, and branch.....	> MBHCLREC (MA16)		R5 R3	
	Otherwise, if maximum blank field length to be compressed is not yet exceeded, branch.....	> MBHCSCNB (MA15)		R6,R3	
MBHCENDB	If the end of the blank field or the maximum length has been reached, register 3 is set up to contain the length of the blank field. If not more than two blank characters, branch to bypass compression.....	> MBHBUMP (MA15)		R3	
	Otherwise, the field is compressed: • The first blank is replaced by an IGS character, (X'1D'). • The blank field count is stored in the second blank. • The part of the line not yet scanned is backed over the remaining blank(s) of the blank field. If ASCII code, pre-translate the one-byte blank length field.				
MBHBUMP	The length count in register 6 is for the subsequent decrement.			R6	
MBHCINCR	Data pointer register 5 is incremented by one to point to the next character, and, if the line to be scanned is not yet exhausted, branch back to find next blank field.....	> MBHCLOOK (MA15)		R5 R6	
	Otherwise, register 3 is set to point immediately past the scanned line.			R3	

Labels	Chart MA16: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
MBHCLREC	(Compressed) line length is calculated in register 3.			R3	
MBHPRNC	The updated record length is stored.		TCRL(IPW$DTC)		
	The carriage control address is saved in the BCA.		TPBLCCAD (IPW$DBC)		
	If terminal type 2780, branch..... >	MB2780PP (MA18)			
	Otherwise, the 2770 byte in register 2 is initialized at 1.			R2	
	Software buffer size in register 0 is converted to hardware buffer size.			R0	
	Register 5 is set to 1 to indicate (2770 punch) displacement.			R5	
MB2770PR	Print displacement in register 5 is initialized at 3.			R5	
	If this is a 3741 punch task, register 5 is initialized to 1 to prevent 2780 carriage control.			R5	
	The print/punch displacement in register 5 is copied to register 6.			R6	
	If current record is not the first in the buffer, branch..... >	MBHPRNXT (MA16)			
	Otherwise, an STX character (X'02') is moved to the beginning of the data buffer.				
	The initial byte count in register 2 is stored in the appropriate CCW.		IOBCCW4(IPW$DBC)		
MBHPRNXT	The record length is loaded in register 2.			R2	
	If device is not 3741, branch..... >	MBHPRVAR (MA16)			
	If device is punch, set length to 80 and branch..... >	MBHPRVAR (MA16)			
	Otherwise, record length in register 2 is set to value of LCBPRLEN.			R2	
MBHPRVAR	Actual record length is calculated in register 5.			R5	
	Byte count of block already in buffer is added to obtain total block length.			R5	
	If higher than maximum block length, branch to flush buffer..... >	MPFLUSH (MA18)			
	Otherwise, the updated block length is stored back in the write CCW.		IOBCCW4(IPW$DBC)		

Labels	Chart MA17: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
	If the record length as determined by the device record width is equal to the actual record length, branch to bypass blanking out the remainder of the line..... >	MBHPPVAR (MA17)			
	Otherwise, the remainder of the line is blanked out.				
	Register 3 is decremented by one to contain record length in machine format.			R3	
MBHPPVAR	Buffer free data pointer register 15 is incremented by the appropriate print/punch displacement to allow for control information.			R15	
	The data record is moved to the buffer.				
	Data pointer register 15 is updated, and stored back in the BCA.		TPBFDATA (IPW$DBC)	R15	
	Using register 1 as a work register, the current output record count is incremented by one.		TPBRECNT (IPW$DBC)	R1	
	An IRS character (X'1E') is moved to the first available position of the data buffer.				
	If device is a 2770 punch, return to caller.			R14	
MPSETCC	The channel command is loaded in register 1, saved in the BCA, and divided by 8 to create an index for later use.		TPBLCCC(IPW$DBC)	R1	
MPSETBH	Using register 2 as a work register, an 'escape' character (X'27') is inserted behind the carriage control character.			R2	
	If terminal is 2780, the index in register 1 is incremented by 32.			R1	
	The index is now used to insert the appropriate carriage control character in register 1 from a carriage control table.			R1	
	This carriage control character is now stored in the record.				
	If the code type is not ASCII, return to caller.			R14	
	Otherwise, the control character is converted to ASCII by flipping the high-order bit. Return to caller.			R14	

Labels	Chart MA18: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
MB2780PP	An ITB character (X'1F') is inserted after the previous record.				
	Byte count in register 2 is initialized at 0.			R2	
	2780 maximum block length is set up in register 0: 400 minus 1 for the ETB character and 1 for an LRC character for each logical record in the buffer.			R0	
	Branch to..... >	MB2770PR (MA16)			
	Remote Output Forms Change Routine				
MOUNTFRM	Indicate forms change in BCA.		BCAFLAGS(IPW$DBC)		
	Set command code to NOP.		TCCC(IPW$DTC)		
	Check for component select only. If so, ignore it.		TPBRECNT(IPW$DBC)		
MPFLUSH	Link to the appropriate I/O routine and wait for completion.				IPW$WFC Chart AA
	If the task has already been stopped, return to caller.			R14	
	If no intervention is required, branch to simulate re-entry..... >	TPPUT (MA10)			
	If this is the message writer, branch to detach..... >	MSGDONE (MA03)			
	If no ACK0, ACK1 or NACK response was received, branch to..... >	FORMCLOS (MA18)			
SVBF	Check if a buffer was already saved. If not, save buffer address, and if not 2780 and transparent, also data length.		LCBRLBUF(IPW$DLC) LCBWRCNT(IPW$DLC)		
FORMCLOS	The return address is saved, except if from FORMCLOS via MSGOPEN.			R5	
	Link to close the list task..... >	MBHCLOSE (MA19)			
	Issue an IPW$WFI macro to wait for an ACK from the remote printer.				IPW$WFI Chart AA
	If an immediate stop is requested, branch to..... >	SCBF (MA18)			
MSGBAL	If no buffer was saved, branch to proceed..... >	TPPUT (MA10)			
	Otherwise, release the work space.				IPW$RLW Chart AC
	Move the correct code table into the BCA.		BCACODE(IPW$DBC)		
SCBF	If a buffer was saved, release the work space.			R14	IPW$RLW Chart AC
	Return to caller.			R14	

Labels	Chart MA19: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
	Remote Terminal Close Routine				
CLOS	An IPW$CLI call is issued to close the logical interface.				IPW$CLI Chart AC
MSGCLOSE	If the device is not a reader, branch to..... >	LSTCLOSE (MA19)			
	If the CCW read sequence is no longer on, return to caller.			R14	
	Branch to..... >	RDRCLOSE (MA19)			
LSTCLOSE	If the CCW write sequence is no longer on, or if it is not for this task, return to caller.			R14	
	If EOT has been received, branch... >	MBHCLOSE		R4	
	If the output buffer is empty, branch to..... >	WC40			
	The CCB (=BCA) address is copied into register 1, and a IPW$$WFC is issued to wait for I/O completion.			R4,R1	IPW$WFC Chart AA
	If a message is printing, branch... >	WC40		R13	
	If the task has been stepped immediately, branch..... >	MSGSTOP			
	Otherwise the write ETB CCW is changed to write ETX.		IOBCCW5 (IPW$DBC)		
	Link to write ETX..... >	MEXCP (MA22)			
	Link to write last buffer..... >	MEXP (MA22)			

Labels	Chart MA19.1: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
RDRCLOSE	Copy the CCB pointer and wait for I/O completion.			R1	IPW$WFC Chart AA
MBHCLOSE	If this is a reader, branch to..... >	MBHCLOS1			
	If this is not a switched line, branch to..... >	MBHCLOS1			
	If line is not being signed off, branch to..... >	MBHCLOS1			
	Set up disconnect sequence..... >	MCCWINIT (MA20)			
	Branch to..... >	WREOT0			
MBHCLOS1	Link to initialize the CCW prepare sequence..... >	MCCWINIT (MA20)			
	If this is a list or punch task, branch to write EOT..... >	WREOT (MA19)			
	If any messages are pending, or if there is list/punch output, convert the EOT CCW to an enquiry CCW.		IOBCCW4(IPW$DBC)		
WREOT	Link to issue applicable terminal I/O..... >	MEXCP (MA22)			
WREOT0	If this is a reader task, or if a buffer was saved, release the work space.				IPW$RLW Chart AC
	Return to caller.			R14	

Labels	Chart MA20: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
	Remote Terminal CCW Setup This routine initiates the CCW string. Upon entry: • Register 0 points to the first CCW to be executed. • Register 1 contains the type of string to be initialized in its low-order byte. (X'80'=read, X'A0'=write, X'40'=prepare).			R0 R1	
MCCWINIT CCWINIT2	The (virtual) first CCW address, passed in register 0, is converted to real, and stored in the IOB START and IOB RESTART fields in the BCA. The CCW sequence type requested, as passed in register 1, is stored in the BCA. The entire string of 6 CCWs is set to binary zero. The real address of the BCA is calculated in register 1. If the CCW sequence requested is not disconnect sequence, branch to..... >	CCWINIT3	IOBSTART (IPW$DBC) IOBRESTR (IPW$DBC) MSEQTYPE (IPW$DBC) IOBCCW1(IPW$DBC)	R0 R1	
	Otherwise, using register 0 as a work register, the disconnect sequence CCW sequence is built: • The address of the DLE-EOT is stored in the write CCW. • The 3 CCW 's are completed by ORing from MDISCCW.			R0	
CCWINIT3	If CCW sequence requested not prepare sequence, branch..... >	MCCWBRDW (MA21)			
	Otherwise, using register 0 as a work register, the prepare sequence CCW sequence is built: • The address of the mode control byte is stored in the set mode CCW. • The address of the enquiry cnaracter is stored in the write enquiry CCW. • The address of the response field is stored in the read response CCW. • The 5 CCWs are completed by ORing the other CCW fields from MDISCCW.		IOBCCW2(IPW$DBC) IOBCCW4(IPW$DBC) IOBCCW5(IPW$DBC)	R0	
	Return to caller.			R6	

Labels	Chart MA21: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
MCCWBRDW	Using register 0 as a work register, the CCB sequence is built:			R0	
	• The address of the ETB sequence is stored in the write ETB CCW.		IOBCCW5(IPW$DBC)		
	• Real address of the data buffer area is stored in the write data CCW.		IOBCCW4(IPW$DBC)		
	• The proper chaining is set in the write CCW.		IOBCCW4(IPW$DBC)		
	• If not write sequence requested, branch to complete read sequence initialization..... >	MCCWBSCR (MA21)			
	• Otherwise, the address of the enquiry character is stored in the write enquiry CCW.		IOBCCW2(IPW$DBC)		
	• The address of the response field is stored in both read response CCWs.		IOBCCW3(IPW$DBC) IOBCCW6(IPW$DBC)		
	• The other CCW fields are initialized by ORing from field MENBCCW.				
	Return to caller.			R6	
MCCWBSCR	Using register 0 as a work register, the initialization of the read CCW sequence is completed:			R0	
	• The first CCW is ORed in from field MENBCCW.		IOBCCW1(IPW$DBC)		
	• The real address of the data buffer is stored in the read data CCW.		IOBCCW3(IPW$DBC)		
	• The address of the response field is stored in the write response CCW.		IOBCCW2(IPW$DBC)		
	• The buffer length is stored in the read data CCW.		IOBCCW3(IPW$DBC)		
	• Write response and read data CCWs completed by ORing in the remaining fields from MWRSPCCW.		IOBCCW2(IPW$DBC) IOBCCW3(IPW$DBC)		
	• Write and write ETB CCWs are completed by ORing the remaining fields from MWRITCCW.		IOBCCW4(IPW$DBC) IOBCCW5(IPW$DBC)		
	• Last read CCW is copied from the third CCW.		IOBCCW6(IPW$DBC)		
	• Write text sequence is indicated in last CCW.		IOBCCW6(IPW$DBC)		
	Return to caller.			R6	

Labels	Chart MA22: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
	Remote Terminal I/O Interface				
MEXCP	The remote next acknowledgement character is inverted to send or expect.		LCBACK(IPW$DBC)		
	The remote response control block is set up with the inverted acknowledgement.		LCBRCB(IPW$DBC)		
	If code type is EBCDIC, branch to bypass translate..... >	MEXCP1 (MA22)			
TRLOOP EXTRT	Otherwise, using registers 0, 1 and 2 as work registers, the buffer contents are translated to ASCII.			R0,R1 R2	
MERREXCP	(Alternative entry point, to skip ACK swap, and translation)				
MEXCP1	The sense byte is set to zero.		IOBSENS0		
	If the task has not been stopped, branch to..... >	MEXCP2			
	Post LM task control block ECB as well as own TCB-ECB and return to calling routine..... >	R6	TCEB		
MEXCP2	Excp parameter register 1 is loaded with the CCB address, and an SVC 0 is issued to initiate terminal I/O.			R1	
	Return to caller.			R6	

Labels	Chart MA22.1: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
	Remote Command Routine				
COMMAND	The sequence ID (Bytes 79 and 80 of the input record), is inserted in the 2 high-order bytes of register 0 for PMESS.			R0	
	Register 1 is initialized pointing to the last byte of the RJE prefix ('* ..').			R1	
	The command is translated to change all characters <X'40' to X'40'. The command is made to upper case. The subsequent loop will scan the command for its operation code.				
SCANCODE	Register 1 is incremented by one to point to the next column.			R1	
	If next column is blank, branch to scan next..... >	SCANCODE (MA22)			
	Otherwise, if SIGNON command, branch to signon processor..... >	MSIGNON (MA26)			
	If not SIGNON command, and nobody currently signed on, branch to process invalid signon..... >	NOSIGNON (MA23)			
	If SIGNOFF command, branch to process signoff..... >	CSIGNOFF (MA23)			
	If START command, branch to start command processor..... >	MSTART (MA30)			

Labels	Chart MA23: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
	If STOP command, branch to stop command processor..... >	MSTOP (MA31)			
	If GO command, branch to go command processor..... >	MGO (MA33)			
	If SETUP command, branch to setup command processor..... >	MSETUP (MA33)			
	The address of the start command processor is loaded in register 12.			R12	
	If 'S' command, branch to..... >	SHORTCMD			
	The address of the STOP command processor is loaded into register 12.			R12	
	If 'P' command, branch to..... >	SHORTCMD			
	The address of the GO command processor is loaded into register 6.			R6.	
	If 'G' command, branch to..... >	SHORTCMD			
	The address of the SETUP command processor is loaded into register 6.			R6.	
	If 'U' command, branch to..... >	SHORTCMD			
	If any other command, the command processor is invoked:				
	• Register 1 is set to point to the statement begin,			R1	IPW$RSR Chart AB
	• Register 0 is set to point to the TCEB, and an IPW$ICP call is issued to pass the command to the command processor.			R0	IPW$ICP Chart GD
	On return, register 1 is pointed to the TCEB, and an IPW$WFC call is issued to wait for command processing completion.			R1	IPW$WFC Chart AA IPW$RLR Chart AB
	The TCEB is unposted.		TCEB(IPW$DTC)		
	Branch back to resume input processing..... >	GETLOOP (MA01)			

Labels	Chart MA23.1: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
NOSIGNON	Message 1R19I is issued:				
	• Register 1 set to point to the message text,			R1	
	• a dummy remote ID (X'FE') is set in the LCB, and		LCBREMID (IPW$DLC)		
	• a link is made to the message routine... >	PMESS (MA24)		R15	
	On return, the remote ID is cleared.		LCBREMID (IPW$DLC)		
CSIGNOFF	A link is made to force signoff.... >	MABORT (MA24)			
EOFRDR	The LCB is tested for reception of end of text character.				
	If ETX not received, branch to bypass message..... >	NOT14I (MA24)			
	Otherwise, register 1 is set to point to message 1R14I, and a link is made to the message routine to issue the message..... >	PMESS (MA24)		R1 R15	

Labels	Chart MA24: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
NOT14I	Link to close the reader (except for the logical interface)..... >	MBHCLOSE (MA19)		R14	
	Wait for enquiry from terminal. (Note that the task is not detached and that the logical interface is kept open.)				IPW$WFI Chart AA
	If immediate stop required, branch to..... >	PUTL (MA01)			
	On interrupt, link to reopen the reader (except for the logical interface) ..... >	MSGOPEN (MA05)		R14	
	Branch to get next record..... >	GETLOOP (MA01)			
SHORTCMD	Bump register 1 to point to blank after short command, and branch to routine in register 6..... >			R6,R1	
	Remote Message Routine				
PMESS	Register 0 is set to zero to clear the sequence ID in its 2 high-order bytes.			R0	
PMESS2	The current remote ID is copied from the LCB to the low-order byte of register 0.			R0	
	The function indicator is set to X'28' to indicate that the message is to be added to the message queue, and a link is made to the remote message service in the POWER/VS nucleus.		TCMW(IPW$DTC)		IPW$RMS Chart AD
	Return to caller.			R15	
	Remote Signoff Subroutine				
MABORT2	Reader TCB address in LCB is loaded in register 15, and link to cancel reader task..... >	STOPRDWR (MA26)		R15 R14	
MABORT	Writer TCB address is loaded in register 15, and link to cancel writer task..... >	STOPRDWR (MA26)		R15 R14	
	If the terminal disconnect flag is on, indicating that this subroutine has already been in control, branch back to caller.			R6	
	Otherwise, the terminal disconnect flag is set on to request terminal disconnection.		LCBFLAGS (IPW$DLC)		
	List and punch are marked 'stopped'.		LCBLIST(IPW$DLC) LCBPUNCH(IPW$DLC) LCBOUT(IPW$DLC)		
	If nobody currently signed on, return to caller.			R6	

Labels	Chart MA25: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
MABNOBUF	If signoff forced by I/O error, branch around sending signoff message..... >	PLOGMESS (MA25)			
	Otherwise, the address of message 1R18I (signoff forced) is loaded in register 1.			R1	
	If signoff caused by excessive idling time, branch to issue message 1R18I..... >	SOFFMESS (MA25)			
	Otherwise, the signoff code in the LCB is set to central stop.		LCBSCODE (IPW$DLC)		
	The address of message 1R17I (line stopped) is loaded in register 1.			R1	
	If the line has been stopped, branch to issue message 1R17I..... >	SOFFMESS (MA25)			
	Otherwise, the signoff code is set to 'normal signoff'.		LCBSCODE (IPW$DLC)		
	The address of message 1R16I is loaded in register 1.			R1	
	The current remote ID is copied into the message to be issued.				
SOFFMESS	Link to message routine to issue message..... >	PMESS (MA24)		R15	
PLOGMESS	Issue message locally.				
	An IPW$RDC call is issued to obtain the time in register 1.				IPW$RDC Chart AG
	Signoff time is stored in account record.		LCBTIMOFF (IPW$DLC)		
	The account record is written: • The account record address is loaded in register 1, • the account record length (48 bytes) is loaded in register 0, and • an IPW$PAR call is issued to write account record.			R1 R0	IPW$PAR Chart FA IPW$RSW Chart AC IPW$RLW Chart AC
	Return to caller.			R6	

Labels	Chart MA26: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
STOPRDWR	If the reader/writer TCB address in register 15 is zero, indicating that reader/writer is not active any more, return.			R14	
	If task is not waiting for TP-I/O-interrupt, return. Otherwise, mark task 'dispatchable' and return.		TCSF(IPW$DIC)	R14	
TA50	If it is an unrecoverable I/O error, branch to..... >	TA60		R9	
	If task has been stopped, return to >	R6		R6	
	Otherwise, post unrecoverable I/O error to prevent more I/O scheduled.		LCBSCODE IPW$DLC)		
TA60	Stop task by inserting stop code in task status field.		TCTT		
	Post BCA traffic bit.		CCBCOM1 (IPW$DBC)		
	Return..... >	R14		R14	
	Remote Signon Processor				
MSIGNON	Register 5 is initialized with the address of the SIGNON card.			R5	
	Data pointer register 1, currently pointing at the SIGNON command code, is saved.		LCBWORK(IPW$DLC)		
	A link is made to the signoff routine to force signoff of any remote ID currently signed on..... >	MABORT (MA24)		R6	
	The remote ID field in the LCB is initialized with a dummy remote ID (X'FE').		LCBREMID (IPW$DLC)		
	16 Bytes of user information are moved from columns 61 - 76 of the SIGNON card to the LCB.		LCBUSER(IPW$DLC)		
	Data pointer register 1 is restored, and incremented to point to the first blank after the SIGNON command code.			R1	
	The subsequent loop scans the statement for the first SIGNON operand.				

Labels	Chart MA27: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
REMSTART	Register 1 is incremented by one to point to the next column. If blank as well, branch back to continue scan..... >	REMSTART (MA27)		R1	
	Otherwise, register 5 is initialized with characters '00' in the 2 low-order bytes.			R5	
	Register 5 is to contain the character representation of the remote ID about to be checked.			R5	
	Register 3, to be used in the subsequent scanning loop of the remote ID operand, is set to 3 to restrict check to 3 digits.			R3	
	Work registers 15 and 2 are set to zero.			R15	
	Register 2 is to contain the remote ID in binary upon completion.			R2	
REMLOOP	If remote ID character currently checked is not numeric, branch to diagnose invalid remote ID..... >	MSOILEGL (MA28)			
	Register 2, containing (in binary) the equivalent of the part of the remote ID scanned previously, is multiplied by 10 to allow for the addition of the digit just checked.			R2	
	Register 5 contents are shifted left one byte, and the digit just checked is inserted as the new low-order byte.			R5	
	The digit currently checked is converted to binary by setting the zone part to zero.				
	This binary value is then loaded in register 15, and added to the current binary value in register 2.			R2	
	Register 1 is incremented by one to point to the next character.			R1	
	If this is a terminator character (comma or blank), ID processing is complete; branch..... >	REMFOUND (MA28)			
	Otherwise, if 3 digits not yet checked, branch back to scan next digit..... >	REMLOOP (MA27)		R3	

Labels	Chart MA28: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
MSOILEGL	Register 1 is loaded with the address of message 1R21I.			R1	
PASSWNOK	Link to issue message..... >	PMESS2 (MA24)		R15	
	The remote ID in the LCB is cleared again.		LCBREMID (IPW$DLC)		
	Branch to flush the reader..... >	GETLOOP (MA01)			
REMFOUND	If remote ID specified is higher than the maximum, branch to diagnose.... >	MSOILEGL (MA28)			
	If remote ID is zero, ignore SIGNON.	MSOILEGL (MA28)			
	Register 15 is set up with the address of the first LCB in the LCB chain.			R15	
REMCHECK	If the same remote ID as signing on is already active on the LCB currently checked, branch to diagnose..... >	MSOILEGL (MA28)			
	Otherwise, register 15 is updated to address the next LCB.			R15	
	If not yet at end of LCB chain, branch to check next LCB..... >	REMCHECK (MA28)			
	Otherwise, the one-byte binary value of the remote ID is stored in the LCB, inserted in the high-order byte of register 5, whose 3 low-order bytes contain the remote ID in character representation, and register 5 contents are stored in the LCB.		LCBID(IPW$DLC)	R5	
			LCBREMID (IPW$DLC)		
	If no line password has been set, branch to skip password checking... >	PASSWOK (MA29)			
	Otherwise, register 15 is set up with the address of the line password.			R15	
	Register 15 will be used as data pointer in a scan for the line password end.				
	Register 5 is set to zero.			R5	
PSWDLOOP	If end of line password found, branch..... >	PSWDEXEC (MA29)			
	Otherwise, register 5, which is to contain the line password length in machine format, is incremented by one; and, if seven characters have already been checked, branch..... >	PSWDEXEC (MA29)		R5	
	Otherwise, data pointer register 15 is incremented by one, and branch back to check next line password character..... >	PSWDLOOP (MA28)		R15	

Labels	Chart MA29: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
PSWDEXEC	The address of the password specified in the SIGNON card is loaded in register 15.			R15	
	The address of message 1R22I (invalid password) is loaded in register 1.			R1	
	If the SIGNON password and the line password do not match, branch to diagnose..... >	PASSWNOK (MA28)			
PASSWOK	Register 2 is first decremented by one and then multiplied by 8 (remote table entry length) to get the offset in the remote table of the appropriate remote ID.			R2	
	The remote table address is added to obtain the actual address of the remote entry concerned.				
	The remote entry is now copied to the LCB.		LCBRMT(IPW$DLC)		
	If the specified remote ID is marked 'invalid' in the remote table, branch to diagnose..... >	SOILEGL (MA28)			
	Otherwise, according to remote table entry specifications:				
	• ASCII flag and transparency flag are set,		LCBPLINE (IPW$DLC)		
	• SIGNOFF code byte is cleared,		LCBSCODE (IPW$DLC)		
	• SIGNOFF flag is reset,		LCBFLAGS (IPW$DLC)		
	• Signon complete flag is set,		LCBFLAGS (IPW$DLC)		
	• the address of signon message 1R15I is loaded in register 1,			R1	
	• remote ID is copied into the message text				
	Link to issue signon message..... >	PMESS (MA24)		R15	
	An IPW$RDC call is issued to obtain signon time in register 1.				IPW$RDC Chart AG
	Signon time is stored in account record, and branch to resume input processing..... >	GETLOOP (MA01)	LCBTIMON (IPW$DLC)		

Labels	Chart MA30: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
	Start Remote Writer Task				
MSTART	Link to scan the task type specified	SCANTSK (MA32)			
	If MSG was specified, branch to process messages..... >	POSTMSG (MA33)			
	If specified task already started, branch to issue message 1R24I..... >	ERREXIT (MA32)			
	Default class is set to 'A'.		LCBLIST(IPW$DLC)		
	If no class specified, branch to leave default..... >	CLASSOK (MA31)			
	If delimiter of remote ID not a comma, branch to diagnose..... >	INVCLASS (MA31)			
	Register 5 is initialized at 4, being the maximum number of different classes allowed to be specified.			R5	
	Register 1 is incremented to point to the starting delimiter, and saved.		LCBWORK(IPW$DLC)	R1	
	Register 3 is initialized with the address of the first byte of the class field.			R3	
CLASLOOP	Register 1 is incremented to point to the next class specified.			R1	
	If blank, all classes valid, branch..... >	CLASSOK (MA31)			
	If not alphabetic, branch..... >	INVCLASS (MA31)			
	The class character is translated. If translated to zero, branch..... >	INVCLASS (MA31)			
	The starting delimiter address is loaded in register 6, and the last class parameter is compared in a loop with all preceding class parameters, to check for duplication.			R6	
	If duplicate class specified, branch..... >	INVCLASS (MA31)			
	Otherwise, the class character, translated to the appropriate class index, is moved to the class table.				
	Class table entry pointer register 3 is incremented by one, and if less than four class specifications have been checked so far, branch back to check next..... >	CLASLOOP (MA30)		R3 R5	
	Otherwise, if the next class position is blank, branch to continue..... >	CLASSOK (MA31)			

Labels	Chart MA31: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
INVCLASS	Output class index is invalidated.				
	Address of message 1R12I is loaded in register 1.			R1	
	Branch to issue message..... >	ERREXIT (MA32)		R15	
CLASSOK	The appropriate list or punch switch in the LCB is set, and branch to resume input processing..... >	GETLOOP (MA01)	LCBOUT(IPW$DLC)		
	Stop Remote Writer Task				
MSTOP	Link to scan task type specified... >	SCANTSK (MA32)		R6	
	If MSG specified, branch to stop message processing..... >	DOWNMSG (MA33)			
	If specified task was not started previously, branch to issue message 1R23I..... >	ERREXIT (MA32)			
	Otherwise, the stop code (C'S'), is loaded in register 5, and, if no operand follows the task type, branch..... >	STOPOK (MA31)		R5	
	Otherwise, if the task type is not delimited by a comma, branch to diagnose..... >	INVTASK (MA32)			
	The next operand is checked. EOJ (stop at EOJ) and RESTART are the only valid specifications.				
	If one of these is found, the applicable stop code is loaded in register 5 (C'E', and C'R', respectively), and a branch is made to continue..... >	STOPOK (MA31)			
	Otherwise, the address of message 1R11I is loaded in register 1.			R1	
	Branch to issue message 1R11I..... >	ERREXIT (MA32)		R15	
STOPOK	Blank output classes.				
	Load the TCB address into register 15 and check if a forms change is required. If so, load the correct TCB address and check if an output class is available. If not, branch to..... >	STOPOK2 (MA32)		R15	

Labels	Chart MA32: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
	Store stop code in TCB.		TCTT(IPW$DTC)		
	If no immediate stop is required, branch to..... >	GETLOOP (MA01)			
	Activate task, if it waits for TP-I-O-interrupt.		TCSF(IPW$DTC)		
	Reset output switch.		LCBOUT(IPW$DLC)		
	Branch to..... >	MGO2 (MA33)			
STOPOK2	Reset output switch.		LCBOUT(IPW$DLC)		
	Branch to..... >	GETLOOP (MA01)			
	Task Specification Verification				
SCANTSK	Register 1 is incremented to point past the START/STOP command code.			R1	
	Check for punch to 3780 without component select. If so, branch to >	NOPUNTST (MA32)			
	Register 1 is pointed to the next character.			R1	
	If blank, branch to continue scan for task type..... >	SCANTSK (MA32)			
	Otherwise, register 15 is set up to point to the punch class table, register 2 is set up to point to punch switch, and if task type specified is PUN, return.			R6	
NOPUNTST	If not PUN, register 15 is loaded with the address of the list class table, register 2 is set up to point to the list switch, and if task type specified is LST, return.			R15 R2 R6	
	Otherwise, register 15 is set to zero, and if task type specified is MSG, return.			R15 R6	
INVTASK	The address of message 1R13I is loaded in register 1.			R1	
ERREXIT	Link to issue message..... >	PMESS2 (MA24)		R15	
	Branch to resume input processing.. >	GETLOOP (MA01)			
	Address of message 1R23I is loaded in register 1, and branch around loading register 1 again.			R1	
	Address of message 1R24I is loaded in register 1.			R1	
	Link to issue message..... >	PMESS2 (MA24)		R15	
	Branch to resume input processing.. >	GETLOOP (MA01)			

Labels	Chart MA33: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
POSTMSG	The 'no messages' switch is reset.		LCBFLAGS (IPW$DLC)		
	Branch to resume input processing.. >	GETLOOP (MA01)			
DOWNMSG	Link to delete any messages in the message queue..... >	KILLMSG (MA04)		R6	
	The 'no messages' switch is set on.		LCBFLAGS (IPW$DLC)		
	Branch to resume input processing.. >	GETLOOP (MA01)			
MGO	Branch and link to validate task parameter..... >	SCANTSK (MA32)			
	Set up the message pointer in register 6.			R6	
	If the wrong task is being started, branch to issue an error message... >	ERREXIT (MA32)			
	If the delimiter is not blank, branch to..... >	INVTASK (MA32)			
MGO2	Turn off end of forms switch.		BCAFLAGS(IPW$DBC)		
	Restore current TCB address.		BCATCBAD(IPW$DBC)		
	Clear forms TCB address.		BCAFRMCB(IPW$DBC)		
	Branch to get next record..... >	GETLOOP (MA01)			
MSETUP	Branch and link to validate task parameter..... >	SCANTSK (MA32)			
	Set up message pointer in register 6.			R6	
	If the wrong task is being started, branch to issue an error message... >	ERREXIT (MA32)			
	If the delimiter is correct, branch to..... >	SETUPOK (MA33)			
	Otherwise, branch to..... >	INVTASK (MA32)			
PAGELOOP	Check for valid numeric characters.				
	If not correct, branch to issue error message..... >	ERREXIT (MA32)			
SETUPOK	Indicate SETUP command and number of pages in TCB.		TCRS(IPW$DTC)		
	Branch to process as GO command.... >	MGO2 (MA33)			

Labels	Chart MA34: IPW$$TM - Remote Job Entry Routines	Modified Data Fields	Reg. Usage	Calls
INITLINE	This entry point is taken when the central operator issues a PSTART LINE command.			
	On entry the following registers are relevant:			
	1: The logical unit number of the line to be initialized.		R1	
	4: Pointer to the PUB entry of the line to be initialized.		R4	
	5 and 6: Line password set by the central operator (if no password was set register 5 contains blanks).		R5,R6	
	The IPW$$TM base registers are set up in registers 7 and 8.		R7,R8	
	The logical unit number in register 1 is saved in register 15.		R15	
	The length of the Line Control Block (LCB) and Buffer Control Area (BCA) is loaded into register 1 and real storage is reserved for these blocks.		R1	IPW$RSW Chart AC
	LCB and BCA Initialization			
	The pointer to the LCB is set up in register 9.		R9	
	The PUB address in register 4 is stored in the LCB.	LCBPUB(IPW$DLC)		
	The pointer to the BCA is set up in register 4.		R4	
	The LCB address is stored in the BCA.	BUFDCT(IPW$DBC)		
	The BCA address is stored in the LCB.	LCBBUFAD (IPW$DLC)		
	The disposition between the real and virtual addresses of the control blocks is calculated in register 1 and stored in the BCA.	BCADISP(IPW$DBC)	R1	
	The line password set by the central operator is stored in the LCB.	LCBPSWD(IPW$DLC)		
	The logical unit number is stored in the CCB in the BCA.	CCBLUB(IPW$DBC)		
	The logical unit number is set to avoid CCW translation.	CCBLUB(IPW$DBC)		
	The PUB address is loaded into register 6.		R6	

Labels	Chart MA35: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
	The PUB device type is checked if it is a 2703 or ICA.				
	The mode byte is taken from the specification by the user.		LCBMCB(IPW$DBC)		
DEVTYPOK	The message subchain index in the LCB is initialized to X'FF'.		LCBMSG(IPW$DLC)		
	Registers 6 and 15 are set up for scanning the line blocks in the line table.			R6,R15	
LNLOOP	The line blocks are scanned for a matcning line address.				
	If found, branch to..... >	MVCLINE (MA35)			
	Get next line block and branch back to..... >	LNLOOP (MA35)			
MVCLINE	The line features in the line block are moved to the LCB.		LCBFEAT(IPW$DLC)		
	If the line features indicate ASCII support, the ASCII flag is set in the LCB.		LCBPLINE (IPW$DLC)		
	A three second interval timeout is calculated in register 1 and then stored in the LCB.		LCBTMOUT (IPW$DLC)	R1	
	Force variable length, initialize print length and store the line account identifier in the LCB.		LCBPRMT(IPW$DLC) LCBPRLEN(IPW$DLC) LCBRECID(IPW$DLC)		
	If no line password was set by the central operator, the default password in the line block is moved to the LCB.		LCBPSWD(IPW$DLC)		
OKPSWD	The line address in the line block is translated into printable characters and moved to the LCB.		LCBDEVAD (IPW$DLC)		
	Using register 1 as a base register to the BG communications region, the system date is moved from the COMREG to the LCB.		LCBDATE(IPW$DLC)	R1	
	The pointer to the first LCB in the queue is loaded from the permanent area into register 2 and stored in the current LCB.		LCBCHAIN (IPW$DLC)	R2	
	The pointer to the current LCB in register 9 is stored in the permanent area to make it the first in the queue.		CALC(IPW$DPA)		

Labels	Chart MA36: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
	The address of the channel appendage routine is loaded from the permanent area into register 2 and stored in the BCA.		CCBAPP(IPW$DBC)	R2	
	The communication byte in the CCB is set to X'60' to indicate channel appendage and sense information required.		CCBBY3(IPW$DBC)		
	The real address of the first sense byte is calculated in register 2 and stored in the BCA.		BCACCW0(IPW$DBC)	R2	
	The flag byte in the LCB is set to indicate to the line manager that a line is to be started.		LCBFLAGS (IPW$DLC)		
	Activate the line manager.				
DETINIT	Detach the initialization task. • The task is removed from the task selection list. • The task control block storage is released.				IPW$DET Chart AA

Labels	Chart MA36.1: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
LMCS	The first 16 bytes constitute the Section Descriptor: 'LMCS V7M0 '				
	The IPW$$TM base registers are set up in registers 7 and 8.			R7,R8	
	The line manager's base register is set up in register 5.			R5	
LM00	The address of the CCB appendage queue is loaded into register 4 and tested for a channel end condition.			R4	
	If so, branch to..... >	LM20 (MA37)			
LM04	The address of the LCB queue is loaded into register 9 and tested for the end of the chain. If so, branch			R9	
	to..... >	LM12 (MA37)			
	The BCA address is loaded into register 4.			R4	
	The flag byte in the LCB is checked to see if a line has to be started.				
	If so, branch to..... >	MGETLINE (MA47)			
	The flag byte in the LCB is checked to see if a line has to be stopped.				
	If not, branch to..... >	LM09 (MA37)			
	If not all I/O completed, branch to >	LM05		R4	
	If appendage is not active, branch to..... >	LCBSCAN		R4	
LM05	If nobody signed on, branch to..... >	HALTIO		R4	
	If no task active, branch to..... >	HALTIO			
	If not switched line branch to..... >	LM09			
	If nobody called, branch to..... >	HALTIO			

Labels	Chart MA37: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
		(MA47)			
LM09	Register 9 is updated with the next LCB address.			R9	
	Branch to..... >	LM04 (MA36)			
LM12	The task is put in the wait state and an exit is made to task selection.				IPW$WFS Chart AA
	On return from task selection the ECB is unposted.		TCEB+2		
LM16	The TCTT indicator in the TCB is checked to see if a PEND command was given. If not, branch to..... >	LM00 (MA36)			
	Test if all LCBs are released, and if not branch to..... >	LM00 (MA36)			
	The line manager task is detached. • The task is removed from the task selection list. • The task control block storage is released.				IPW$DET Chart AA
LM20	The buffer is removed from the chain and the associated LCB address is loaded into register 9.			R9	
LM36	If channel appendage not active, branch to..... >	LCBSCAN			
	Reset retry counter to zero.		BCAUCCNT		
	If no unit check occured, branch to >	LM24 (MA37)			
	If no sense was received, branch to >	MFORCERL (MA46)			
	If PREP sequence, branch to..... >	LM24 (MA37)			
	Increment the retry counter and check it if reached the limit of 30. If so, branch to..... >	MFORCERL (MA46)	BCAUCCNT	R2	
LM24	The flag byte in the LCB is checkedfor completion of signoff processing. If so, branch to..... >	MGETLINE (MA47)			
	If it is not switched line, branch to..... >	LM38			
	If intervention required, branch to..... >	MFORCERL			
	The virtual address of the last executed CCW is calculated in register 2 and moved to the BCA.		BCACOMD(IPW$DBC)	R2	

Labels	Chart MA38: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
LM38	The CSW status bytes are checked for disastrous errors, and if any have occurred, a branch is made to log them..... >	MFORCERL (MA46)			
	If the device status indicates unit check or unit exception, and if the residual count is not the same as the initial specification, and if the last executed CCW is not a read CCW, branch to..... >	LM40 (MA39)			
	Set the first sense byte to X'03', to simulate error.		IOBSENS0 (IPW$DBC)		
	Indicate unit check in the device status byte.		CCBSTA(IPW$DBC)		

Labels	Chart MA39: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
LM40	If the CCW is not of a prepare sequence type, a branch is made to the BSC read/write sequence processing routine..... >	MBCERDWR (MA42)			
	If the flag in the LCB indicates that a line is to be stopped, a branch and link is made to the signoff routine..... >	MABORT2 (MA24)			
LM50	If the device status indicates unit exception, branch to..... >	MBSCPRUE (MA44)			
	If the CCW command is a disable or set mode command, branch to retry.. >	MRETRY (MA43)			
	To bypass the disable and set mode commands the next time the channel program is executed, the restart address in the BCA is reset.		IOBRESTR (IPW$DBC)		
	If the device status indicates unit check, branch to..... >	MBSCPRUC (MA39)			
	The error counter in the LCB is reset to zero.		LCBERRCT (IPW$DLC)		
	The response to the last executed CCW is checked:				
	If ENQ, branch to start the reader..... >	MSTARTRD (MA40)			
	If ACK0 or ACK1, branch to start the printer or punch..... >	MSTARTPP (MA40)			
	If NAK, branch to write EOT..... >	MBSCPTST (MA40)			
	If DLE-EOT, branch to..... >	MFORCERL			
	Otherwise, branch to log the error and retry..... >	MRETRYL (MA43)			
MBSCPRUC	The first sense byte is checked:				
	If command reject, branch to log the error..... >	MFORCERL (MA46)			
	If neither intervention required nor timeout, branch to log the error and retry..... >	MRETRY			
	If the CCW command is not a read command, branch to retry..... >	EOTWRITE			
	If SIGNOFF processing has been initiated, branch to..... >	EOTWRIT1		R9	
TIMEOUT	If the user does not want signoff forcing, branch to..... >	MBSCPTST (MA40)			

Labels	Chart MA40: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
	Increment the timeout counter by one and store it in the LCB.		LCBTMCNT (IPW$DLC)	R1	
	If the limit of the timeout counter has been exceeded, the signoff code in the LCB is set and a branch and link made to the signoff routine... >	MABORT2 (MA24)			
	The timeout counter is checked.		LCBTMCNT		
MBSCPTST	The timeout statistics are updated.		LCBIDLE		
	If line is not a switched line, branch to..... >	MBSCPTS0			
	If remote is signed on, branch to.. >	MBSCPTS0			
	If not timed out 7 times before remote signed on, branch to..... >	MRETRY			
	Set to sign off.		LCBFLAGS		
	Branch to disconnect line..... >	EOTWRITE			
MBSCPTS0	Check whether an ENQ/EOT has to be sent. If not, branch to..... >	MBSCPNOP (MA40)			
	If it is a 3741 device type, branch to..... >	LPCC		R9	
	The ENQ and EOT in the CCW are alternated.		IOBCCW4+3 (IPW$DBC)		
	Check whether an EOT has to be sent. If so, branch to..... >	EOTWRITE (MA40)			

Labels	Chart MA40.1: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
MBSCPNOP	Set the command code to NOP.		IOBCCW4(IPW$DBC)		
	Repeat the NOP sequence 6 times to enable a line bid from the terminal.		IOBCCW4+7 (IPW$DBC)		
	This sequence is executed if a writer task is interrupted for any reason. It enables the remote user to read cards by simply making the output device not ready and the reader ready (hot reader support). The allowed time within which the reader must be readied is 6x3 = 18 seconds.				
	Check if anything has to be sent. If not branch to retry..... >	MRETRY (MA43)			
LPCC	If any messages have to be transmitted or if punch or list output is available, branch to..... >	ENQWRITE (MA40)			
EOTWRITE	Check if SIGNOFF processing has been initiated.				
	If not, branch to..... >	MRETRY (MA43)			
EOTWRIT1	Otherwise set SIGNOFF completed.		LCBFLAGS (IPW$DLC)		
	If line is not a switched line, branch to..... >	MRETRY (MA43)			
	Set up disconnect sequence.	MCCWINIT		R6	
	Start the I/O.	MEXCP1		R6	
	Branch to..... >	LM00			

Labels	Chart MA40.2: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
ENQWRITE	Set the command code to enquiry write and branch to..... >	MRETRY (MA43)	IOBCCW4(IPW$DBC)		
MSTARTRD	The flag byte in the LCB is checked for signoff processing. If so, the enquiry is ignored and a branch made to..... >	MBSCPTST (MA40)			
	The reader TCB address is loaded into register 14 and the reader identifier into register 15.			R14,R15	
	Branch to start the reader task.... >	STARTUP (MA41)			
MSTARTPP	The command type is checked and, if necessary, set to write.		IOBCCW4(IPW$DBC)		
	The list/punch TCB address is loadedinto register 14 and the list identifier into register 15. If there are any messages to be transmitted, branch to start the list task..... >	STARTUP (MA41)		R14,R15	

Labels	Chart MA41: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
	The output switch in the LCB is checked for list output. If there is list output available in the list queue, branch to start a writer task..... >	STARTUP (MA41)			
	If punch output available in the output queue, branch to ignore unsollicited ACK..... >	TIMEOUT (MA39)			
STARTUP	The timeout counter in the LCB is reset to zero.		LCBTMCNT (IPW$DLC)		
	The previous TCB address is loaded into register 6 to see if any task was interrupted. If so, branch to reactivate the task..... >	POSTTASK (MA42)		R6	
ATTC	Reserve real storage for the TCB.				IPW$RSW Chart AC
	If no storage was obtained, branch to simulate timeout..... >	MBSCPTST (MA40)			
	The TCB is initialized:				
	• Set block identifier		TCBI(IPW$DTC)		
	• Store RDR/LST/PUN identifier		TCTI(IPW$DTC)		
	• Set up line address		TCCU(IPW$DTC)		
	• Set remote identifier		TCRI(IPW$DTC)		
	• Set termination switch to blanks		TCTT(IPW$DTC)		
	• Pass registers			R3-R8	
	• Pass TCB address		TCSV(IPW$DTC)		
	• Set up input class pointer		TCCT(IPW$DTC)		
	• Set class list delimiter to X'FF'		TCCT+4(IPW$DTC)		
	Load the reader entry point into register 3.			R3	
	If a reader is to be started, branch to attach the task..... >	PATT (MA42)			

Labels	Chart MA42: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
	Set up registers for building the task class list in the TCB. R2 = class list pointer, R6 = pointer to class displacements, R15 = pointer to class table (either punch or list task).			R2,R6 R15	
FORMSID	If a printer is to be started, the forms number is stored in the TCB.		TCGW(IPW$DTC)		
FILCLASS	The task class list is build in the TCB (maximum of 4 classes).				
CLASDONE	Load the list/punch entry point into register 3.			R3	
PATT	Attach the task.				IPW$ATT Chart AA
	The current TCB address is saved in the LCB.		LCBBUFAD (IPW$DLC)		
	Branch to..... >	LM00 (MA36)			
POSTTASK	If not in 'I' state, branch to..... >	TIMEOUT (MA39)			
	If starting the message writer, branch to..... >	ATTC (MA41)			
	If starting a reader, branch to.... >	NOFORMS (MA42)			
	If forms change required, ignore ACK, and branch to..... >	TIMEOUT (MA39)			
NOFORMS	Reactivate interrupted task.		TCSF(IPW$DTC)		
	Branch to..... >	LM00 (MA36)			
	BSC Read/Write Sequence Processing				
MBCERDWR	Check channel and device status:				
MBCENSF	If unit exception, branch to..... >	MBSCRWUE (MA44)			
	If unit check, branch to..... >	MBSCRWUC (MA44)			
	Reset the error counter in the LCB.		LCBERRCT (IPW$DLC)		
	Check the sequence type:				
	If SOH-ENQ, branch to..... >	MFORCERL (MA46)			
	If write sequence, branch to..... >	MBCEHDWR (MA43)			
	Check the data bytes:				
	If STX or DLE-STX, branch to..... >	MBCEHDRD (MA43)			
	If ENQ, branch to..... >	MRETRY (MA43)			
	If SOH, branch to..... >	MBCENULL (MA43)			

Labels	Chart MA43: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
		(MA45)			
MWRNAK	Set up a NAK response in the remote response control block in the BCA.		LCBRCB(IPW$DBC)		
	Branch to write the enquiry..... >	MWRENQTR (MA45)			
MBCEHDRD	The address of the last data byte is calculated in register 15, and if the byte indicates ENQ, branch to..... >	MWRNAK (MA43)		R15	
	The address of the last data byte is stored in the BCA.		BUFEWF(IPW$DBC)		
	The ETX switch in the LCB is reset. If the termination byte indicates ETX, the ETX switch is set.		LCBFLAGS (IPW$DLC)		
	The buffer size is calculated in register 15, and the first data byte s checked for DLE. If so, the buffer length is reduced by one. Then the buffer size is compared with the residual length, and if data is present, branch to..... >	MNORMAL (MA44)		R15	
MBCENULL	The address of the restart CCW is loaded into register 1 and stored in the BCA.		IOBSTART (IPW$DBC)	R1	
	Branch and link to read the next block..... >	MEXCP (MA22)			
	Branch to..... >	LM00 (MA36)			
MBCEHDWR	Check the response characters:				
	If not DLE, branch to..... >	MBSCHNAK (MA45)			
	If ACK, branch to..... >	MNORMAL (MA44)			
	If WACK, branch to..... >	MWRENQ (MA45)			
	Check the command type to see if the enquiry has already been written. If not, branch to..... >	MWRENQ (MA45)			
	The ACK is inverted in the BCA and checked for a count check.		LCBRCB+1 (IPW$DBC)		
	If not, branch to write ENQ and log..... >	MWRENQIR (MA45)			
	Branch to retry..... >	MRETRY			

Labels	Chart MA44: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
MBSCRWUE	If the command is a write command, branch to..... >	MRESTART (MA45)			
	The EOT switch in the BCA is turned on.		BCAFLAGS (IPW$DBC)		
MNORMAL	The address of the first carriage control is stored in the BCA.		TPBLCCAD (IPW$DBC)		
	The address of the first data byte is stored in the BCA.		TPBFDATA (IPW$DBC)		
	The address of the first CCW is reset.		IOBSTART (IPW$DBC)		
	The communication byte is set to indicate RJE RDR/LST/PUN ready.		CCBCOM1(IPW$DBC)		
	Branch to..... >	LM00 (MA36)			
MBSCRWUC	If the sense byte indicates intervention required, branch to... >	MBINTREQ (MA45)			
	If the command is an enable command, branch to..... >	MBINTREQ (MA45)			
	If the sequence has a programmable interface, branch to..... >	MCWRNAKL (MA43)			
	Set up register 1 for scanning the unit check table.			R1	
MBUCSRCH	The command is checked against the entries of the unit check table.				
	If the condition is not in the table, branch to..... >	MFORCERL (MA46)			
	If a match is found, the sense bits are checked and if all selected bits are off, continue the scan by branching to..... >	MBUCSRCH (MA44)			
	The error sequence displacement is loaded into register 1, and a branch is made to the error routine.			R1	
MBSCPRUE	If the command is a write, branch to>	MRESTART			
	If DLE-EOT received, branch to..... >	MFORCERL			
	Otherwise, branch to..... >	MRENTRY			

Labels	Chart MA45: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
MRESTART	The address of the next CCW is loaded into register 2.			R2	
	If the command code is a read, branch to..... >	MNEWSTRT (MA45)			
	If the command code is not a TIC, branch to get the next CCW..... >	MRESTART (MA45)			
	Otherwise, the address of the next read CCW is loaded into register 2, and a branch made to..... >	MNEWSTRT (MA45) (MA45)		R2	
MRDTXT	The address of the read CCW is loaded into register 2, and a branch made to..... >	MNEWSTRT (MA45)		R2	
MWRENQIR	Increase count for invalid responses.		LCBINRSP (IPW$DLC)	R2	
MBCEHENQ	The command type is checked, and if it was a response to an ENQ, branch to..... >	MWRENQ (MA45)			
MWRENQ	The address of the write CCW is loaded into register 2.			R2	
MNEWSTRT	The real CCW address is calculated in register 2.			R2	
	Branch to re-execute the channel program..... >	MRETRY2 (MA44)			
MBSCHNAK	If a NAK not received, branch to... >	MWRENQIR			
	If a previous EOT has been received, branch to..... >	MWRENQ			
	Increase invalid response count by 1.		LCBINRSP (IPW$DLC)	R2	

Labels	Chart MA45.1: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
MRETRY	Setup the restart address of the CCW's.			R2	
MRETRY2	Store the address of the restart CCW's in the BCA.		IOBSTART (IPW$DBC)		
	Branch and link to initiate I/O.... >	MEXCP1			
	Return to..... >	LM00			
MBINTREQ	Branch and link to the error log routine..... >	MCERRLOG (MA45)			
	Set up a NAK response in the remote response control block in the BCA.		LCBRCB(IPW$DBC)		
	Branch to retry..... >	MRETRY (MA43)			
MCERRLOG	If only a timeout record and trace option not on for this line, then return to caller via register 6.				
MCERRLO	The first sense byte is checked for timeout or intervention required. If not, branch to..... >	ERRLOG (MA46)			
	If the error has changed, branch to..... >	MCESAMER			
	If it is not the first error, branch to..... >	ERRLOG			
	The error counter is incremented by one and checked for the maximum. If the maximum has not yet been reached, return to caller via link register 6.		LCBERRCT (IPW$DLC)	R6	

Labels	Chart MA46: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
ERRLOG	The SYSREC record is build:				
	• Clear the SYSREC record area		MCEHEADR		
	• Set the length of the data part of the record to 64		MCELNGTH		
	• Set the record identification to X'34' to indicate type 2 I/O device.		MCETYPE		
	• The device dependent switch is set to X'00' to indicate unit check.		MCESWT1		
	• Move the job name.		MCEJOBID		
	• If this is an end-of-day record, branch to skip the I/O related information..... >	MCEMOVS1 (MA46)			
	• Move CCW address.		MCECCWAD		
	• Move failing CCW address.		MCECCW		
	• Move device and channel status bytes.		MCESTAT		
	• Move residual count.		MCERESID		
	If this is not an unrecoverable I/O error, branch to..... >	MCEMOVS1			
	• Move sense bytes and retry count.		MCESENS0		
MCEMOVS1	Move initial sense byte.		MCESENS1		
MCELB12	• Set number of status bytes to 2.		MCESENSE		
	• Set number of device dependent data words to 2.		MCEDWD		
	• Set length of statistical data to 0.		MCESDCAR		
	• Move line address.		MCELINE		
	• Move physical unit address.		MCEPHCUA		
	• Move remote id.				
	• Move device type from PUB.		MCEDEVCE		
	• Move PUB2 usage count.		MCESIOCN		
	The address of the SYSREC record area is loaded into register 1, and the record is written via SVC 44.			R1	
	The error counter in the LCB is reset to zero.		LCBERRCT (IPW$DLC)		
	Return to caller via link register 6.			R6	
MFORCERL	The signoff code in the LCB is set to X'08' to indicate that there has been an unrecoverable error on the line.		LCBSCODE (IPW$DLC)		
	Branch and link to the signoff routine to force signoff..... >	MABORT2 (MA24)		R6	
	If the line is not operational, branch to..... >	MDOSTOP			
	If line is switched, branch to..... >	MFORCER0			

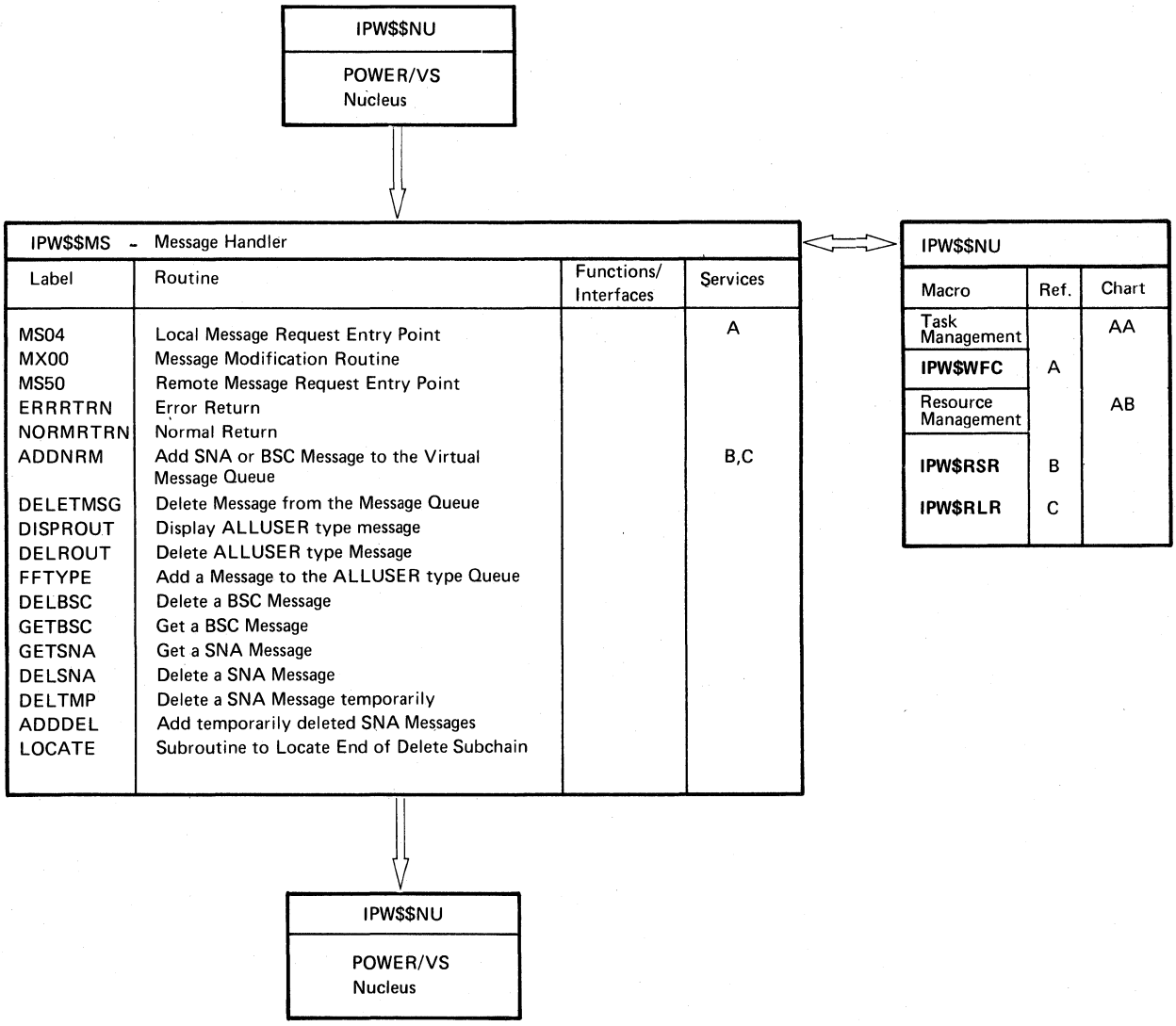
Labels	Chart MA47: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
	Set LCBFLAGS to force line stop.		LCBFLAGS(IPW$DLC)		
MFORCER0	If any line is to be closed, branch to..... >	MNORMAL (MA44)			
	If line is not switched, branch to. >	MGETLINE			
	If disable has been issued, branch to..... >	MGETLINE			
	Set line to signed off.		LCBFLAGS		
	Branch to..... >	EOTWRITE			
MDOSTOP	Force line stop.		LCBFLAGS		
MGETLINE	Branch and link to delete any remaining messages..... >	KILLMSG (MA04)			
	The flag byte in the LCB is checked to see if a line has to be stopped. If so, branch to..... >	HALTIO (MA47)			
	The flag byte is reset to zero.		LCBFLAGS (IPW$DLC)		
	Clear remote ID.		LCBREMID (IPW$DLC)		
	The maximum buffer size of 536 is loaded into register 1 and stored in the LCB.		LCBFSIZ(IPW$DCL)	R1	
	The terminal type indicator in the LCB is checked for EBCDIC or USASCII code and according to that the address of the proper code table is loaded into register 1 and stored in the BCA.		BCACODE(IPW$DBC)	R1	
	Registers 0 and 1 are set up for CCW initialization and control is given to the CCW initialization routine.. >	MCCWINIT (MA20)		R0,R1, R6	
	On return the command code of the fourth CCW is changed to a PREP command.		IOBCCW4(IPW$DBC)		
	If line is switched, the command code of the fourth CCW is set to NOP.				
	Control is given to the I/O interface routine to execute the channel program and initialize the line.... >	MERREXCP (MA22)		R6	
	Branch to..... >	LM00 (MA36)			

Labels	Chart MA47.1: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
HALTIO	The CCB pointer in register 4 is loaded into register 1.			R1	
	Halt I/O via SVC 25.				
STOPLINE	Point to DISABLE CCW.		IOBSTART	R1	
	Unchain CCW from CCW chain.		IOBCCW1		
	Reset channel end appendage.		CCBBY3		
	If line is not operational, branch to..... >	STOPL1			
	Otherwise force I/O complete.		CCBCOM1		
	Branch to..... >	MDOLOG			
STOPL1	Setup disable command using SVC0 and wait for completion.				
MDOLOG	Prepare to write EOD record on SYSREC.				
	Branch to mainline..... >	LM00			

Labels	Chart MA48: IPW$$TM - Remote Job Entry Routines		Modified Data Fields	Reg. Usage	Calls
LCBSCAN	The LCB pointer is initialized.			R1	
LCBBUMP	The address of the next LCB is calculated in register 1.			R1	
	If this LCB is to be deleted, it is removed from the chain.		LCBCHAIN(IPW$DLC)		
	Otherwise, branch back to..... >	LCBBUMP (MA48)			
	The message 1R02I LINE cuu STOPPED is issued to the operator.				IPW$WTO Chart AD
	The CCB pointer in register 4 is loaded into register 1.			R1	
	The logical unit is unassigned. The LCB pointer is loaded into register 1 and the LCB and BCA space is released.			R1	IPW$ULP Chart GC IPW$RLW Chart AC
	Branch to..... >	LM16 (MA37)			
INIT	This routine is entered from IPW$$I2 at POWER/VS initialization time.				
	Save registers 3-7.				
	Using registers 3-7 as work registers, the line and remote blocks from the POWER/VS generation table are moved into virtual storage.			R3-R7	
	Restore registers 3-7 and return to IPW$$I2.				

CHART MB: IPW$$MS - MESSAGE HANDLER (16 PARTS)

Chart MB00: IPW$$MS - Message Handler, General Flow and Macro Calls



Labels	Chart MB01: IPW$$MS - Message Handler		Modified Data Fields	Reg. Usage	Calls
MSCS	The first 16 bytes constitute the section descriptor: 'MSCS V10M0 ' On entry, the following register contents are relevant:				
	0 - Service work register. Address BIND parameters for MSG 1V34I or RJE-ID for message 1V33I.			R0	
	1 - Service work register			R1	
	2 - Service work register			R2	
	3 - Service work register			R3	
	4 - Message control block		IPW$DMS	R4	
	5 - Message table address			R5	
	6 - Message entry address			R6	
	7 - Internal link register			R7	
	8 - SNA unit control block		IPW$DSU	R8	
	9 - Line control block		IPW$DLC	R9	
	10 - POWER/VS nucleus		IPW$DPA	R10	
	11 - Task control block		IPW$DTC	R11	
	12 - Reserved for nucleus use			R12	
	13 - Linkage register save area		IPW$DSV	R13	
	14 - Return address to caller			R14	
	15 - Message handler base address			R15	
	If a remote message request is to be serviced, control is passed to..... >	MS50			
	Local Message Request Entry Point				
	Addressability of the message control block is established in register 4.			R4	
	If the message request word does not contain the address of a TCB, the address of the current TCB is loaded into register 5.			R5	
MS04	The message address and message length are retrieved from the message request word in the TCB and loaded into registers 2 and 3.			R2,R3	
	Then the length of the message is checked, and if it is too long, it is set to the maximum of 72 characters.				
MS08	The message text is moved into the output area of the message control block.		MMMA(IPW$DMS)		
	The message reply word is checked (or a reason code. If logon-reason code not present, branch to..... >	MS10			
	Save reason code for message modification.		MMR0		

Labels	Chart MB02: IPW$$MS - Message Handler		Modified Data Fields	Reg. Usage	Calls
MS10	The address of the message is loaded into register 6, and a link is made to the message modification routine to modify certain message fields if necessary..... >	MX00		R6	
MS12	The resulting message text is checked for trailing blanks and length, and, if necessary, set to the maximum of 72 characters.			R1,R3	
MS16	The message length is set in the write CCW in the message control block.		MMWT(IPW$DMS)		
	If request is for termination, or not an active spool management request, or spool management not present, then branch to..... >	MS18			
	If message expects a reply, then branch to..... >	MS19			
	Load address of user buffer and indicate message logged.		SPPB(SPL)		
	If a return message is requested, then branch to..... >	MS17			
	If a message is pending, then send this message by branching to..... >	MS32			
	If message returned is not 'IR88I', then set error indicator.		SPER(SPL)		
MS17	Place first 44 characters of POWER/VS message text at offset 28 in user-supplied buffer.				
	If this is not an '1QnnI' message, branch to..... >	MS32			
	The reply request word in the TCB is checked to determine whether the message expects a reply. If not, branch to..... >	MS28			
	The length of the expected reply is retrieved from the reply request word in the TCB and loaded into register 3. Then the length is checked, and if it is too long, set to the maximum of 72 characters.			R3	

Labels	Chart MB02.1: IPW$$MS - Message Handler		Modified Data Fields	Reg. Usage	Calls
MS20	The reply length is set in the read CCW.		MMRD(IPW$DMS)		
	The command code of the write CCW is set to X'01' (write with no carriage return).		MMWT(IPW$DMS)		
	The command code of the read CCW is set to X'0A'.		MMRD(IPW$DMS)		
MS24	The input area is set to blanks.		MMMI(IPW$DMS)		
MS28	The 'Blank Compression Routine' is called to delete duplicate blanks within a message.				
	Brancn..... >	COMPRESS		R7	
	The message is issued via SVC 0, and an IPW$WFC macro is issued to wait for completion.				IPW$WFC Chart AA
	If no reply was expected, branch to..... >	MS32			
	The unit exception bit in the CCB is checked to see whether the operator has pressed the CANCEL key. If so, the message is reissued by branching back to..... >	MS24			

Labels	Chart MB03: IPW$$MS - Message Handler		Modified Data Fields	Reg. Usage	Calls
	The command code of the read CCW is set to X'03' (NOP).		MMCD(IPW$DMS)		
	The command code of the write CCW is set to X'09' (write with carriage return).		MMWT(IPW$DMS)		
	The operator's reply is converted to uppercase characters and moved to the caller's input area.				
MS32	Return to caller via register 14.			R14	
	Message Modification Routine				
	Upon entry to this routine, register 3 contains the length of the message, register 6 contains the address of the message, and register 7 contains the return address.			R3,R6, R7	
MX00	Calculate new message length in register 3 for scanning. Put new offset for scan in register 6. Test if end of message reached. Return to caller via register 7.			R3 R6 R7	
MX01	The message text is scanned for a message modifier character via a translate and test instruction, and if a modifier is found, the address where the message has to be modified will be in register 1, and the message is modified via the branch table at..... >	MX12			
	Return to caller via register 7.			R7	
MX12	Message Identifier / Message Modification				
	t Task identifier..... >	MX20			
	c Unit record address..... >	MX24			
	r RJE identifier..... >	MX28			
	j job name..... >	MX32			
	u User information..... >	MX36			
	n Job number..... >	MX40			
	f Forms ID..... >	MX44			
	s Sublibrary..... >	MX48			
	x Number..... >	MX52			
	l Line address..... >	MX56			
	d RJE identifier..... >	MX60			
	m Logical unit name..... >	MX64			
	q RPL request..... >	MX68			
	e RTNCD,FDB2..... >	MX72			
	a Sense..... >	MX76			
	o Command code..... >	MX80			
	p SNA stop code..... >	MX84			
	b CCB address..... >	MX88			
	y Logon reason code..... >	MX92			
	v RJE identifier..... >	MX96			
	w Bind data..... >	MX100			
	i Application ID..... >	MX104			

Labels	Chart MB03.1: IPW$$MS - Message Handler	Modified Data Fields	Reg. Usage	Calls
	h forms overlay identifier (flash). > MX110 k paper thread request..... > MX112 z forms identifier..... > MX116 g abend reason code..... > MX120			

Labels	Chart MB04: IPW$$MS - Message Handler		Modified Data Fields	Reg. Usage	Calls
MX20	Using register 5 as a base address for the TCB, the task identifier (TCTI) is inserted in the message text. If it is a SNA task, the SNA identifier is added. Continue modification scan..... >	MX00		R5	
MX24	Using register 5 as a base address for the TCB, the physical device identifier (TCCU) is inserted in the message text. If it is a SNA task, 'SNA' is replaced by the remid. Continue modification scan..... >	MX00		R5	
MX28	Using register 5 as a base address for the TCB, the RJE identifier (TCRI) is inserted in the message text. Continue modification scan..... >	MX00		R5	
MX32	Using register 2 as a base address for the queue record, the job name (QRNM) is inserted in the message text. Continue modification scan..... >	MX00		R2	
MX36	Using register 2 as a base address for the queue record, the user information (QRUI) is inserted in the message text. Continue modification scan..... >	MX00		R2	
MX40	Using register 2 as a base address for the queue record, the job number (QRNO) is converted to decimal and inserted in the message text. Continue modification scan..... >	MX00		R2	
MX44	Using register 2 as a base address for the queue record, the forms identifier (QRFI) is inserted in the message text. Continue modification scan..... >	MX00		R2	

Labels	Chart MB05: IPW$$MS - Message Handler		Modified Data Fields	Reg. Usage	Calls
MX48	Using register 2 as a base address for the source library work area, the SLI sublib (SLSL) and bookname (SLBM) are inserted in the message text.			R2	
	Continue modification scan..... >	MX00			
MX52	The three-digit number passed in register 4 (MMR4) by the issuing task is loaded into register 2, converted to decimal, and inserted in the message text.			R2,R4	
	Continue modification scan..... >	MX00			
MX56	Using register 9 as a base address for the line control block, the line address (LCBDEVAD) is inserted in the message text.			R9	
	Continue modification scan..... >	MX00			
MX60	Using register 9 as a base address for the line control block, the RJE identifier (LCBREMID) is inserted in the message text.			R9	
	Continue modification scan..... >	MX00			
MX64	Using register 2 as a base address for the logical unit control block, the logical unit name (LULU) is inserted in the message text.			R2	
	Continue modification scan..... >	MX00			
MX68	Using register 2 as a base address for the RPL pointed to by register 6 (MMR6), the RPL request type (RPLREQ) is inserted in the message text. (The RPL request type may be: SETLOGON, OPNDST, INQUIRE, CLSDST, SEND, RECEIVE, RESETSR, or SESSIONC.)			R2,R6	
	Continue modification scan..... >	MX00			

Labels	Chart MB06: IPW$$MS - Message Handler		Modified Data Fields	Reg. Usage	Calls
MX72	Using register 2 as a base address for the RPL pointed to by register 6 (MMR6), the RPL return code (RPLRTNCD) and the RPL feedback field (RPLFDB2) are converted to decimal and inserted in the message text.			R2,R6	
	Continue modification scan..... >	MX00			
MX76	Using register 2 as a base address for the RPL pointed to by register 6 (MMR6), the RPL system sense bytes (RPLSSEI and RPLSSMI) are converted to decimal and inserted in the message text.			R2,R6	
	Continue modification scan..... >	MX00			
MX80	Using register 2 as a base address for the RPL pointed to by register 6 (MMR6), the operator command code is inserted in the message text.			R2,R6	
	Continue modification scan..... >	MX00			
MX84	Using register 2 as a base address for the SNA work area (WACB), the SNA stop code for an SNA inbound, or outbound processor is inserted in the message text.			R2	
	Continue modification scan..... >	MX00			
MX88	The address of the CCB, located in MMB7 is set into the message. Branch to..... >	MX00	IPW$DMM	R7	
MX92	The reason code for the failure of SNA logon requests, passed in register 0, is set into the message. Branch..... >	MX00	IPW$DMM	R0,R2	
MX96	The RJE-ID passed in register 0 from SNA manager is set into the message. Branch..... >	MX00		R0	
MX100	The BIND parameters are set into the message. Address is passed in register 0.			R0,R2, R3	
	Branch on register m..... >			R7	
MX104	The application ID from the VTAM ACB, located in the SNCB, is set into the message. Branch..... >	MX00		R2	
MX110	Using register 2 as a base for the queue record, the forms overlay identifier (flash-id) is inserted into the message text.			R1,R2	
	Continue modification scan..... >	MX00			

Labels	Chart MB06.1: IPW$$MS - Message Handler		Modified Data Fields	Reg. Usage	Calls
MX112	Using register 2 as a base for the queue record, the paper thread request is inserted in the message text. The paper thread request may be: • BTS - burster trimmer stacker • CFS - continuous form stacker Continue modification scan..... >	MX00		R1,R2	
MX116	Using register 2 as a base for the queue record, the forms identifier is inserted into the message text. Continue modification scan..... >	MX00		R1,R2	
MX120	The POWER/VS abend reason code contained in register 0 (MMR0) is converted to printable format and inserted into the message text. Continue modification scan..... >	MX00		R1	
COMPRESS	Delete duplicate blanks out of a message after all variables have been set by message modification. Address of message is passed in register 1, length in register 0. Branch on register 7..... >			R3,R4 R5,R6, R14 R7	
MS50	Remote Message Request Entry Point Addressability of the message control block is set up in register 4. The function indicator is saved. The message request word in the TCB is reset to X'00'. The parameter registers are loaded into registers 0 and 1. The address of the remote message table is loaded into register 5. The address of the logical unit control block, if any, is loaded into register 8. The pointer to the ALLUSER-TYPE table is set in register 6. The function indicator is set in register 2.		MSFI(IPW$DMS) TCMW(IPW$DTC)	R4 R0,R1 R5 R8 R6 R2	

Labels	Chart MB07: IPW$$MS - Message Handler		Modified Data Fields	Reg. Usage	Calls
MS54	The requested function is selected via the following branch table:				
	Display ALLUSER message..... >	DISPROUT			
	Delete ALLUSER message..... >	DELROUT			
	Add ALLUSER message..... >	FFTYPE			
	Get BSC message..... >	GETBSC			
	Delete temporary SNA message..... >	DELTMP			
	Delete BSC message..... >	DELBSC			
	Add temporary deleted SNA message.. >	ADDDEL			
	Delete SNA message..... >	DELSNA			
	Get SNA message..... >	GETSNA			
	Add SNA or BSC message..... >	ADDNRM			
	Error Return				
ERRRTRN	Register 1 is set to zero to indicate that the requested function could not be performed.			R1	
	Normal Return				
NORMRTRN	Registers 0 and 1 are saved, and a return to caller is made via register 14.		MSR0(IPW$DMS)	R0,R1 R14	
	Add SNA or BSC Message to the Virtual Message Queue				
ADDNRM	Check if the remote ID is bigger than the number of BSC remotes. If so, branch to..... >	ADDSNA		R2	
SCANLCB	Using register 9 as LCB pointer, the LCB chain is scanned for a matching remote ID. After all LCBs have been scanned, a branch is made to add an SNA message..... >	ADDSNA		R9	
	When a matching remote ID is found, the function indicator is set to X'29' to indicate a BSC message.		MSFI(IPW$DMS)		
	The LCB flags are checked to determine whether any messages are wanted. If not, the message is ignored and a return to caller is made by branching to ..... >	NORMRTRN			

Labels	Chart MB08: IPW$$MS - Message Handler		Modified Data Fields	Reg. Usage	Calls
ADDMSG	The free chain index (MSFC) in the message control block is checked for a free entry. If there is none available, a branch is made to..... >	DELETMSG			
	The free entry index is saved in register 7, and the displacement in the message table is calculated in register 6. The new free entry index is then stored in the message control block using register 2 as a work register, and the end of subchain indicator is set in the message table. Then the message length is obtained in register 2 and checked. If it is too long, it is set to the maximum of 59 characters.		MSFC(IPW$DMS) MIND	R7 R2 R2	
TXTOK	The previous message text in the message table is cleared, and the new message text is moved into it, as well as the message length and sequence identifier.		MTXT MLEN MSID		
	The start address of the BSC subchain, or the SNA subchain is loaded into register 3.			R3	
MSGSCAN	The subchain is scanned for more entries and the displacement of a possible next entry is calculated in register 2.			R2	
ENDSCAN	When no more entries are to be scanned the free entry index, which was saved in register 7, is stored in the last entry pointed to by register 3.				
	Save the message address in register 6.		SAVMSGAD	R6	
	The length of the message is obtained in register 3, the address of the message in register 6, and register 5 is set up for modification. Then a link is made to the message modification routine at..... >	MX01		R3,R5, R6	

Labels	Chart MB08.1: IPW$$MS - Message Handler		Modified Data Fields	Reg. Usage	Calls
	Reload message address into register 6.			R6	
	Save work registers registers 14 to 6.		FW9		
	Delete trailing blanks.				
	Save message length without trailing blanks.		MLEN	R1	
	Put length in register 14 and message address in register 1.				
	Branch to multiple blank deletion routine..... >	COMPRESS			
	Save number of deleted blanks from register 4 in register 2.			R2	
	The original message length is reduced by the number of deleted blanks and restored.		MLFN	R2,R7	
	A check is made if this was a BSC message. If so, the message counter is incremented by one, and a branch is made to exit..... >	NORMRTRN	LCBMSCTR (IPW$DLC)	R2	

Labels	Chart MB09: IPW$$MS - Message Handler		Modified Data Fields	Reg. Usage	Calls
SUCBCTR	The SNA message counter is incremented by one, and a check is made whether this was the first SNA message. If not, branch to..... >	NOPOST	SUMN(IPW$DSU)	R2	
	Otherwise, using register 2 as a base address for the SNA control block, the SNA manager work ECB is posted, and the select indicator is set to C'S'.		SNEB(IPW$DSN) LUSL(IPW$DSU)		
NOPOST	A branch is made to..... >	RLSUCB1			
	Check if the remote ID is bigger than the number of BSC remotes. If not, branch to..... >	ERRRTRN			
ADDSNA	A check is made whether the SNA control block is present. If not, a branch is made to exit..... >	ERRRTRN			
	A check is made whether a message number is specified in register 1. If not, branch to..... >	NOMSGNR		R1	
	The displacement of the message in the message module is calculated in register 1.			R1	
NOMSGNR	Register 8 is initiated for the first SNA unit control block.			R8	
SCANSU	The SNA unit control block chain is scanned for a matching remote ID. If none is found, branch to exit..... >	RLSUCB			
	If a matching remote ID is found, the function indicator is set to X'30' to indicate SNA message.		MSFI(IPW$DMS)		
	Scan LUCBs for this remote ID to find a session to process the message.				
	If no session logged off, or logoff in process, branch..... >	RLSUCB			
	If session useable, branch..... >	ADDMSG			
RLSUCB	Register 1 is set to zero to indicate an error return.			R1	

Labels	Chart MB10: IPW$$MS - Message Handler		Modified Data Fields	Reg. Usage	Calls
RLSUCB1	A branch is made to exit..... >	NORMRTRN			
DELETMSG	The message counter in register 3 is set to zero.			R3	
	The LCB pointer (CALC) is loaded into register 9, and checked for an empty LCB chain. If so, branch to check the SNA unit control block chain..... >	SCANSU2		R9	
SCANLCB2	The LCB chain is scanned, and thehighest message counter is obtained in register 9.			R9	
SCANSU2	Set register 8 to zero to indicate no SNA.			R8	
	The SNA control block pointer (CASM) is loaded into register 2. If the pointer is zero, branch to ignore SNA..... >	DELMSG2		R2	
	Using register 2 as a base address for the SNA control block, the address of the first active SNA unit control block is loaded into register 8.			R8	
	If no SNA unit control block is active, branch to..... >	DELMSG2			
SCANSU3	The SNA unit control block chain is scanned, and the highest message counter is obtained in register 3.			R3	
DELMSG2	The message subchain address (LCBMSG for BSC, or SUMC for SNA) is loaded into register 2, and the message number (LBCMSCTR for BSC, or SUMN for SNA) is loaded into register 3.			R2,R3	

Labels	Chart MB11: IPW$$MS - Message Handler		Modified Data Fields	Reg. Usage	Calls
DELMSG3	Using registers 6 and 7 as work registers, all messages are deleted from the subchain, and the space is returned to the free list chain.			R6,R7	
	The entry for the next message is calculated in register 6.			R6	
	The new free list pointer is stored and the subchain index is set to X'FF' to indicate end of subchain.		MSFC(IPW$DMS) MIND		
	The previous message text in the message table is cleared and the number of messages that have been deleted is converted to decimal. Then message 1R20I nnn MESSAGES DELETED is moved into the message table.			R2	
	The message counter is set to 1, and a branch is made to add messages to the virtual message queue again.... >	ADDNRM			
DISPROUT	Display ALLUSER Type Messages				
	If the current entry is the first one (register 1 zero), then register 1 is initialized with the ALLUSER table address.			R1	
DISPRT2	Register 1 is updated to point to the next entry. When all entries have been processed, a branch is made to exit..... >	ERRRTRN		R1	
	Otherwise, if the current entry is not in use, a branch is made to get the next entry address..... >	DISPRT2			
	If the current entry is in use, register 1 is pointed to the message descriptor byte, and a branch is made to exit..... >	NORMRTRN			

Labels	Chart MB12: IPW$$MS - Message Handler		Modified Data Fields	Reg. Usage	Calls
DELROUT	Delete ALLUSER Type Messages				
	The address of the message descriptor byte is loaded into register 8. If all messages are to be deleted (register 8 zero), branch to..... >	DELRT3		R8	
	Otherwise, the corresponding queue entry is calculated in register 8. If the message number is invalid, branch to exit..... >	ERRRTRN		R8	
	A check is made whether the request is from the central operator (register 0 zero). If so, the user ID does not have to be checked, and a branch is made to..... >	DELRT2			
	The user ID is validated, and if it is invalid, a branch is made to exit>	ERRRTRN			
DELRT2	The queue entry is released (remote ID set to X'FF') and a branch is made to exit..... >	NORMRTRN			
DELRT3	Using register 6 as a pointer to the queue entries, all queue entries are scanned for a matching remote ID. If a matching ID is found, the queue entry is released by setting it to X'FF'. When all entries have been scanned, a branch is made to exit.. >	NORMRTRN		R6	
FFTYPE	Add a Message to the ALLUSER Type Message Queue				
	Register 6 is updated to point to the next queue entry. If the end of the queue is reached, a branch is made to exit..... >	ERRRTRN		R6	
	When a free entry is found, the originator's ID is stored. Then the message length is checked, and if it is too long, it is set to the maximum of 59 characters.				

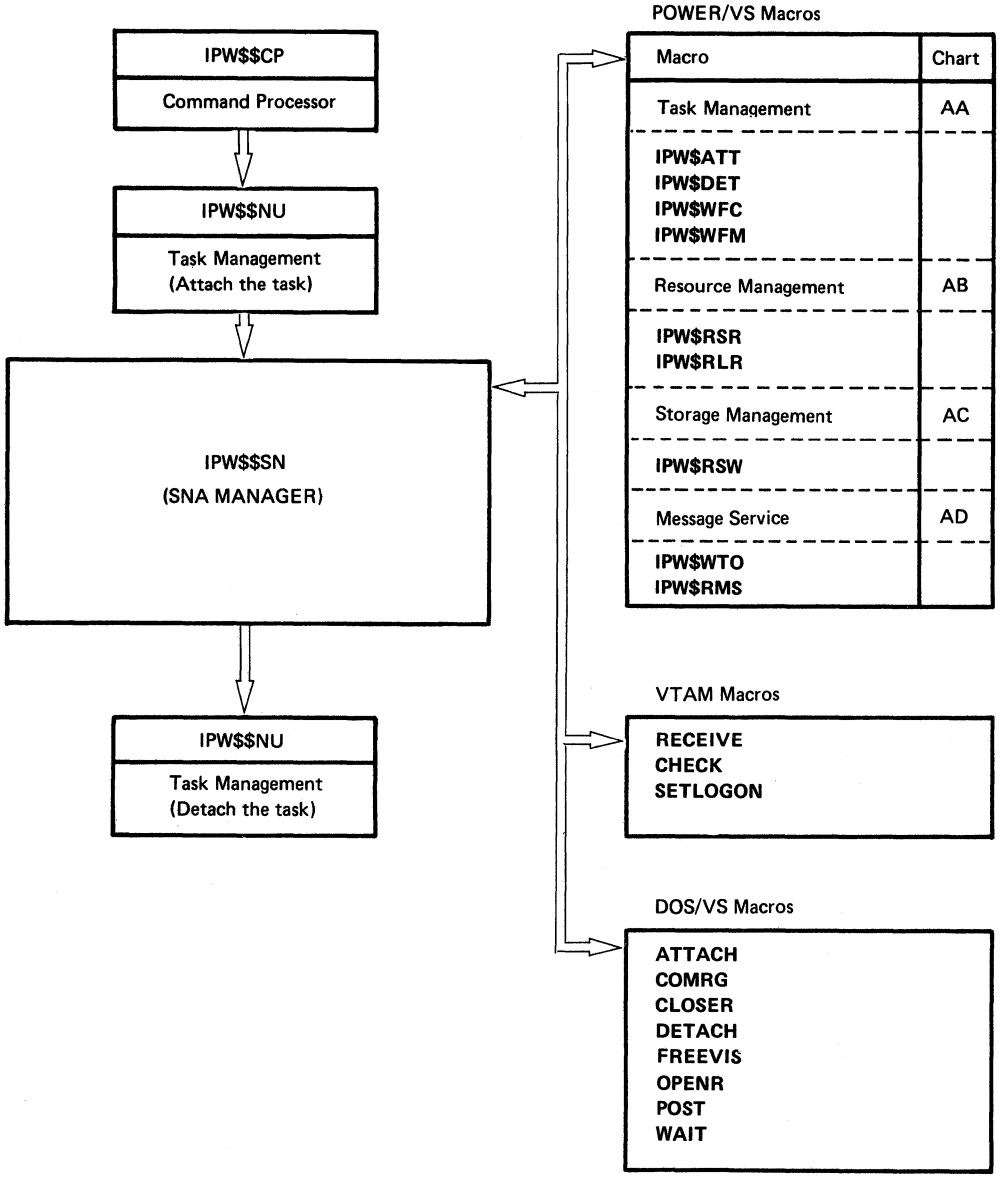
Labels	Chart MB13: IPW$$MS - Message Handler		Modified Data Fields	Reg. Usage	Calls
	The previous message text in the message table is cleared, and the message text and length are moved in. Then a branch is made to exit..... >	NORMRTRN			
DELBSC	Delete BSC Message				
	The address of the message to be deleted is loaded into register 6.			R6	
	The subchain index (MIND) is saved in register 2, and then updated with the free chain index.		MIND	R2	
	Then the free chain index is updated with the message subchain index, and the subchain index, which was saved in register 2 is stored in the message subchain index.		MSFC(IPW$DMS) LBCMSG(IPW$DLC)		
	The BSC message counter is obtained in register 3, decremented by one, and stored again.		LCBMSCTR(IPW$DLC)	R3	
	A branch is made to exit..... >	NORMRTRN			
GETBSC	Get BSC Message				
	A check is made whether there are any BSC messages. If not, a branch is made to exit..... >	ERRRTRN			
	The message index (LCBMSG) is obtained in register 1, and a branch is made to..... >	GETMSG		R1	
GETSNA	Get SNA Message				
	A check is made whether there are any SNA messages. If not, a branch is made to exit..... >	ERRRTRN			
	The message index (SUMC) is obtained in register 1, and register 5 is incremented by one to get the second entry byte.			R1,R5	
GETMSG	The proper message displacement is calculated in register 1, and a branch is made to exit..... >	NORMRTRN		R1	

Labels	Chart MB14: IPW$$MS - Message Handler		Modified Data Fields	Reg. Usage	Calls
DELSNA	Delete SNA Message				
	A check is made to see whether the temporary delete chain is empty. If so, a branch is made to exit..... >	ERRRTRN			
	A link is made to locate the end of the delete subchain..... >	LOCATE			
	Upon return, register 2 points to the last entry.				
	The last entry is set to point to the next free entry.				
	The free chain index is updated with the temporary delete chain index.		MSFC(IPW$DMC)		
	The temporary delete chain index is set to X'FF'.		SUMD(IPW$DSU)		
	A branch is made to exit..... >	NORMRTRN			
DELTMP	Delete SNA Message Temporarily				
	A check is made to see whether the subchain is empty. If so, a branch is made to exit..... >	ERRRTRN			
	A link is made to locate the end of the delete subchain..... >	LOCATE			
	Upon return, register 2 points to the last entry.				
	The displacement of the message to be deleted is calculated in register 6.			R6	
	The last entry is updated with the subchain index.				
	The subchain index is updated with the entry of the message to be deleted.		SUMC(IPW$DSU)		
	The entry of the message to be deleted is set to X'FF'.				
	The message counter is obtained in register 3, decremented by one, and stored again.		SUMN(IPW$DSU)	R3	
	A branch is made to exit..... >	NORMRTRN			

Labels	Chart MB15: IPW$$MS - Message Handler		Modified Data Fields	Reg. Usage	Calls
ADDDEL	Add Temporarily Deleted SNA Message				
	A check is made to see whether the temporary delete chain is empty. If so, a branch is made to exit..... >	ERRRTRN			
	A link is made to locate the end of the delete chain..... >	LOCATE			
	Upon return, register 2 points to the last entry, and register 3 contains the number of messages in the delete subchain.				
	The last entry is updated with the subchain index.				
	The subchain index is updated with the temporary delete chain index.		SUMC(IPW$DSU)		
	The temporary delete chain index is set to C'FF'.		SUMD(IPW$DSU)		
	The actual number of messages is calculated in register 3 and stored again.		SUMN(IPW$DSU)	R3	
	A branch is made to exit..... >	NORMRTRN			
LOCATE	Locate End of Delete Subchain				
	The index to the first entry is obtained in registers 2 and 6.			R2,R6	
	A check is made to see whether the temporary delete chain is empty. If so, branch to..... >	ENDELC			
LOCLOOP	The message counter in register 3 is incremented by one, and the index of the next entry is obtained in register 2. If the end of the chain has not yet been reached, register 6 is set to point to the next entry and a branch is made back to..... >	LOCLOOP		R2,R3 R6	
ENDELC	Return to caller via register 7.			R7	

CHART MC: IPW$$SN - SNA MANAGER (42 PARTS)

Chart MC00: IPW$$SN - SNA Manager, General Flow and Macro Calls



SNA MANAGER (IPW$$SN) ORGANIZATION

Chart MC1

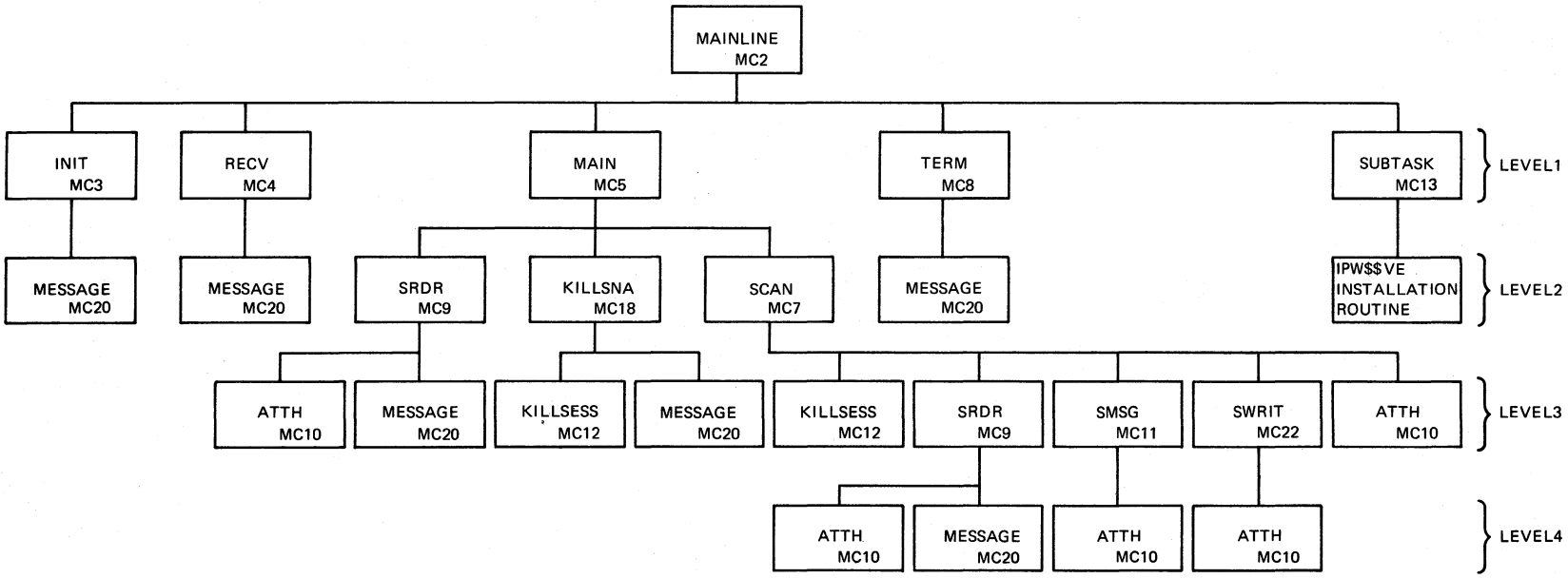
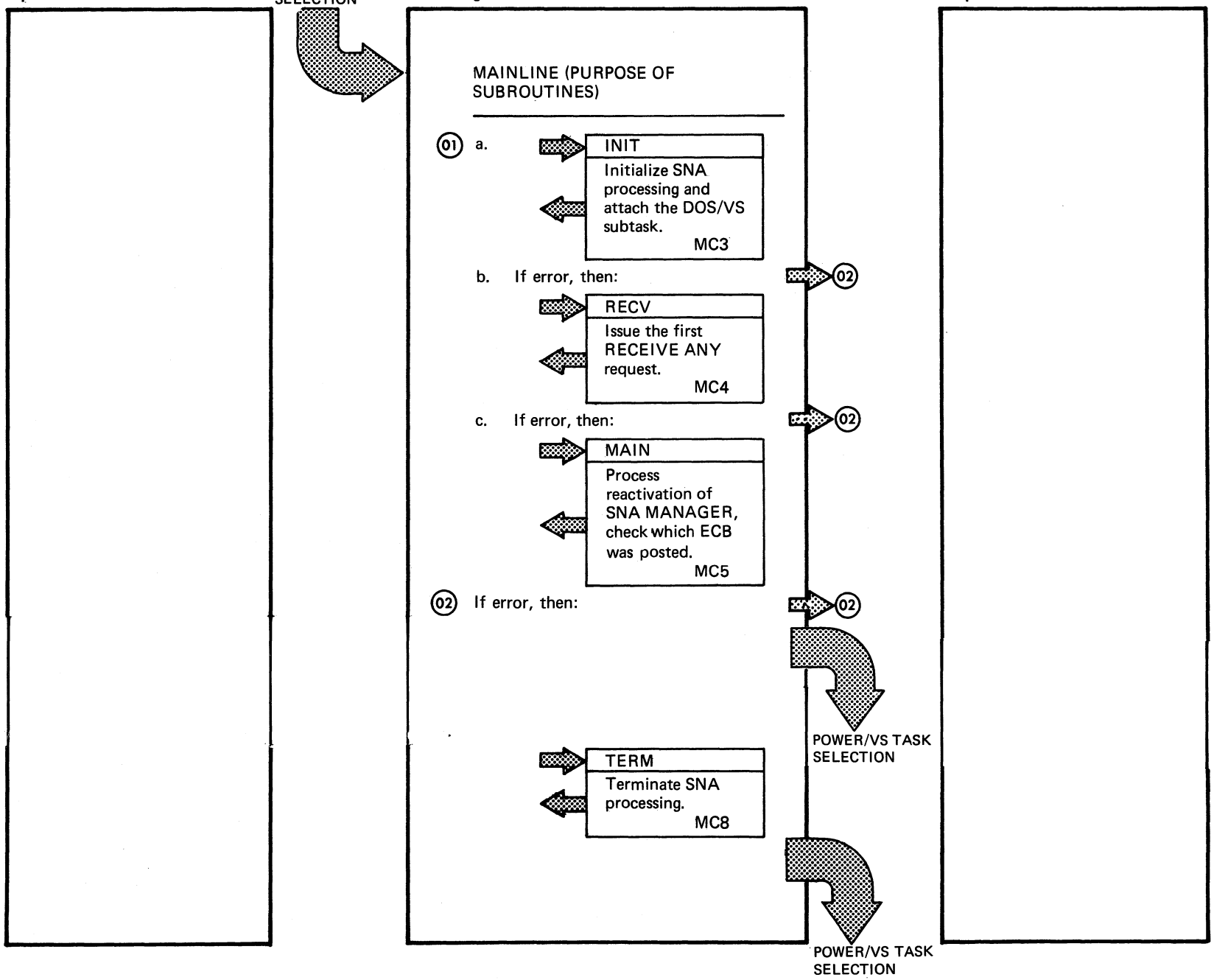


Chart MC1


Input
POWER/VS TASK SELECTION
Processing
Output
MAINLINE (PURPOSE OF SUBROUTINES)
01
a.
INIT
Initialize SNA processing and attach the DOS/VS subtask.
MC3
b. If error, then:
02
RECV
Issue the first RECEIVE ANY request.
MC4
c. If error, then:
02
MAIN
Process reactivation of SNA MANAGER, check which ECB was posted.
MC5
02
If error, then:
02
POWER/VS TASK SELECTION
TERM
Terminate SNA processing.
MC8
POWER/VS TASK SELECTION

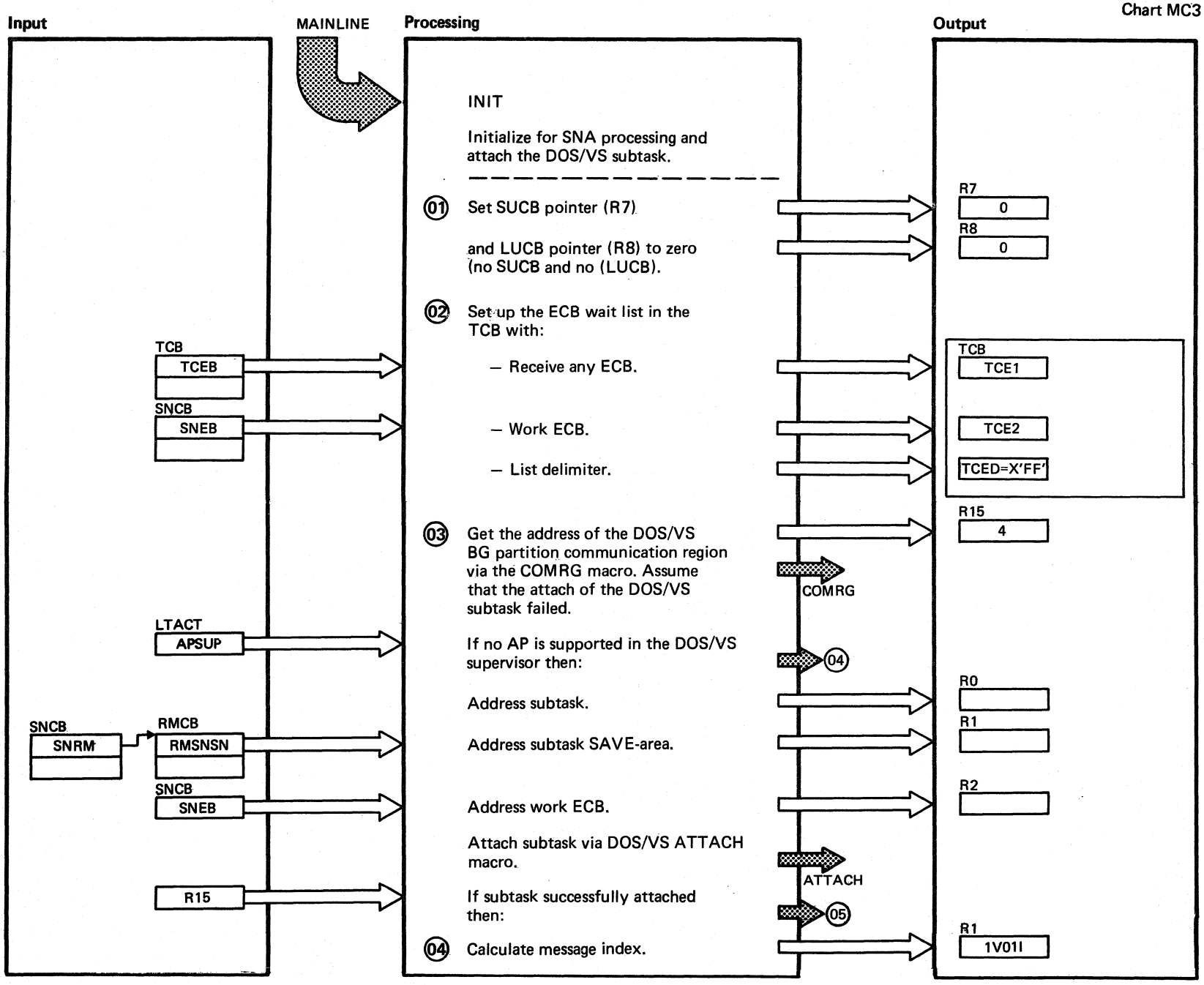


Chart MC3
Part 1 of 3

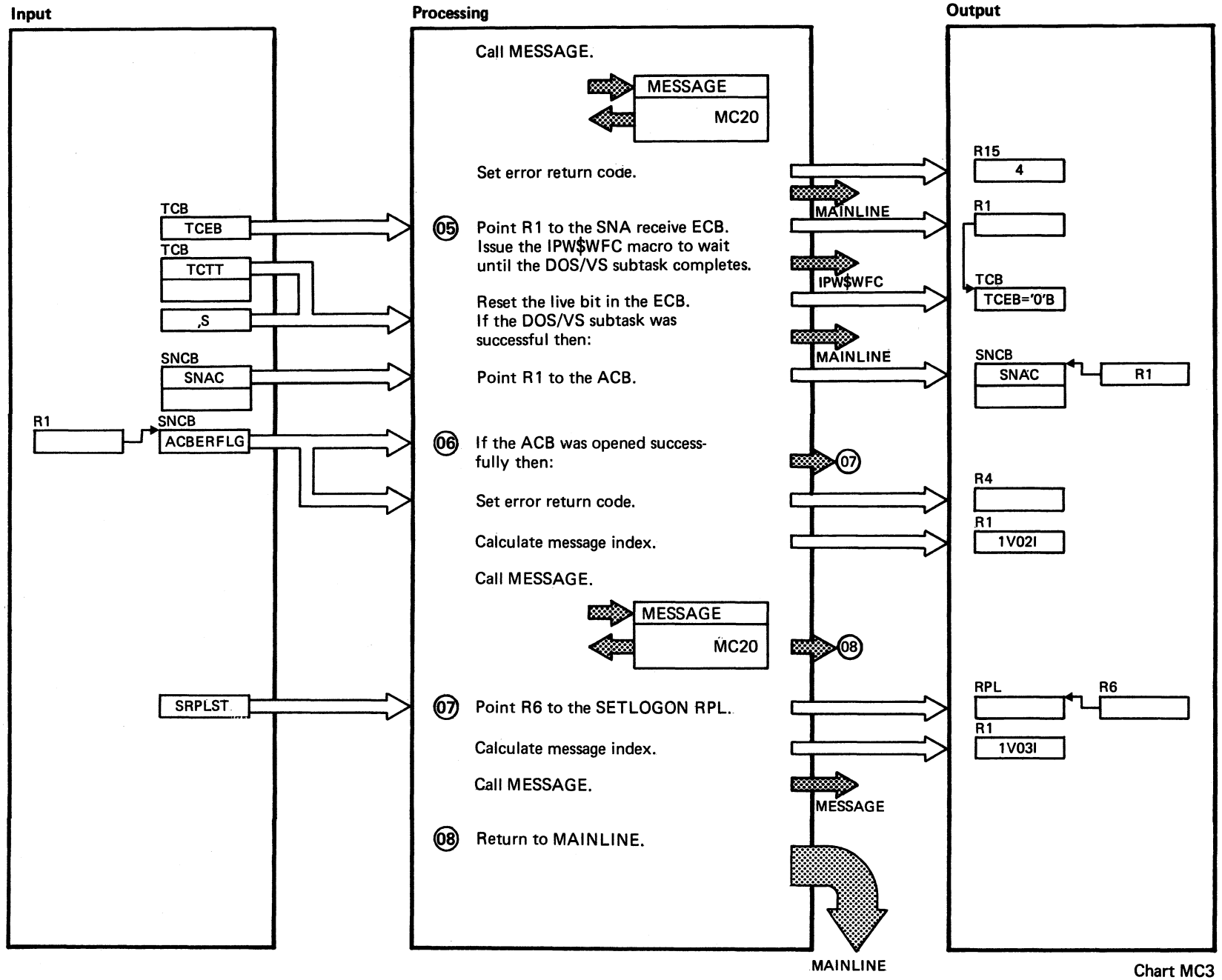


Chart MC3
Part 2 of 3

Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(05) 1V011 NO SUBTASK AVAILABLE FOR RJE, SNA. (07) 1V021 VTAM OPEN FAILURE RTNCD = xxx. (08) 1V031 ERROR ON REQUEST RTNCD, FDBK = 'xx, yy' SENSE = sss.			

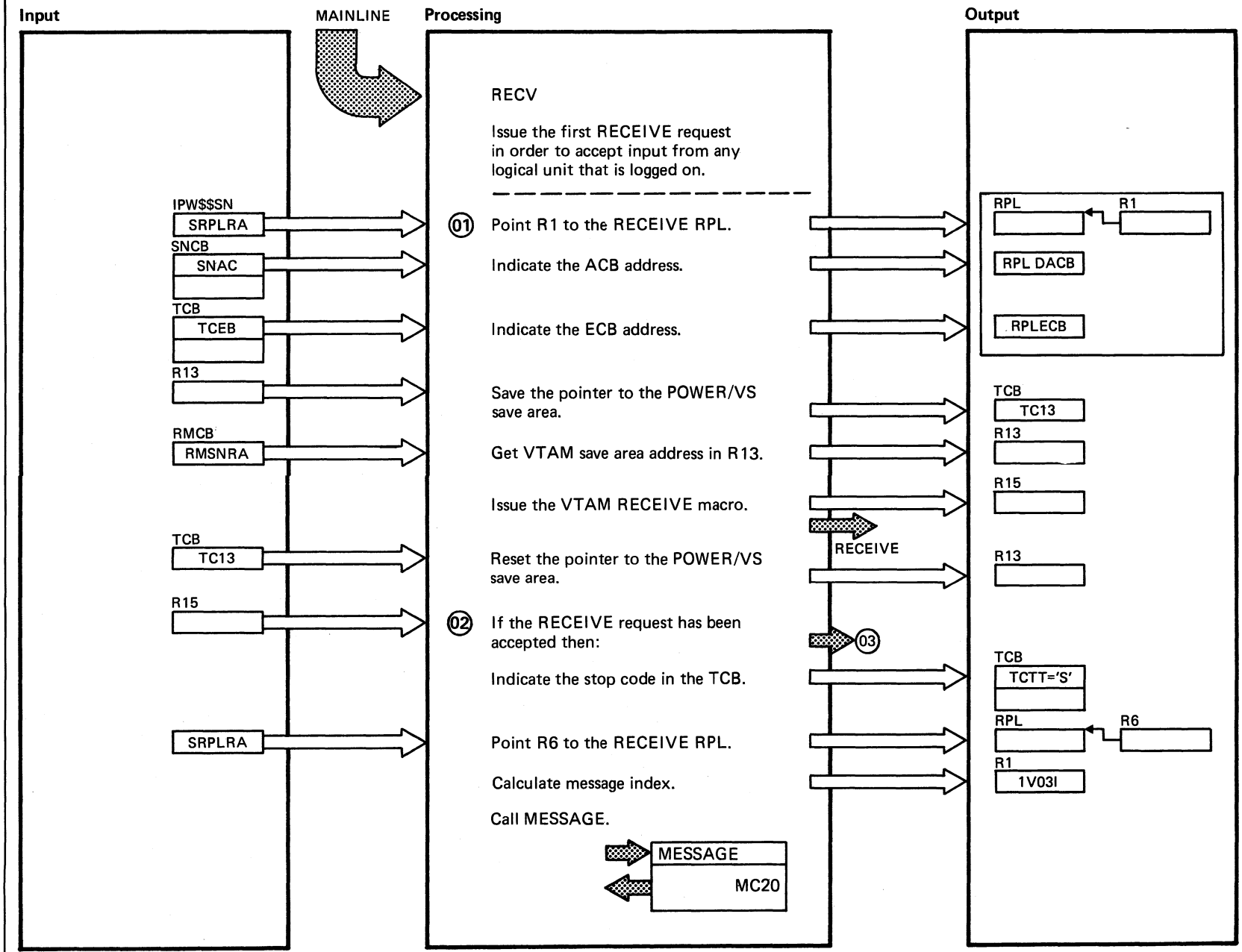
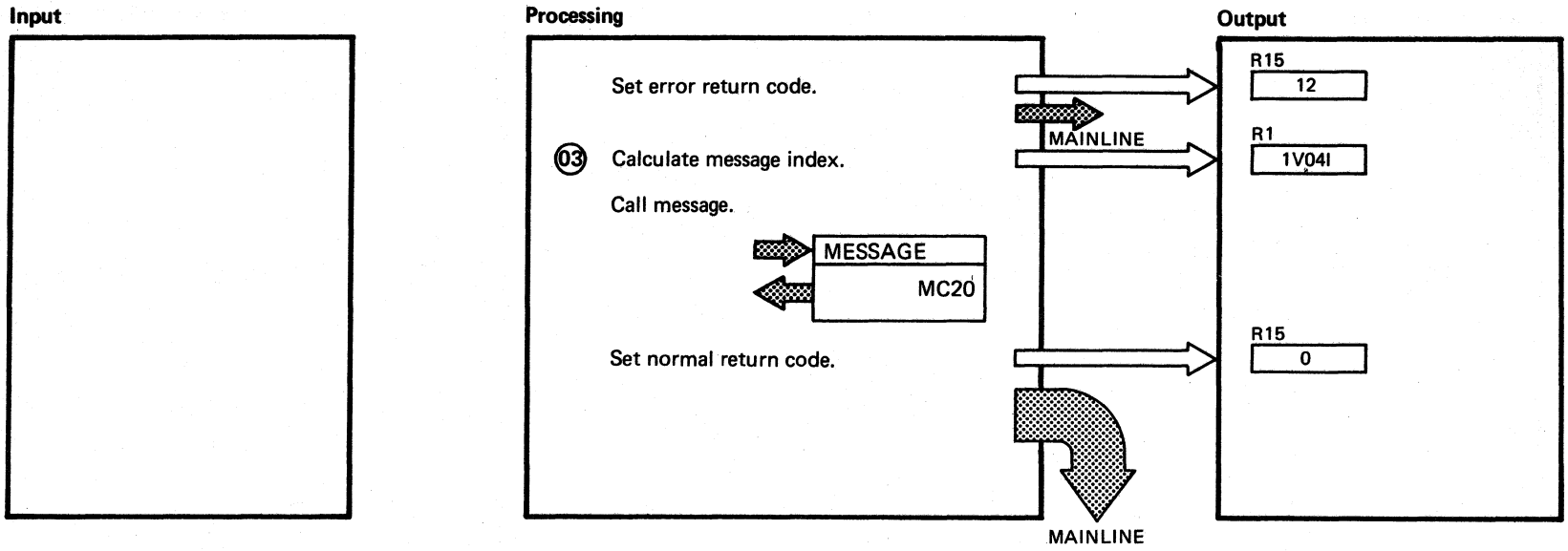


Chart MC4
Part 1 of 2



Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(02) 1V03I ERROR ON RECEIVE RTNCD, FDBK = xx, yy, SENSE = ssss.			
(03) 1V04I RJE, SNA STARTED.			

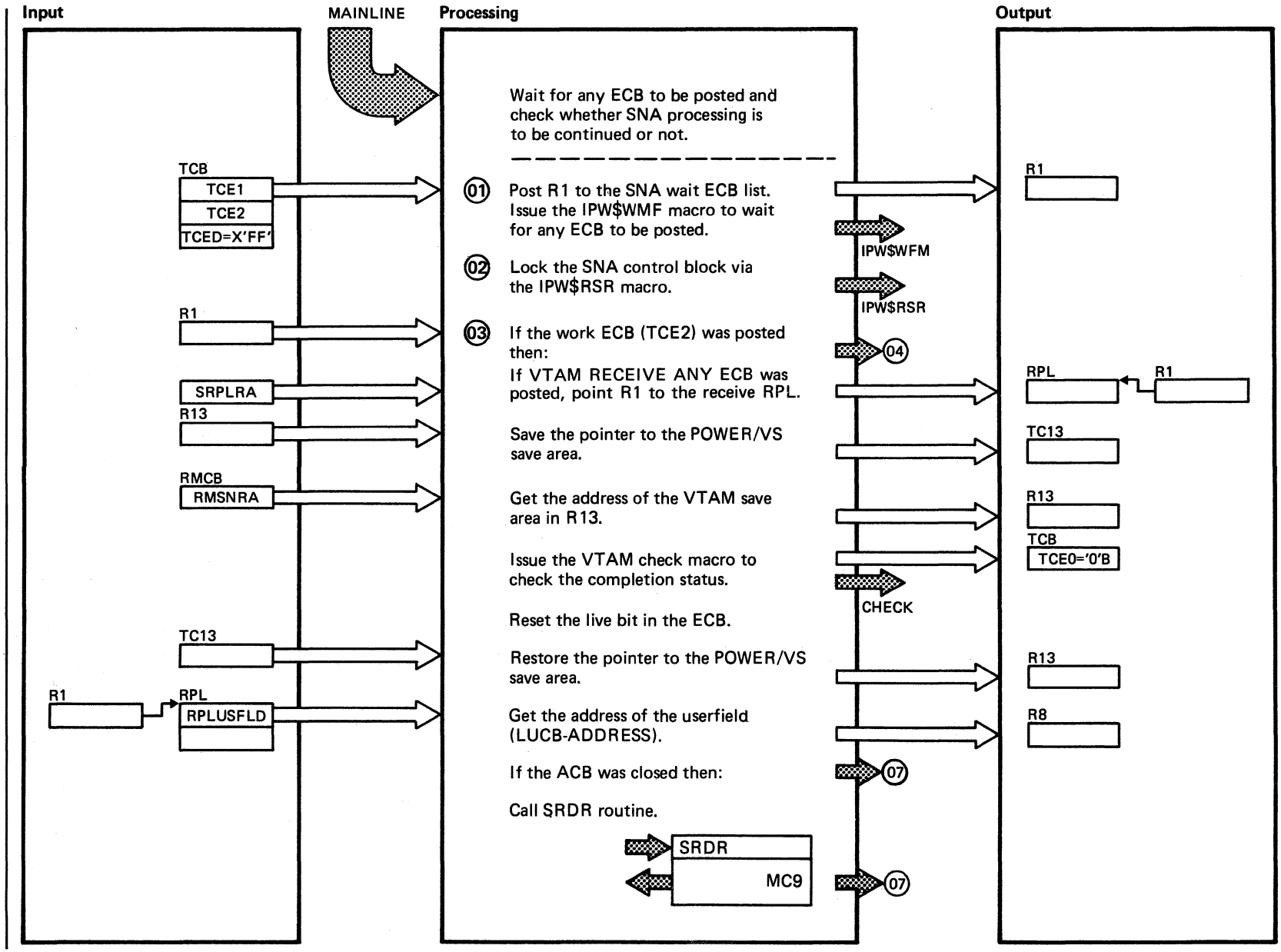
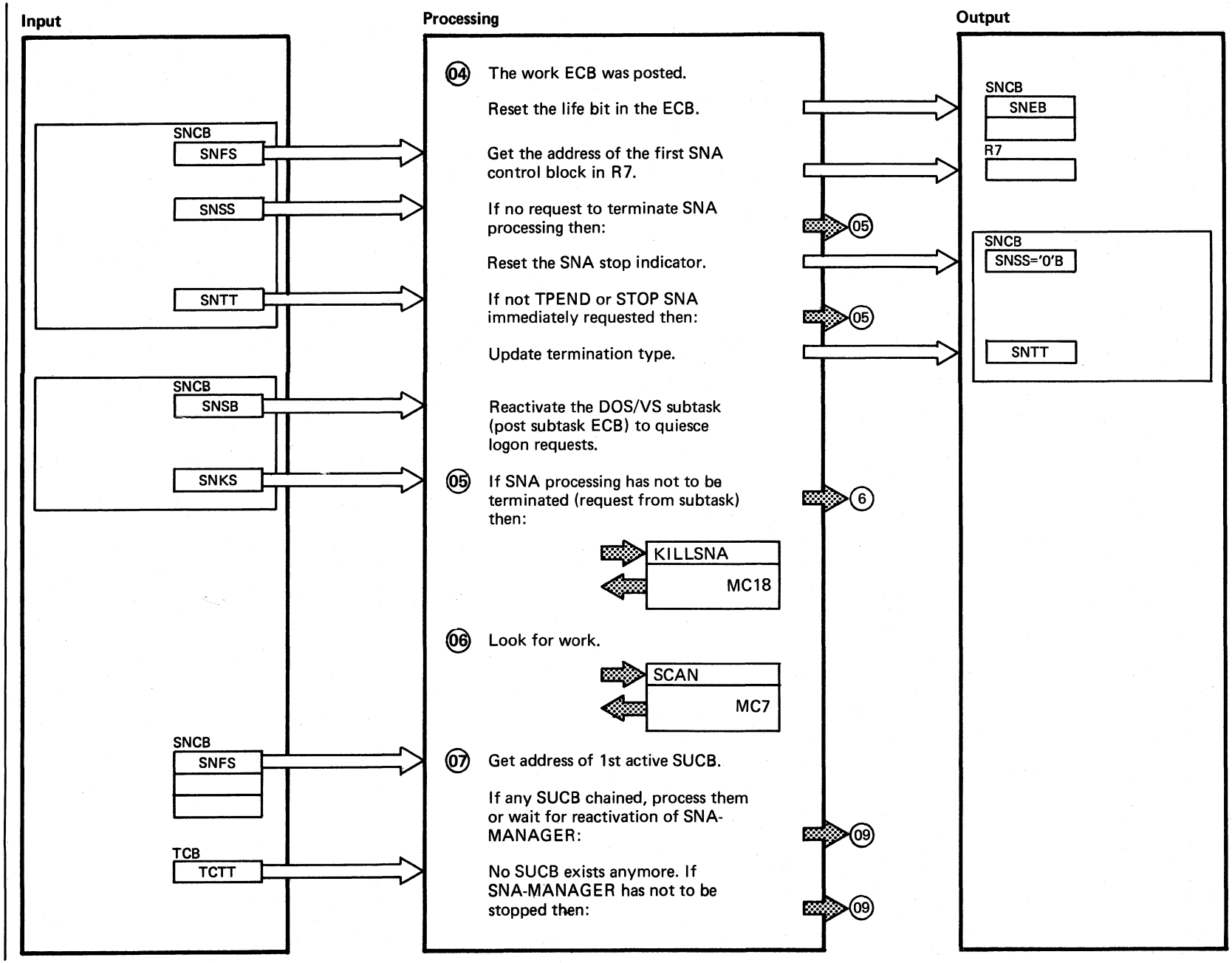
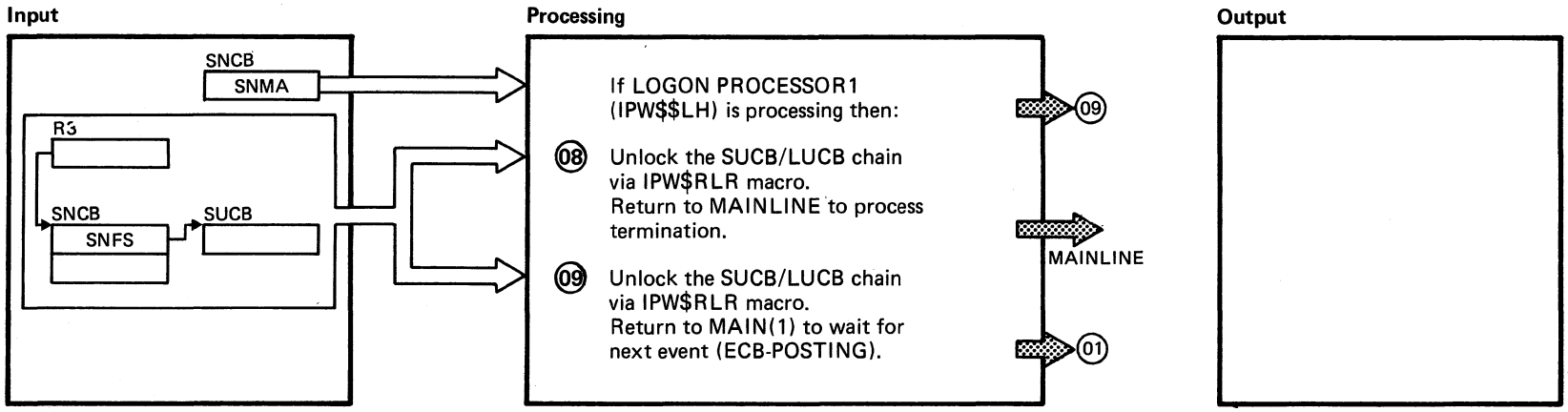
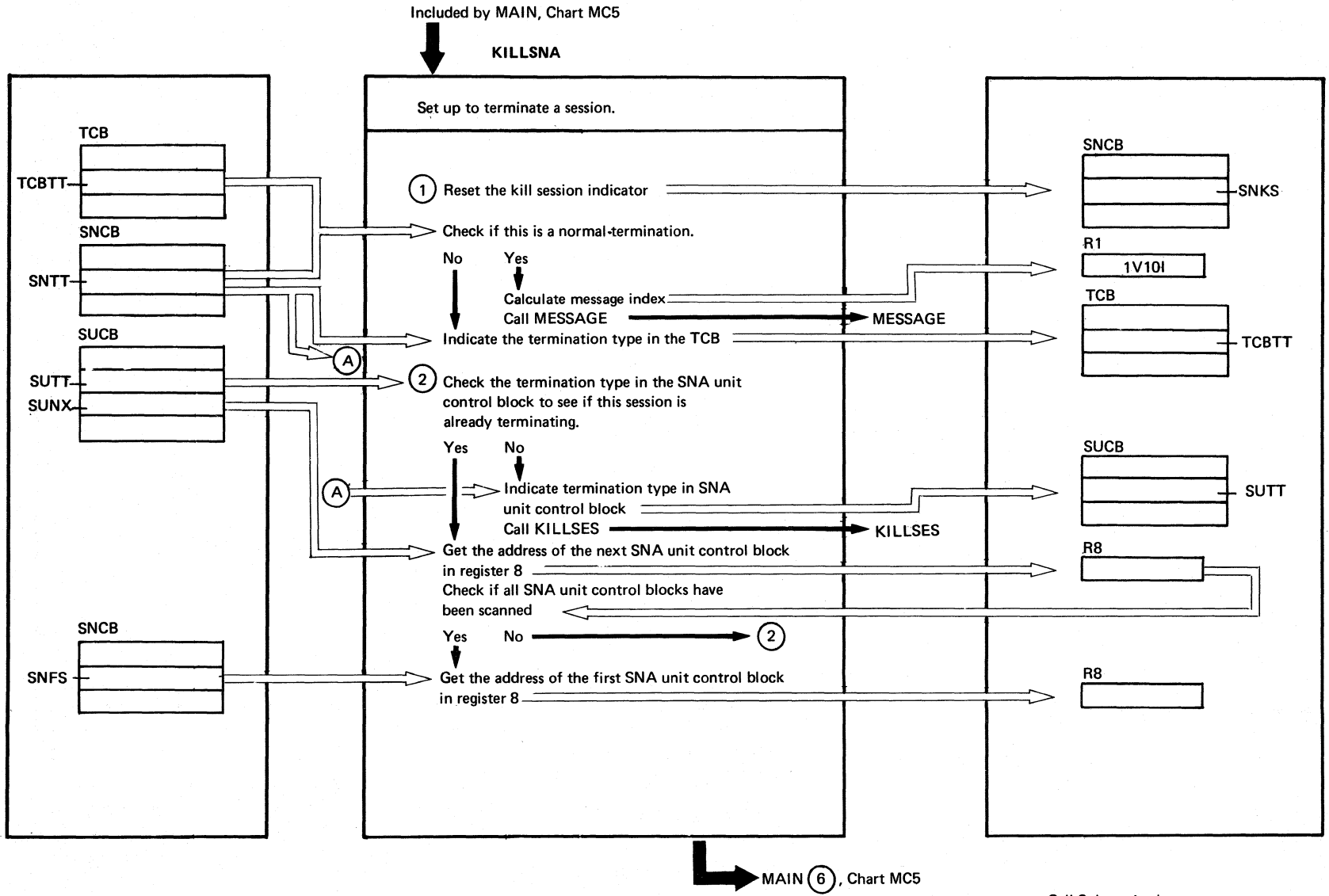


Chart MC5
Part 1 of 3


Input
Processing
Output
SNCB
SNFS
SNSS
SNTT
SNCB
SNSB
SNKS
SNCB
SNFS
TCB
TCTT
04 The work ECB was posted.
Reset the life bit in the ECB.
Get the address of the first SNA control block in R7.
If no request to terminate SNA processing then:
Reset the SNA stop indicator.
If not TPEND or STOP SNA immediately requested then:
Update termination type.
Reactivate the DOS/VS subtask (post subtask ECB) to quiesce logon requests.
05 If SNA processing has not to be terminated (request from subtask) then:
KILLSNA
MC18
06 Look for work.
SCAN
MC7
07 Get address of 1st active SUCB.
If any SUCB chained, process them or wait for reactivation of SNA-MANAGER:
No SUCB exists anymore. If SNA-MANAGER has not to be stopped then:
05
05
6
09
09
SNCB
SNEB
R7
SNCB
SNSS='0'B
SNTT


Input
Processing
Output
SNCB
SNMA
R3
SNCB
SNFS
SUCB
If LOGON PROCESSOR1
(IPW$$LH) is processing then:
08
Unlock the SUCB/LUCB chain
via IPW$RLR macro.
Return to MAINLINE to process
termination.
09
Unlock the SUCB/LUCB chain
via IPW$RLR macro.
Return to MAIN(1) to wait for
next event (ECB-POSTING).
09
MAINLINE
01



Extended Description	Include Segment	Call Subroutine/ Macro	Chart
1 1V10I RJE,SNA IS IN SHUTDOWN PERIOD		MESSAGE	MC11
2 Terminate the session with the logical unit related to this SNA unit control block.		KILLSES	MC12

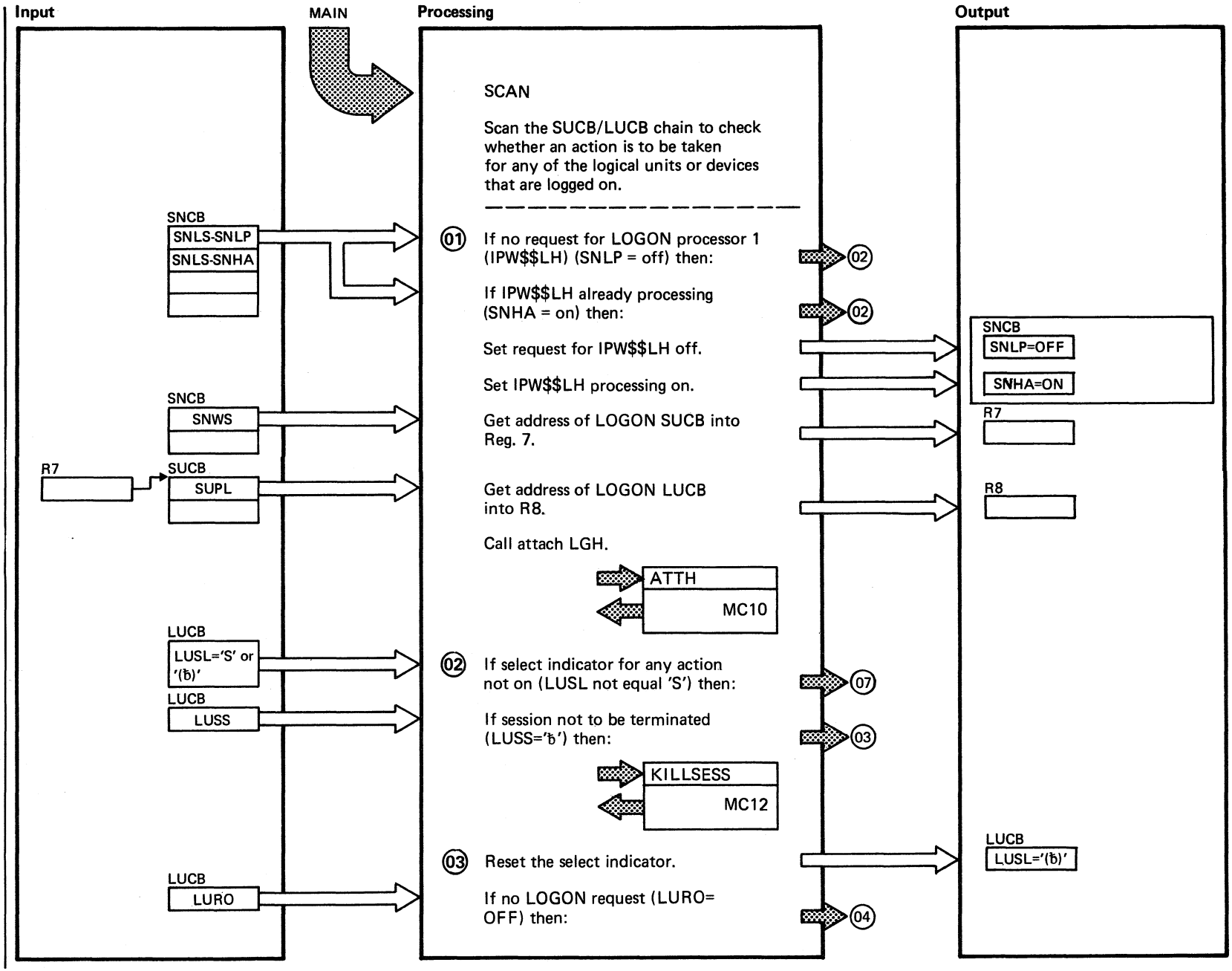
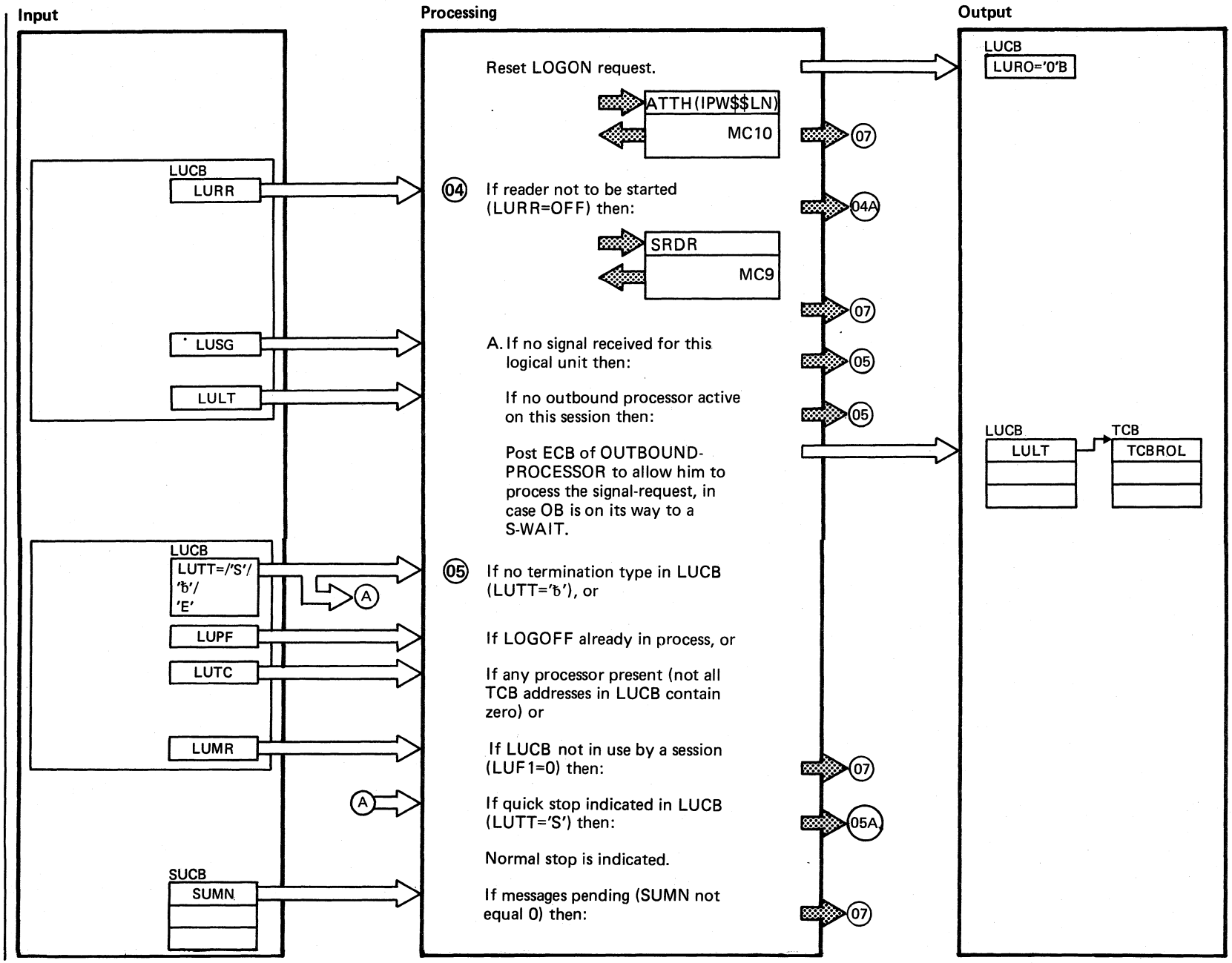
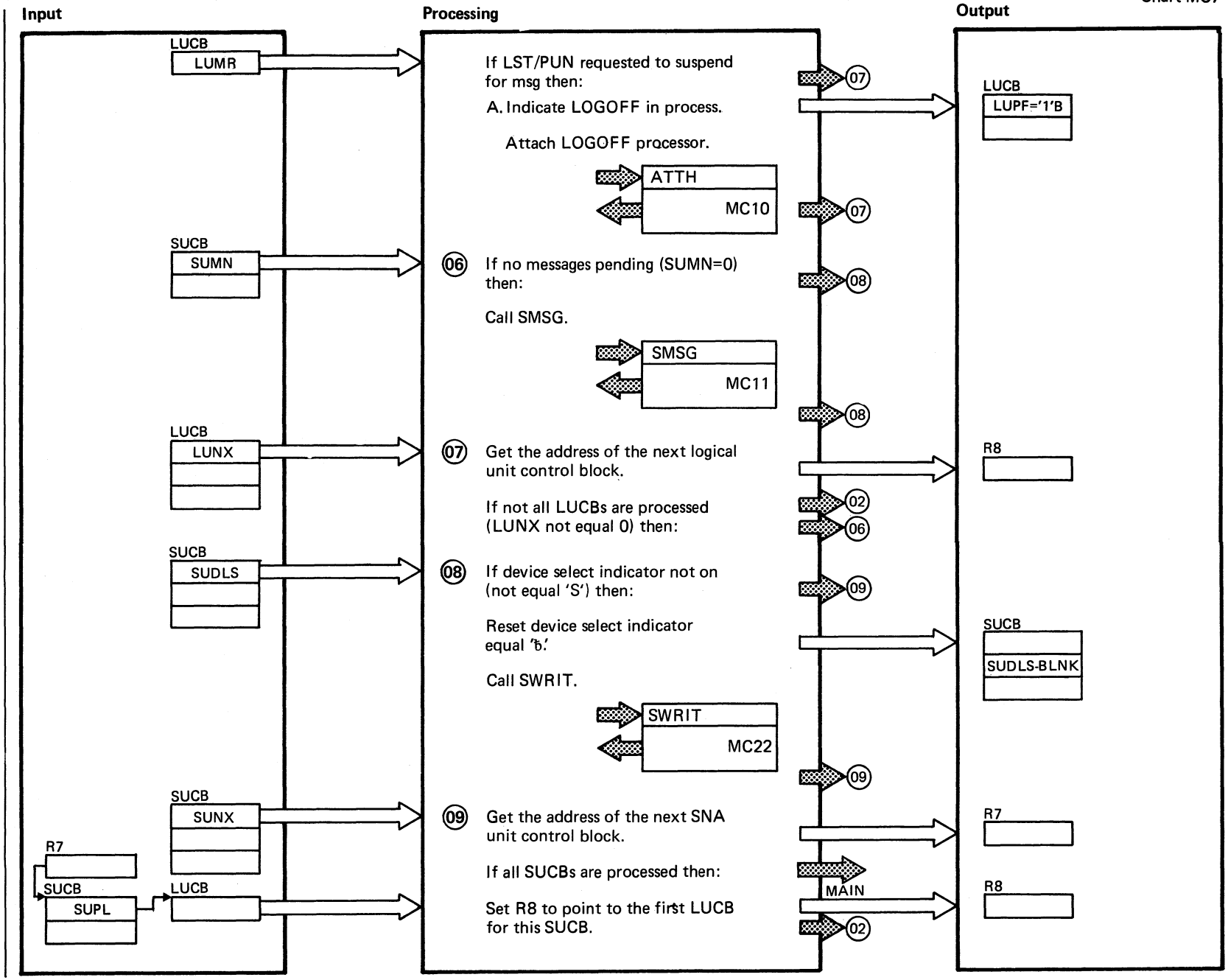
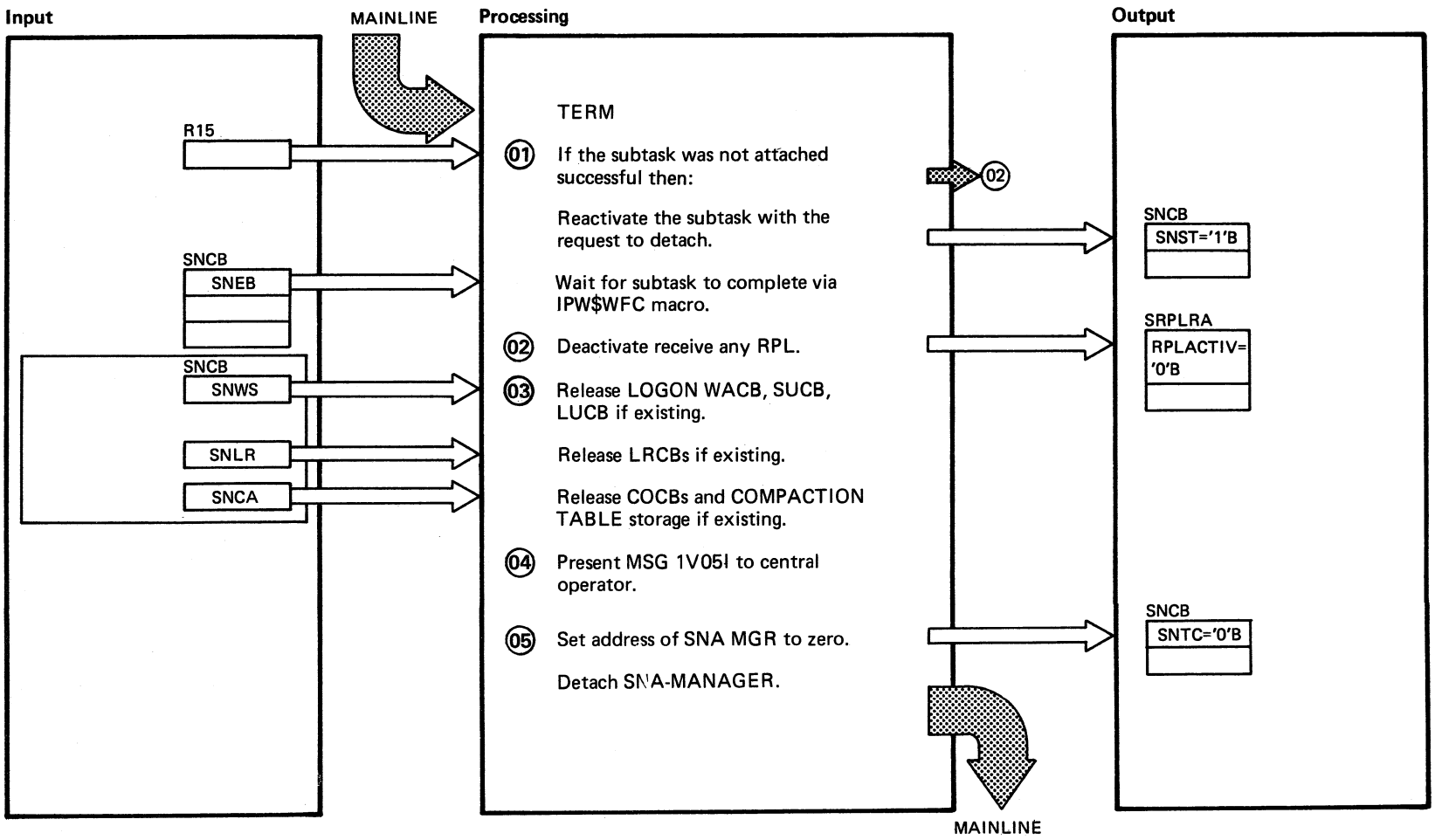


Chart MC7
Part 1 of 4


Input
Processing
Output
LUCB
LURR
LUSG
LULT
LUCB
LUTT=/'S'/
'b'/
'E'
LUPF
LUTC
LUMR
A
SUCB
SUMN
Reset LOGON request.
ATTH(IPW$$LN)
MC10
07
04
If reader not to be started
(LURR=OFF) then:
04A
SRDR
MC9
07
A. If no signal received for this
logical unit then:
05
If no outbound processor active
on this session then:
05
Post ECB of OUTBOUND-
PROCESSOR to allow him to
process the signal-request, in
case OB is on its way to a
S-WAIT.
05
If no termination type in LUCB
(LUTT='b'), or
If LOGOFF already in process, or
If any processor present (not all
TCB addresses in LUCB contain
zero) or
If LUCB not in use by a session
(LUF1=0) then:
07
If quick stop indicated in LUCB
(LUTT='S') then:
05A
Normal stop is indicated.
If messages pending (SUMN not
equal 0) then:
07
LUCB
LURO='0'B
LUCB
LULT
TCB
TCBROL


Input
Processing
Output
LUCB
LUMR
SUCB
SUMN
LUCB
LUNX
SUCB
SUDLS
SUCB
SUNX
R7
SUCB
SUPL
LUCB
If LST/PUN requested to suspend for msg then:
A. Indicate LOGOFF in process.
Attach LOGOFF processor.
ATTH
MC10
06
If no messages pending (SUMN=0) then:
Call SMSG.
SMSG
MC11
07
Get the address of the next logical unit control block.
If not all LUCBs are processed (LUNX not equal 0) then:
08
If device select indicator not on (not equal 'S') then:
Reset device select indicator equal 'b'.
Call SWRIT.
SWRIT
MC22
09
Get the address of the next SNA unit control block.
If all SUCBs are processed then:
Set R8 to point to the first LUCB for this SUCB.
07
07
08
08
02
06
09
09
MAIN
02
LUCB
LUPF='1'B
R8
SUCB
SUDLS-BLNK
R7
R8

Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(01) IPW$$LH loops through the LRCBs. When 1 LRUB is processed, IPW$$LH scans for the next one. Only when all LRUBs are processed, SNHA is set off. Now SNA-MANAGER can attach IPW$$LH again upon request. (02) "Look for Work" routine of SNA MANAGER scans all LUCBs for the select indicator 'S' in LUSL. This major scan argument indicates that an action for this LUCB has to be started. Further analysis of the action bytes in the LUCB will cause the appropriate task to be attached. If LURO is set, SNA MANAGER will attach Logon Processor 2 IPW$$LN to process stage 2 of the LOGON process. The SUCB contains a summary of outbound card/print data flow which is indicated in the BIND data. This summary is used to check in the SNA MANAGER if at least one session is bound, which allows card or print outbound data flow for an MLU work station. (05) Attach logoff processor. (06) Call " Start Message Processor" routine. (08) Queue management selects a printer in the device list, if print output is available. To select a specific printer, it must meet following criteria: - printer must be started (SUHS = ON) - printer must be available (SUHU = ON) - and classes of output and printer must match. Then queue management sets following indicator: - device not available (SUHU = OFF) - output available (SUHU = ON) - select device indicator (SUDLS = 'S').	SMSG		



Extended Description	Include Segment	Call Subroutine/ Macro	Chart
04 1V05I RJE, SNA terminated.			

Chart MC9

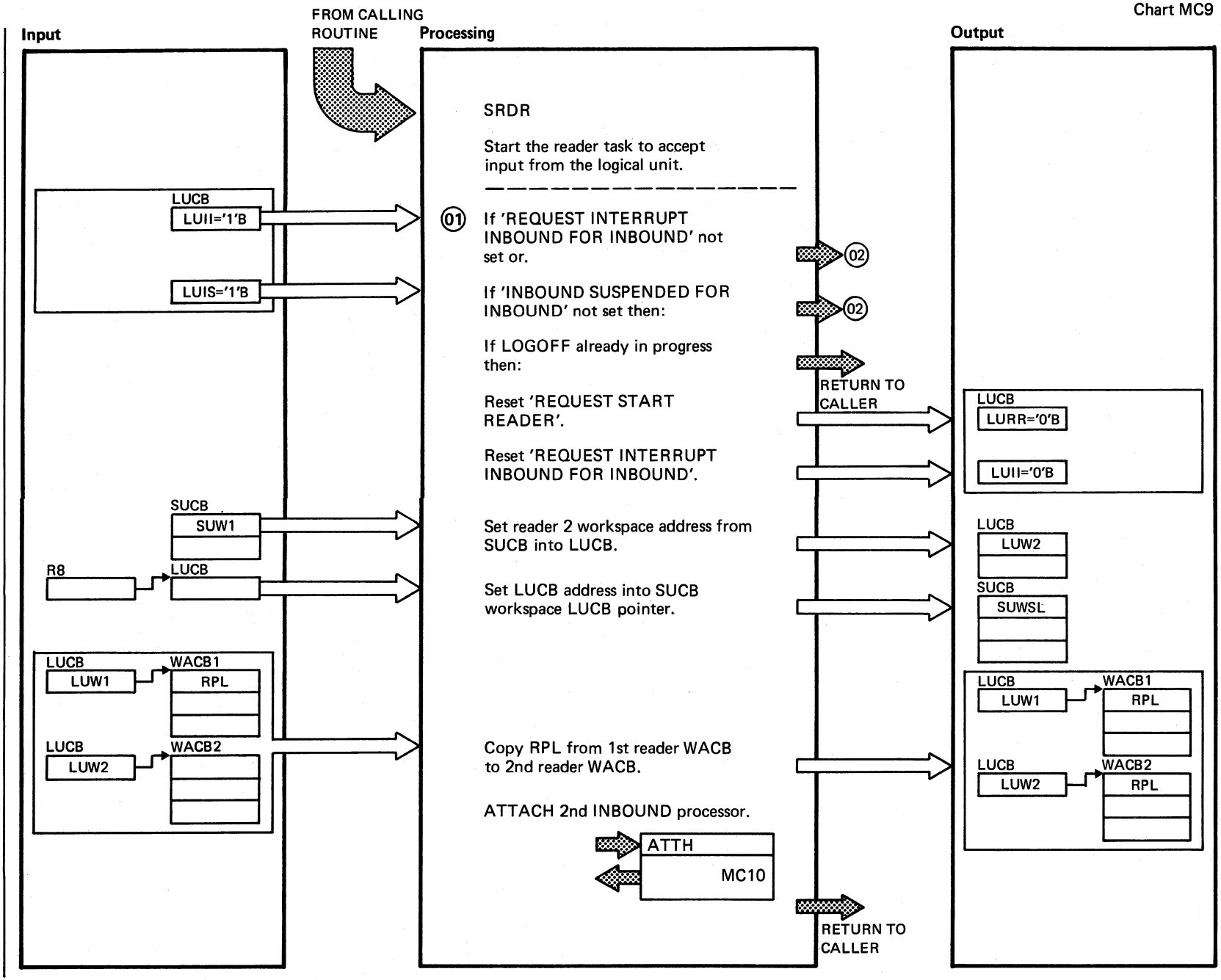


Chart MC9
Part 1 of 3

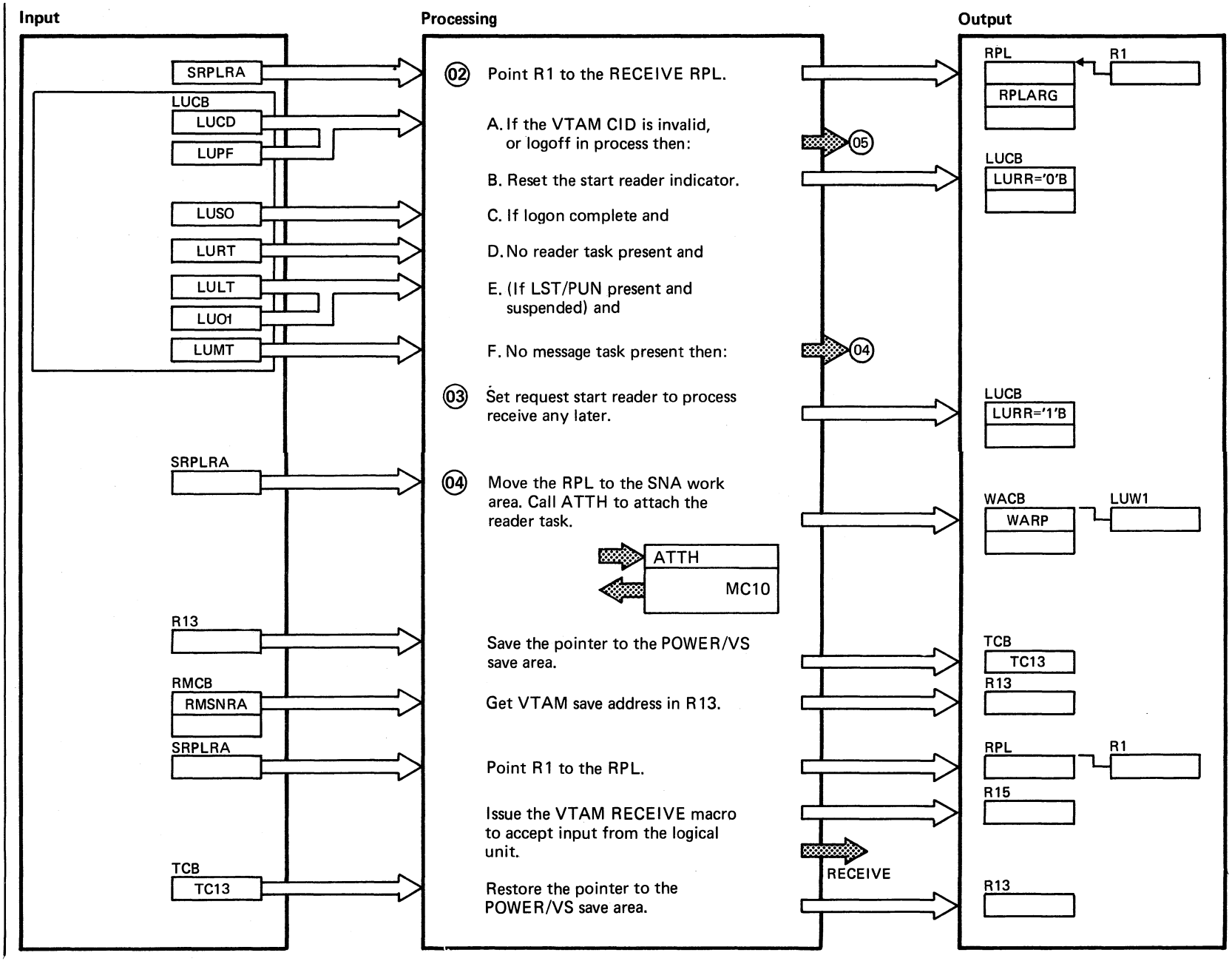
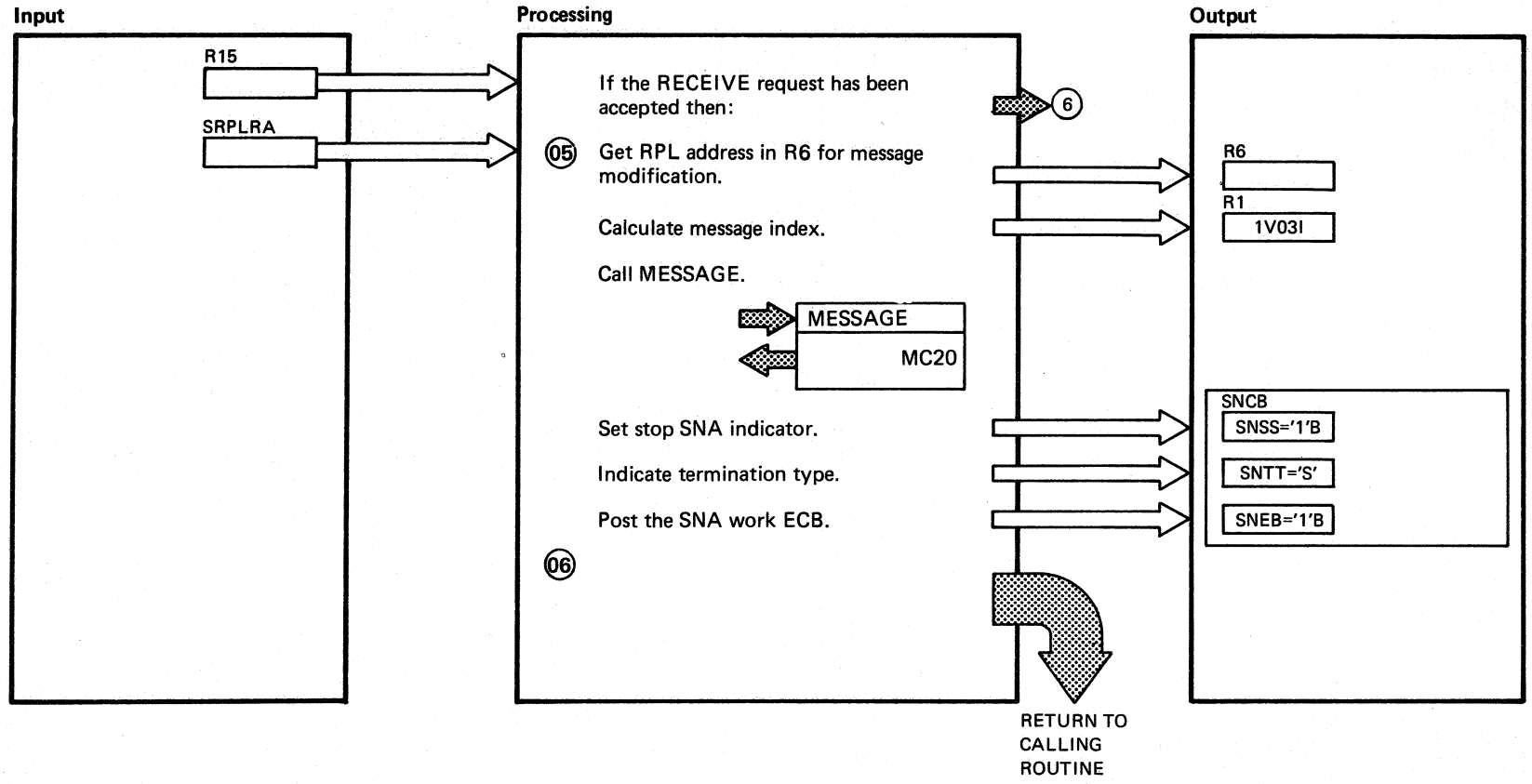
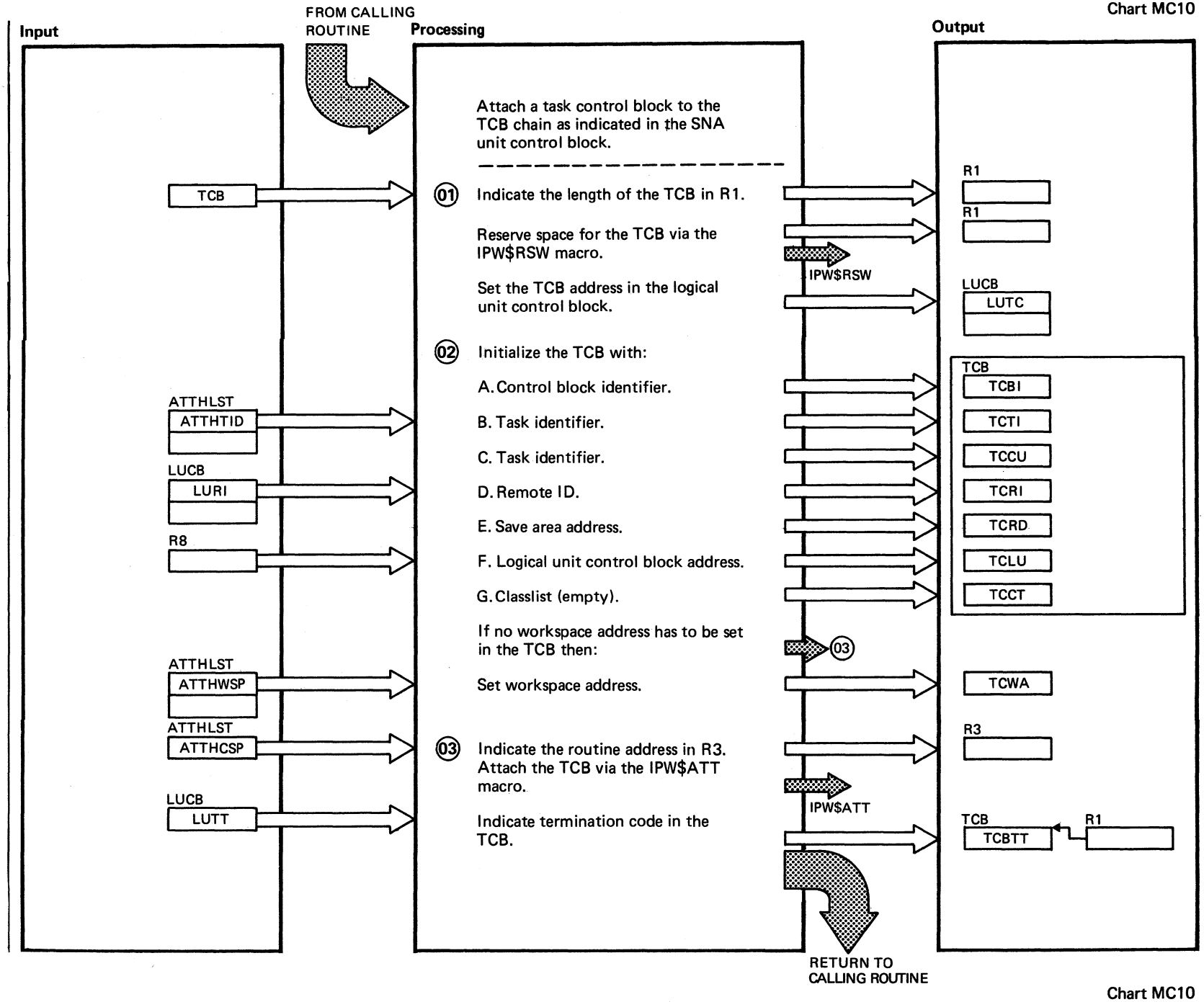


Chart MC9
Part 2 of 3



Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(01) On entry, R8 points to the LUCB. (05) 1 V031 ERROR ON RECEIVE RTNCD, FDBK = xx, yy SENSE = ssss.			


Input
FROM CALLING ROUTINE
Processing
Output
Attach a task control block to the TCB chain as indicated in the SNA unit control block.
TCB
01 Indicate the length of the TCB in R1.
R1
R1
Reserve space for the TCB via the IPW$RSW macro.
IPW$RSW
Set the TCB address in the logical unit control block.
LUCB
LUTC
02 Initialize the TCB with:
TCB
TCBI
ATTHLST
ATTHTID
A. Control block identifier.
B. Task identifier.
TCTI
LUCB
LURI
C. Task identifier.
TCCU
D. Remote ID.
TCRI
R8
E. Save area address.
TCRD
F. Logical unit control block address.
TCLU
G. Classlist (empty).
TCCT
If no workspace address has to be set in the TCB then:
03
ATTHLST
ATTHWSP
Set workspace address.
TCWA
ATTHLST
ATTHCSP
03 Indicate the routine address in R3. Attach the TCB via the IPW$ATT macro.
R3
IPW$ATT
LUCB
LUTT
Indicate termination code in the TCB.
TCB
TCBTT
R1
RETURN TO CALLING ROUTINE

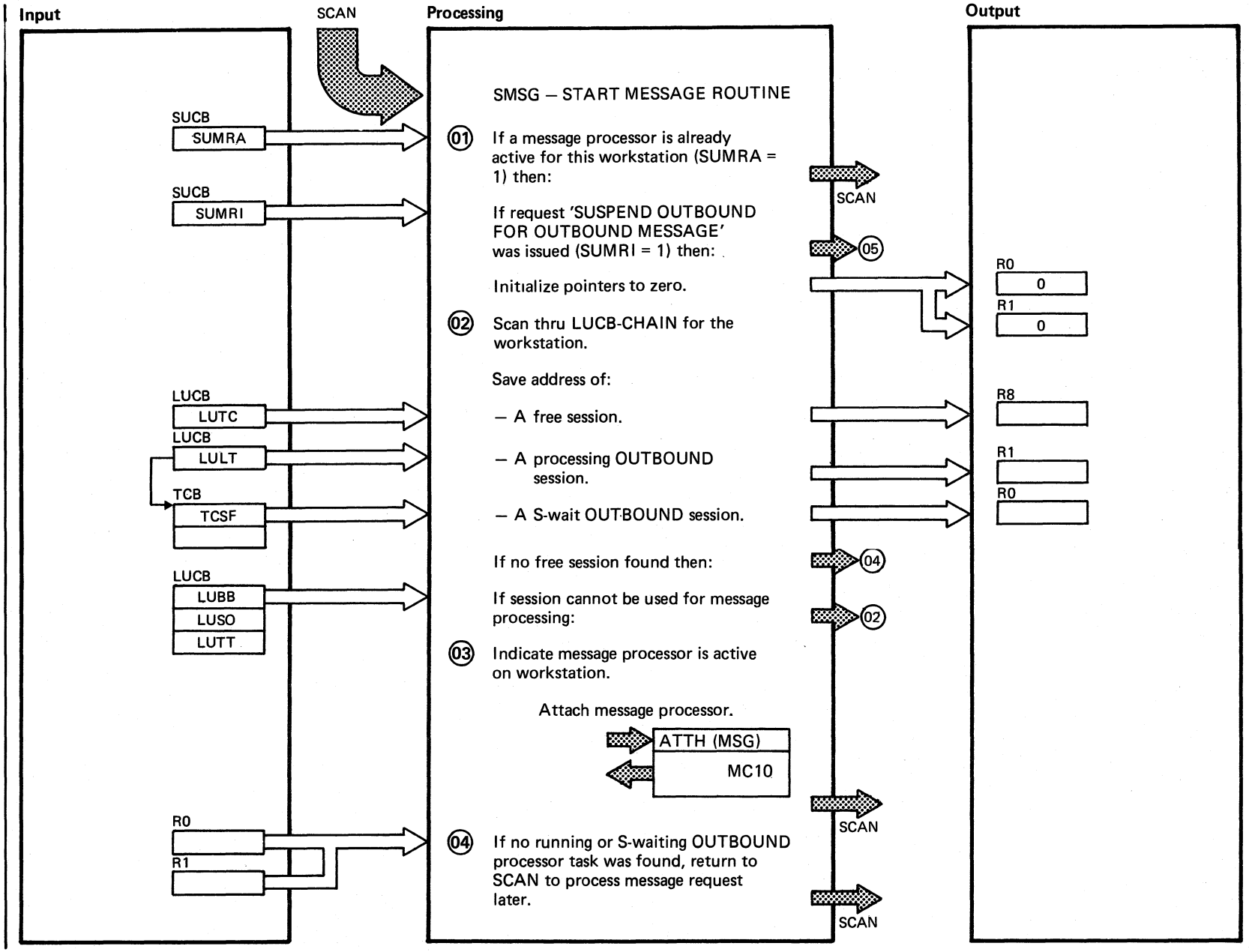
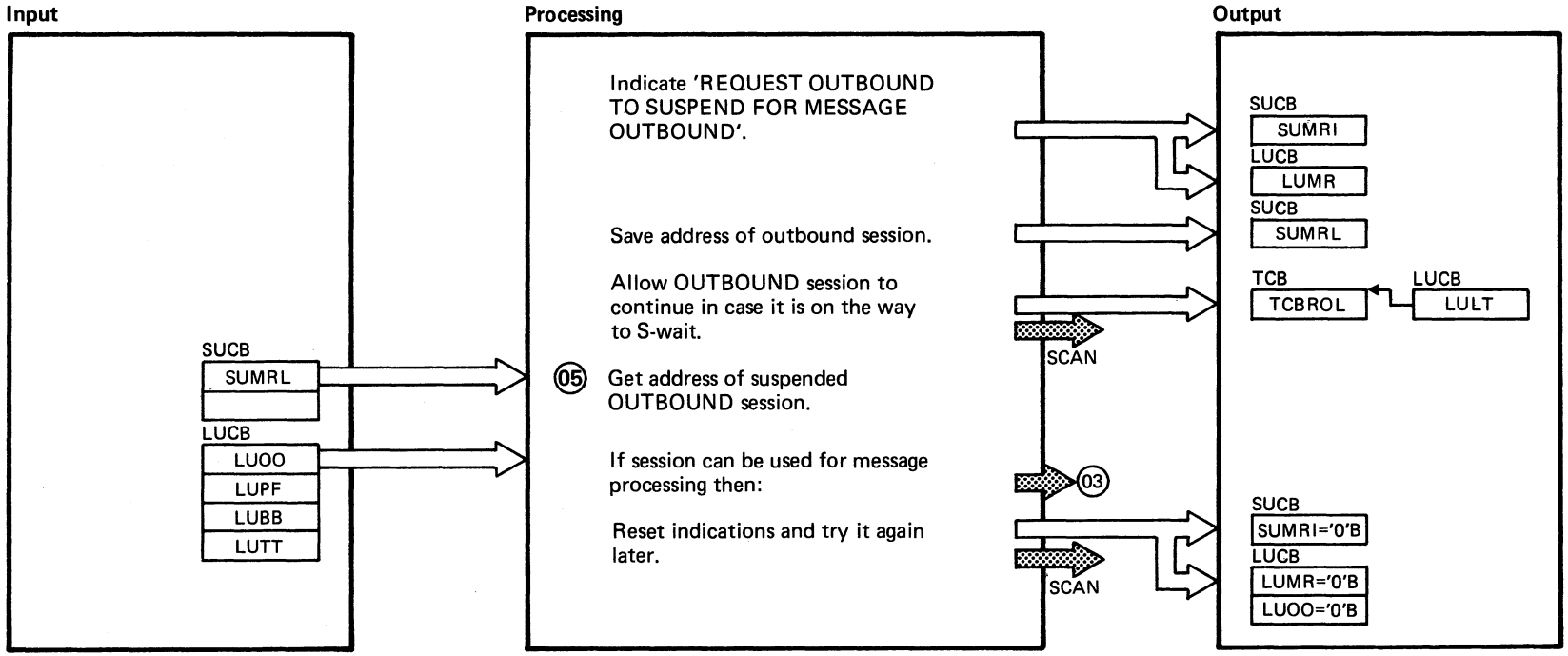


Chart MC11
Part 1 of 2



Extended Description

	Include Segment	Call Subroutine/ Macro	Chart
(01) The 'SCAN' - Routine checked the 'SUMN' - field in the SUCB for the number of messages queued. IF content of SUMN is greater zero, that means messages are queued, SCAN calls the SMSG routine.			

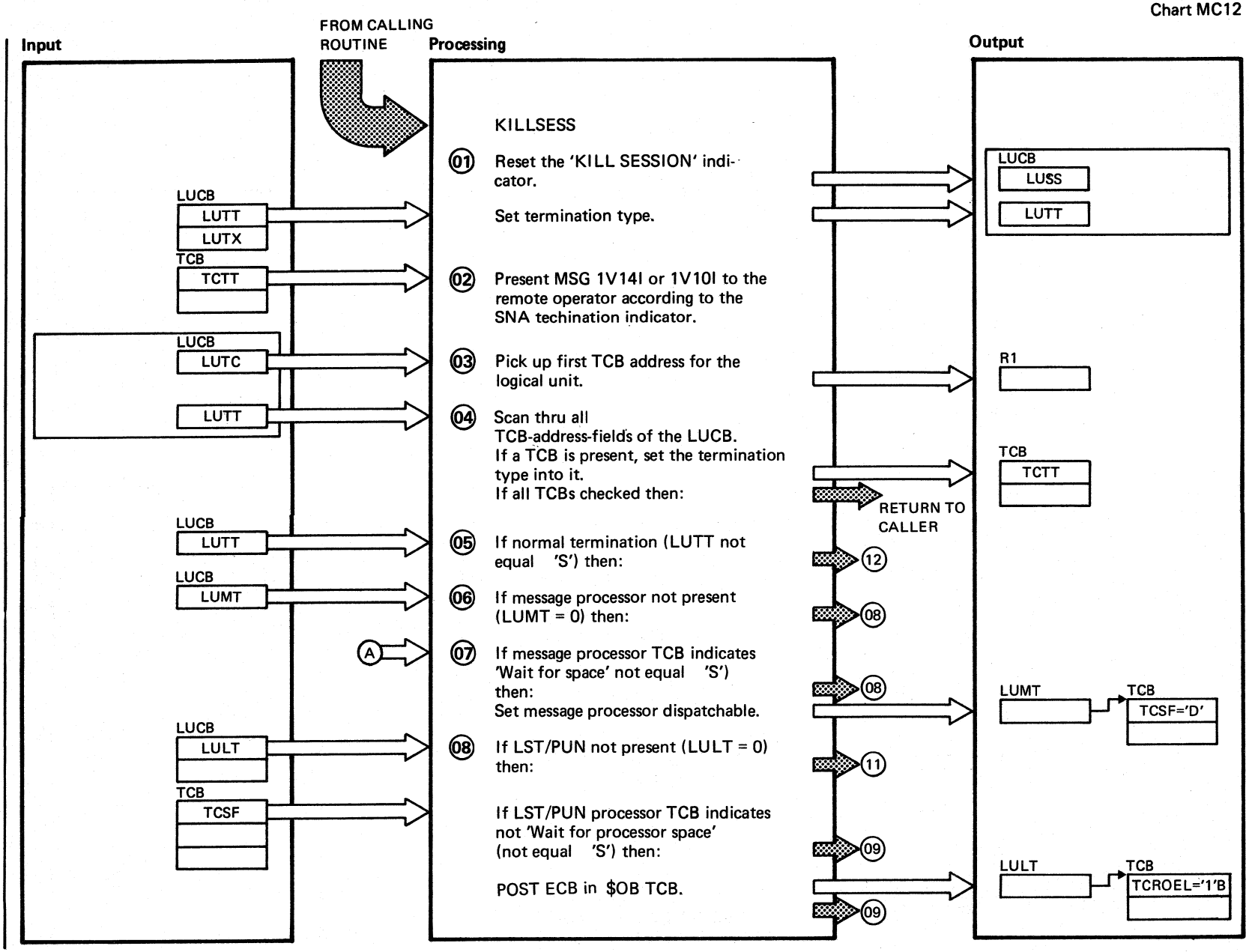
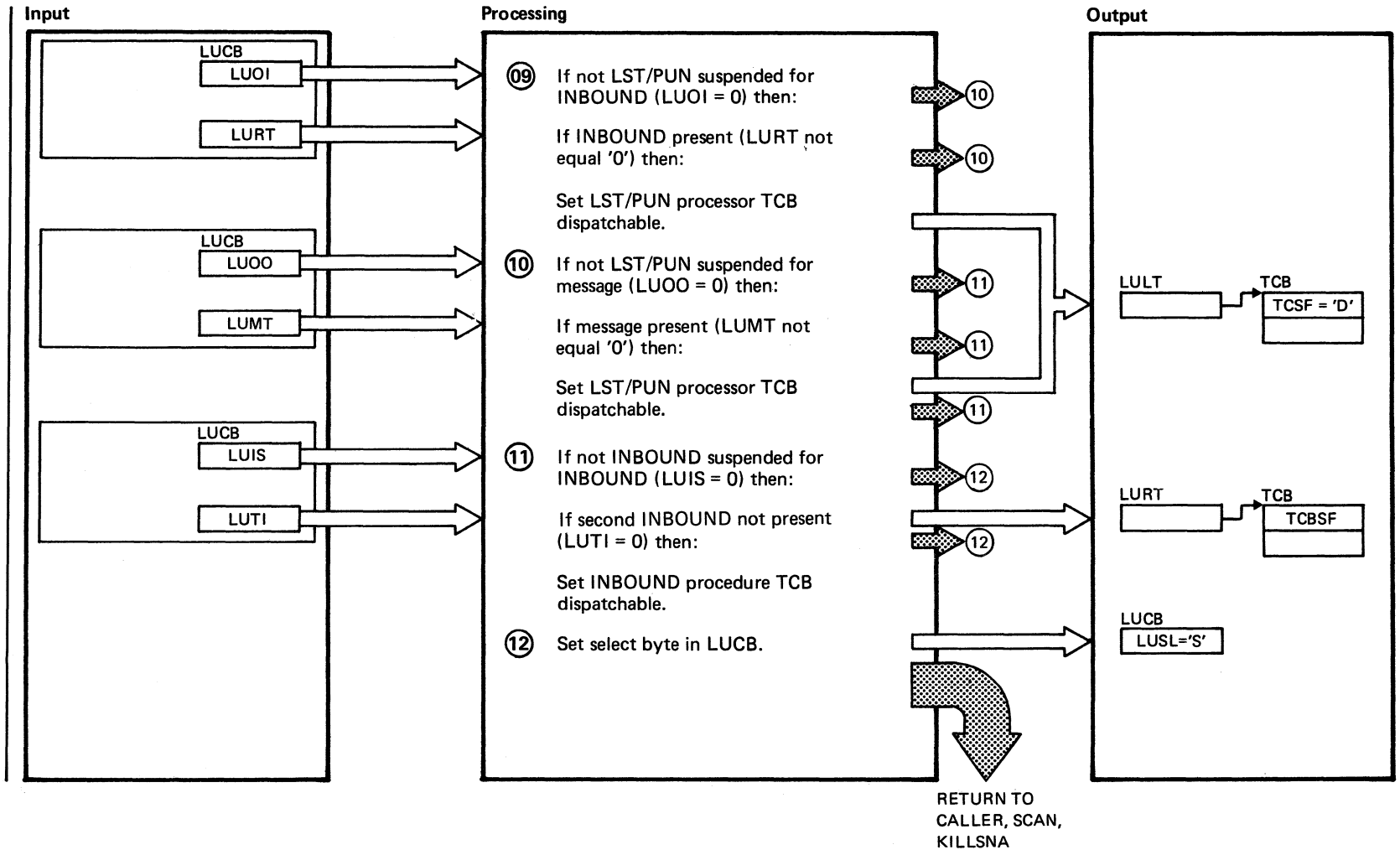
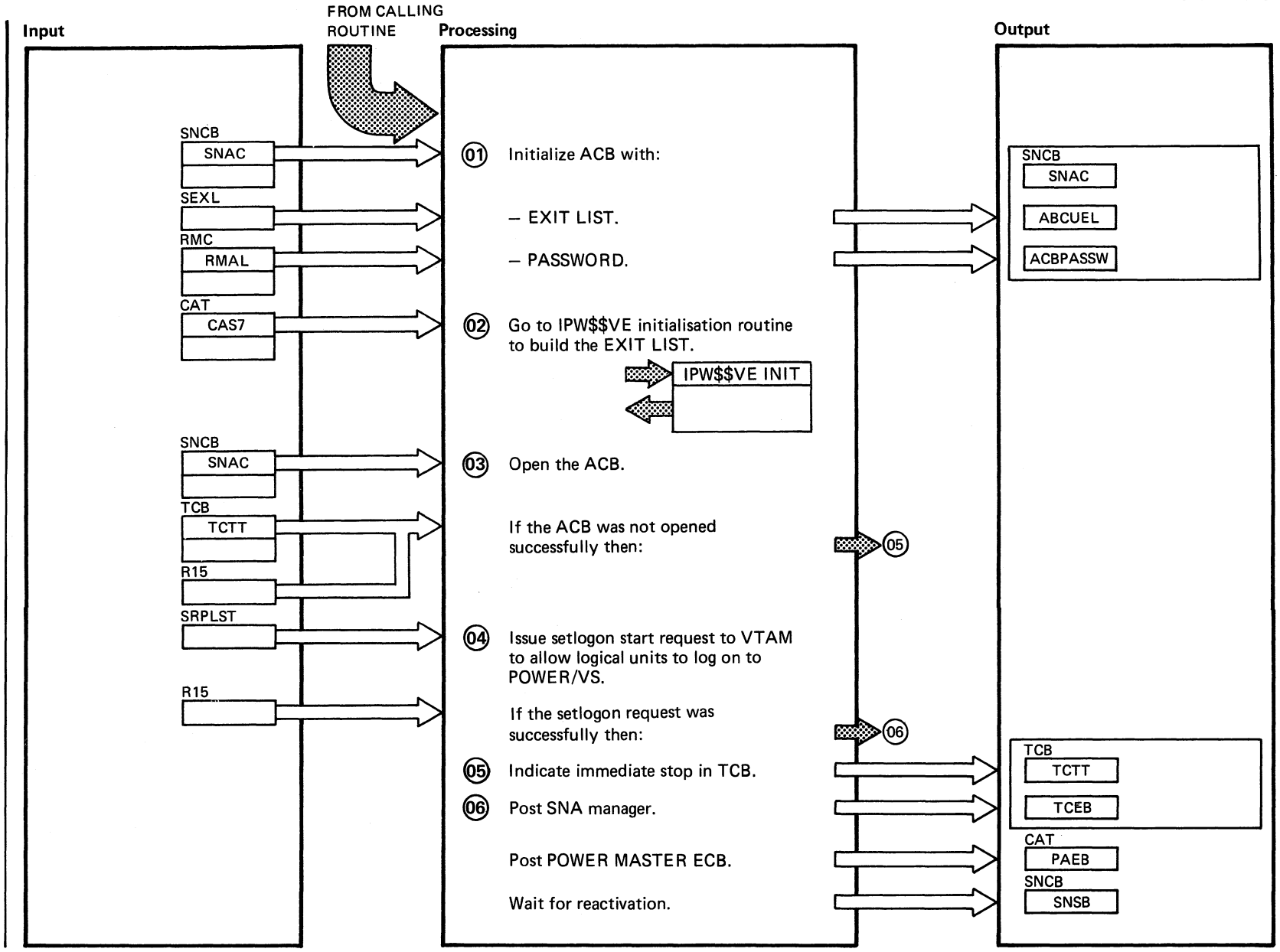
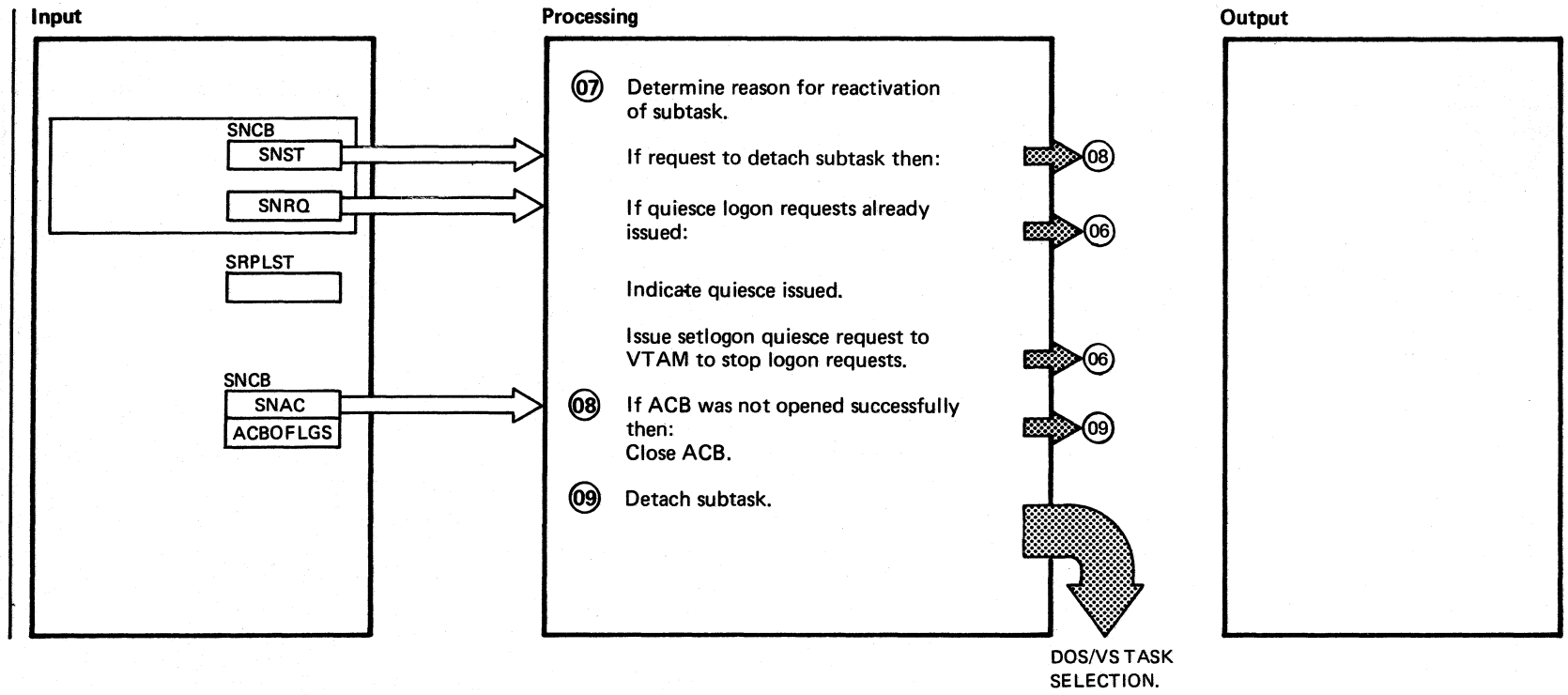


Chart MC12
Part 1 of 3


Input
Processing
Output
LUCB
LUOI
LURT
LUCB
LUOO
LUMT
LUCB
LUIS
LUTI
09 If not LST/PUN suspended for INBOUND (LUOI = 0) then:
If INBOUND present (LURT not equal '0') then:
Set LST/PUN processor TCB dispatchable.
10 If not LST/PUN suspended for message (LUOO = 0) then:
If message present (LUMT not equal '0') then:
Set LST/PUN processor TCB dispatchable.
11 If not INBOUND suspended for INBOUND (LUIS = 0) then:
If second INBOUND not present (LUTI = 0) then:
Set INBOUND procedure TCB dispatchable.
12 Set select byte in LUCB.
10
10
11
11
11
12
12
LULT
TCB
TCSF = 'D'
LURT
TCB
TCBSF
LUCB
LUSL='S'
RETURN TO CALLER, SCAN, KILLSNA

Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(02) 1V10I RJE, SNA is in shutdown period. 1V14I SESSION is in shutdown period.			

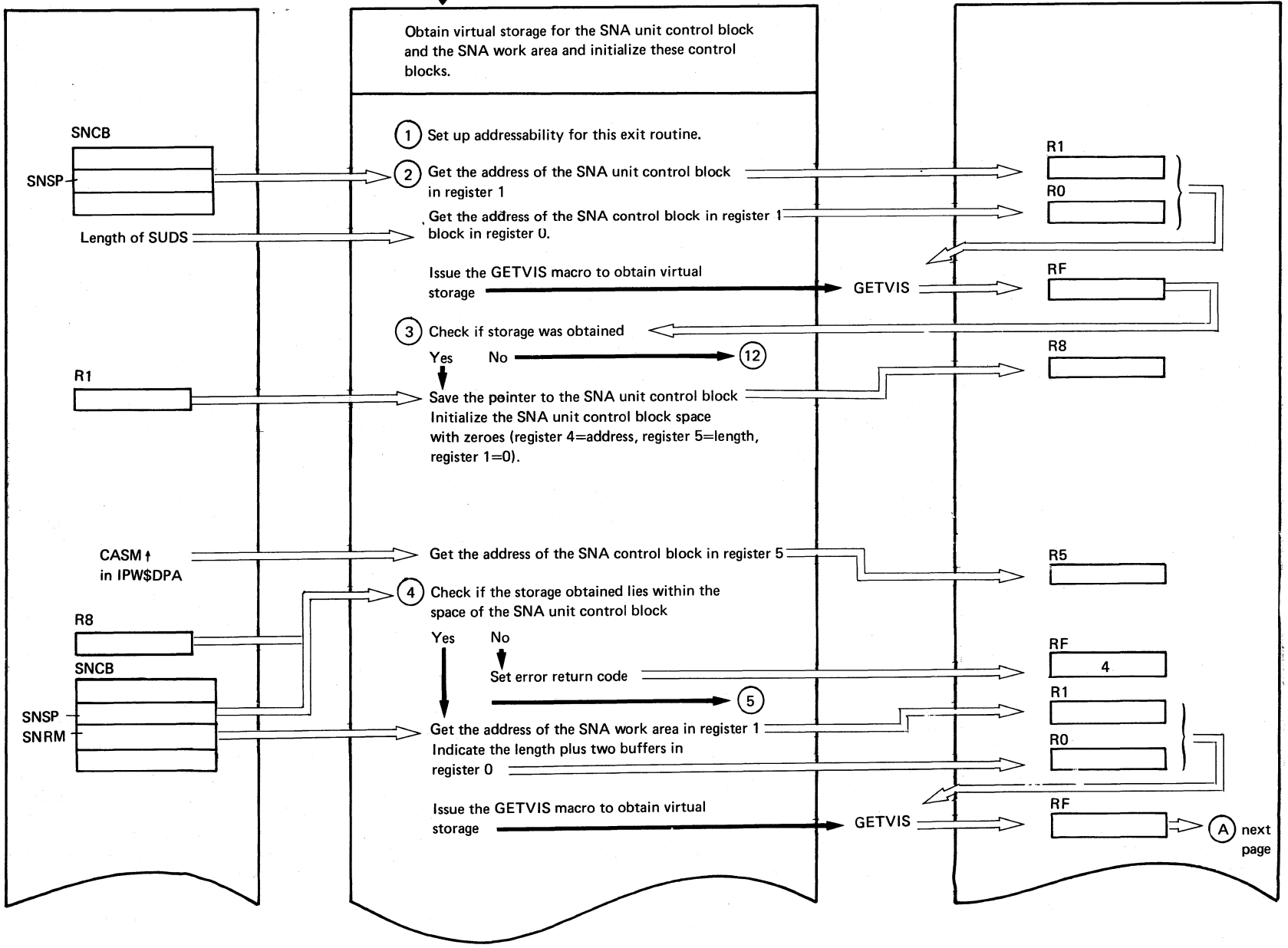

Input
FROM CALLING ROUTINE
Processing
Output
SNCB
SNAC
SEXL
RMC
RMAL
CAT
CAS7
SNCB
SNAC
TCB
TCTT
R15
SRPLST
R15
01 Initialize ACB with:
– EXIT LIST.
– PASSWORD.
02 Go to IPW$$VE initialisation routine to build the EXIT LIST.
IPW$$VE INIT
03 Open the ACB.
If the ACB was not opened successfully then:
05
04 Issue setlogon start request to VTAM to allow logical units to log on to POWER/VS.
If the setlogon request was successfully then:
06
05 Indicate immediate stop in TCB.
06 Post SNA manager.
Post POWER MASTER ECB.
Wait for reactivation.
SNCB
SNAC
ABCUEL
ACBPASSW
TCB
TCTT
TCEB
CAT
PAEB
SNCB
SNSB


Input
Processing
Output
SNCB
SNST
SNRQ
SRPLST
SNCB
SNAC
ACBOFLGS
07 Determine reason for reactivation of subtask.
If request to detach subtask then:
If quiesce logon requests already issued:
Indicate quiesce issued.
Issue setlogon quiesce request to VTAM to stop logon requests.
08 If ACB was not opened successfully then:
Close ACB.
09 Detach subtask.
08
06
06
09
DOS/VS TASK SELECTION.

Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(1) The DOS/VS subtask is attached in the INIT routine, and is executed to open the VTAM ACB and to issue a SETLOGON START request, to allow LOGON requests to POWER/VS. As VTAM does some of its processing under the PIB of the program which issued the open ACB request, page faults in VTAM have some effect on the performance of this task. To minimize this effect on the main task, the DOS/VS subtask will be used. The subtask waits, after execution of the SETLOGON START, for reactivation by the maintask to disable LOGON requests and to close the ACB, when RJE, SNA processing is terminating. (3) (5) (7) (8) (9) (10) (11) This piece of coding forms an infinite wait loop which is only terminated when SNA processing is to be terminated.		OPENR(DOS/VS) SETLOGON(VTAM) POST(DOS/VS) WAIT(DOS/VS) SETLOGON(VTAM) POST(DOS/VS) CLOSER(DOS/VS) DETACH(DOS/VS)	

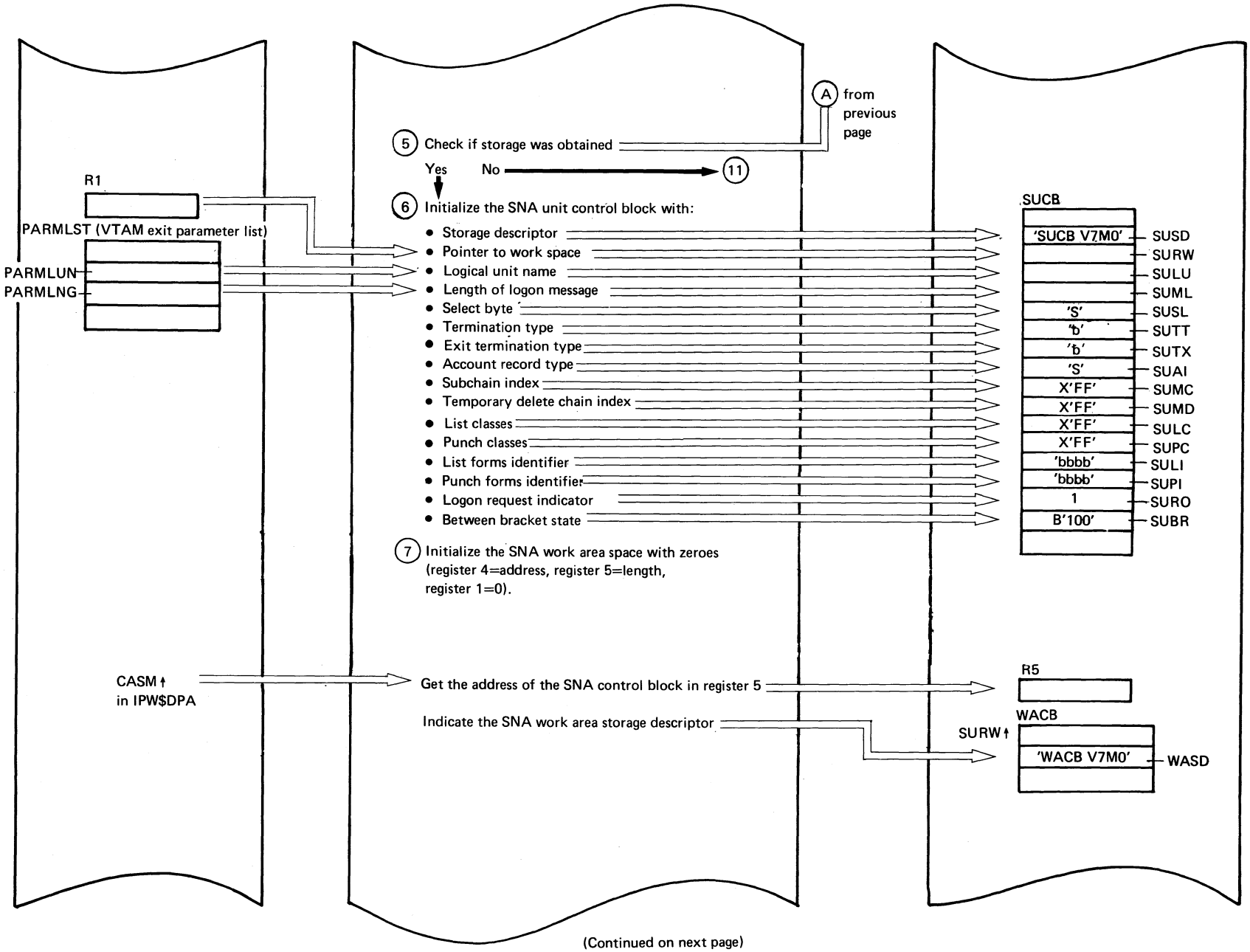
Called by VTAM

LGON (Part 1 of 4)



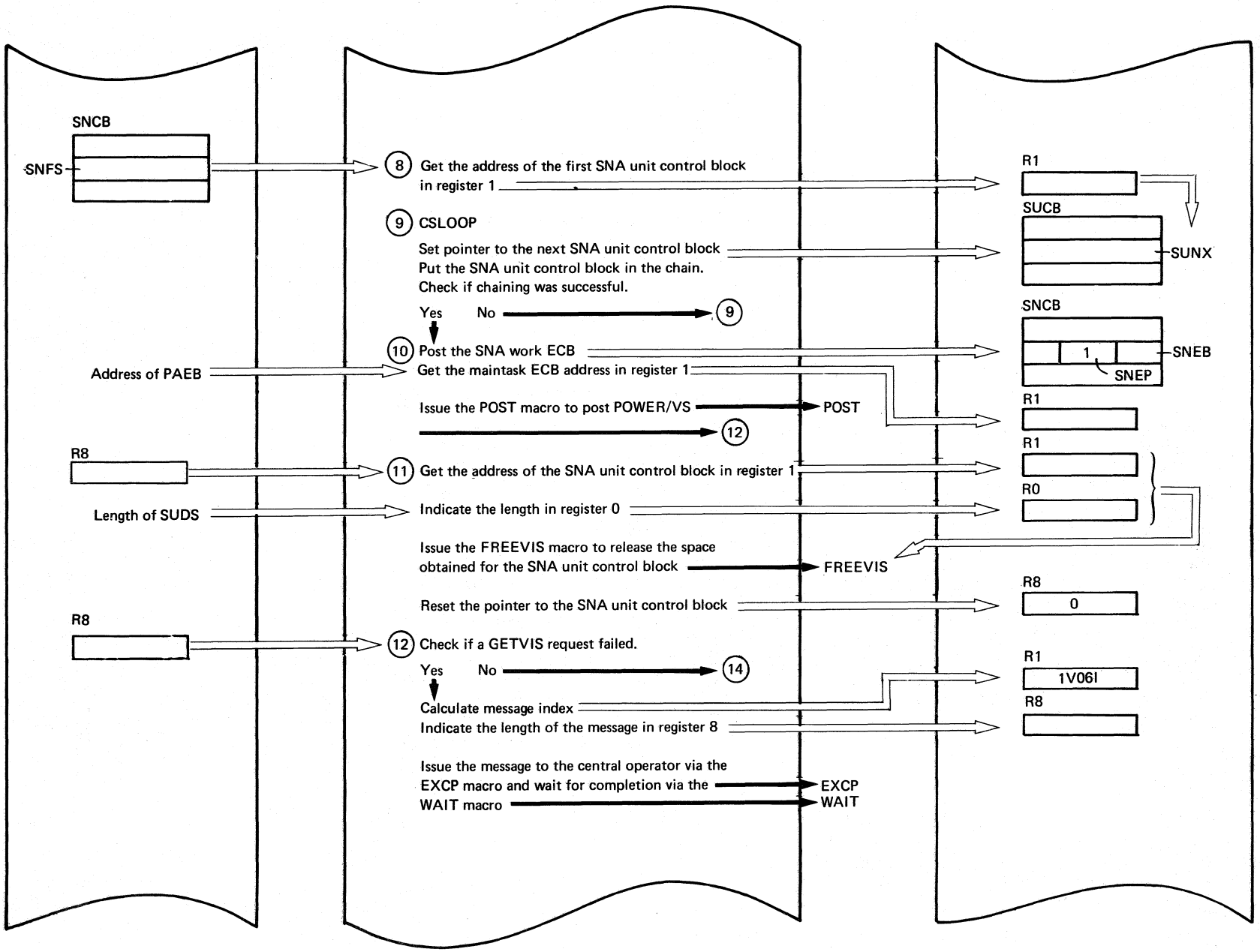
(Continued on next page)

LGON (Part 2 of 4)



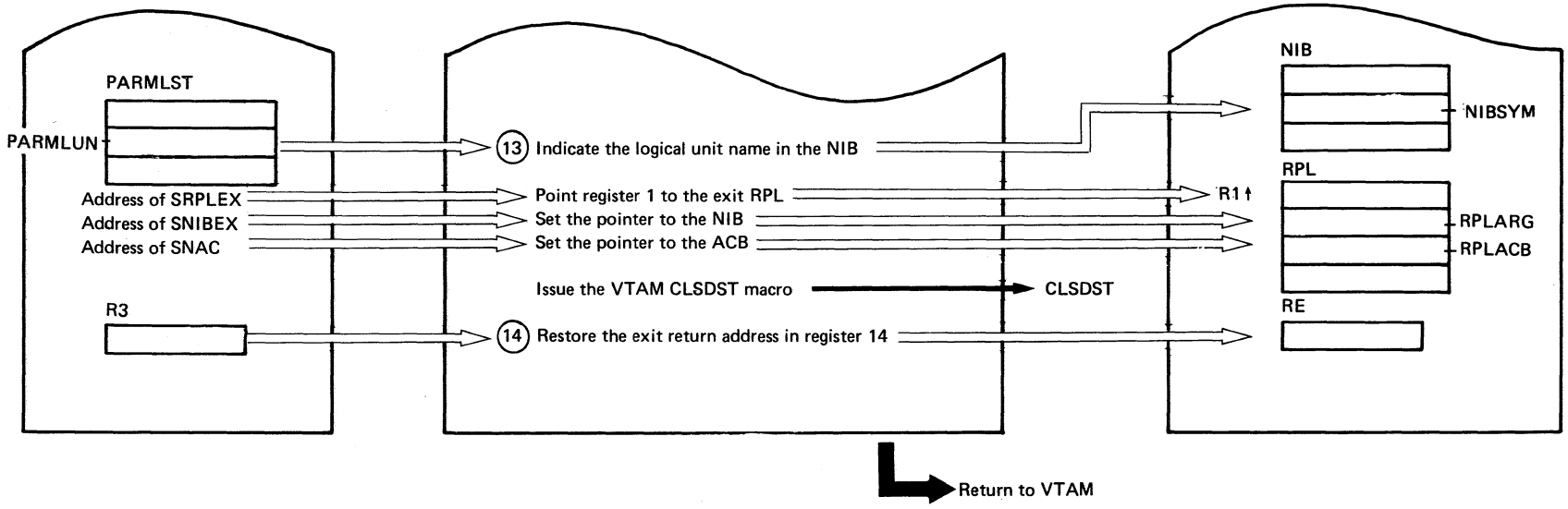
(Continued on next page)

LGON (Part 3 of 4)

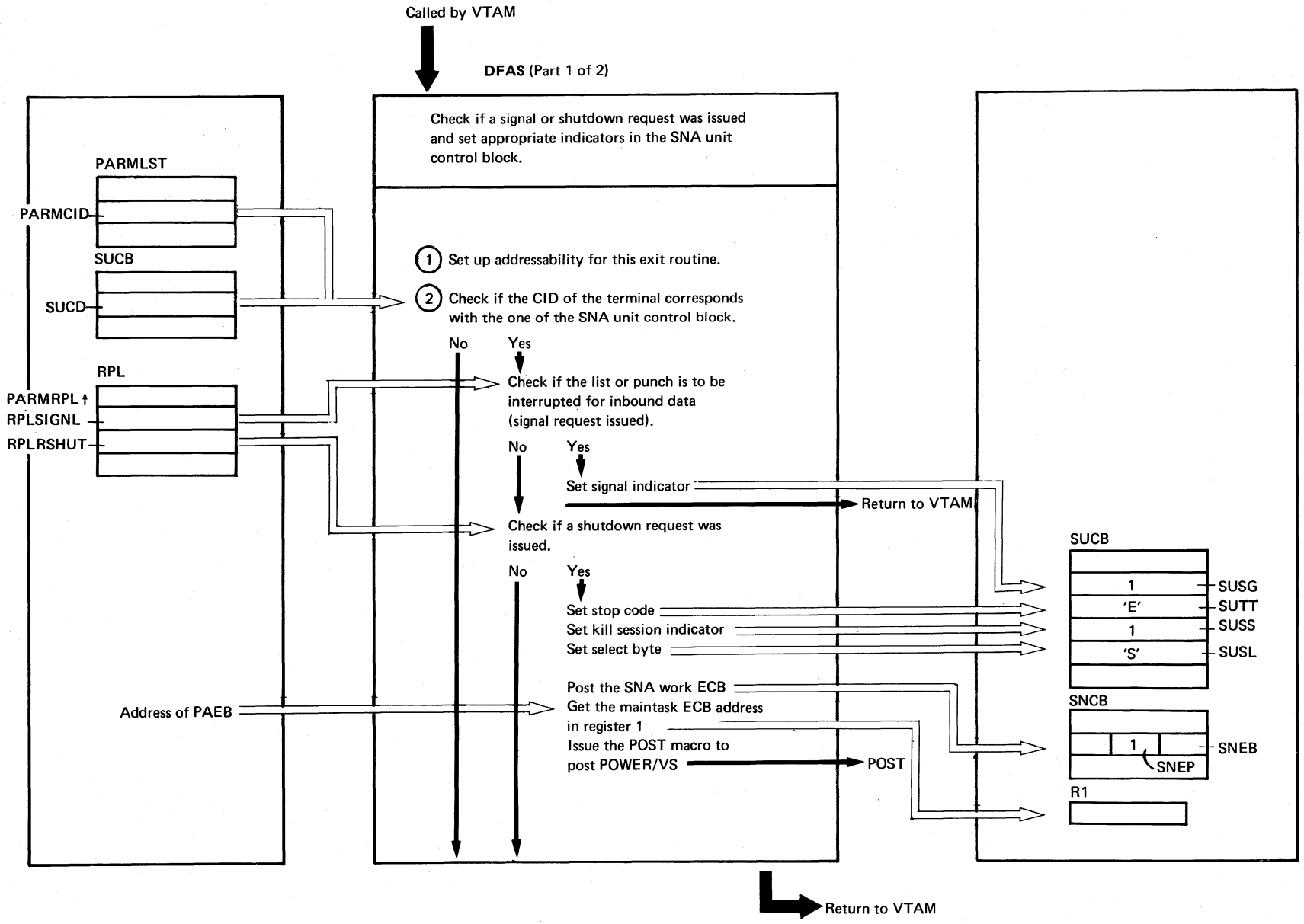


(Continued on next page)

LGON(Part 4 of 4)

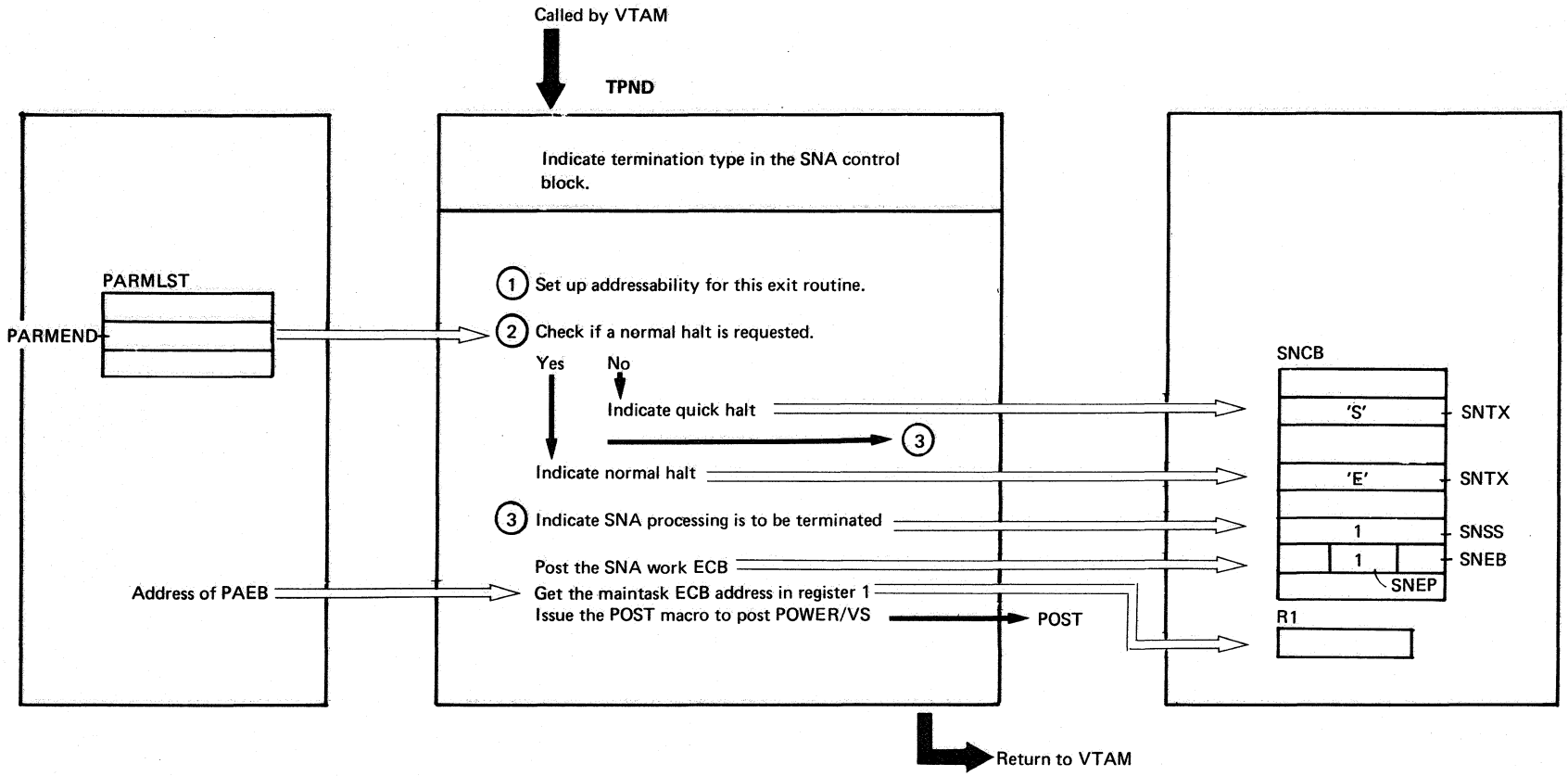


Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(1) On entry to this routine, register 1 contains the address of a 4-word parameter list (PARMLST: ACB address, Address of logical unit name, Reserved, Length of logon message), register 14 contains the VTAM return address and register 15 contains the address of the logon exit routine. Register 2 is used to save the parameter list address (from register 1) Register 3 is used to save the VTAM return address (from register 14) Register 5 is used as an SNA control block pointer Register 8 is used as an SNA unit control block pointer (initialized to 0) Register 9 is used as base address of the SNA manager Register 10 is used as base for the POWER/VS nucleus Register 13 is used for the VTAM save area address			
(2) (4)		GETVIS(DOS/VS)	
(10)		POST(DOS/VS)	
(11)		FREEVIS(DOS/VS)	
(12) 1V06I UNABLE TO LOGON luname		EXCP(DOS/VS) WAIT(DOS/VS)	

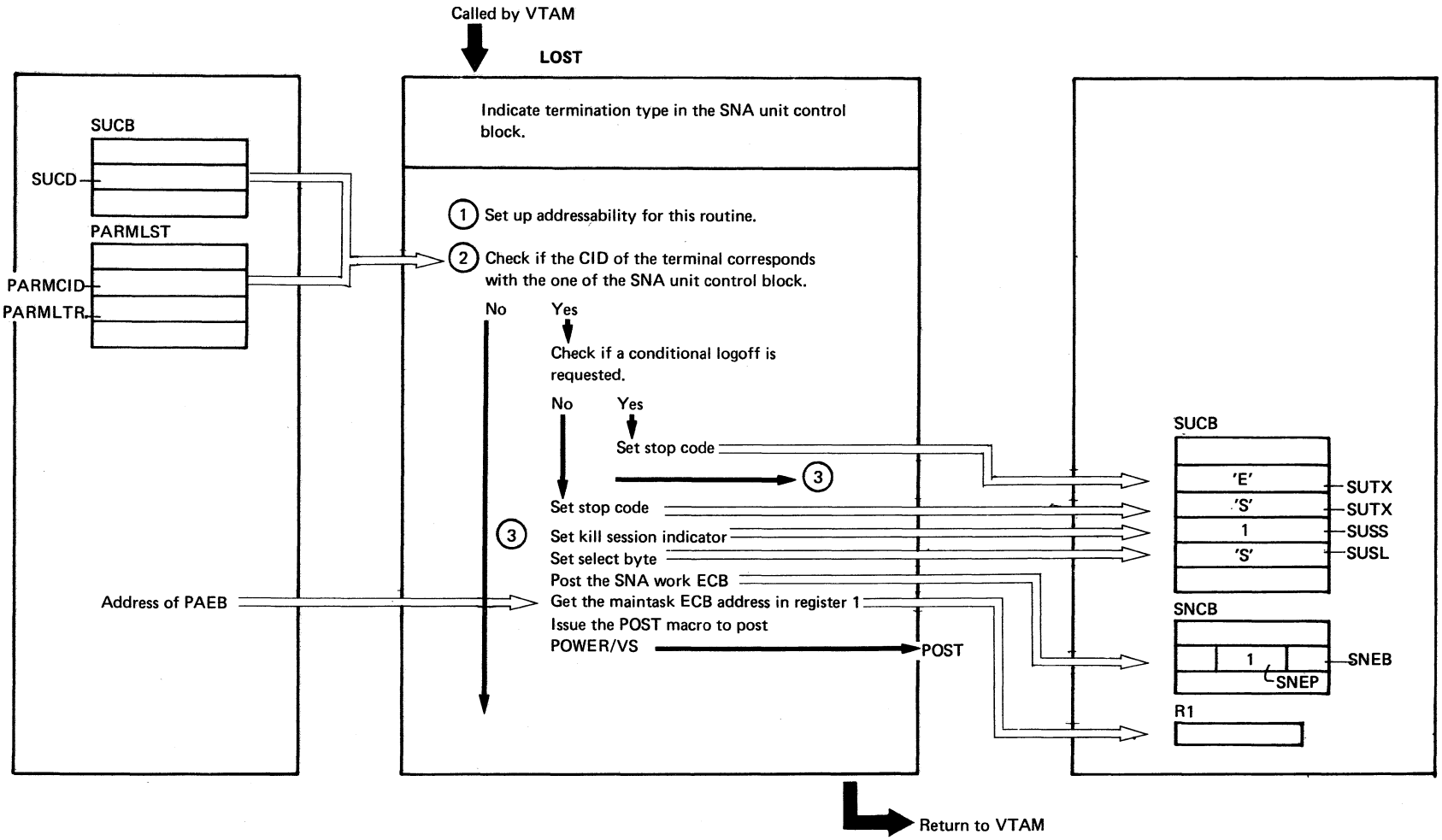

Called by VTAM
DFAS (Part 1 of 2)
Check if a signal or shutdown request was issued and set appropriate indicators in the SNA unit control block.
PARMLST
PARMCID
SUCB
SUCD
RPL
PARMRPL
RPLSIGNL
RPLRSHUT
1 Set up addressability for this exit routine.
2 Check if the CID of the terminal corresponds with the one of the SNA unit control block.
No
Yes
Check if the list or punch is to be interrupted for inbound data (signal request issued).
No
Yes
Set signal indicator
Return to VTAM
Check if a shutdown request was issued.
No
Yes
Set stop code
Set kill session indicator
Set select byte
Post the SNA work ECB
Get the maintask ECB address in register 1
Issue the POST macro to post POWER/VS
POST
Address of PAEB
SUCB
1
'E'
1
'S'
SUSG
SUTT
SUSS
SUSL
SNCB
1
SNEP
SNEB
R1
Return to VTAM

DFAS(Part 2 of 2)

Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(1) On entry to this routine register 1 contains the address of a 5-word parameter list (PARMLST: ACB address, CID of terminal, SNA unit control block address, Number of bytes received, Address of read-only RPL), register 14 contain the VTAM return address and register 15 contains the address of the DFASY exit routine.) Register 5 is used as an SNA control block pointer Register 8 is used for the address of the SNA unit control block Register 9 is used as base address of the SNA manager Register 10 is used as base for the POWER/VS nucleus (2)		POST(DOS/VS)	



Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(1) On entry to this routine register 1 contains the address of a 2-word parameter list (PARMLST: ACB address, Reason for shutdown (0 = normal halt, 4 = quick halt), register 14 contains the VTAM return address and register 15 contains the address of the TPEND exit routine). Register 5 is used as an SNA control block pointer Register 9 is used as base address of the SNA manager Register 10 is used as base for the POWER/VS nucleus			
(3)		POST(DOS/VS)	



Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(1) On entry to this routine register 1 contains the address of a 4-word parameter list (PARMLST: ACB address, CID of terminal, SNA unit control block address, Reason for logoff (32 = conditional logoff), register 14 contains the VTAM return address and register 15 contains the address of the LOSTERM exit routine. Register 5 is used as an SNA control block pointer Register 8 is used for the address of the SNA unit control block Register 9 is used as base address of the SNA manager Register 10 is used as base for the POWER/VS nucleus (3)		POST(DOS/VS)	

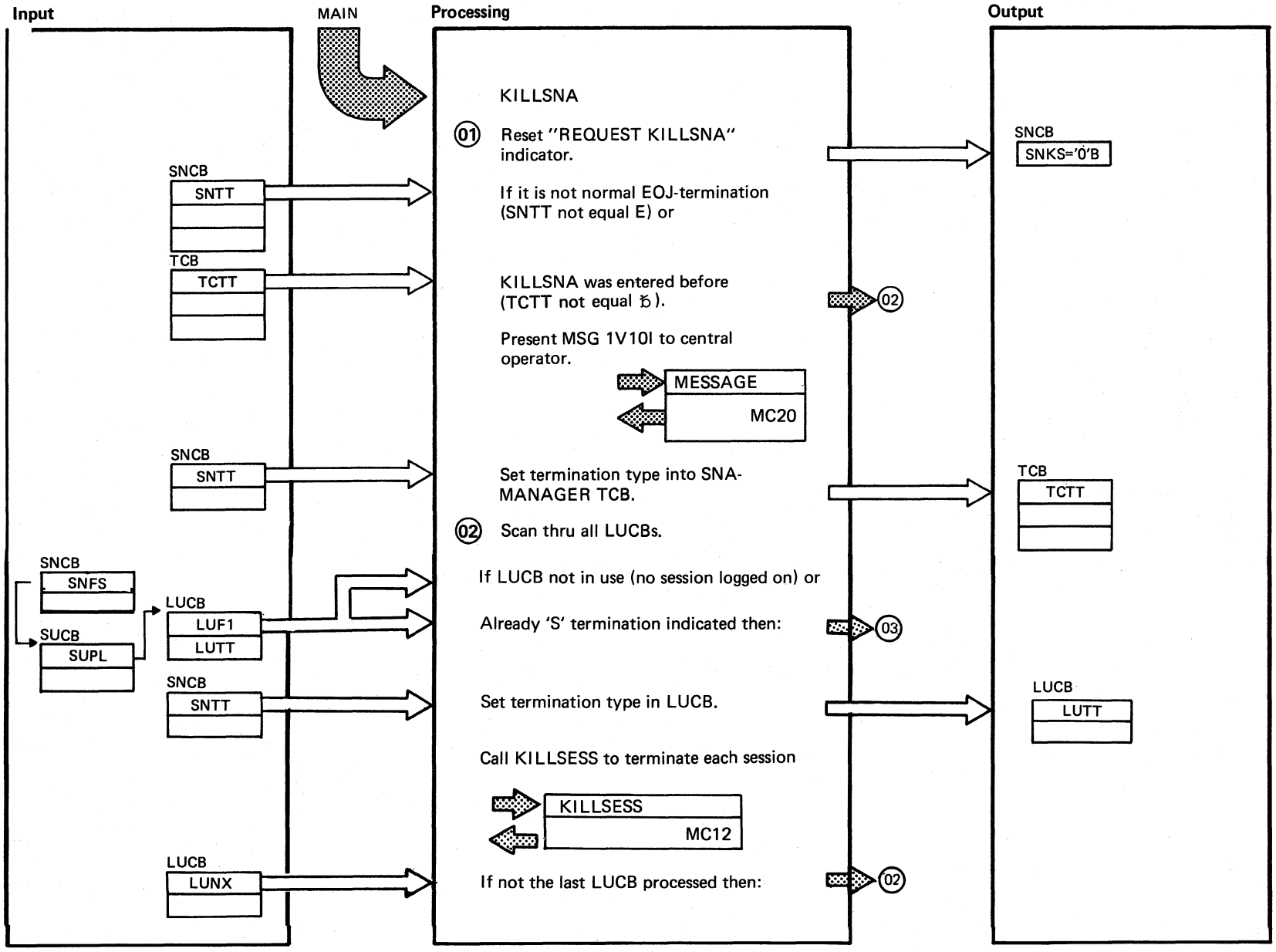
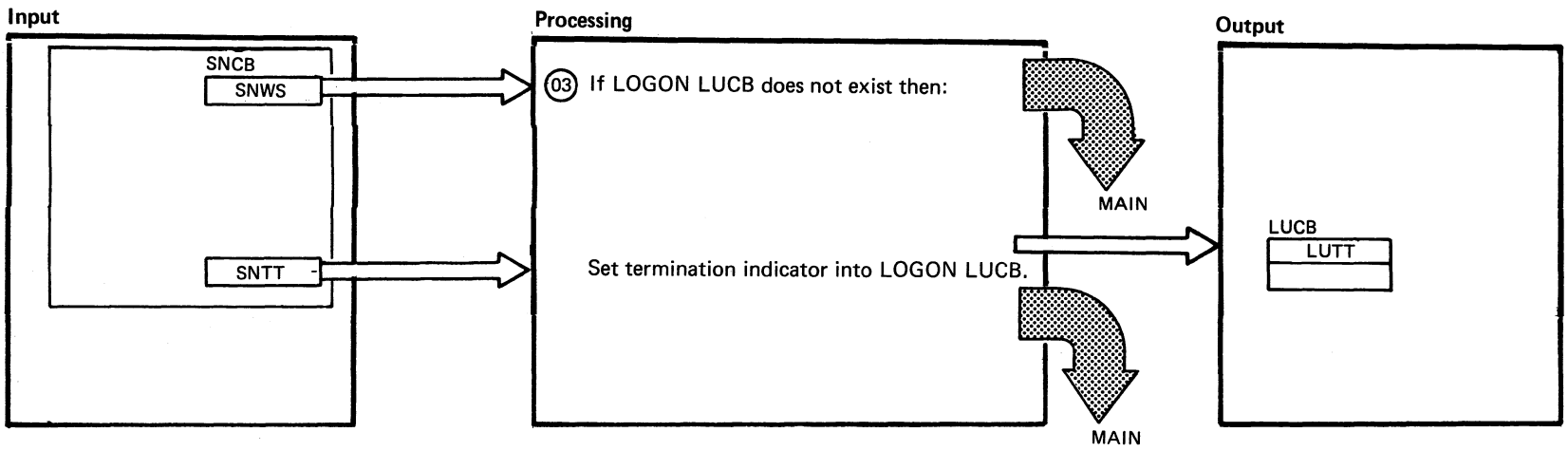
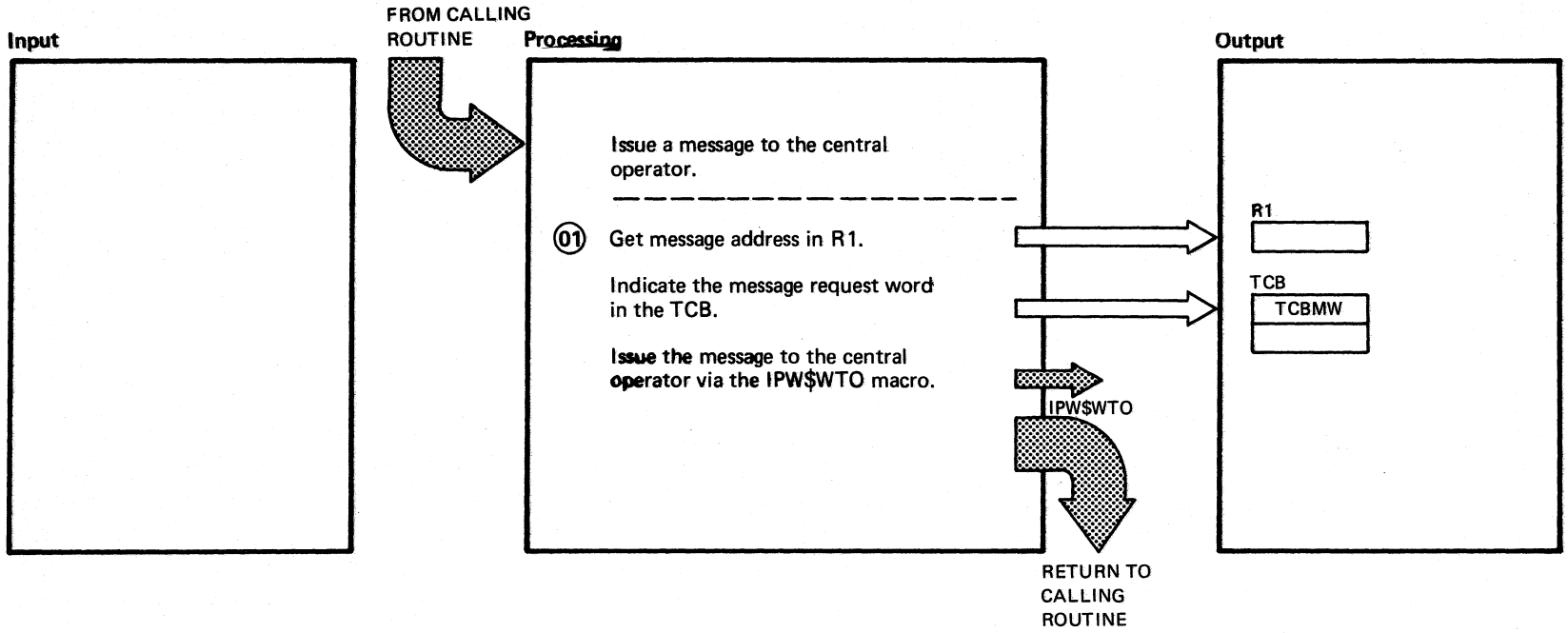


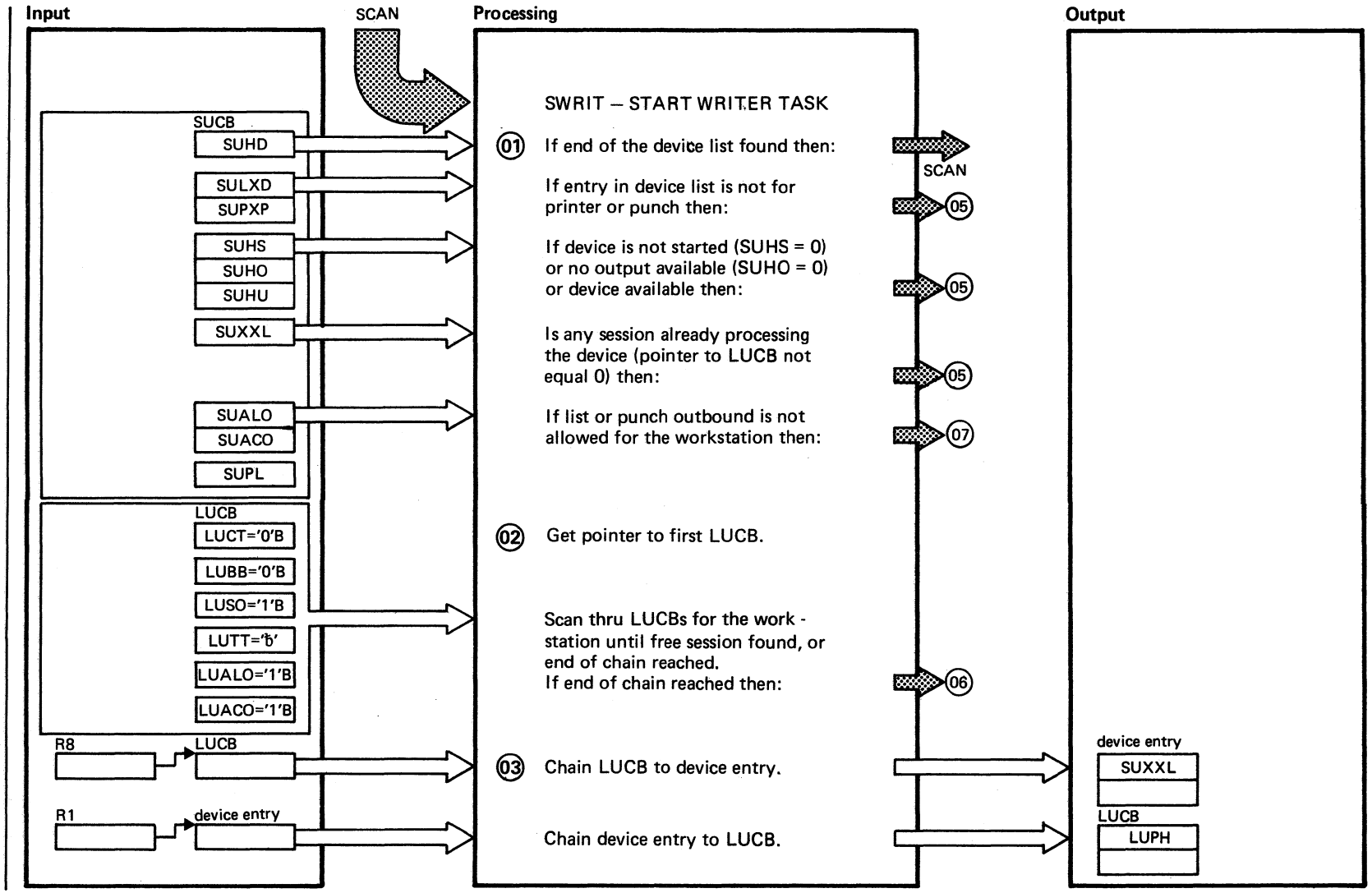
Chart MC18
Part 1 of 2

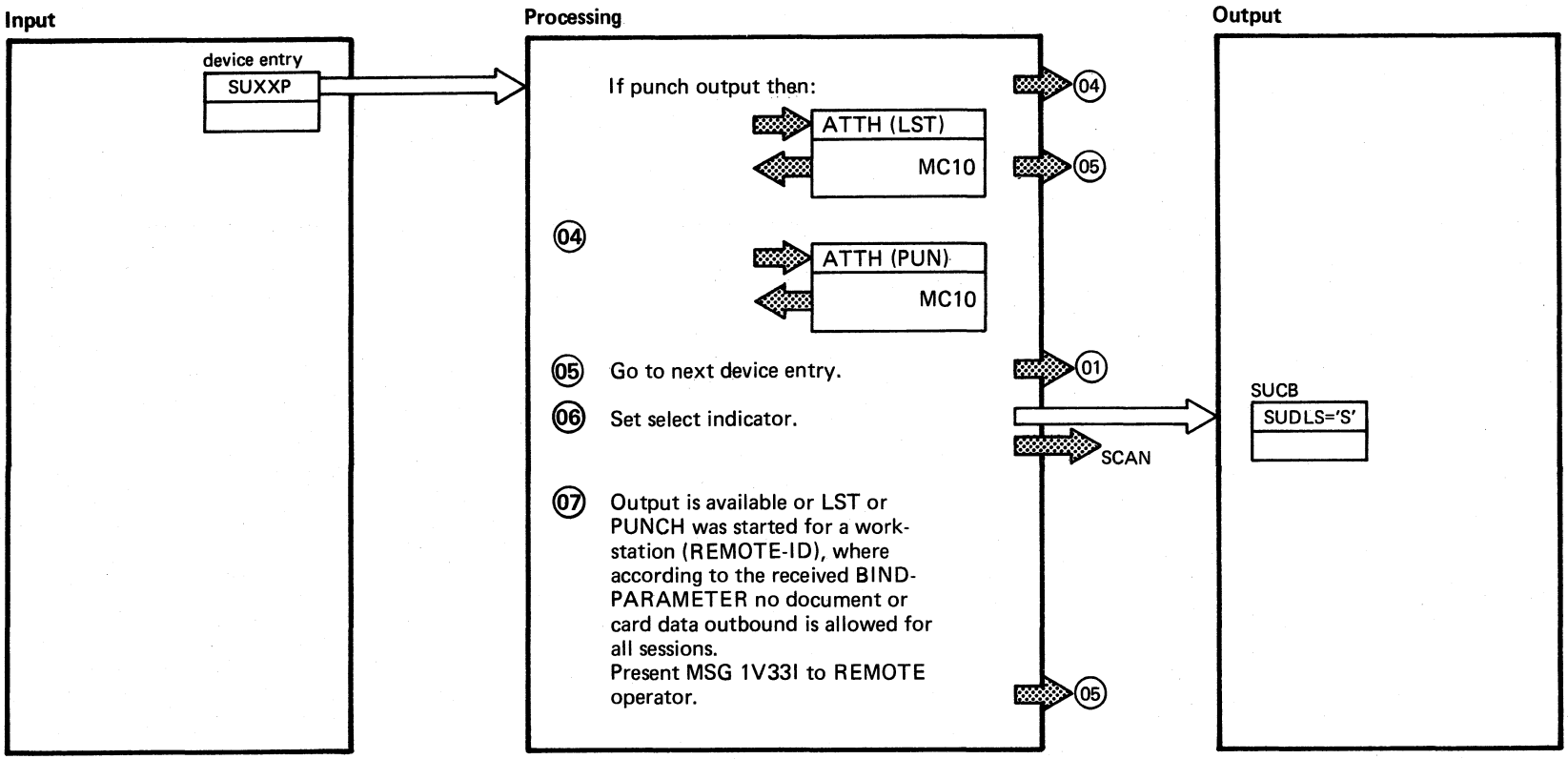


Extended Description	Include Segment	Call Subroutine/ Macro	Chart
01 1V10I RJE,SNA is in shutdown period.			



Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(01) Upon entry R1 is assumed to contain the message index, which is used to obtain the message address in the message, definition module (IPW$$MD). If the message is to be modified, register 4 contains the binary value, and register 6 the RPL address. The IPW$WTO macro uses registers 0,1,2 and 3.		IPW$WTO POWER/VS	


Input
SCAN
Processing
Output
SWRIT – START WRITER TASK
SUCB
SUHD
SULXD
SUPXP
SUHS
SUHO
SUHU
SUXXL
SUALO
SUACO
SUPL
LUCB
LUCT='0'B
LUBB='0'B
LUSO='1'B
LUTT='b'
LUALO='1'B
LUACO='1'B
R8
LUCB
R1
device entry
01 If end of the device list found then:
SCAN
If entry in device list is not for printer or punch then:
05
If device is not started (SUHS = 0) or no output available (SUHO = 0) or device available then:
05
Is any session already processing the device (pointer to LUCB not equal 0) then:
05
If list or punch outbound is not allowed for the workstation then:
07
02 Get pointer to first LUCB.
Scan thru LUCBs for the work - station until free session found, or end of chain reached.
If end of chain reached then:
06
03 Chain LUCB to device entry.
Chain device entry to LUCB.
device entry
SUXXL
LUCB
LUPH



Extended Description

Include Segment	Call Subroutine/ Macro	Chart

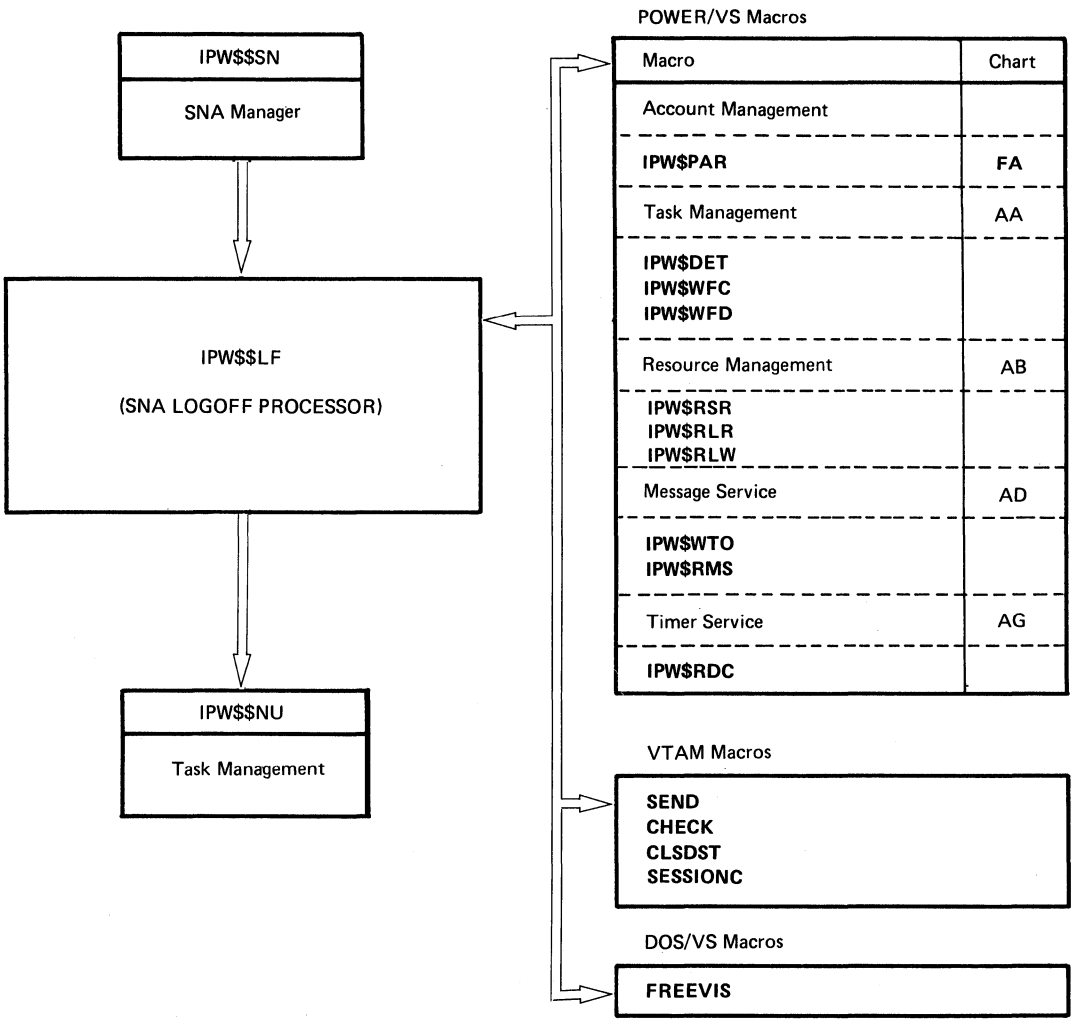
(01) All devices are checked one after each other, in case action is indicated in several devices before SNA MANAGER gets control.

(06) No free session is found to print or punch. The output available remains on, so when the SNA-MANAGER gets control again, it will scan for a free session again. Next scan will take place when e.g. a processor detaches and posts SNA-MANAGER.

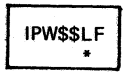
(07) 1V33I REMOTE =xxx output for nonwriter work station.

CHART ME: IPW$$LF - SNA LOGOFF PROCESSOR (9 PARTS)

Chart ME00: IPW$$LF - SNA Logoff Processor, General Flow and Macro Calls



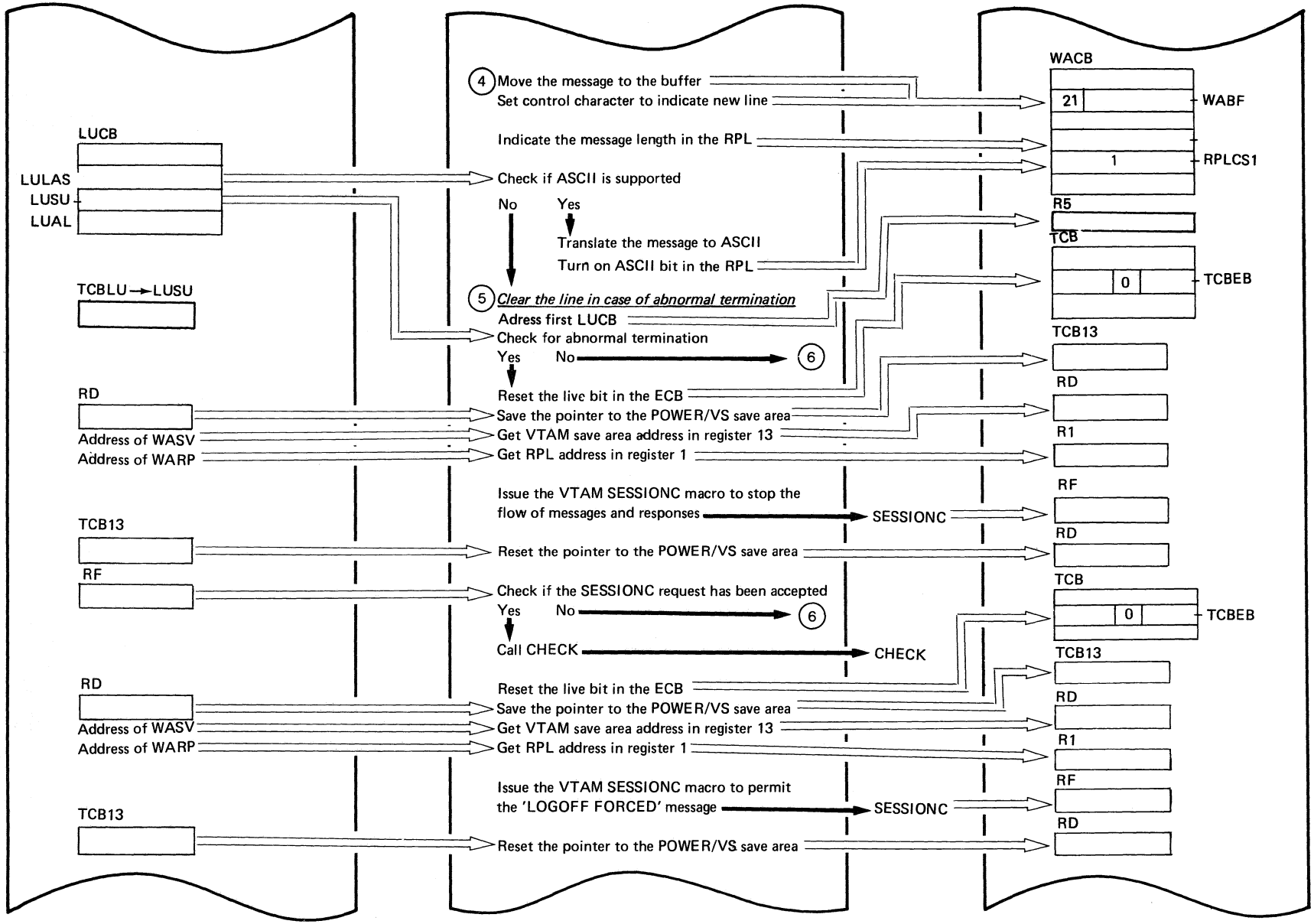
SNA LOGOFF PROCESSOR (IPW$$LF) ORGANIZATION



* The SNA LOGOFF processor consists of only 1 segment, which contains the following functions:

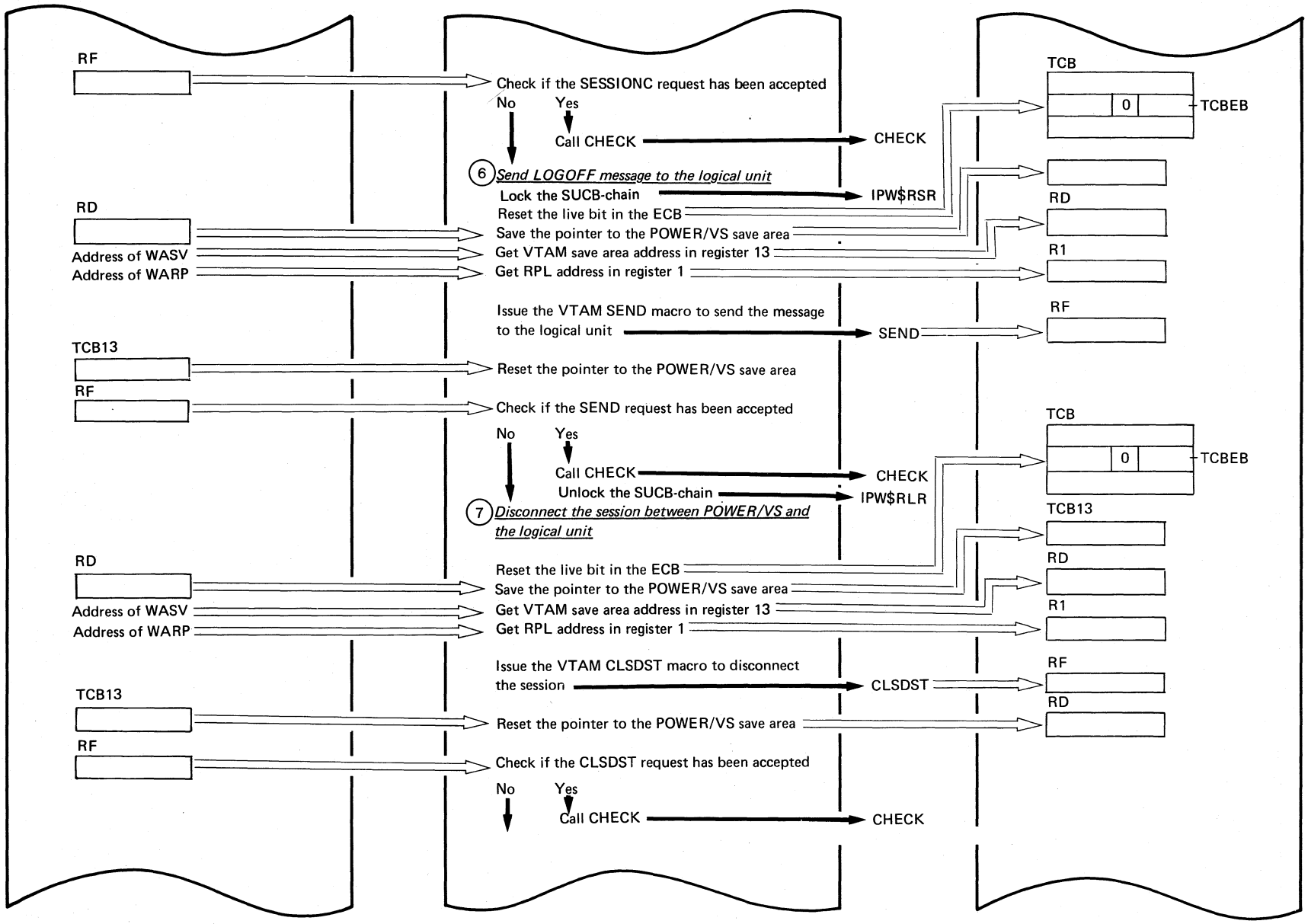
- Initialize work area, RPL, and message buffer
- Clear the line in case of an abnormal termination
- Send a LOGOFF message to the logical unit
- Disconnect the session between POWER/VS and the logical unit
- Write an account record
- Send a LOGOFF message to the central operator
- Release the space of the SNA unit control block and work area
- Subroutine to check the RPL

IPW$$LF (Part 2 of 7)



Continued on next page

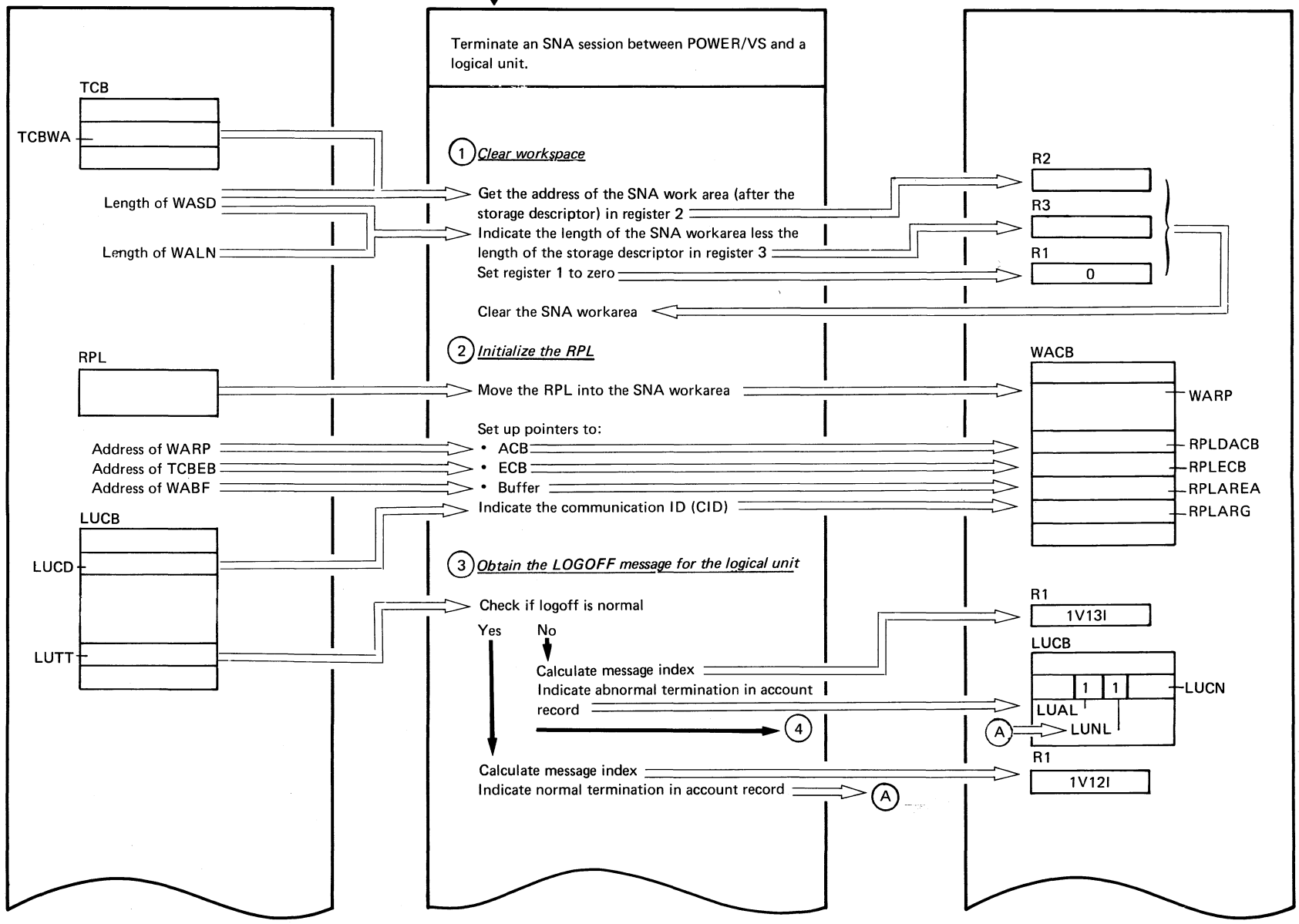
IPW$$LF (Part 3 of 7)



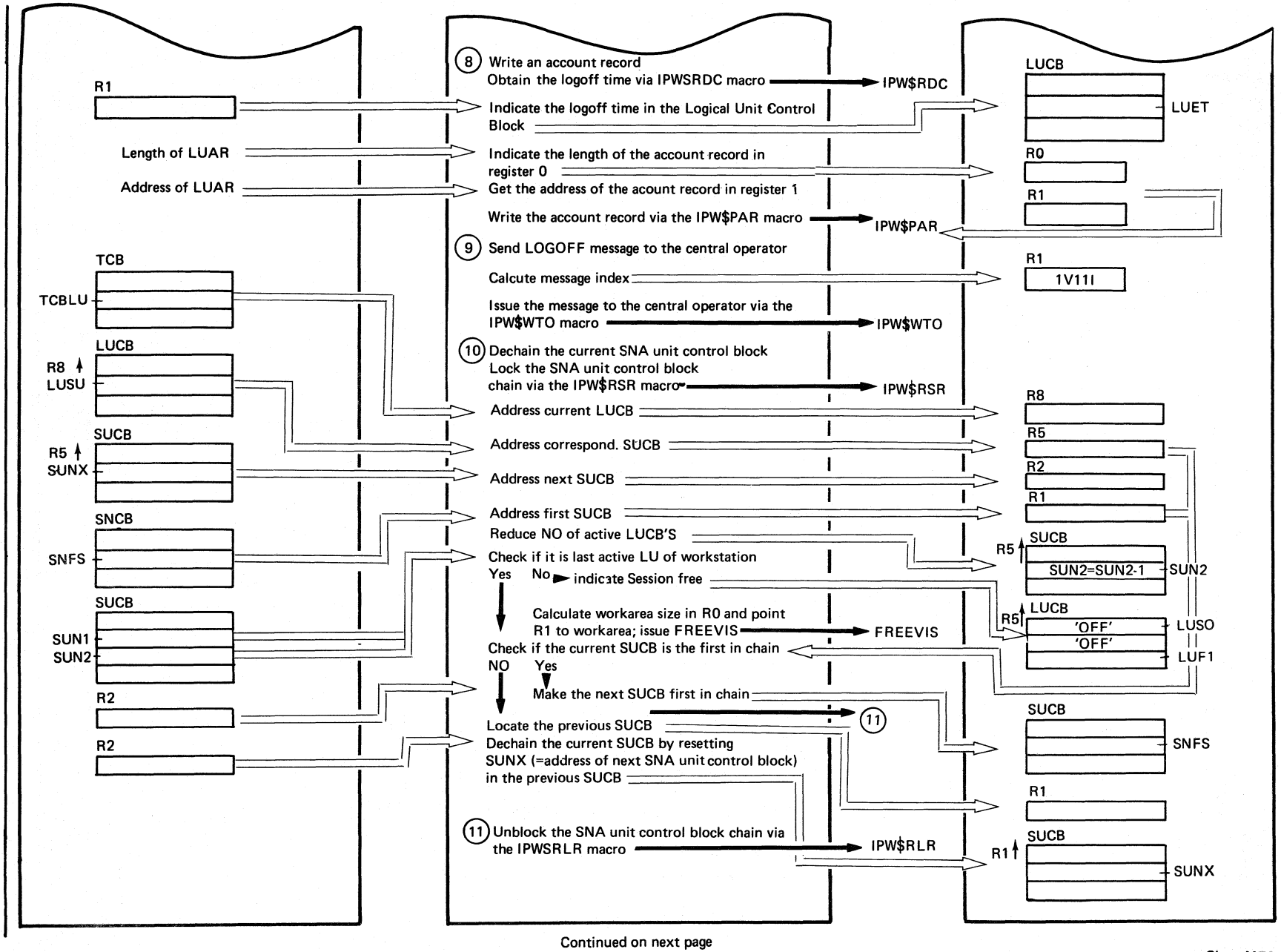
Continued on next page

Attached by IPW$$SN

IPW$$LF (Part 1 of 7)

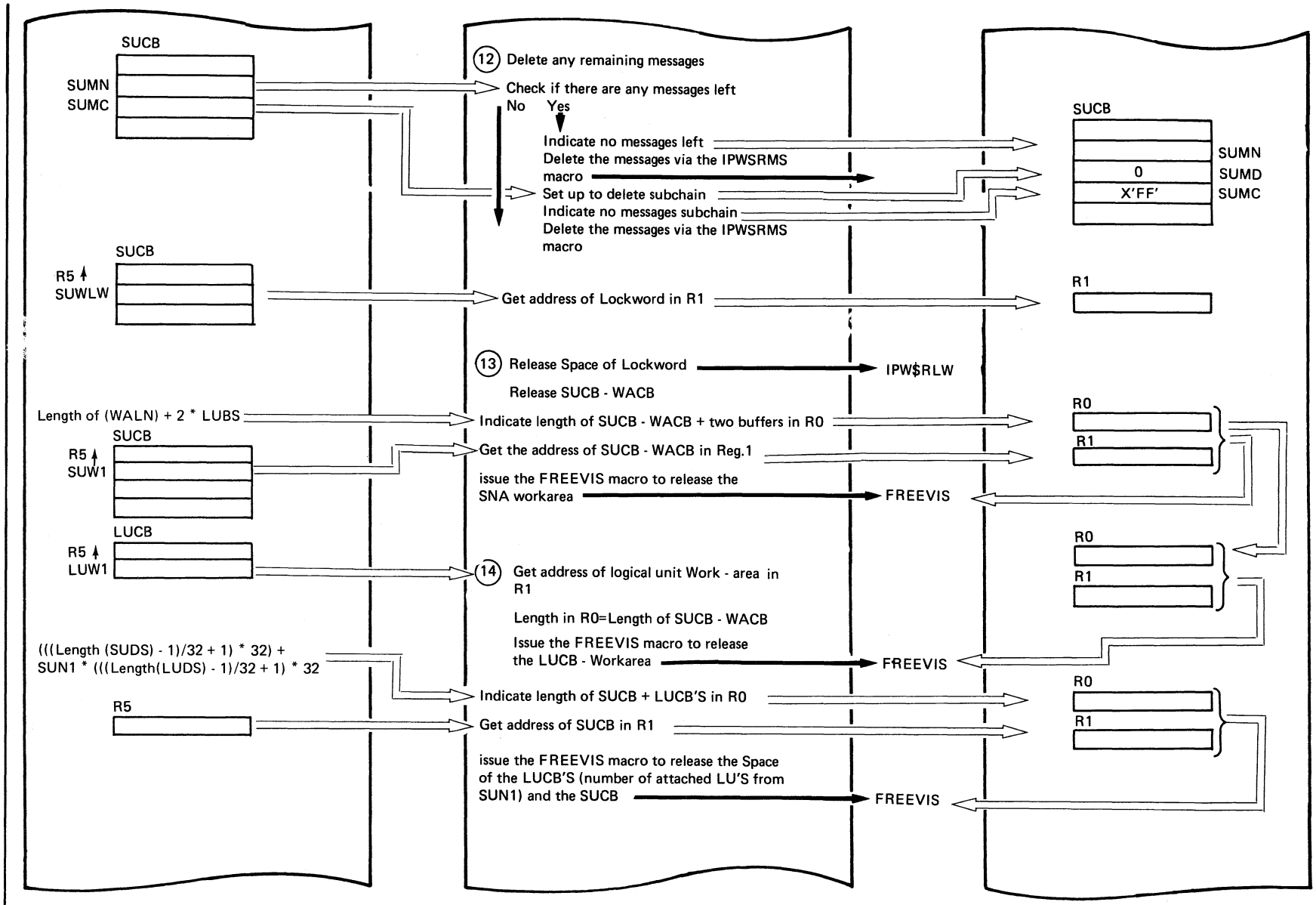


Continued on next page



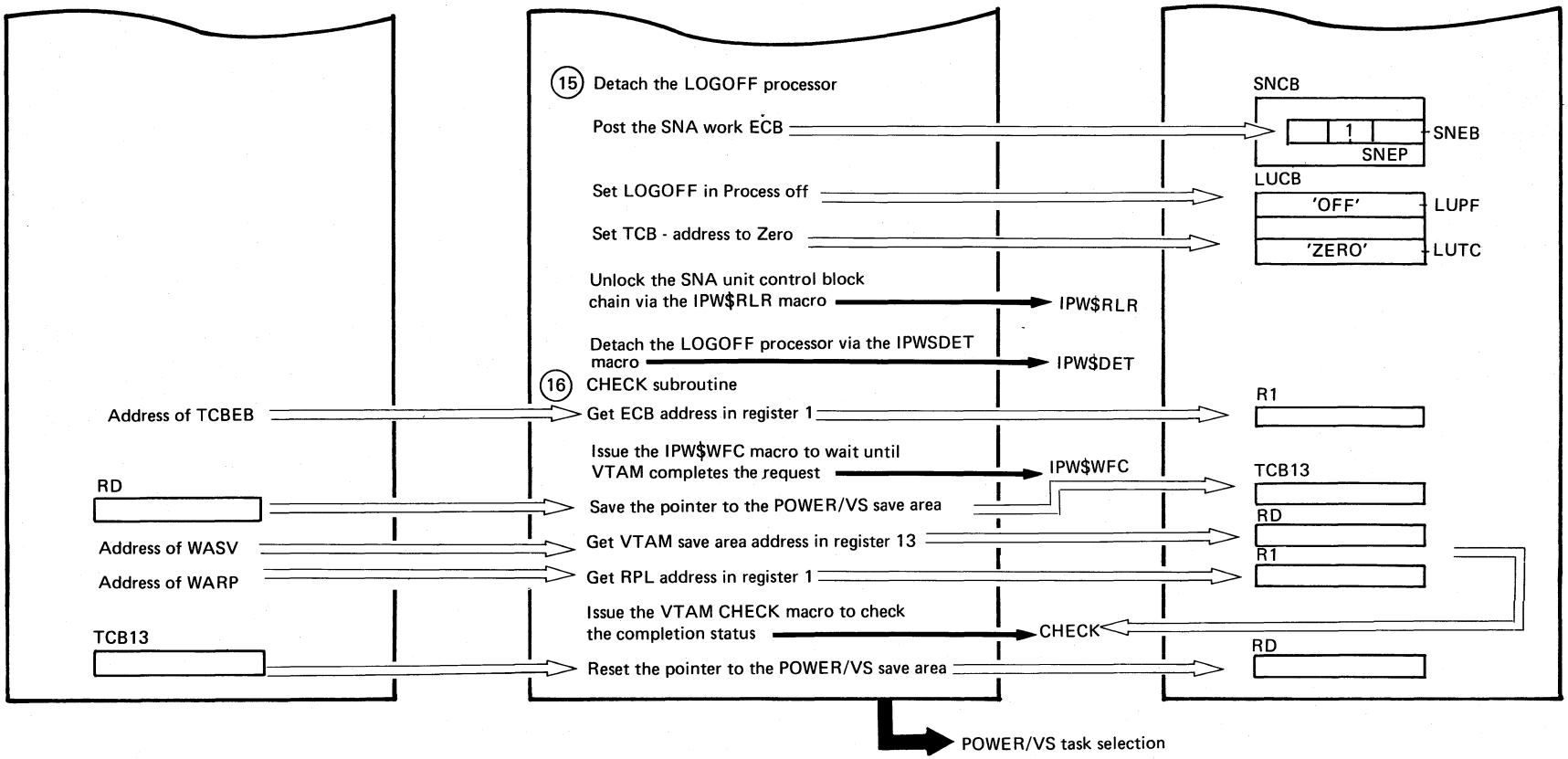
Continued on next page

IPW$$LF (Part 5 of 7)



Continued on next page

IPW$$LF (Part 6 of 7)

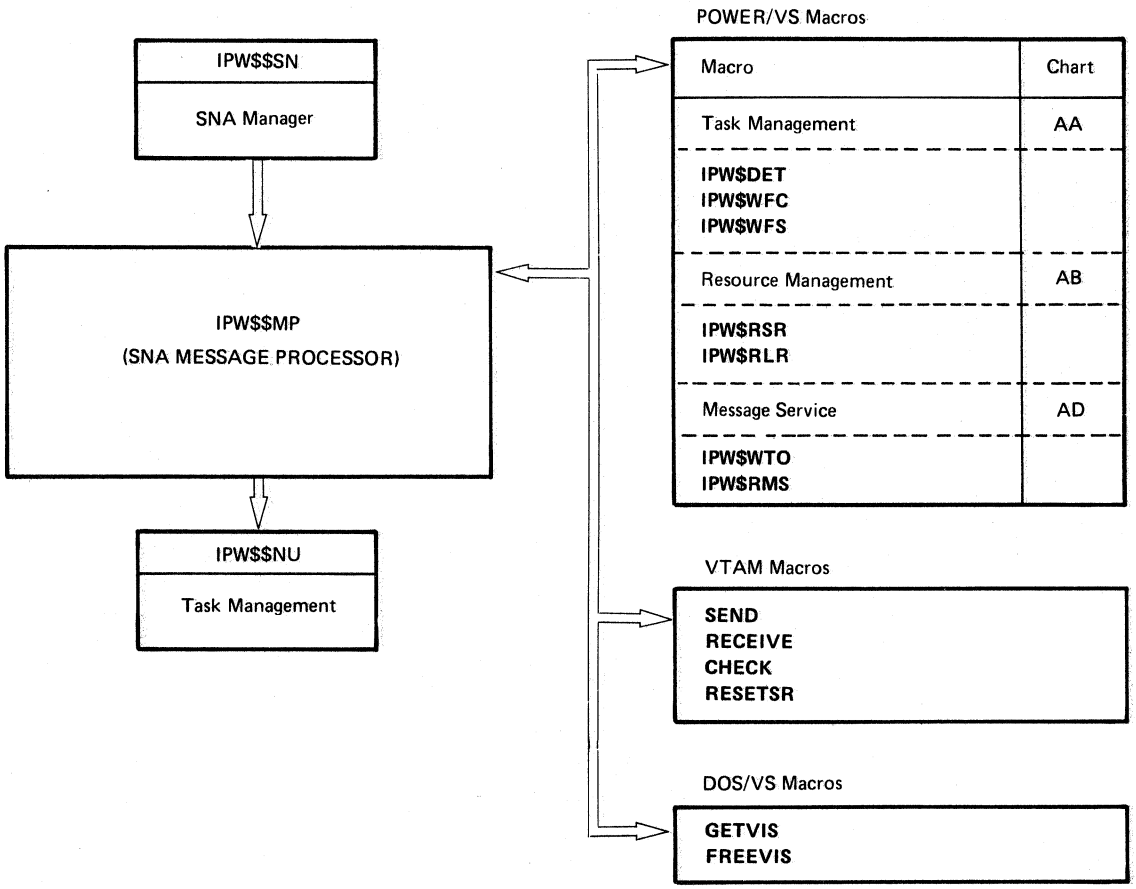


IPW$$LF (Part 7 of 7)

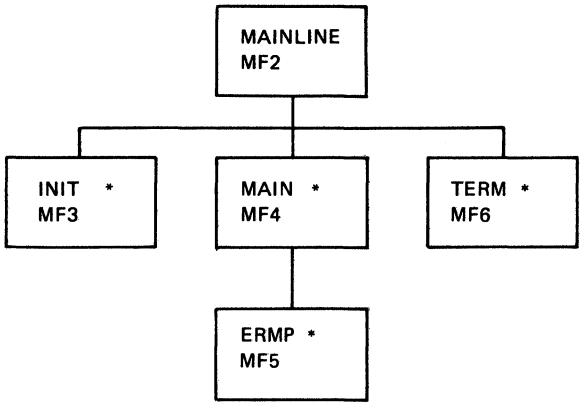
Extended Description	Include Segment	Call Subroutine/Macro	Chart
(3) 1V13I LOGOFF FORCED 1V12I LOGOFF COMPLETED			
(5)		SESSIONC(VTAM) CHECK	ME2
(6)		SEND(VTAM) CHECK	ME2
(7)		IPWRSR, IPWRLR CLSDST(VTAM)	
(8)		CHECK IPW$RDC(POWER/VS) IPW$PAR(POWER/VS)	ME2
(9) 1V11I REMOTE rrr LOGGED OFF FROM POWER ON luname The IPW$WTO macro uses registers 0, 1, 2, and 3		IPW$WTO(POWER/VS)	
(10) (11) The IPW$RSR and IPW$RLR macros use registers 2 and 3		IPW$RSR(POWER/VS) FREEVIS(DOS/VS) IPW$RLR(POWER/VS)	
(12)		IPW$RMS(POWER/VS)	
(13) (14) The FREEVIS macro uses registers 0, 1, and 15		IPW$RLW FREEVIS(DOS/VS)	
(15)		IPW$DET(POWER/VS) IPW$RLR	
(16)		IPW$WFC(POWER/VS) CHECK(VTAM)	

CHART MF: IPW$$MP - SNA MESSAGE PROCESSOR (18 PARTS)

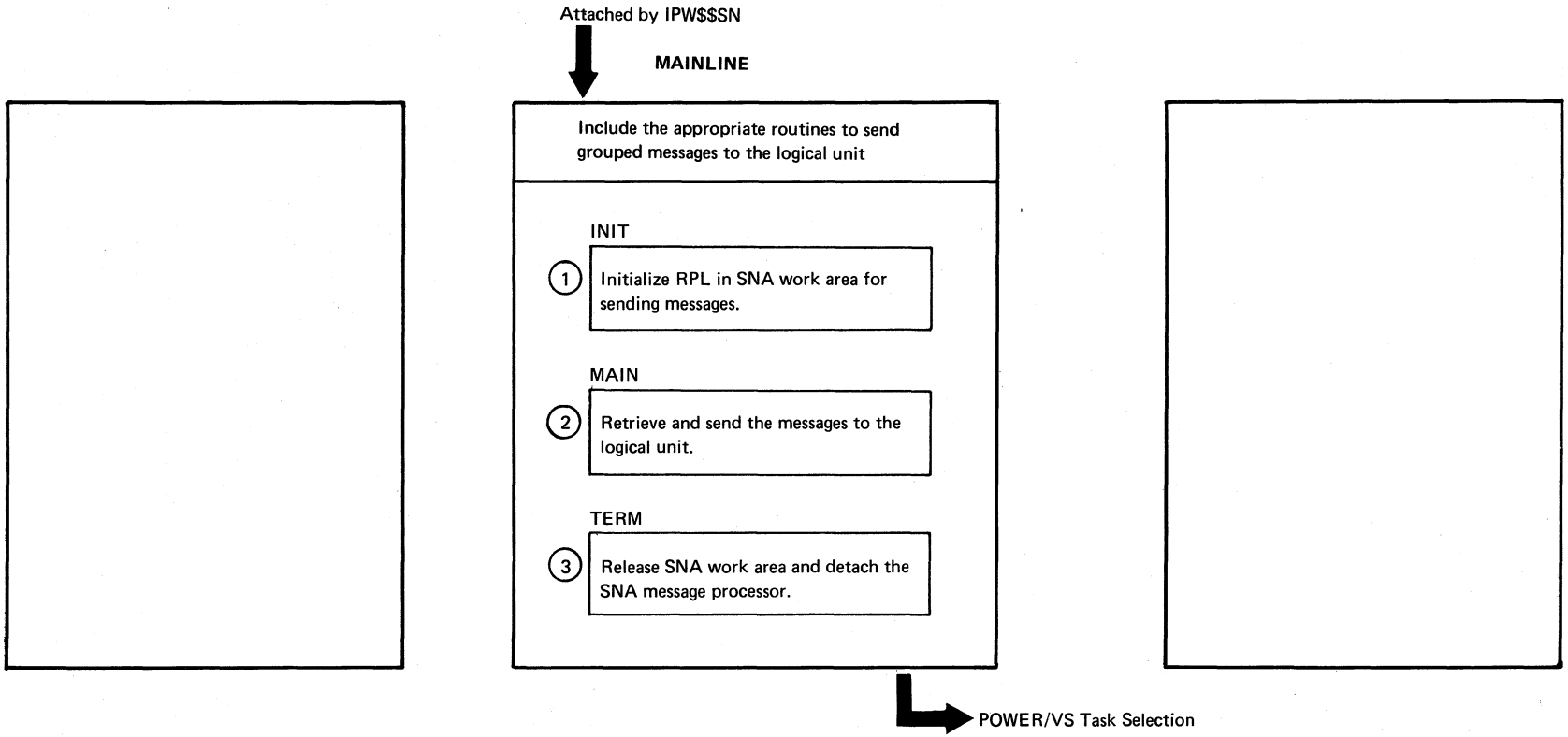
Chart MF00: IPW$$MP - SNA Message Processor, General Flow and Macro Calls



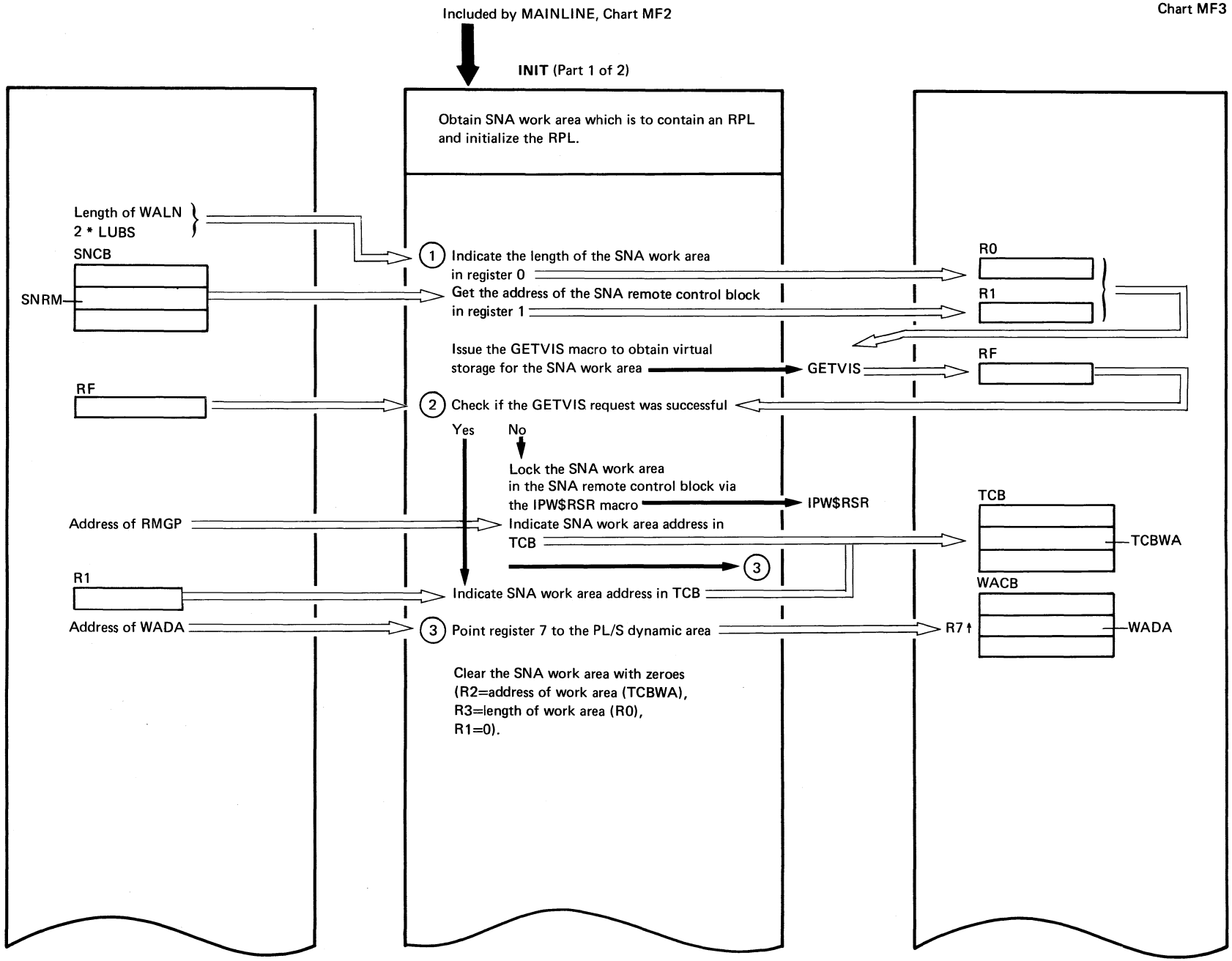
SNA MESSAGE PROCESSOR (IPW$$MP), ORGANIZATION



* Not a Subroutine. Code included via PLS INCLUDE statement in the next higher level.

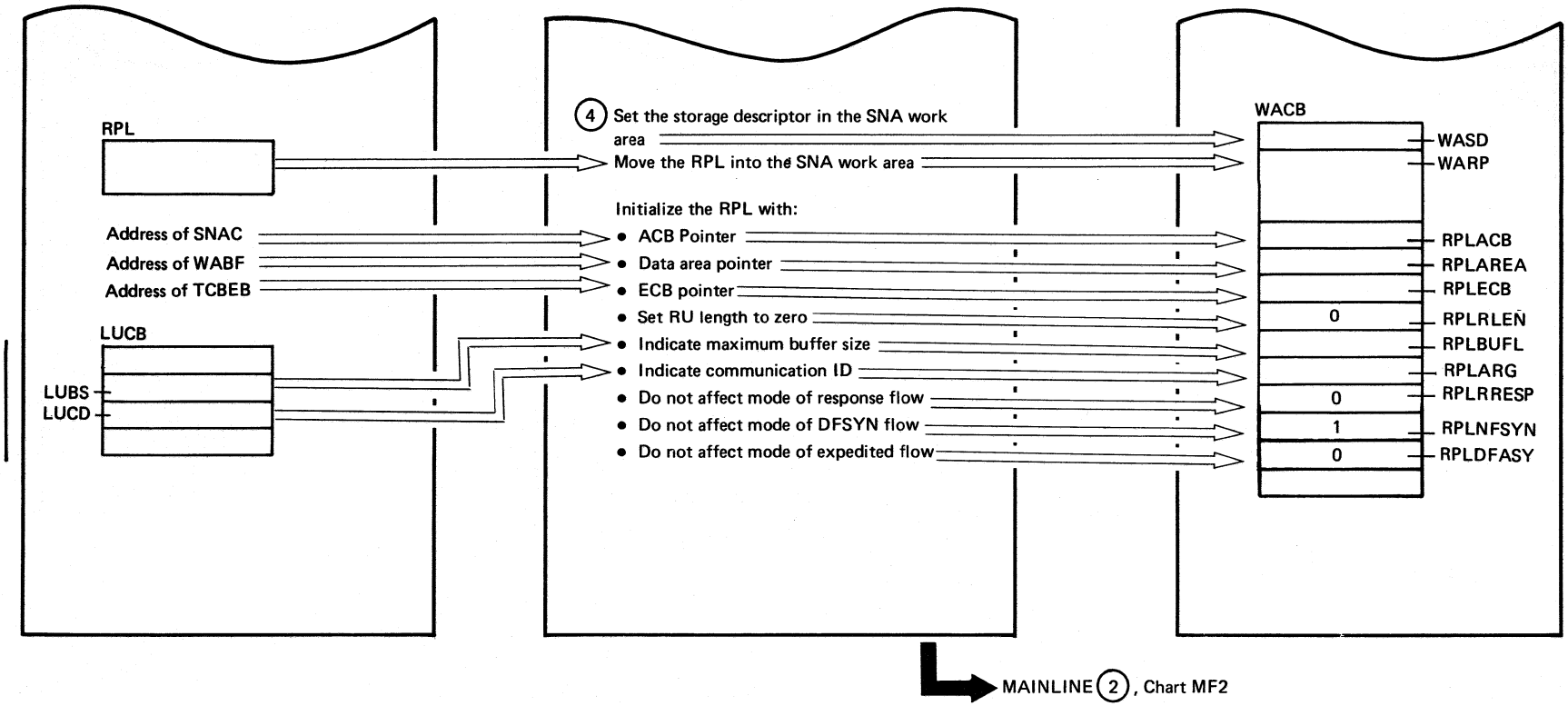


Extended Description	Include Segment	Call Subroutine	Chart
(1)	INIT		MF3
(2)	MAIN		MF4
(3)	TERM		MF6



Continued on next page

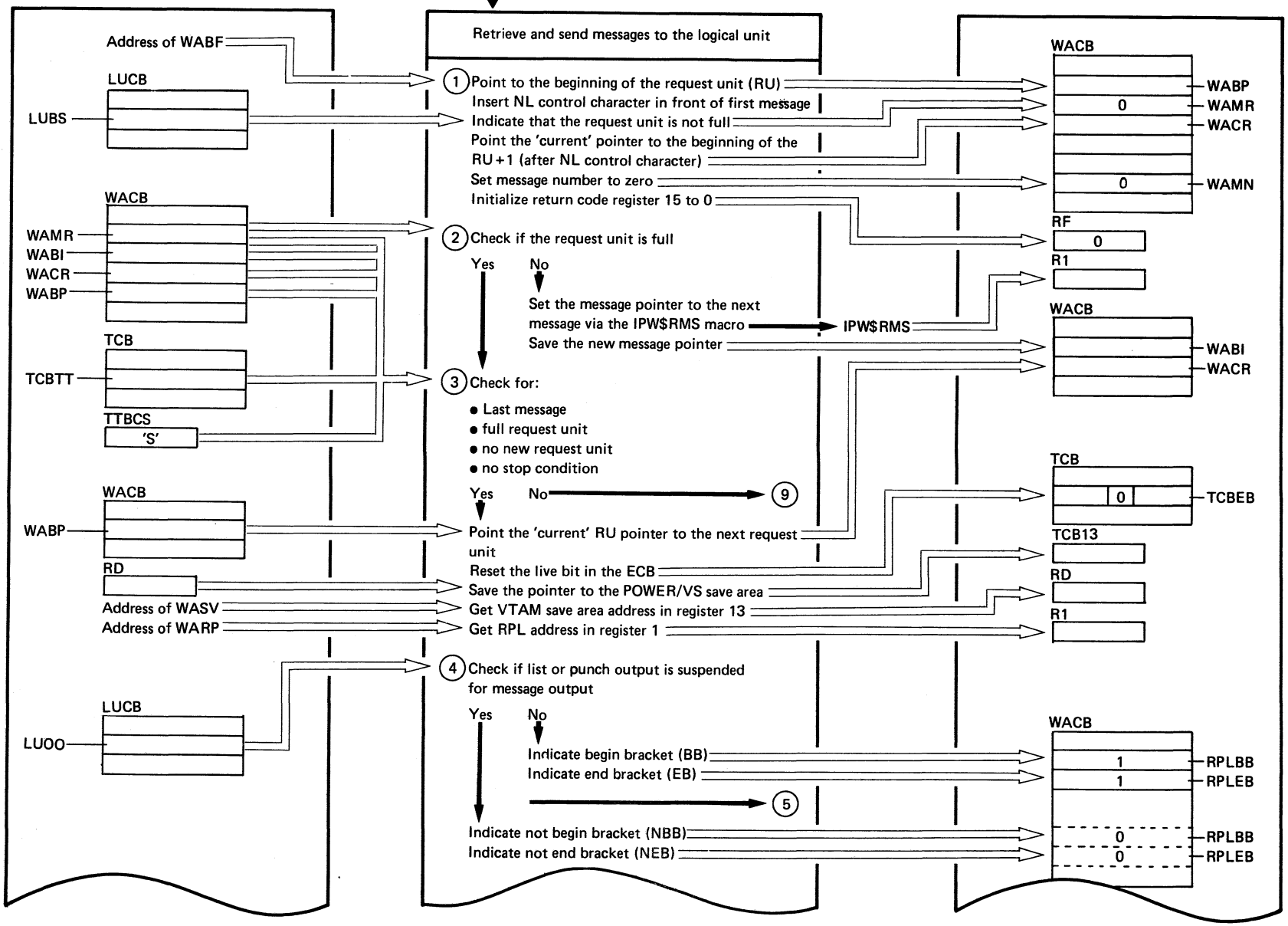
INIT (Part 2 of 2)



Extended Description	Include Segment	Call Subroutine/Macro	Chart
(1) The GETVIS macro uses registers 0 and 1.		GETVIS(DOS/VS)	
(2) The IPW$RSR macro uses registers 2 and 3.		IPW$RSR(POWER/VS)	

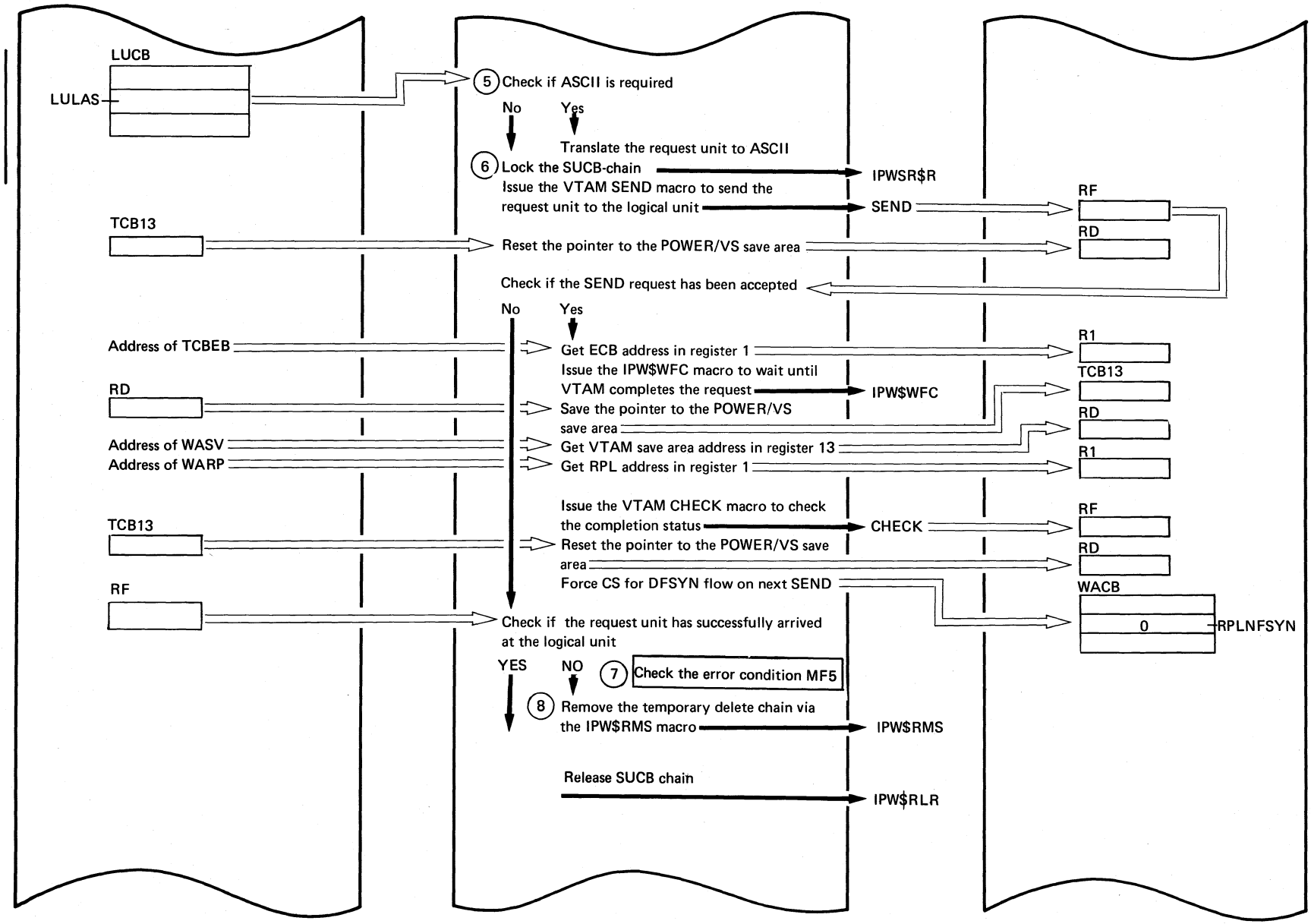
Included by MAINLINE, Chart MF2

MAIN (Part 1 of 4)



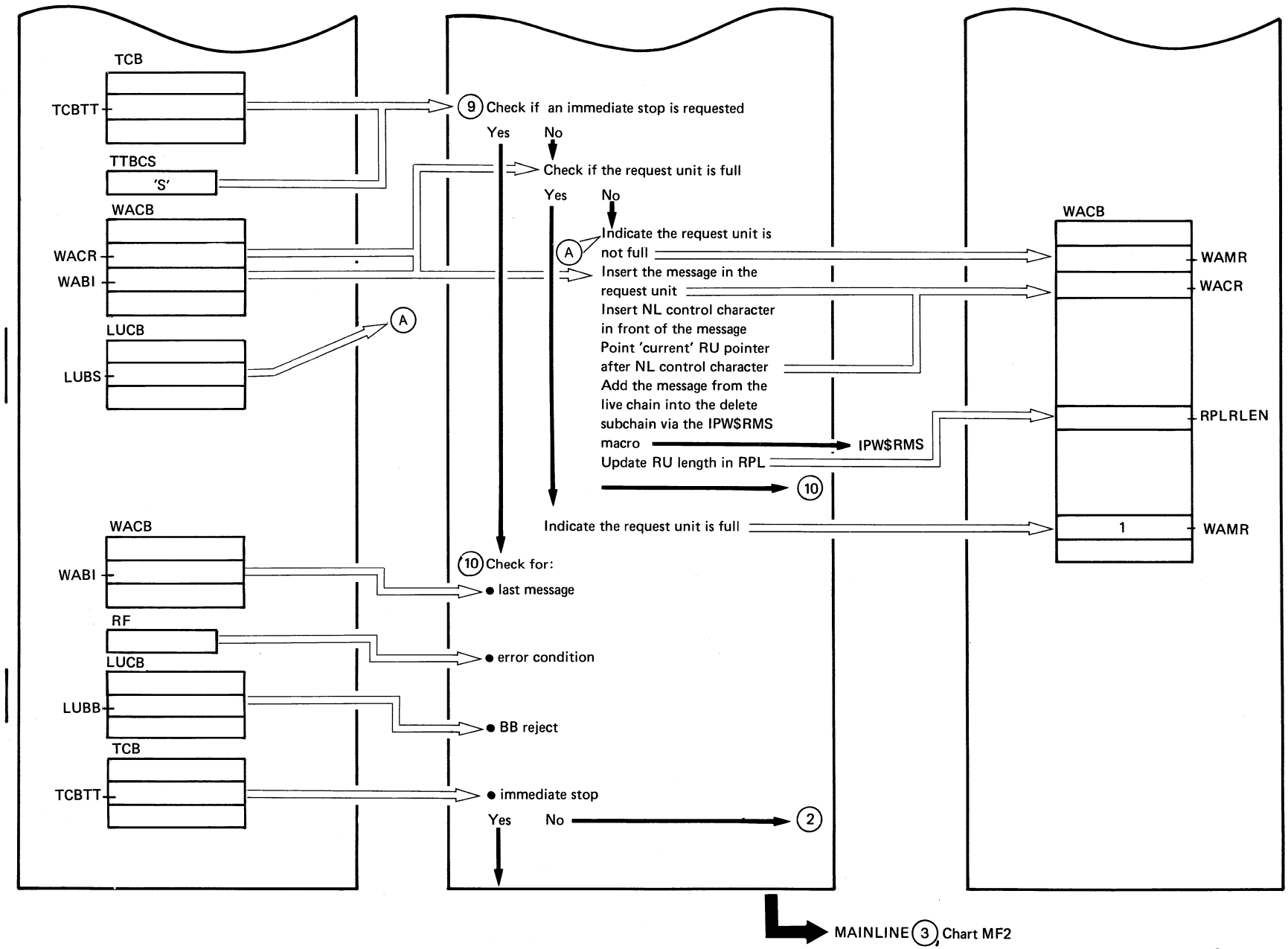
Continued on next page

MAIN (Part 2 of 4)



Continued on next page

MAIN (Part 3 of 4)

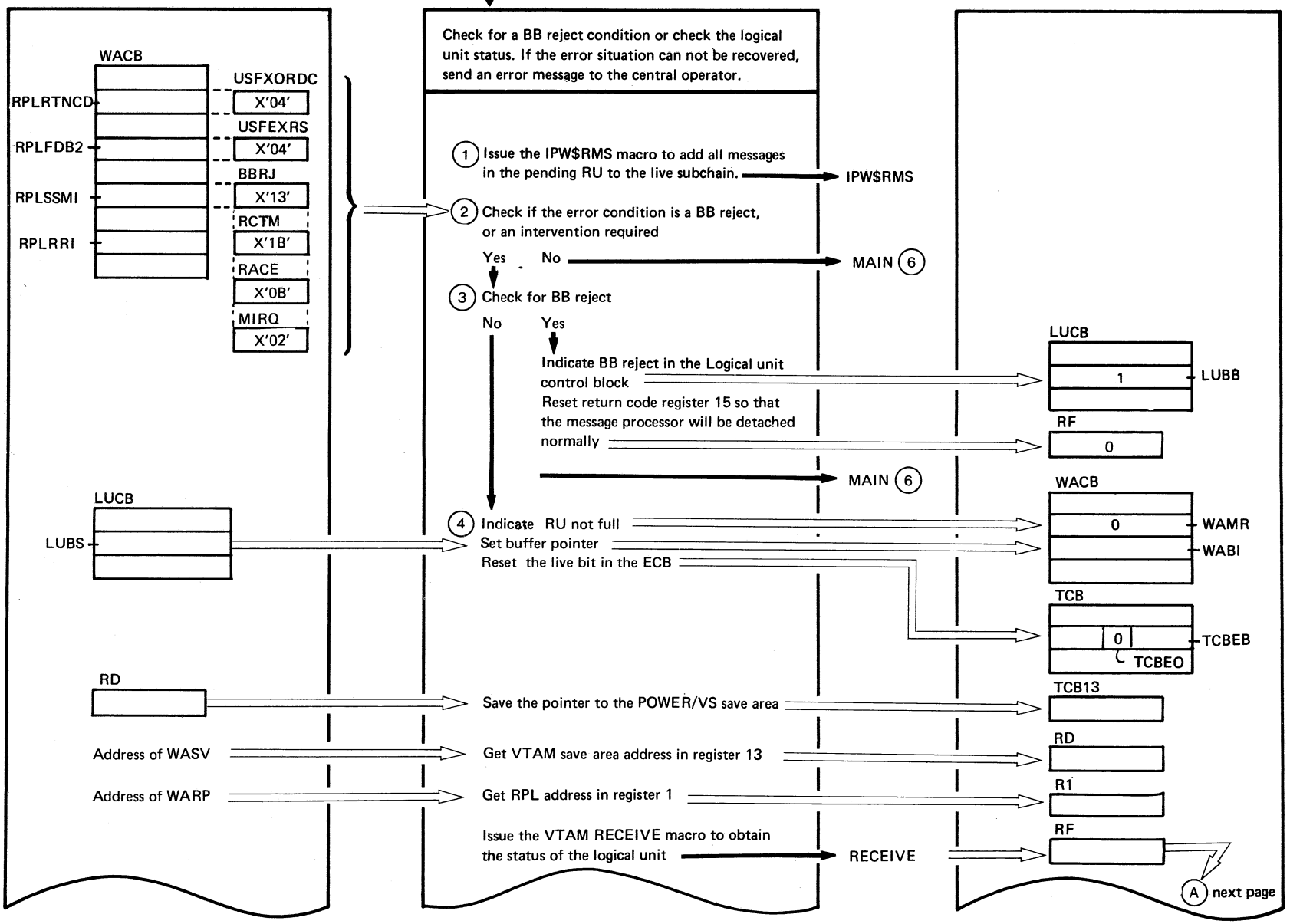


MAIN (Part 4 of 4)

Extended Description	Include Segment	Call Subroutine/Macro	Chart
(1) WABP always points to the beginning of the request unit (RU) WABI always points to the last message retrieved WAMR indicates whether the request unit is full or not (0=not full) WACR indicates the current position in the request unit			
(2) The IPW$RMS macro uses register 0, 1, 2, and 3		IPW$RMS(POWER/VS)	
(6)		IPW$RSR (POWER/VS) SEND(VTAM) IPW$WFC(POWER/VS) CHECK(VTAM) IPW$RMS(POWER/VS)	
(7)	ERMP		MF5
(8)		IPW$RMS(POWER/VS) IPW$RLR (POWER/VS)	

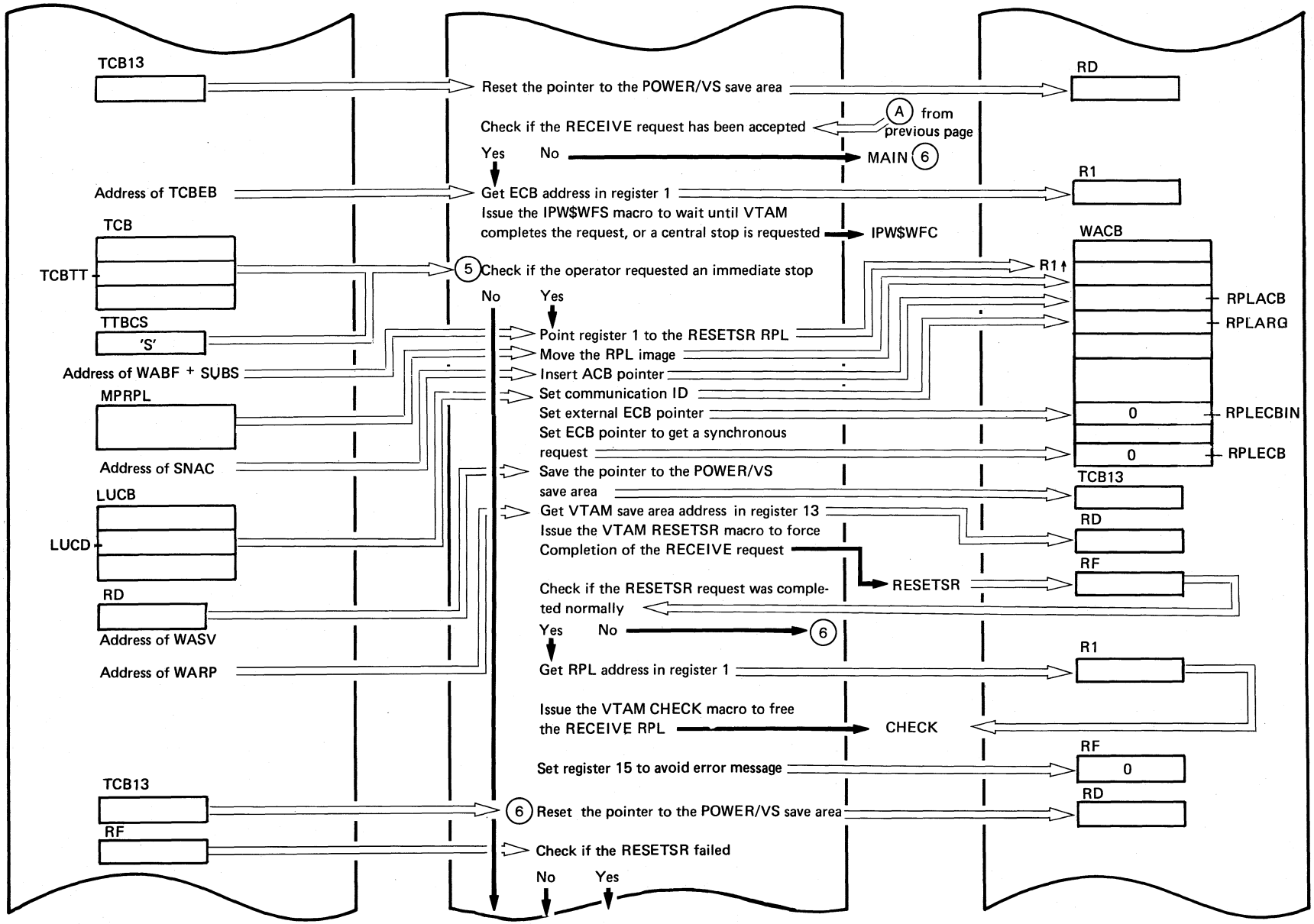
Included by MAIN, Chart MF4

ERMP (Part 1 of 5)



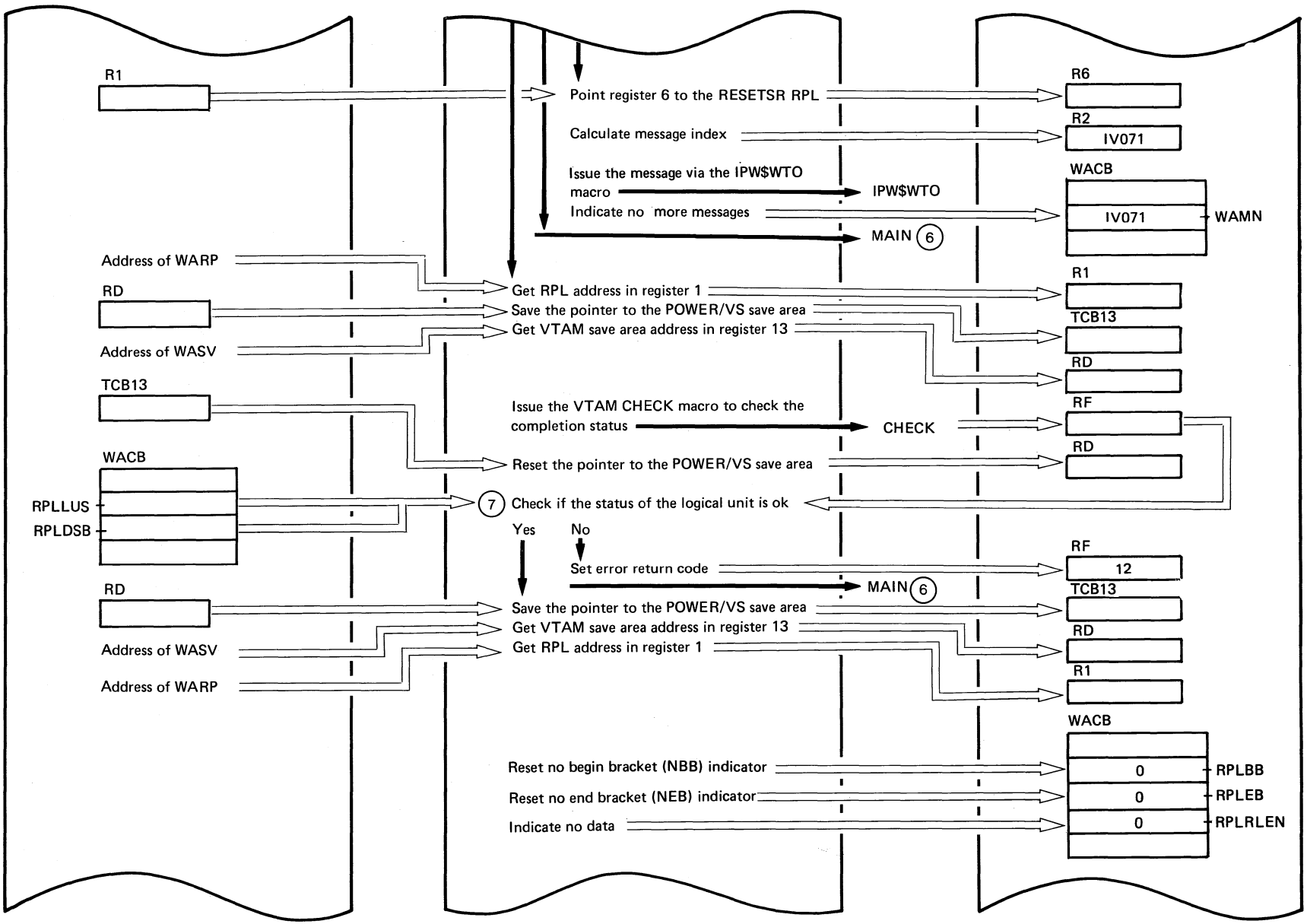
Continued on next page

ERMP (Part 2 of 5)



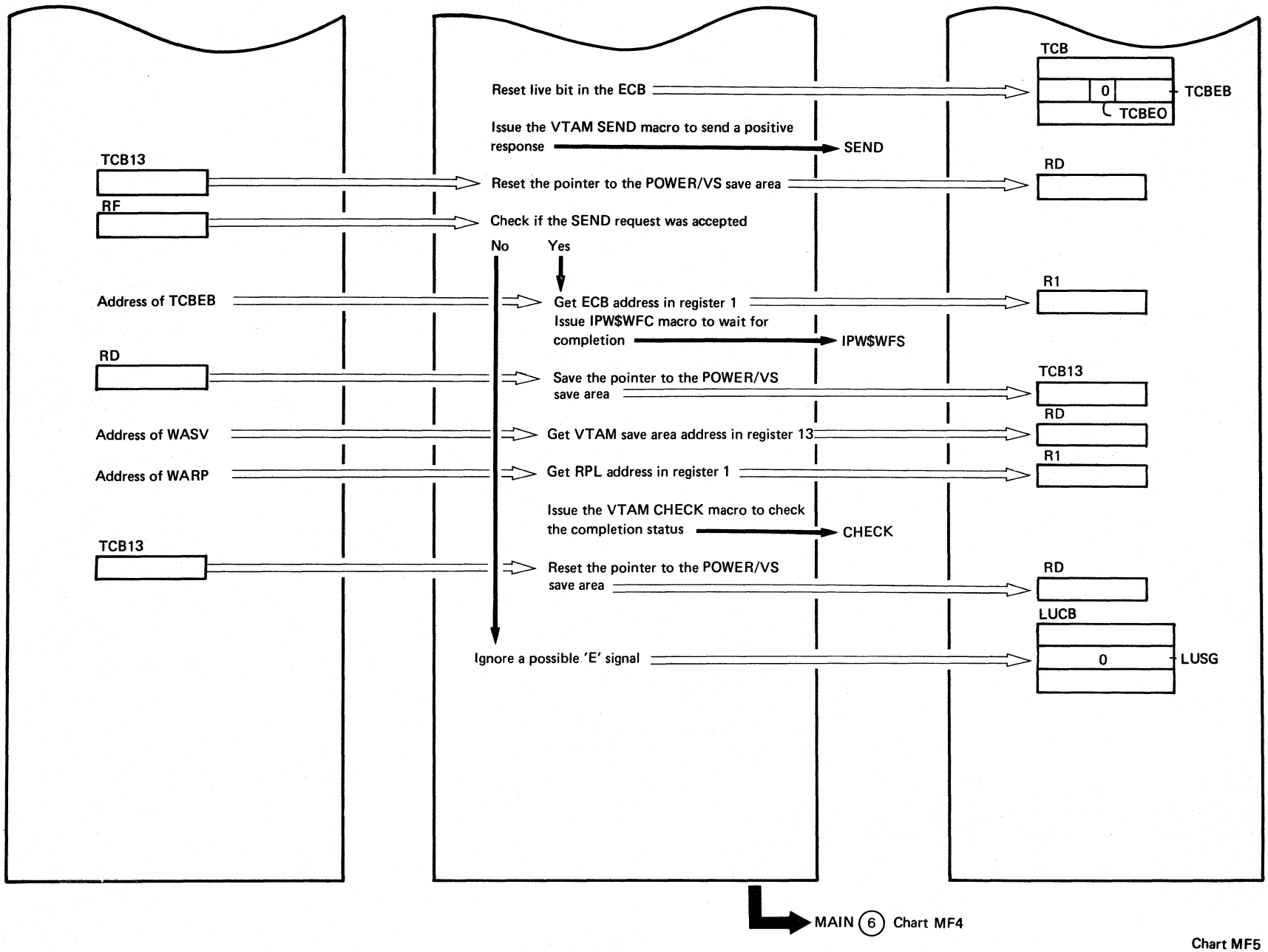
Continued on next page

ERMP (Part 3 of 5)



Continued on next page

ERMP (Part 4 of 5)

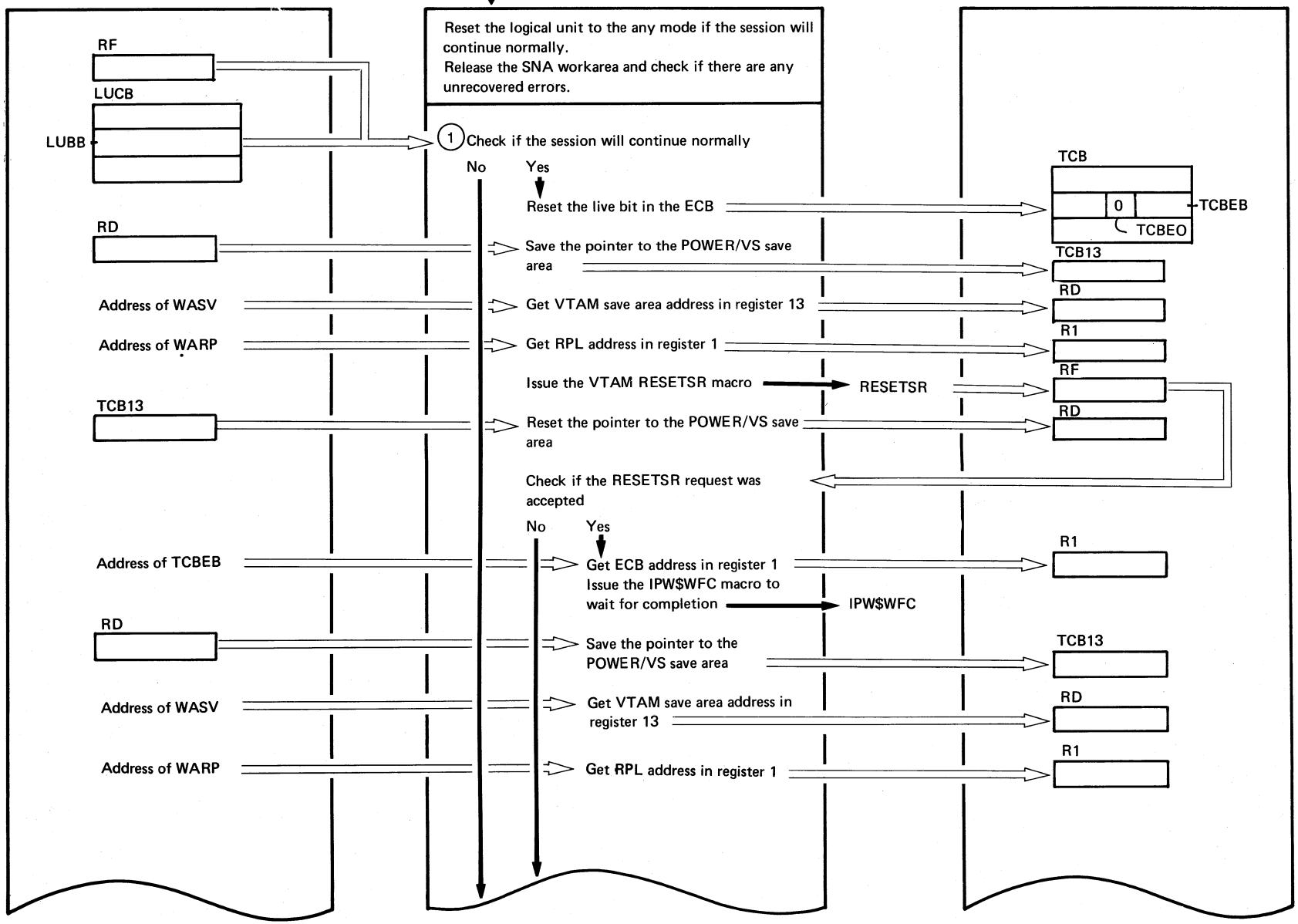


ERMP (Part 5 of 5)

EXtended Description	Include Segment	Call Subroutine/Macro	Chart
(1)		IPW$RMS (POWER/VS)	
(4)		RECEIVE (VTAM) IPW$WFS (POWER/VS)	
(5)		RESETSR (VTAM) CHECK (VTAM)	
(6) 1V07I ERROR ON request RTNCD,FDBK2=xx,yy SENSE=xxxx ON luname		IPW$WTO (POWER/VS)	
(7)		SEND (VTAM) IPW$WFC (POWER/VS) CHECK (VTAM)	

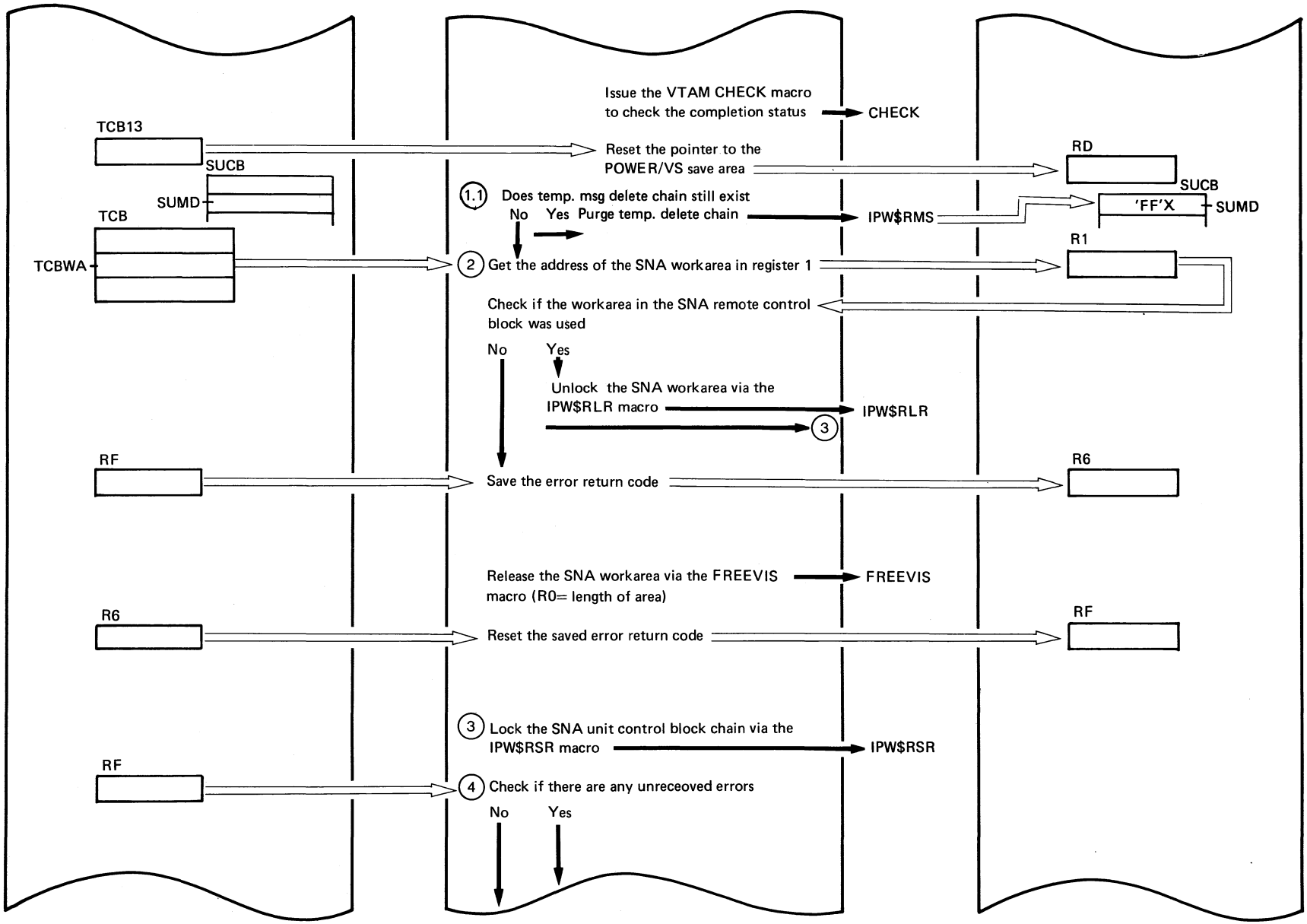
Included by MAINLINE, Chart MF2

TERM (Part 1 of 4)



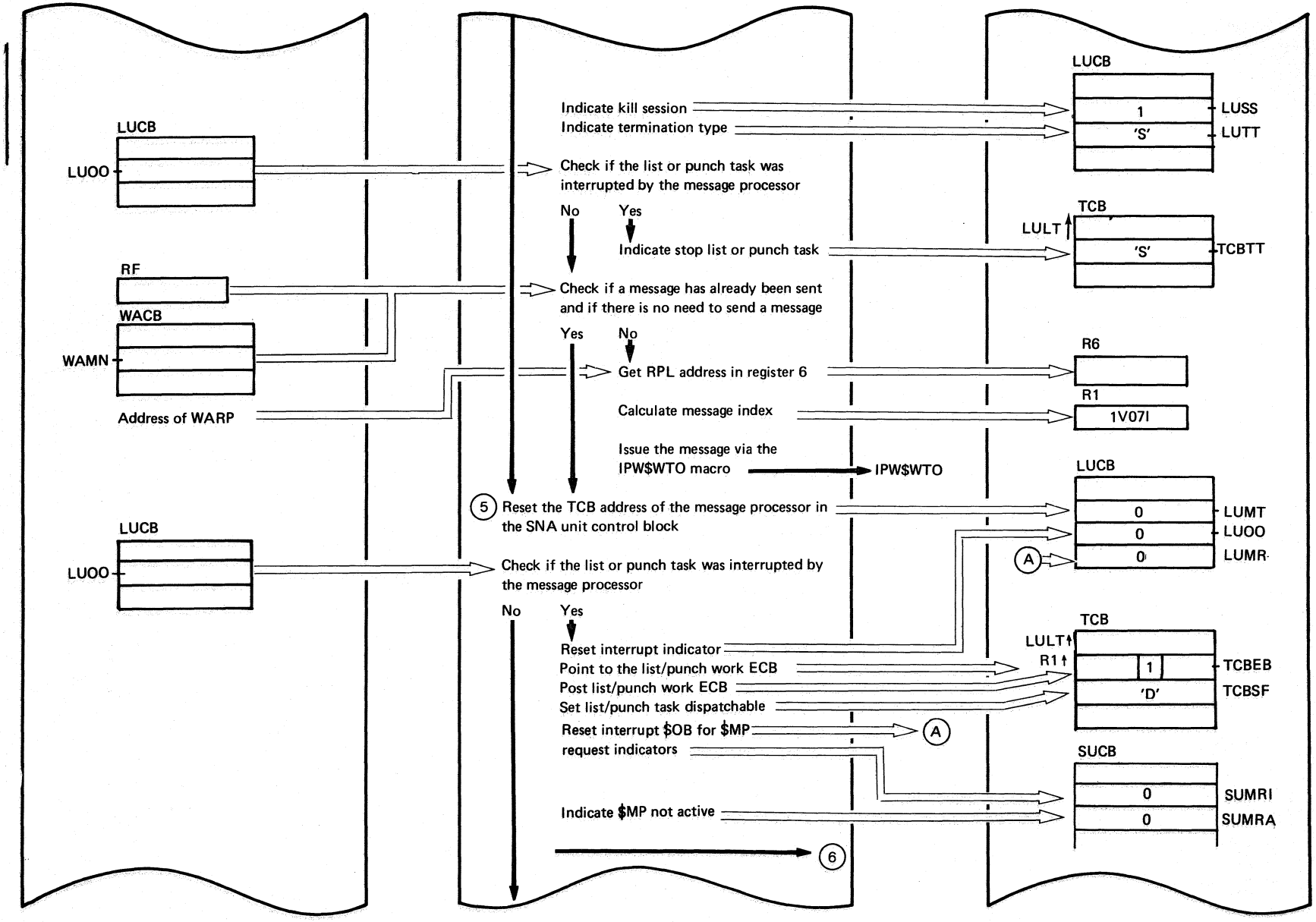
Continued on next page

TERM (Part 2 of 4)



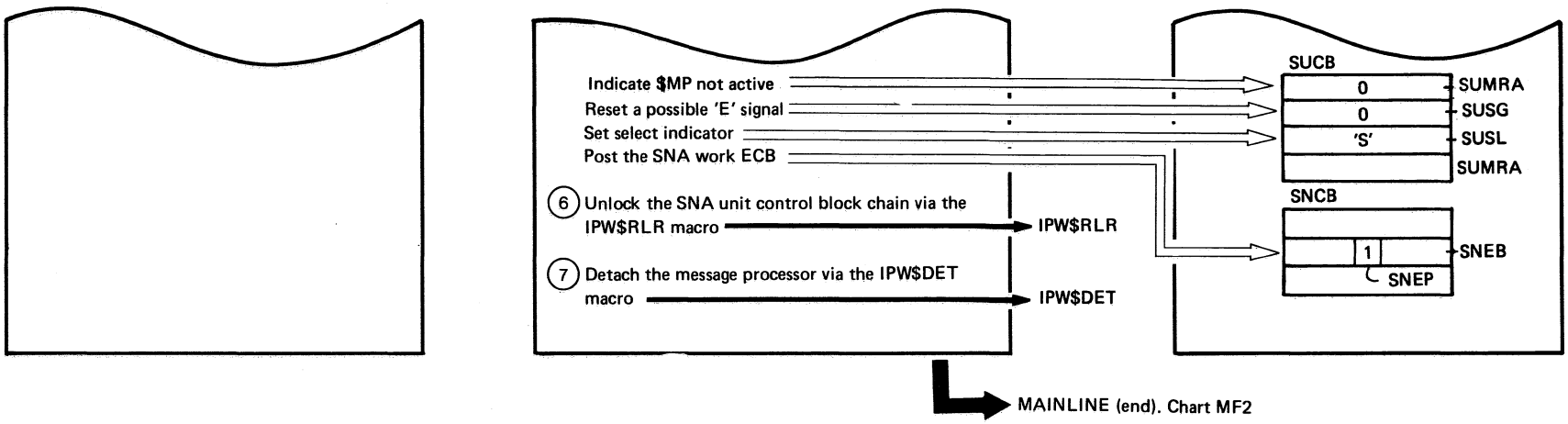
Continued on next page

TERM (Part 3 of 4)



Continued on next page

TERM (Part 4 of 4)



Extended Description	Include Segment	Call Subroutine/Macro	Chart
(1)		RESETSR(VTAM) IPW$WFC(POWER/VS) CHECK(VTAM)	
(1.1) If an error during send occured, the msgs should not be removed from the msg queue. The temp. delete msg chain still exists during termination and must be purged to prevent deleting msgs from the msg queue.			
(2)		IPW$RLR(POWER/VS) FREEVIS(DOS/VS)	
(3)		IPW$RSR(POWER/VS)	
(4) 1V07I ERROR ON 'request' RTNCD,FDBK2=xx,yy SENSE=xxxx 'luname'		IPW$WTO(POWER/VS)	
(6)		IPW$RLR(POWER/VS)	
(7)		IPW$DET(POWER/VS)	

CHART MG: IPW$$IB - SNA INBOUND PROCESSOR (84 PARTS)

Chart MG00: IPW$$IB - SNA Inbound Processor, General Flow and Macro Calls

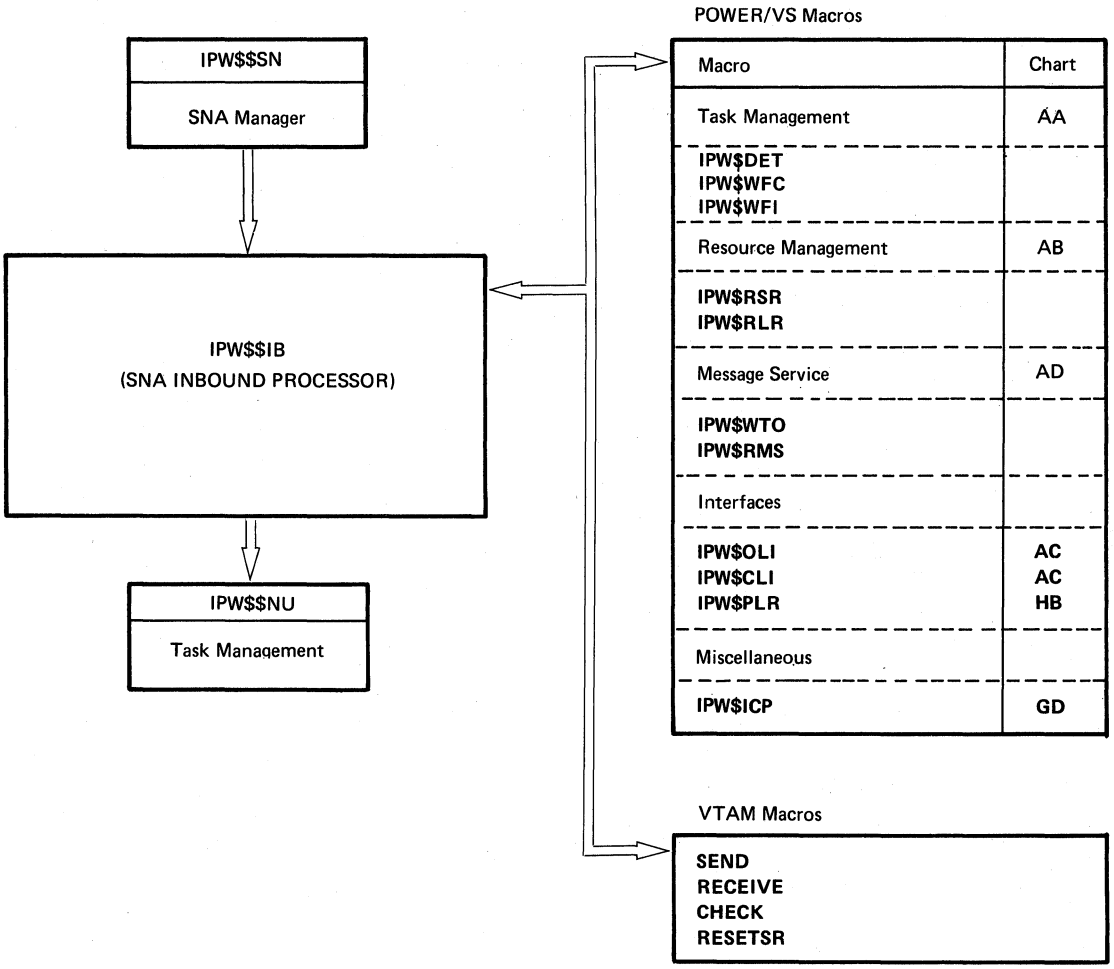
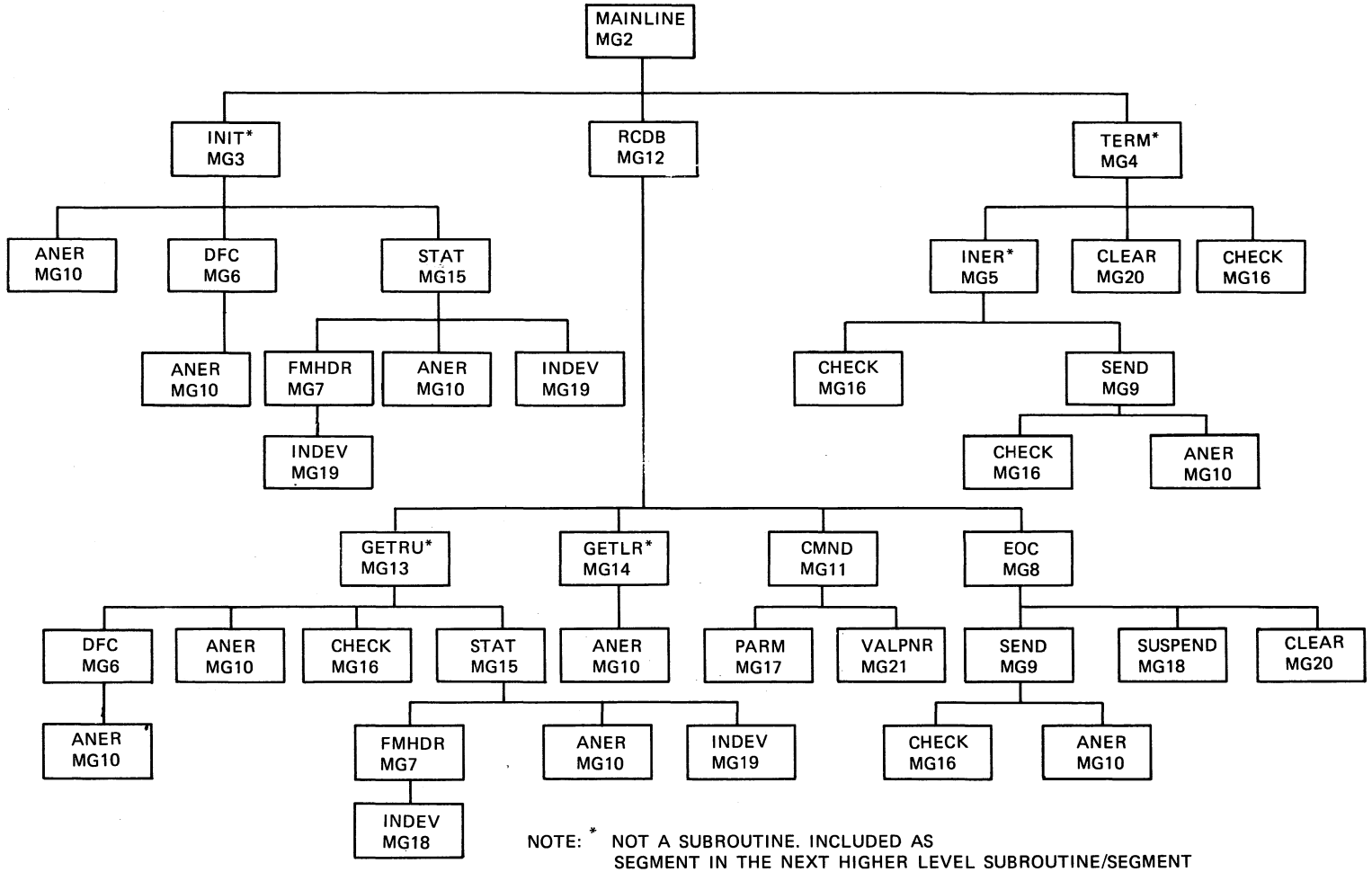


Chart MG1

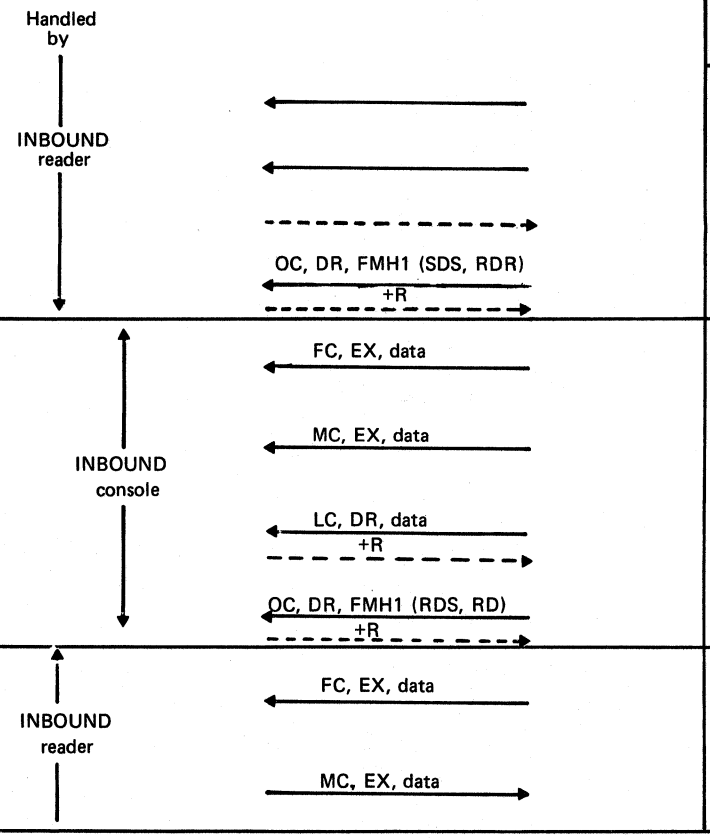
SNA INBOUND PROCESSOR (IPW$$IB), ORGANIZATION



NOTE: * NOT A SUBROUTINE. INCLUDED AS SEGMENT IN THE NEXT HIGHER LEVEL SUBROUTINE/SEGMENT

Chart MG1

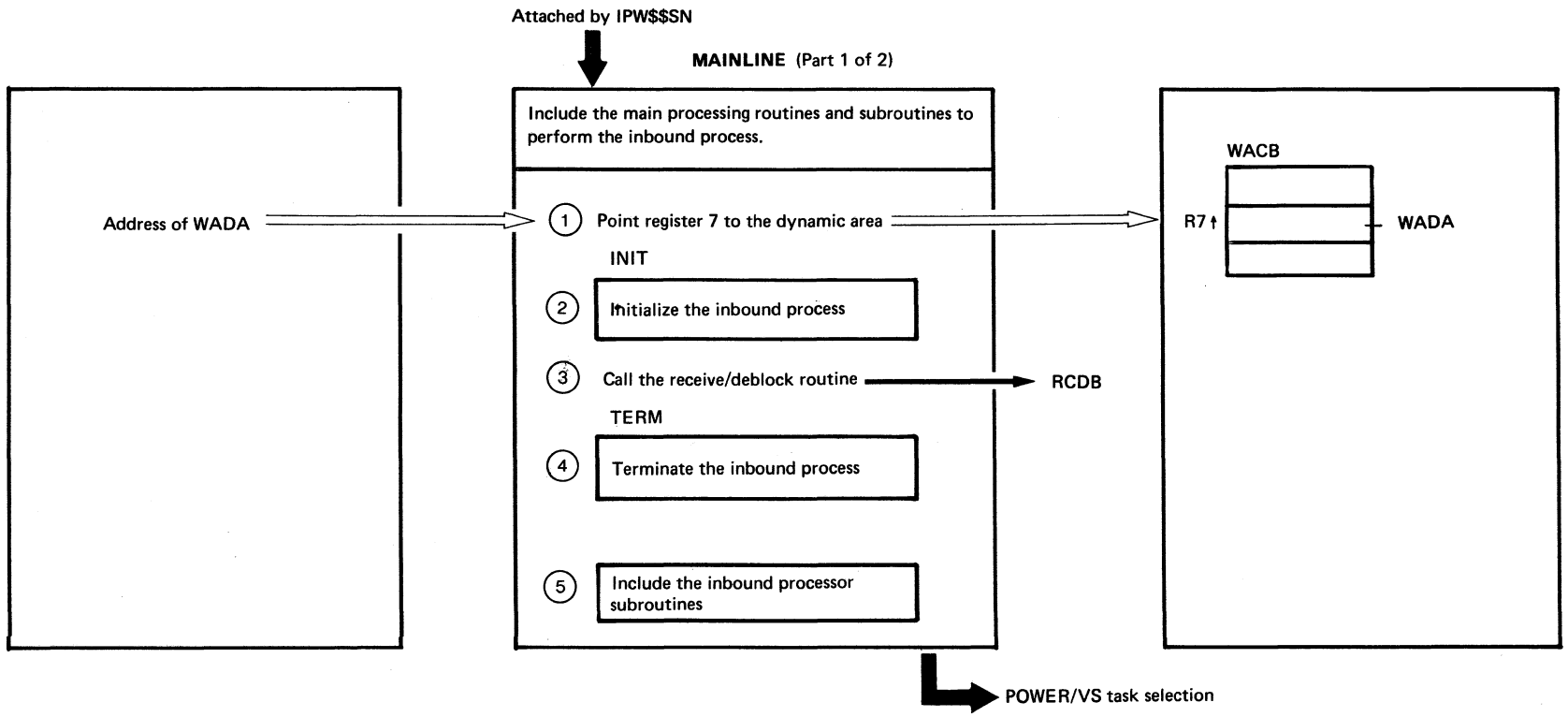
DS State Protocol for Interrupt INBOUND reader for INBOUND console



	Reader		Console	
	DS	CS	DS	CS
	I	I	B	
	I	EP		
		B		
OC, DR, FMH1 (SDS, RDR)				
+R	S	B	B	
FC, EX, data			B	I
MC, EX, data			B	I
LC, DR, data				EP
+R			B	B
OC, DR, FMH1 (RDS, RD)				
+R	SR			B
FC, EX, data	I	I	B	B
MC, EX, data	I	I	B	B

Abbreviations: SNA defined:

CS ..... chain state
DR ..... definite response mode
DS ..... data stream state
EX ..... exception response
FC ..... first of chain
FMH1 .. function management header type 1
LC ..... last of chain
MC ..... middle of chain
OC ..... only chain
+R ..... positive response
RDR ..... reader
RDS ..... resume data stream
SDS ..... suspend data stream
B ..... between state
I ..... in state
S ..... suspend state
EP ..... end (between) state pending
SR ..... suspend state with resume requested


Attached by IPW$$SN
MAINLINE (Part 1 of 2)
Include the main processing routines and subroutines to perform the inbound process.
Address of WADA
1 Point register 7 to the dynamic area
INIT
2 Initialize the inbound process
3 Call the receive/deblock routine
RCDB
TERM
4 Terminate the inbound process
5 Include the inbound processor subroutines
WACB
R7
WADA
POWER/VS task selection

MAINLINE (Part 2 of 2)

Extended Description	Include Segment	Call Subroutine/Macro	Chart
(2) First page of IPW$$IB (Main procedure):	INIT		MG3
(3) Receive request units (RUS), deblock them to logical records and pass them to the logical reader (IPW$$LR) or command handler (CMND). RCDB subroutine is included in the WORKINGSET (see ref. 5).		RCDB	
		IPW$$LR	HB
(4)		CMND	MG11
	TERM		MG4
(5) Analyze errors		ANER	MG10
– Second page of IPW$$IB:			
• Data flow control command handler		DFC	MG6
• Function management header validation		FMHDR	MG7
• End of chain handler		EOC	MG8
• Interrupt IB wait routine		SUSPEND	MG18
• Clear device characteristics field in SUCB		CLEAR	MG20
• Send responses or requests		SEND	MG9
– Third page of IPW$$IB: (RJE command handling):		CMND	
• RJE command handling		PARM	
• Analyze first command operand		VALPNR	
• Validate page numbers for SETUP and RESTART command			
– Fourth page of IPW$$IB (Working Set)			MG12
• Receive and deblock RUS		RCDB	MG15
• Validate bracket, data stream, and chain states		STAT	MG19
• Initialize OB device characteristic fields in SUCB		INDEV	MG16
• Issue the VTAM check macro		CHECK	

Included by MAINLINE. Chart MG2

INIT (Part 1 of 2)

Initialize the SNA workarea and RPL and check the receive any RPL.

LUCB
LUIS
LUOI
LUPH

(1) Initialize the reader class list in the TCB
If a rdr or writer is interrupted save ptr to SUCB device entry

(2) Initialize the SNA workarea with:
- Entry address for terminator — Address of RTRN
- Residual count
- Address of receive buffer — Address of WABF
- Address of buffer in process — Address of WABF + SUBS
- Processing switches
- Status byte
- Sense byte
- Error message number
- Data stream state flags
- Chain state flags
- FMH

(3) Point register 1 to the RPL — Address of WARP
Initialize the RPL with:
- Buffer size
- ECB address

LUCB
LUSS

Address of TCBEB

(3.1) If a 0-RU or a RDR interrupted → (6)

WACB
R1↑
RPLRTNCD
RPLLEN

(4) Check if an error occurred on CHECK or RECEIVE any

No

Yes
Call ANER → ANER MG10

→ (6)

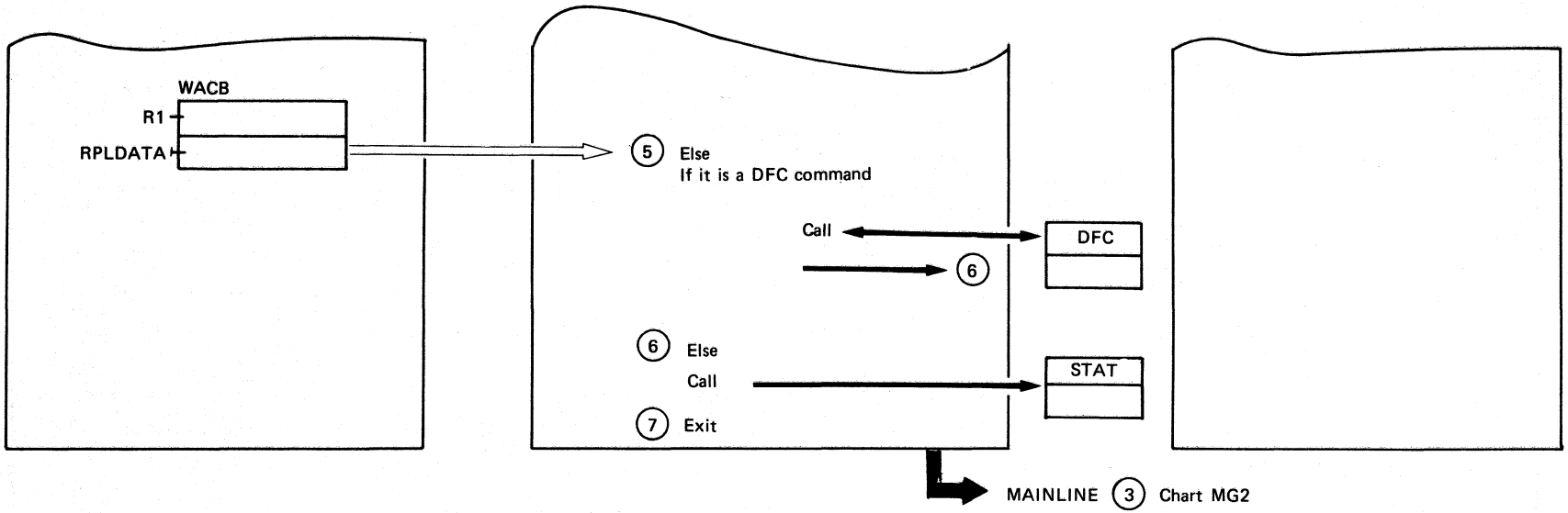
LUCB
LUIS

TCB
'A' — TCBAC

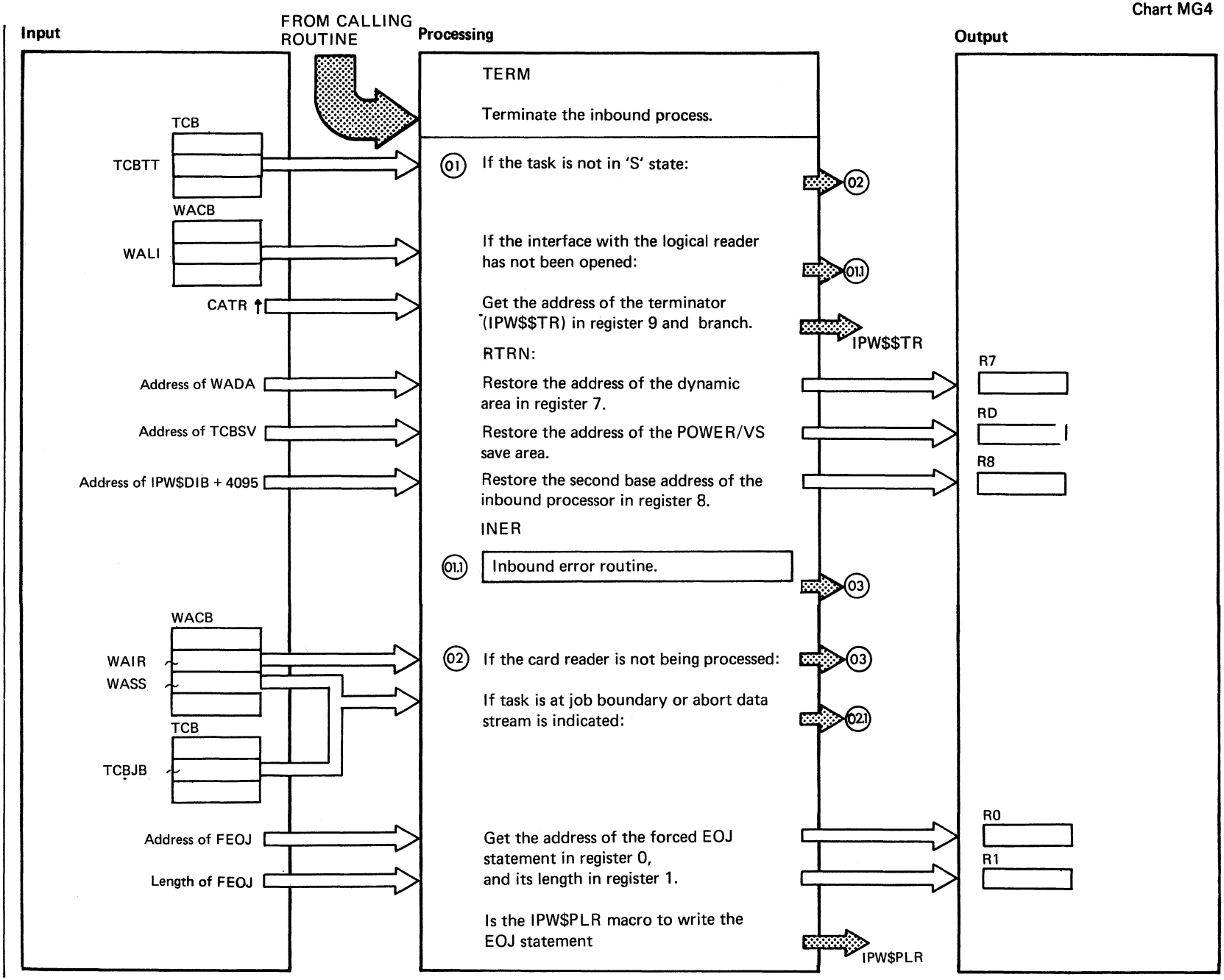
WACB
R1↑

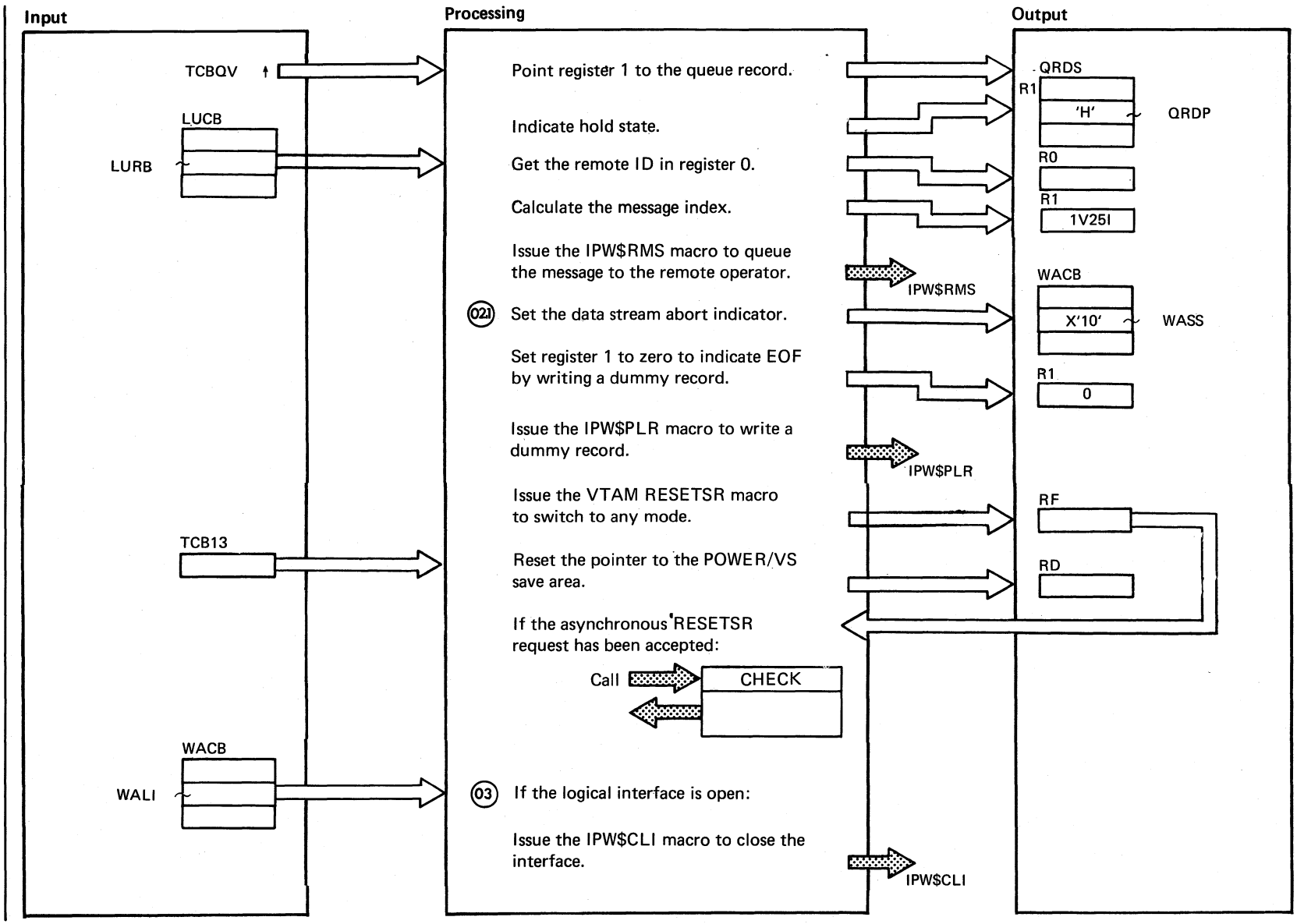
	WAPH
	WAER
0	WARC
	WABI
	WABP
0	WASW
0	WAST
0	WASN
0	WAMN
X'80'	WASS
X'80'	WACS
0	WAFM
	WARP
	RPLBUFL
	RPLECB

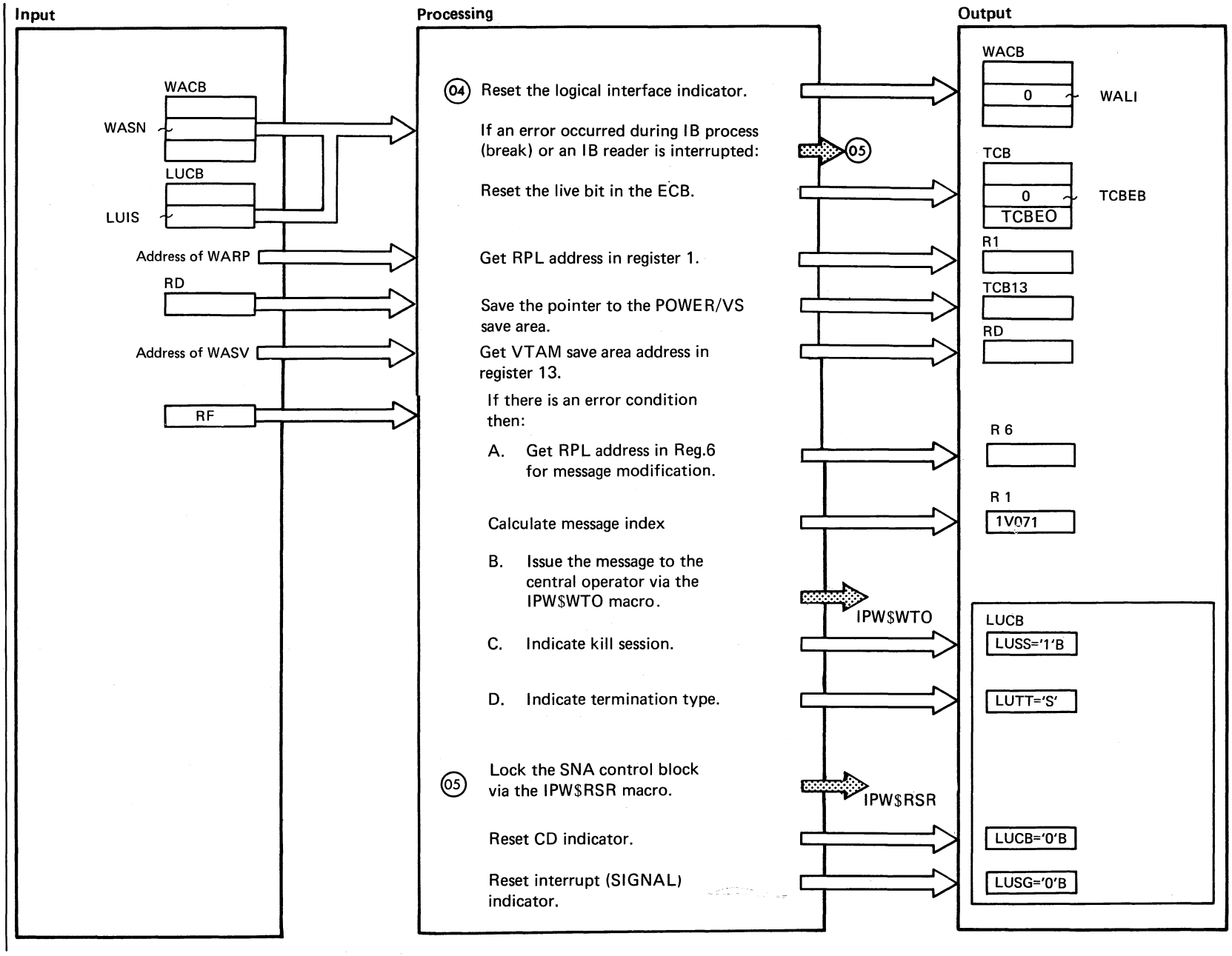
Continued on next page

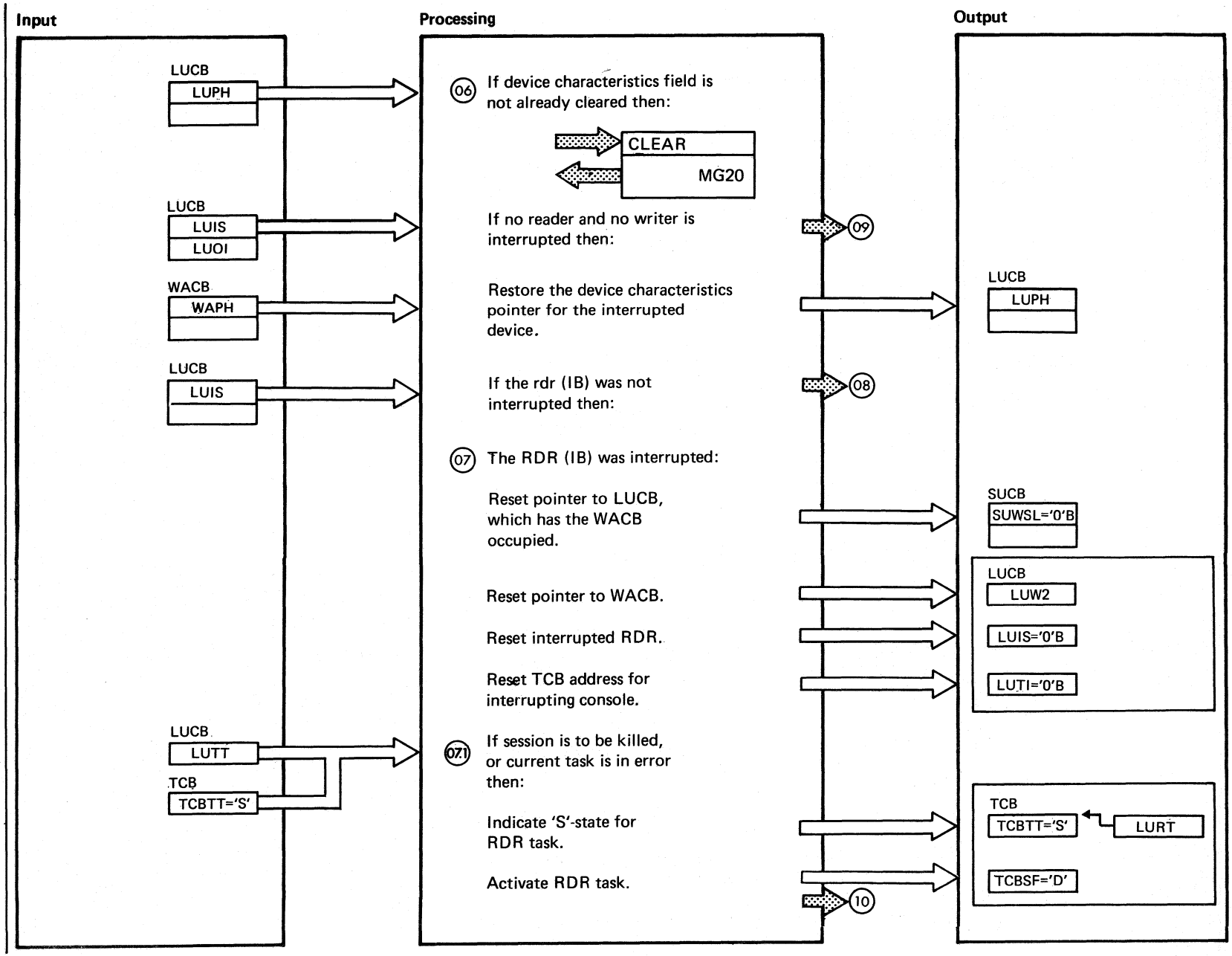


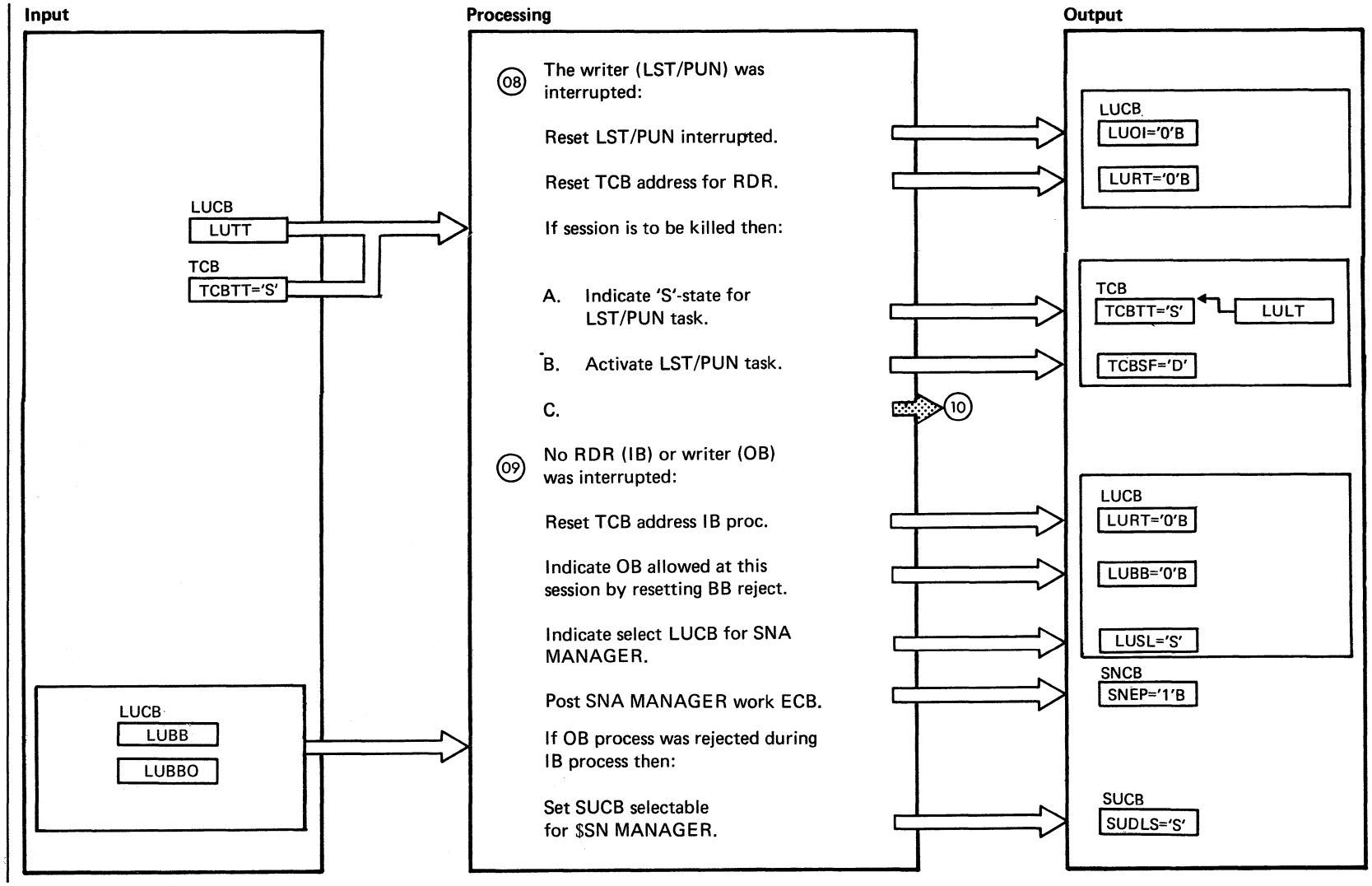
Extended Description	Include Segment	Call Subroutine/Macro	Chart
(3.1) Note: Any data associated with REC ANY must be taken out of VTAM with REC SPEC. 0 RUS were not kept; they must be analyzed immediately. REC ANY was not issued, when IB interrupted IB reader. (LUIS=ON)		ANER	MG10
(4)			
(5) Data flow control command handler		DFC	MG6
(6) It must be a 0RU. Validate and maintain bracket data stream and chain states.		STAT	MG15


Input
FROM CALLING ROUTINE
Processing
Output
TERM
Terminate the inbound process.
TCB
TCBTT
WACB
WALI
CATR
Address of WADA
Address of TCBSV
Address of IPW$DIB + 4095
01 If the task is not in 'S' state:
02
If the interface with the logical reader has not been opened:
01.1
Get the address of the terminator (IPW$$TR) in register 9 and branch.
IPW$$TR
RTRN:
Restore the address of the dynamic area in register 7.
Restore the address of the POWER/VS save area.
Restore the second base address of the inbound processor in register 8.
R7
RD
R8
INER
01.1 Inbound error routine.
03
WACB
WAIR
WASS
02 If the card reader is not being processed:
03
If task is at job boundary or abort data stream is indicated:
02.1
TCB
TCBJB
Address of FEOJ
Length of FEOJ
Get the address of the forced EOJ statement in register 0, and its length in register 1.
R0
R1
Is the IPW$PLR macro to write the EOJ statement
IPW$PLR


Input
Processing
Output
TCBQV
LUCB
LURB
TCB13
WACB
WALI
Point register 1 to the queue record.
Indicate hold state.
Get the remote ID in register 0.
Calculate the message index.
Issue the IPW$RMS macro to queue the message to the remote operator.
02
Set the data stream abort indicator.
Set register 1 to zero to indicate EOF by writing a dummy record.
Issue the IPW$PLR macro to write a dummy record.
Issue the VTAM RESETSR macro to switch to any mode.
Reset the pointer to the POWER/VS save area.
If the asynchronous RESETSR request has been accepted:
Call
CHECK
03
If the logical interface is open:
Issue the IPW$CLI macro to close the interface.
IPW$RMS
IPW$PLR
IPW$CLI
QRDS
R1
'H'
QRDP
R0
R1
1V25I
WACB
X'10'
WASS
R1
0
RF
RD


Input
Processing
Output
WACB
WASN
LUCB
LUIS
Address of WARP
RD
Address of WASV
RF
04 Reset the logical interface indicator.
If an error occurred during IB process (break) or an IB reader is interrupted:
05
Reset the live bit in the ECB.
Get RPL address in register 1.
Save the pointer to the POWER/VS save area.
Get VTAM save area address in register 13.
If there is an error condition then:
A. Get RPL address in Reg.6 for message modification.
Calculate message index
B. Issue the message to the central operator via the IPW$WTO macro.
IPW$WTO
C. Indicate kill session.
D. Indicate termination type.
05 Lock the SNA control block via the IPW$RSR macro.
IPW$RSR
Reset CD indicator.
Reset interrupt (SIGNAL) indicator.
WACB
0
WALI
TCB
0
TCBEB
TCBEO
R1
TCB13
RD
R 6
R 1
1V071
LUCB
LUSS='1'B
LUTT='S'
LUCB='0'B
LUSG='0'B


Input
Processing
Output
LUCB
LUPH
LUCB
LUIS
LUOI
WACB
WAPH
LUCB
LUIS
LUCB
LUTT
TCB
TCBTT='S'
06 If device characteristics field is not already cleared then:
CLEAR
MG20
If no reader and no writer is interrupted then:
09
Restore the device characteristics pointer for the interrupted device.
If the rdr (IB) was not interrupted then:
08
07 The RDR (IB) was interrupted:
Reset pointer to LUCB, which has the WACB occupied.
Reset pointer to WACB.
Reset interrupted RDR.
Reset TCB address for interrupting console.
07.1 If session is to be killed, or current task is in error then:
Indicate 'S'-state for RDR task.
Activate RDR task.
10
LUCB
LUPH
SUCB
SUWSL='0'B
LUCB
LUW2
LUIS='0'B
LUTI='0'B
TCB
TCBTT='S'
LURT
TCBSF='D'


Input
Processing
Output
LUCB
LUTT
TCB
TCBTT='S'
LUCB
LUBB
LUBBO
08
The writer (LST/PUN) was interrupted:
Reset LST/PUN interrupted.
Reset TCB address for RDR.
If session is to be killed then:
A. Indicate 'S'-state for LST/PUN task.
B. Activate LST/PUN task.
C.
10
09
No RDR (IB) or writer (OB) was interrupted:
Reset TCB address IB proc.
Indicate OB allowed at this session by resetting BB reject.
Indicate select LUCB for SNA MANAGER.
Post SNA MANAGER work ECB.
If OB process was rejected during IB process then:
Set SUCB selectable for $SN MANAGER.
LUCB
LUOI='0'B
LURT='0'B
TCB
TCBTT='S'
LULT
TCBSF='D'
LUCB
LURT='0'B
LUBB='0'B
LUSL='S'
SNCB
SNEP='1'B
SUCB
SUDLS='S'

Input

Processing

(10) Unlock the SNA control block via the IPW$RLR macro.

(11) Detach the SNA IB processor task via the IPW$DET macro.

IPW$RLR

IPW$DET

MAINLINE

Output

Extended Description

Include Segment	Call Subroutine/ Macro	Chart
IPW$RSR (POWER/ VS)		MG 5

(05)

(06) If the device characteristcs field is already cleared (normally it will be done when data stream is set in between data stream state - EOC subroutine), the pointer to it (LUPH) must be -0-. When an error occured or the processor was set externally into STOP immediate state, this field is not cleared yet.

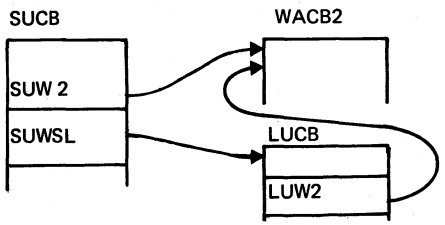
(07) To free the WACB, which got the console task via the SNA manager, the following ptrs must be cleared:

A. SUWSL .... Pointer to LUCB which uses second LUCB.

B. LUW2... interrupting IB workspace address obtained from SUCB.

Note:

WACB chaining during interrupt IB for IB:



Extended Description	Include Segment	Call Subroutine/ Macro	Chart

(07.1) If an error occurs on the second interrupt level, all processors will be stopped, but not the session. The end bracket will be sent by the OB -processor (see table chart MG4 - 09)

(08) Normal IB processor termination occurs. Only the SNA manager (IPW$$SN) must be activated to perform a scan through all LUCBS according to a SUCB. He has to look for further work.

(10)

(11)

IPW$DET (POWER/ VS)

IPW$RLR (POWER/ VS)

IB - TERM

ERROR handling

Interrupt Level	Level: Proc.	Level: Device	No interrupt	1st: Int. IB console for IB reader	1st: Int. OB for IB	2nd: Int. OB for IB reader for IB console
2nd	IB	Console	X	– Response 0 RU, EB	X	– Response (2)
1st	IB	Reader	Purge state – Resp. 0 RU, EB	'S' TCBT (1)	Purge state – Resp. 0 RU	'S' (3)
		Console	– Resp. 0 RU, EB	X	– Resp. 0 RU	
	OB		X	X	Continue	'S' 0 RU, EB

Notes:

(1) If IB found itself between bracket and in 'S'-state, it detaches itself after clean up (close logical interface etc.) immediately.

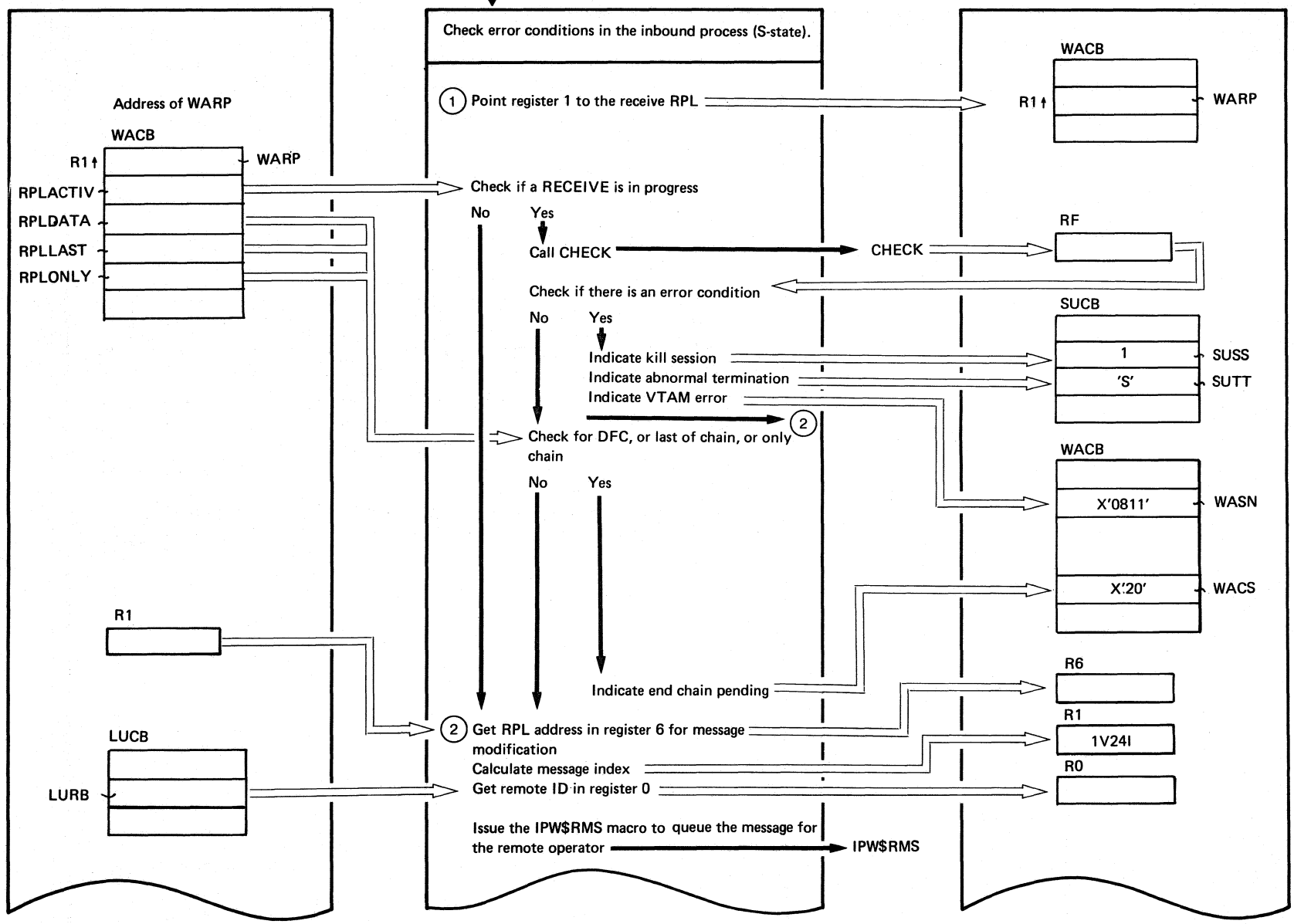
(2) To send a 0 RU will be suppressed, when IB and OB are interrupted.

(3) An indicator (WATI) tells the IB-reader, that an error occured at the second level interrupt. Therefore 0 RU sending will be suppressed.

Chart MG5

Included by TERM, Chart MG4

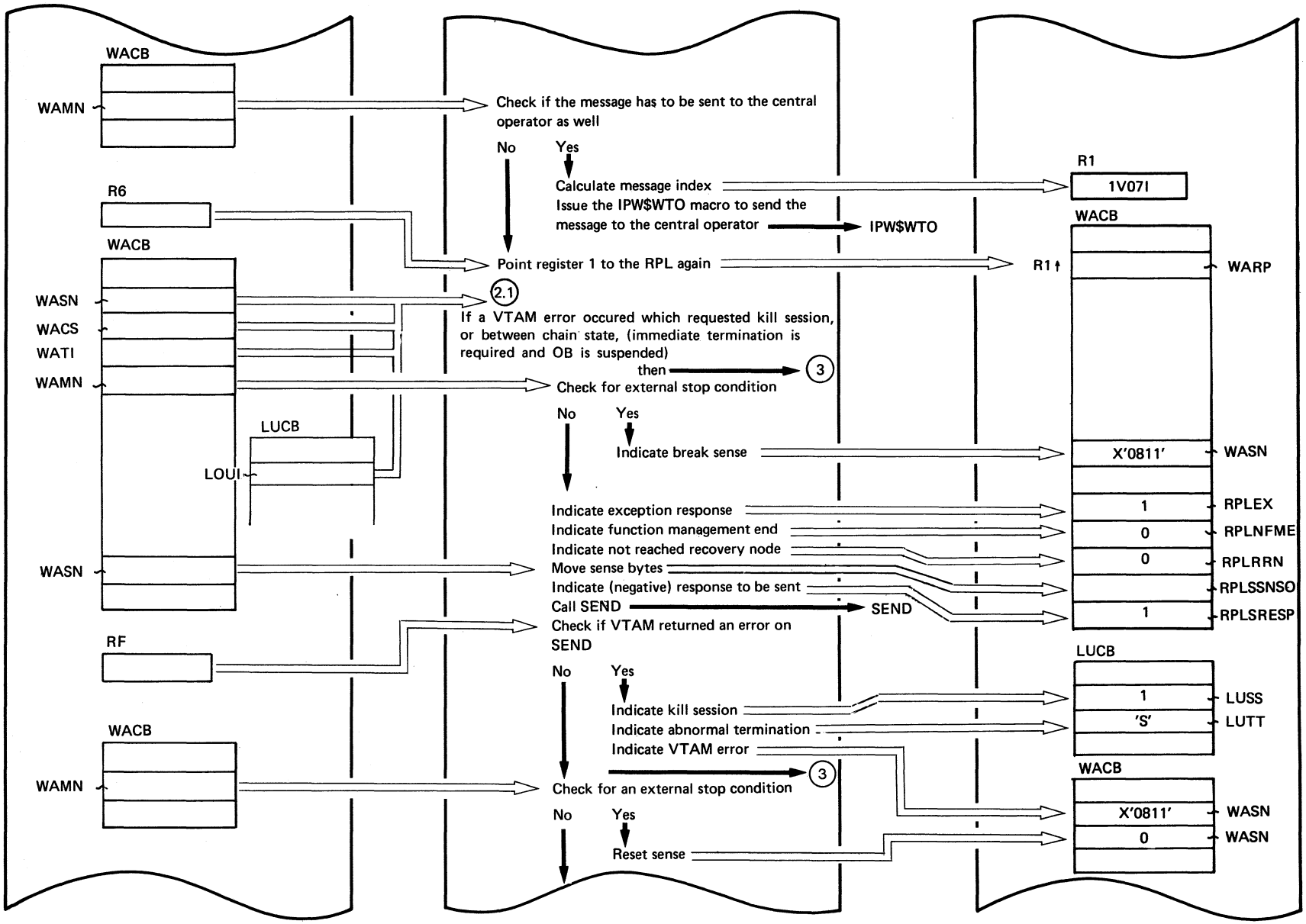
INER (Part 1 of 7)



Continued on next page

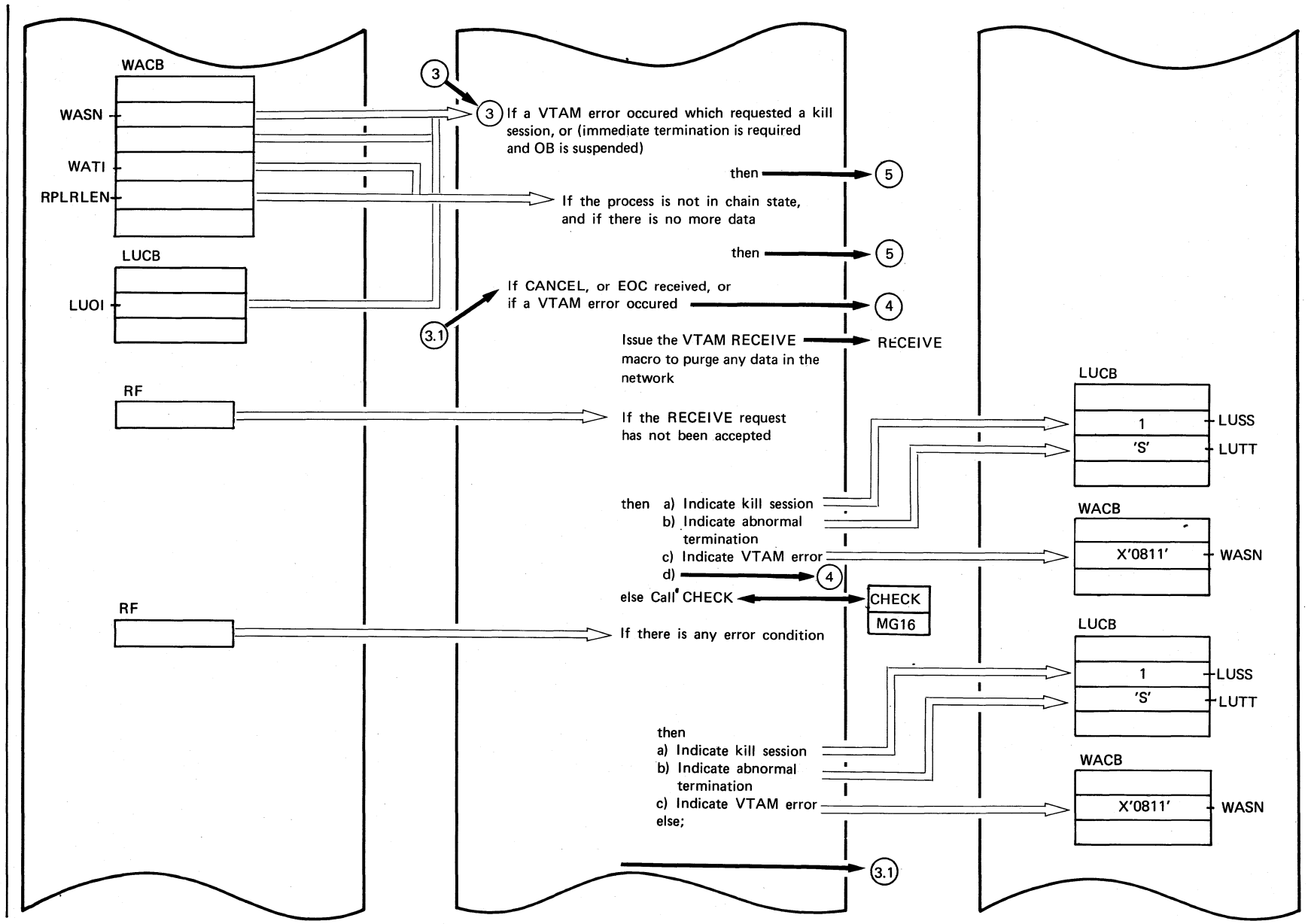
Chart MG5

INER (Part 2 of 7)

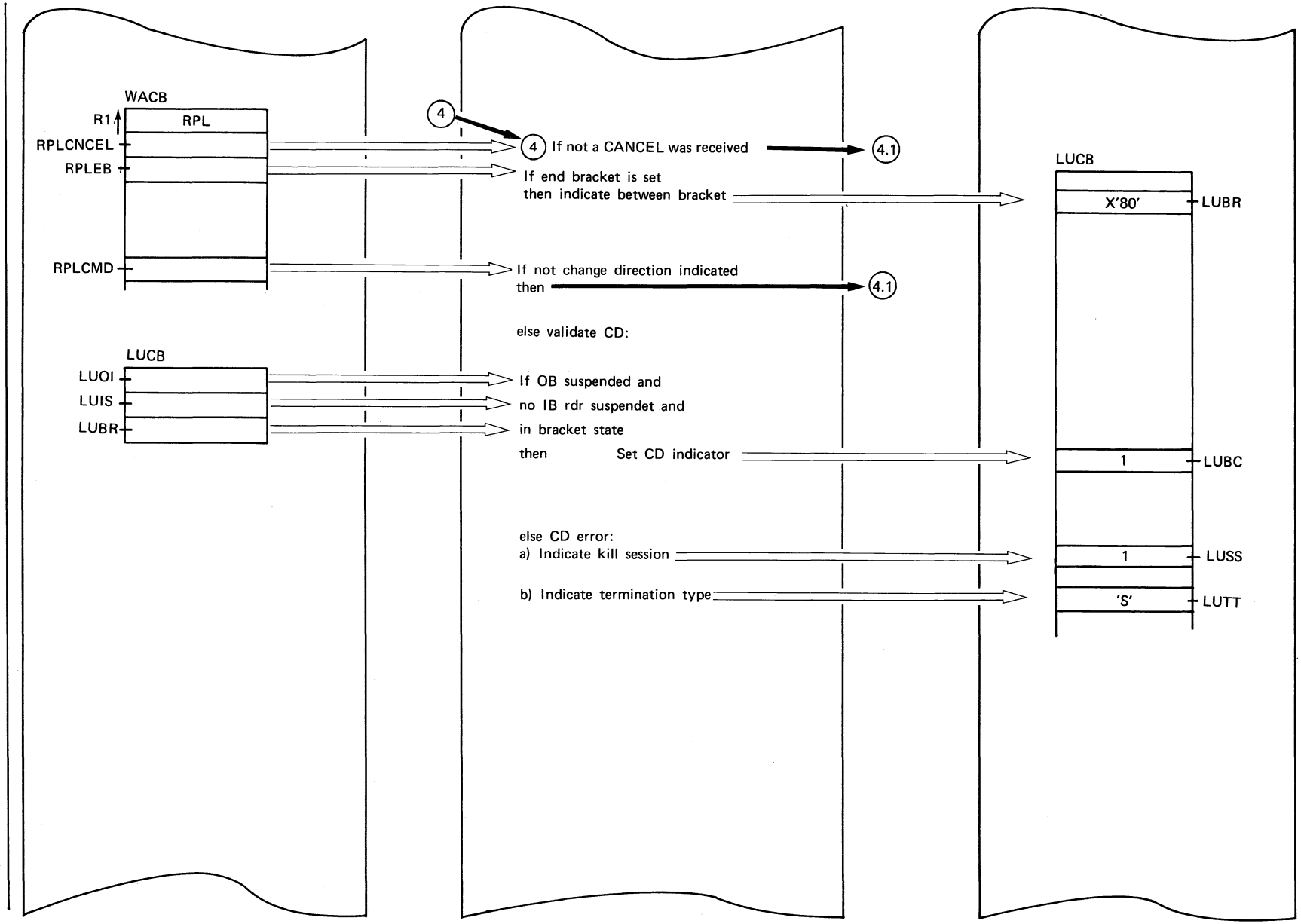


Continued on next page

INER (Part 3 of 7)

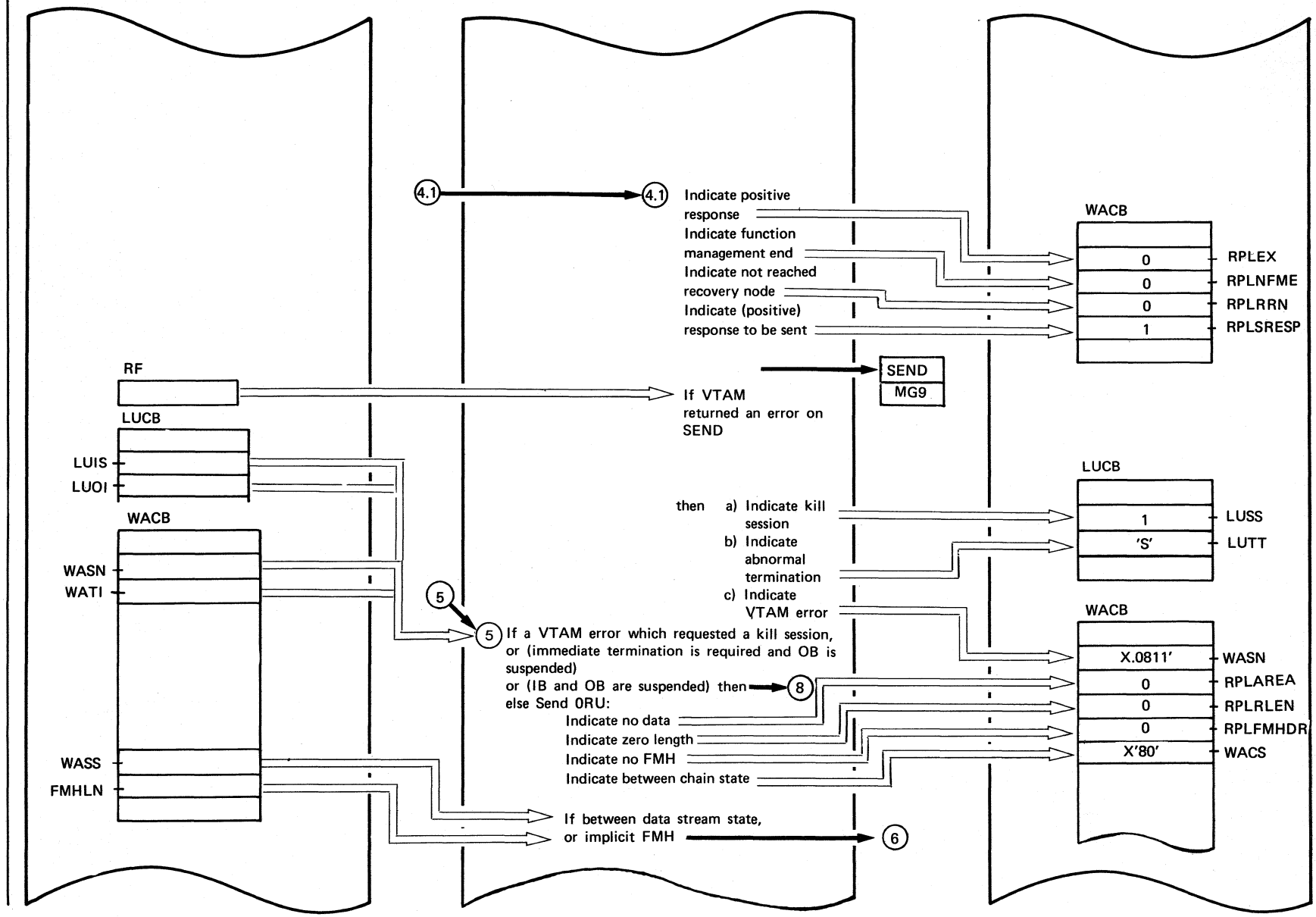


Continued on next page



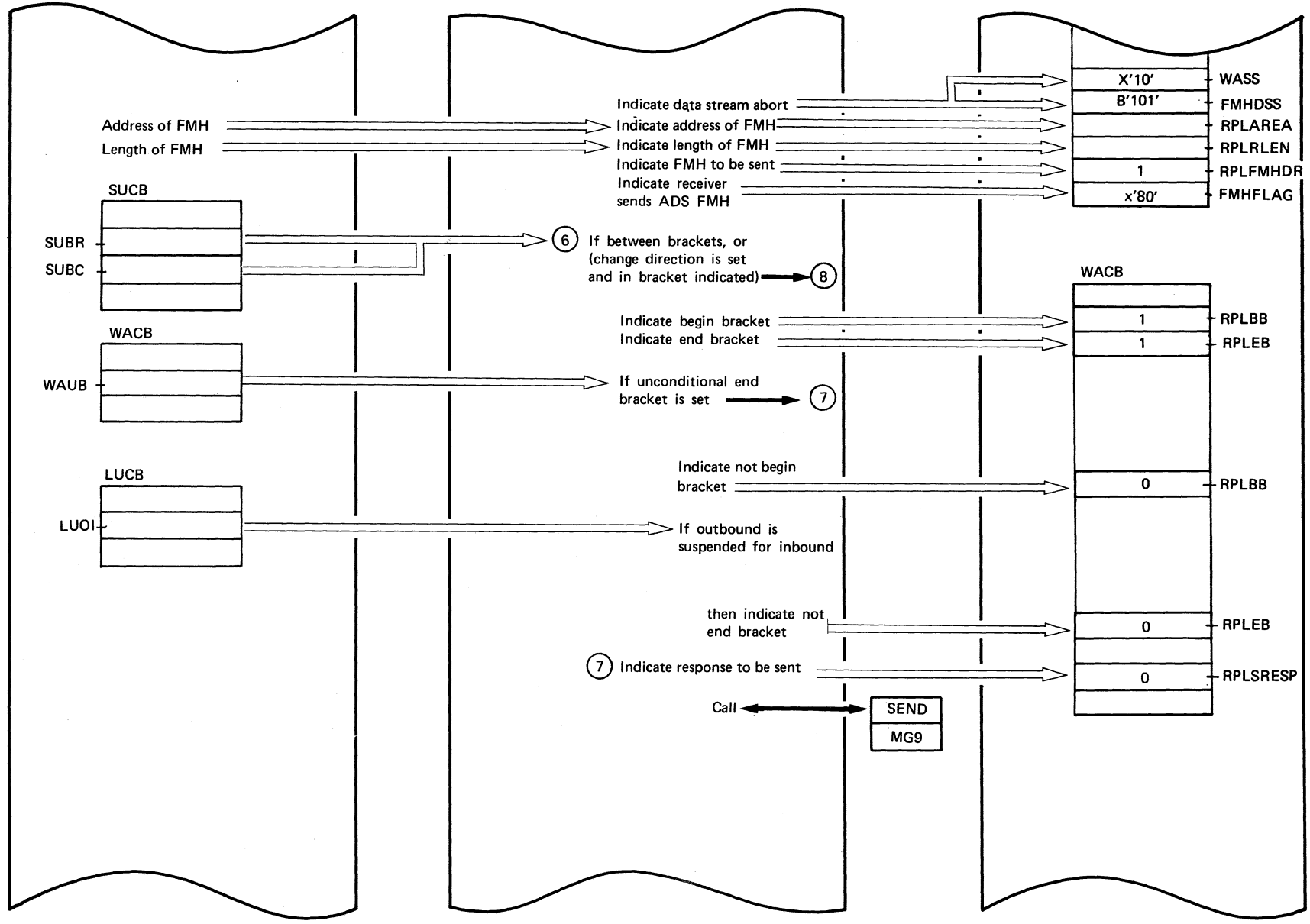
Continued on next page

INER (Part 5 of 7)



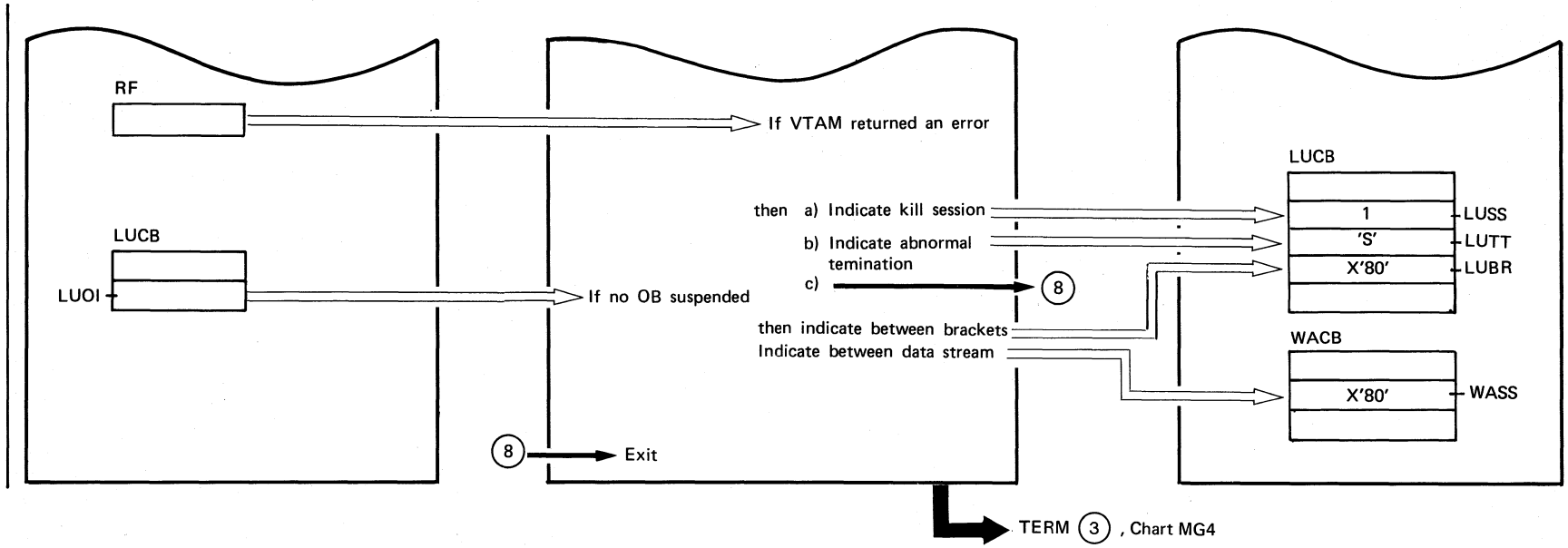
Continued on next page

INER (Part 6 of 7)

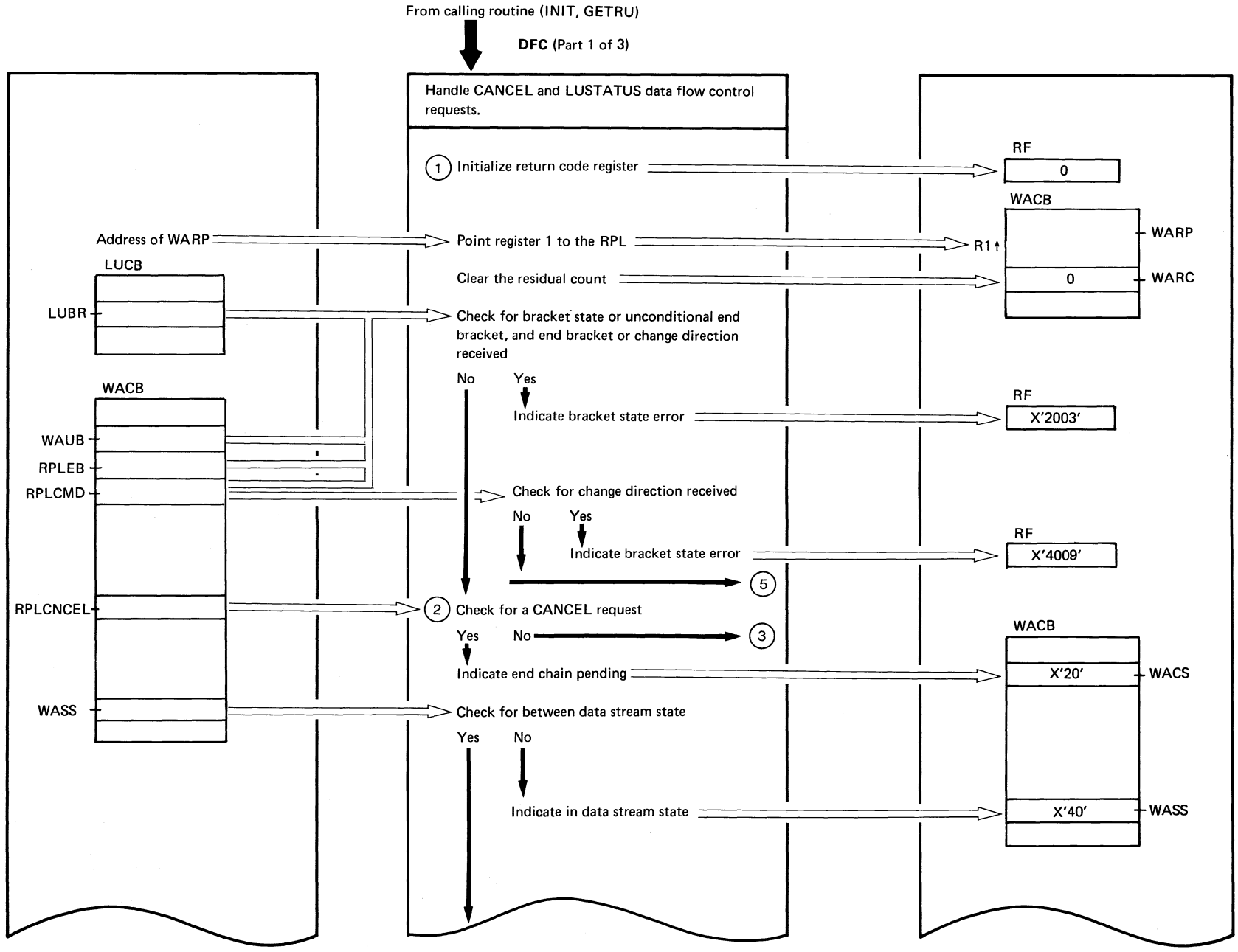


Continued on next page

INER (Part 7 of 7)

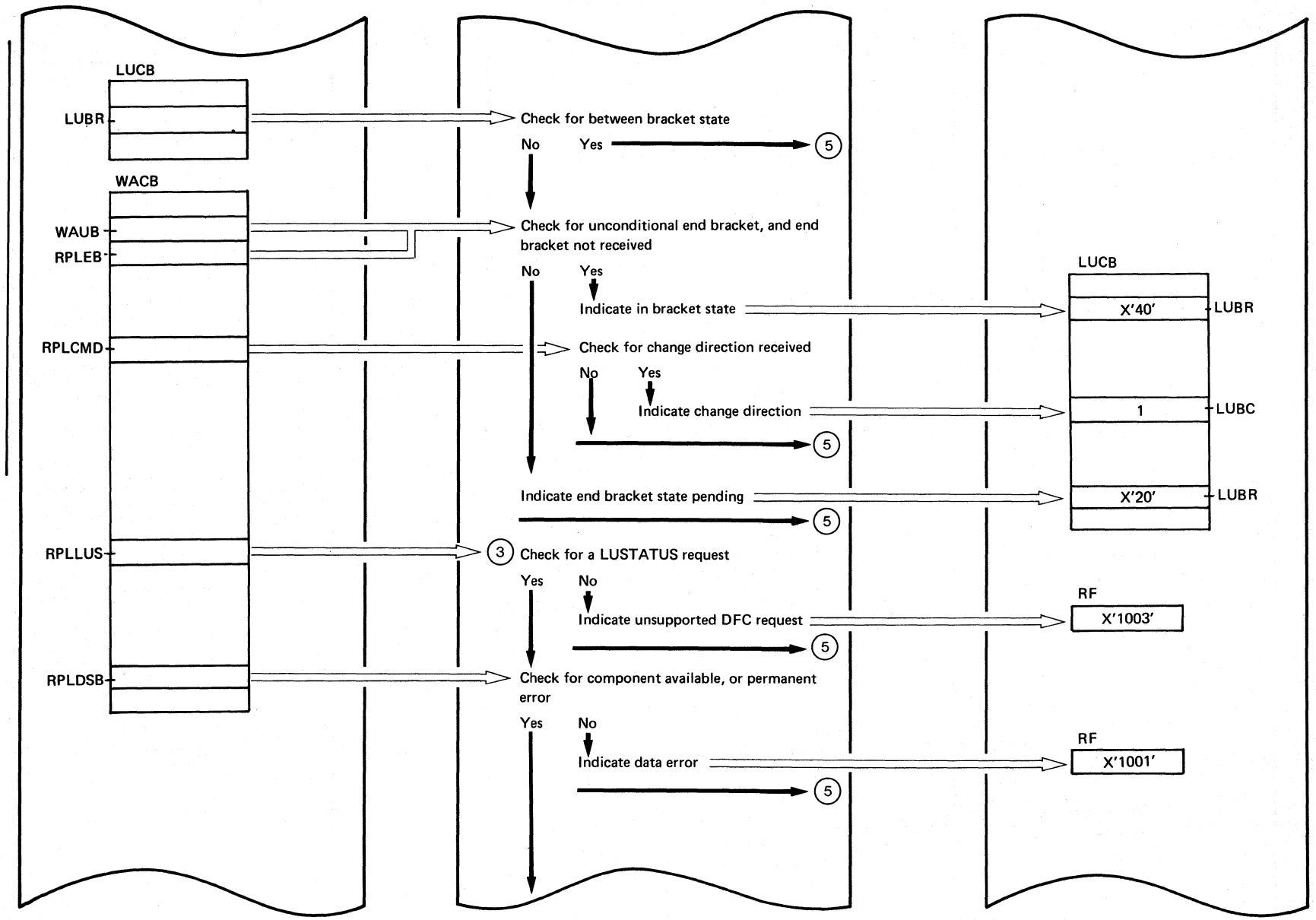


Extended Description	Include Segment	Call Subroutine/Macro	Chart
(1)		CHECK	MG16
(2) 1V24I ttt TERMINATED REASON=xxxx 1V07I ERROR ON 'request' RTNCD,FDB2=xx,xx SENSE=xxxx ON 'luname'		IPW$RMS(POWER/VS) IPW$WTO(POWER/VS)	
(3)		SEND	MG9
		RECEIVE(VTAM)	
		CHECK	MG16
(4) If a CANCEL was received an FME must be sent		SEND	MG9
(5)		SEND	MG9



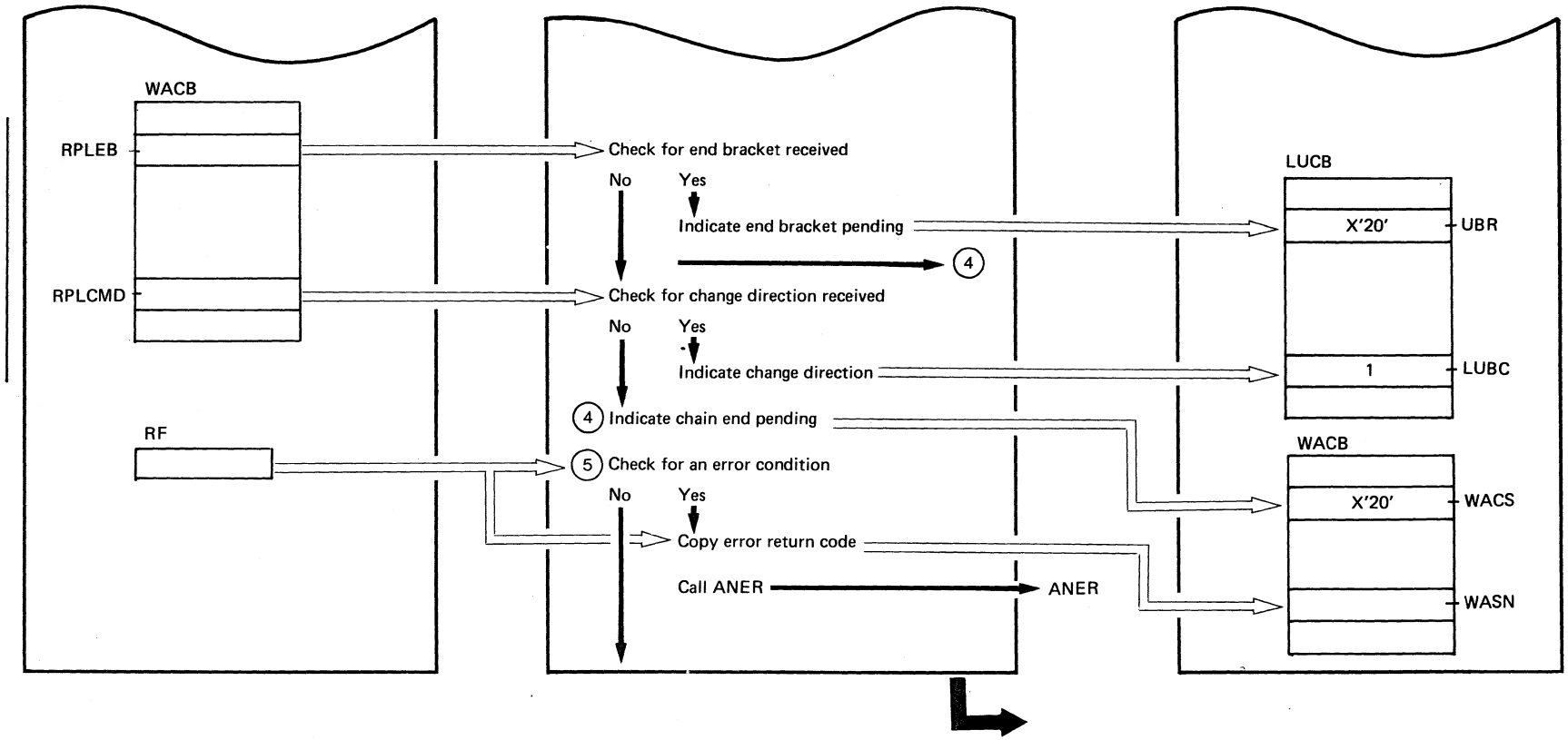
Continued on next page

DFC (Part 2 of 3)

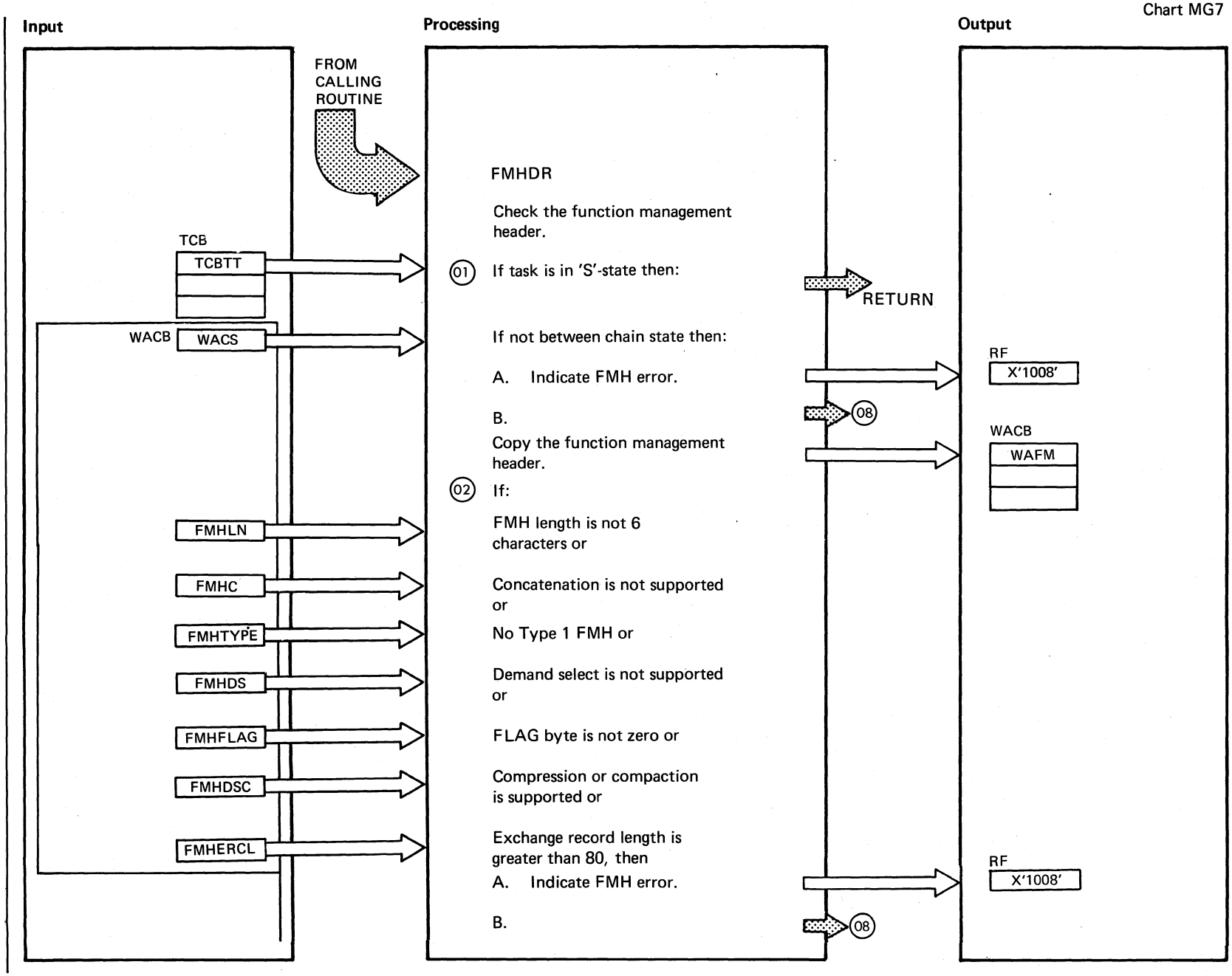


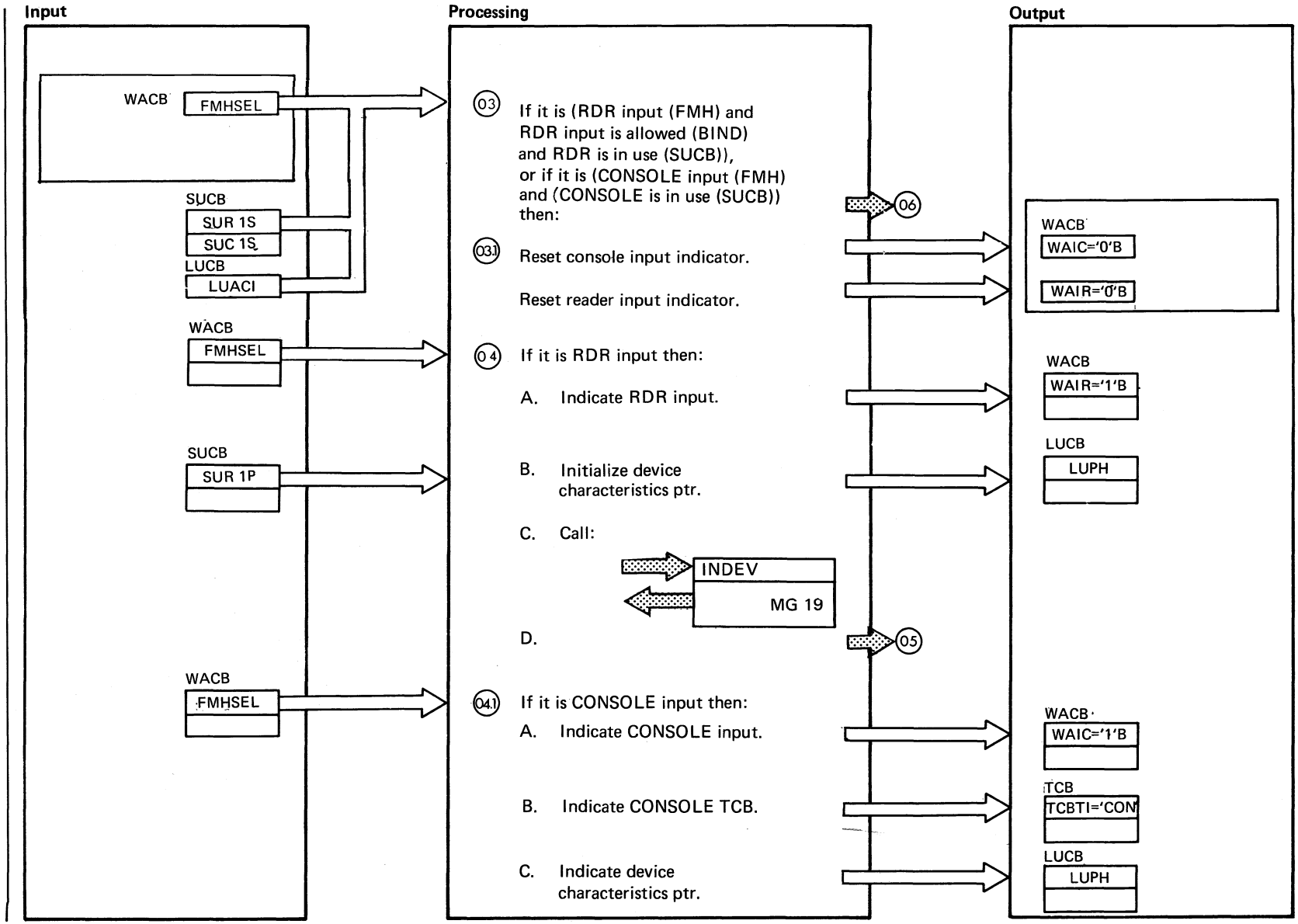
Continued on next page

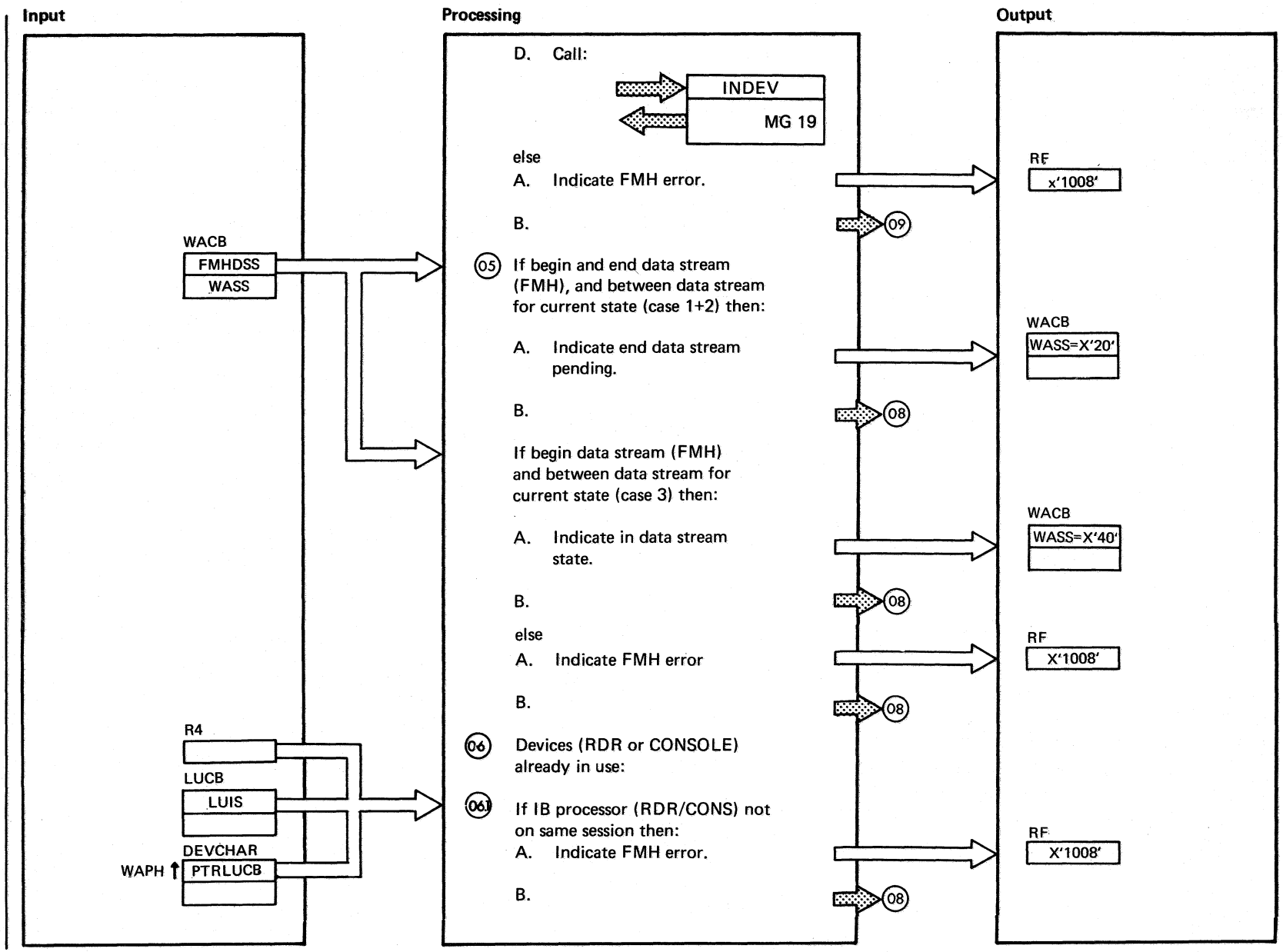
DFC (Part 3 of 3)

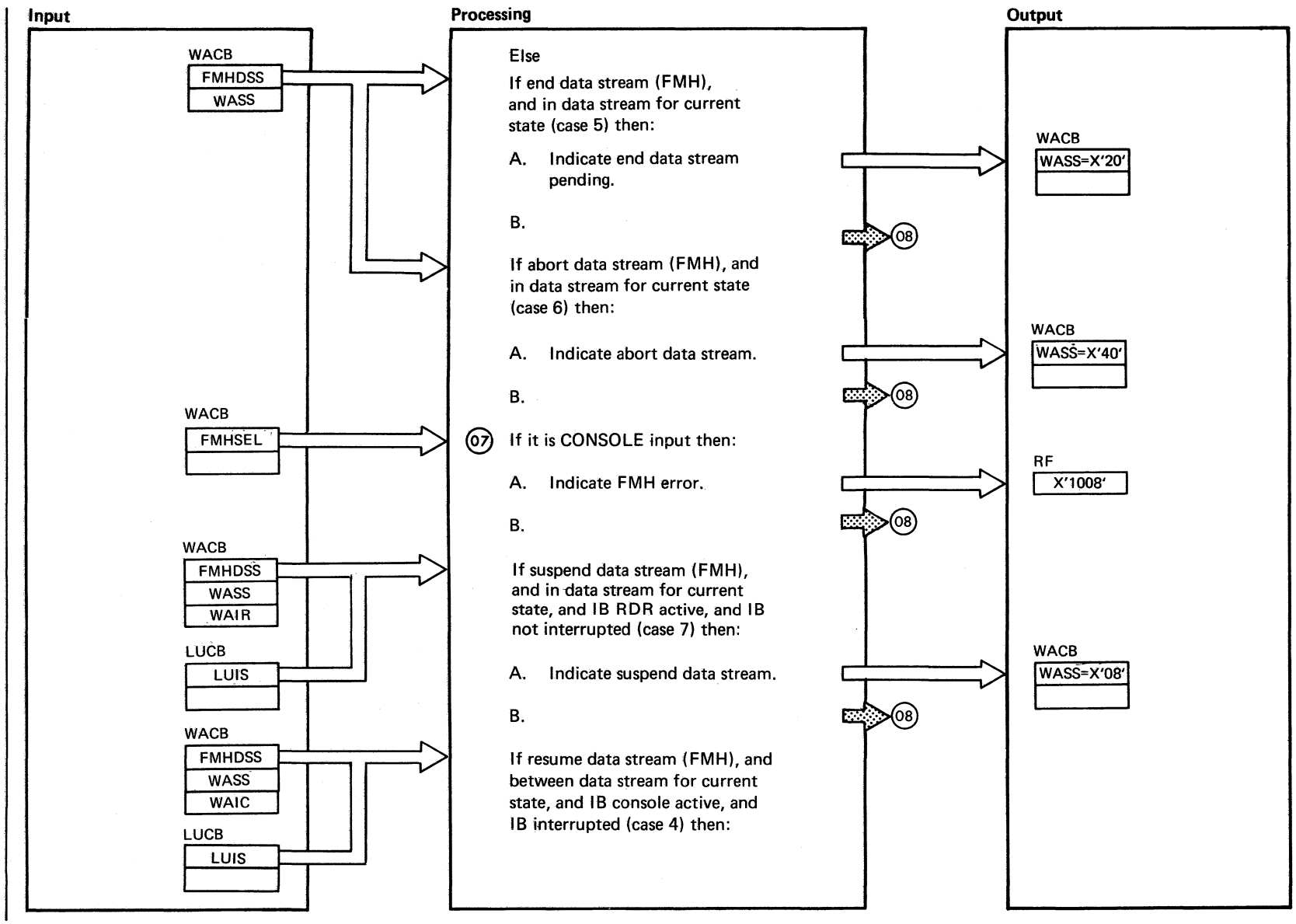


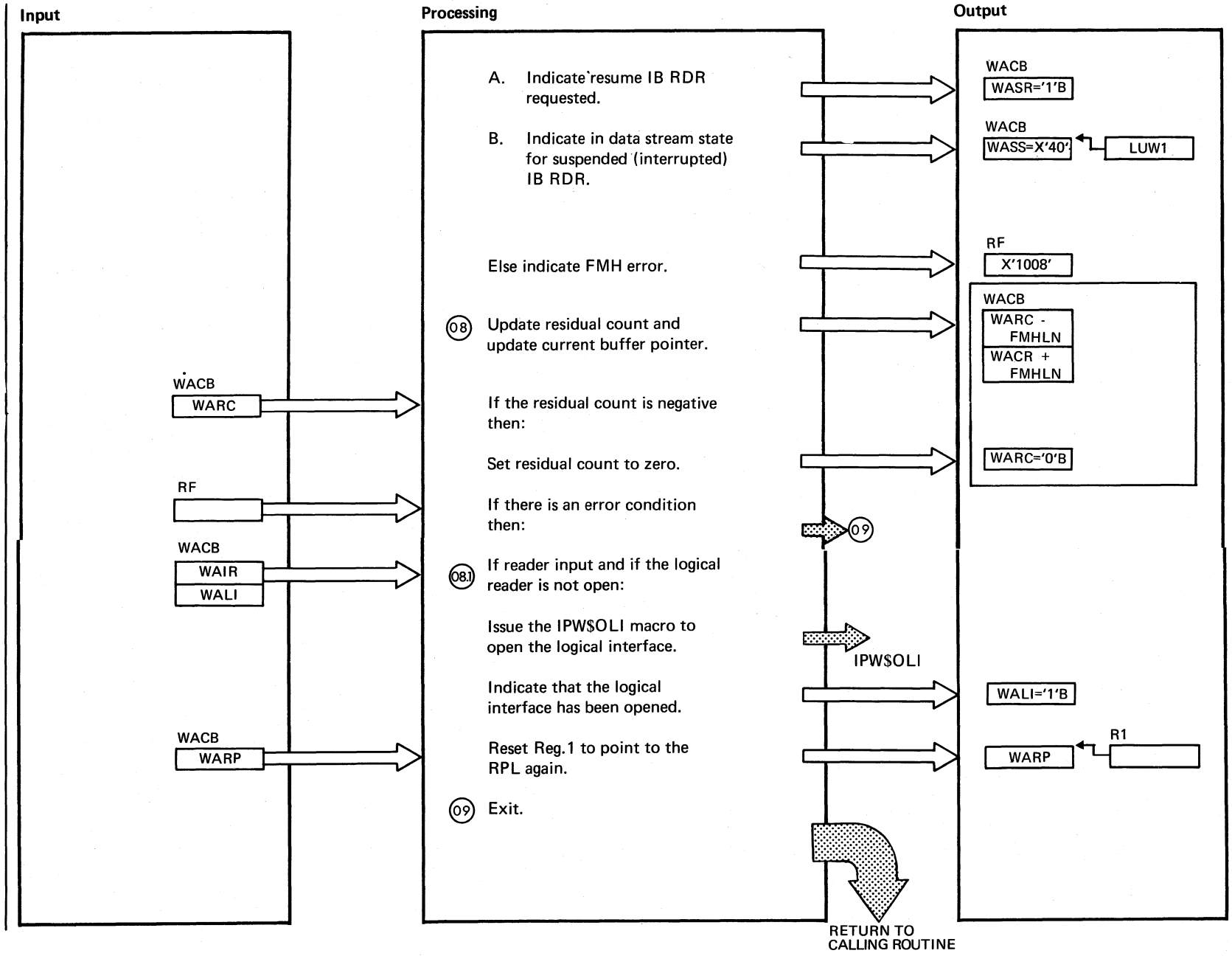
Extended Description	Include Segment	Call Subroutine/Macro	Chart
(5)		ANER	MG10


Input
Processing
Output
FROM CALLING ROUTINE
FMHDR
Check the function management header.
TCB
TCBTT
01 If task is in 'S'-state then:
RETURN
WACB
WACS
If not between chain state then:
RF
X'1008'
A. Indicate FMH error.
08
B.
Copy the function management header.
WACB
WAFM
02 If:
FMHLN
FMH length is not 6 characters or
FMHC
Concatenation is not supported or
FMHTYPE
No Type 1 FMH or
FMHDS
Demand select is not supported or
FMHFLAG
FLAG byte is not zero or
FMHDSC
Compression or compaction is supported or
FMHERCL
Exchange record length is greater than 80, then
RF
X'1008'
A. Indicate FMH error.
B.
08


Input
Processing
Output
WACB
FMHSEL
SUCB
SUR 1S
SUC 1S
LUCB
LUACI
WACB
FMHSEL
SUCB
SUR 1P
WACB
FMHSEL
03
If it is (RDR input (FMH) and RDR input is allowed (BIND) and RDR is in use (SUCB)), or if it is (CONSOLE input (FMH) and (CONSOLE is in use (SUCB)) then:
06
03.1
Reset console input indicator.
Reset reader input indicator.
WACB
WAIC='0'B
WAIR='0'B
04
If it is RDR input then:
A. Indicate RDR input.
WACB
WAIR='1'B
B. Initialize device characteristics ptr.
LUCB
LUPH
C. Call:
INDEV
MG 19
D.
05
04.1
If it is CONSOLE input then:
A. Indicate CONSOLE input.
WACB
WAIC='1'B
B. Indicate CONSOLE TCB.
TCB
TCBTI='CON'
C. Indicate device characteristics ptr.
LUCB
LUPH


Input
Processing
Output
D. Call:
INDEV
MG 19
else
A. Indicate FMH error.
B.
09
RF
x'1008'
WACB
FMHDSS
WASS
05 If begin and end data stream (FMH), and between data stream for current state (case 1+2) then:
A. Indicate end data stream pending.
B.
08
WACB
WASS=X'20'
If begin data stream (FMH) and between data stream for current state (case 3) then:
A. Indicate in data stream state.
B.
08
WACB
WASS=X'40'
else
A. Indicate FMH error
B.
08
RF
X'1008'
R4
LUCB
LUIS
06 Devices (RDR or CONSOLE) already in use:
061 If IB processor (RDR/CONS) not on same session then:
A. Indicate FMH error.
B.
08
RF
X'1008'
DEVCHAR
WAPH
PTRLUCB


Input
Processing
Output
WACB
FMHDSS
WASS
WACB
FMHSEL
WACB
FMHDSS
WASS
WAIR
LUCB
LUIS
WACB
FMHDSS
WASS
WAIC
LUCB
LUIS
Else
If end data stream (FMH), and in data stream for current state (case 5) then:
A. Indicate end data stream pending.
B.
If abort data stream (FMH), and in data stream for current state (case 6) then:
A. Indicate abort data stream.
B.
07 If it is CONSOLE input then:
A. Indicate FMH error.
B.
If suspend data stream (FMH), and in data stream for current state, and IB RDR active, and IB not interrupted (case 7) then:
A. Indicate suspend data stream.
B.
If resume data stream (FMH), and between data stream for current state, and IB console active, and IB interrupted (case 4) then:
WACB
WASS=X'20'
08
WACB
WASS=X'40'
08
RF
X'1008'
08
WACB
WASS=X'08'
08


Input
Processing
Output
A. Indicate resume IB RDR requested.
B. Indicate in data stream state for suspended (interrupted) IB RDR.
Else indicate FMH error.
08 Update residual count and update current buffer pointer.
If the residual count is negative then:
Set residual count to zero.
If there is an error condition then:
08.1 If reader input and if the logical reader is not open:
Issue the IPW$OLI macro to open the logical interface.
Indicate that the logical interface has been opened.
Reset Reg.1 to point to the RPL again.
09 Exit.
WACB
WARC
RF
WACB
WAIR
WALI
WACB
WARP
WACB
WASR='1'B
WACB
WASS=X'40'
LUW1
RF
X'1008'
WACB
WARC - FMHLN
WACR + FMHLN
WARC='0'B
09
IPW$OLI
WALI='1'B
R1
WARP
RETURN TO CALLING ROUTINE

Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(03) If it is not the first FMH which is arrived by this procedure the branch is made. LUCI indicates, that card inbound is allowed. It will be set during logon according to the BIND - parameters. (03.1) It is a BDS or BDS/EDS FMH, because RDR or CONSOLE has not been active. (04.1) In case of console input, the TCB indentification field (TCBTI) will be changed from RDR1 to CON1. For implicit function management headers (FMH) it will be done in subroutine STAT. (05) The case number indentifies an entry in the atached decision table, chart MG 09. Case 2 is also verified by subroutine INDEV (no second RDR allowed, that means it is not possible to interrupt IB rdr for IB rdr) (06) In this case LUPH also points to the appopriate device characteristics field (SUCB). (06.1) R4 always points to the LUCB on which the IB processor TCB is associated. PTRLUCB must also point to the same LUCB. PTRLUCB references the pointer to the LUCB in the SUCB device characteristies field. (07) It is not allowed to interrupt IB console. A check must be made, if a second device (RDR or CONSOLE) tried to send data. When this happened, the device was already started and a BDS or BDS, EDS was sent.			

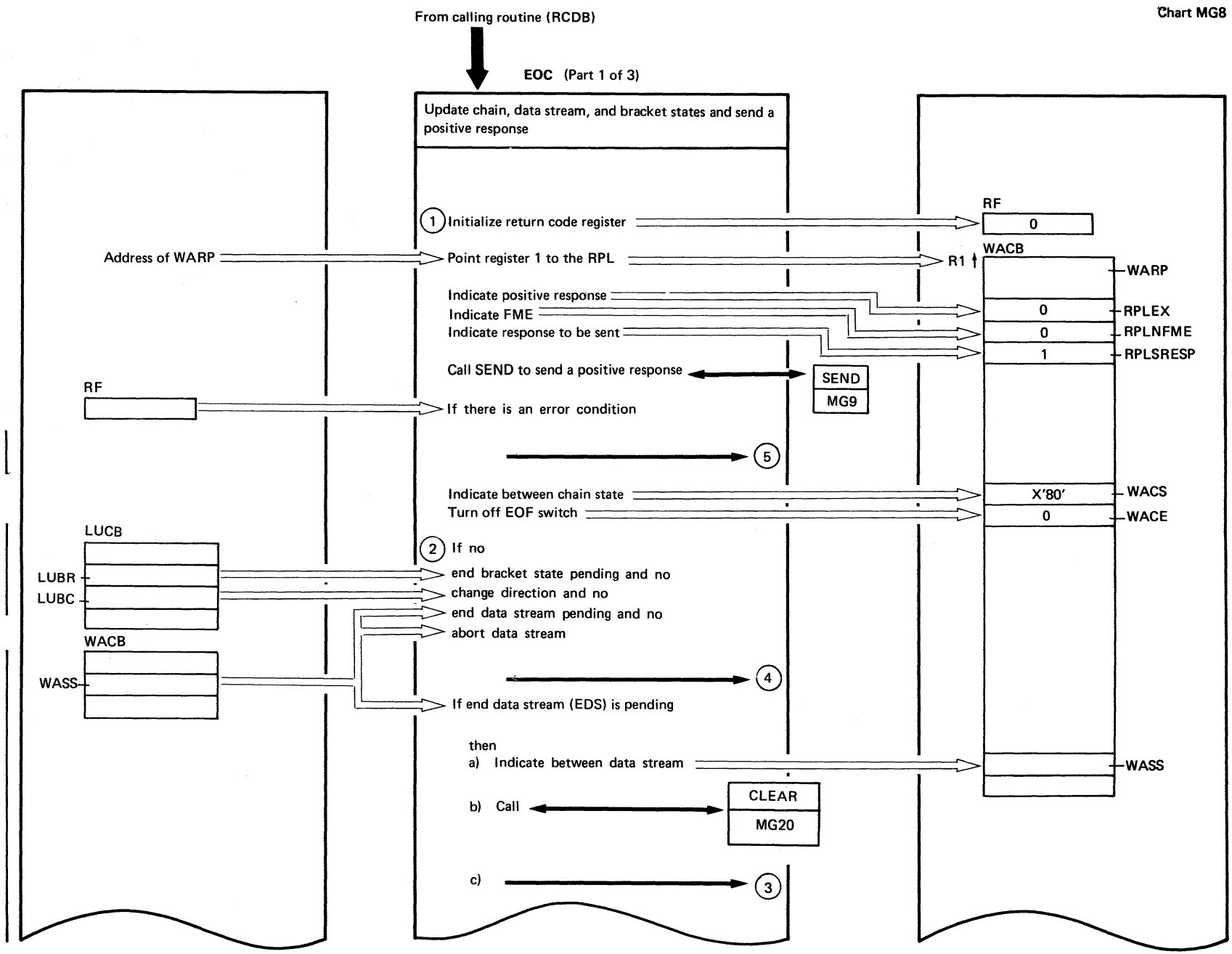
Extended Description

Include Segment	Call Subroutine/ Macro	Chart
IPW$OLI POWER/VS		

(081)

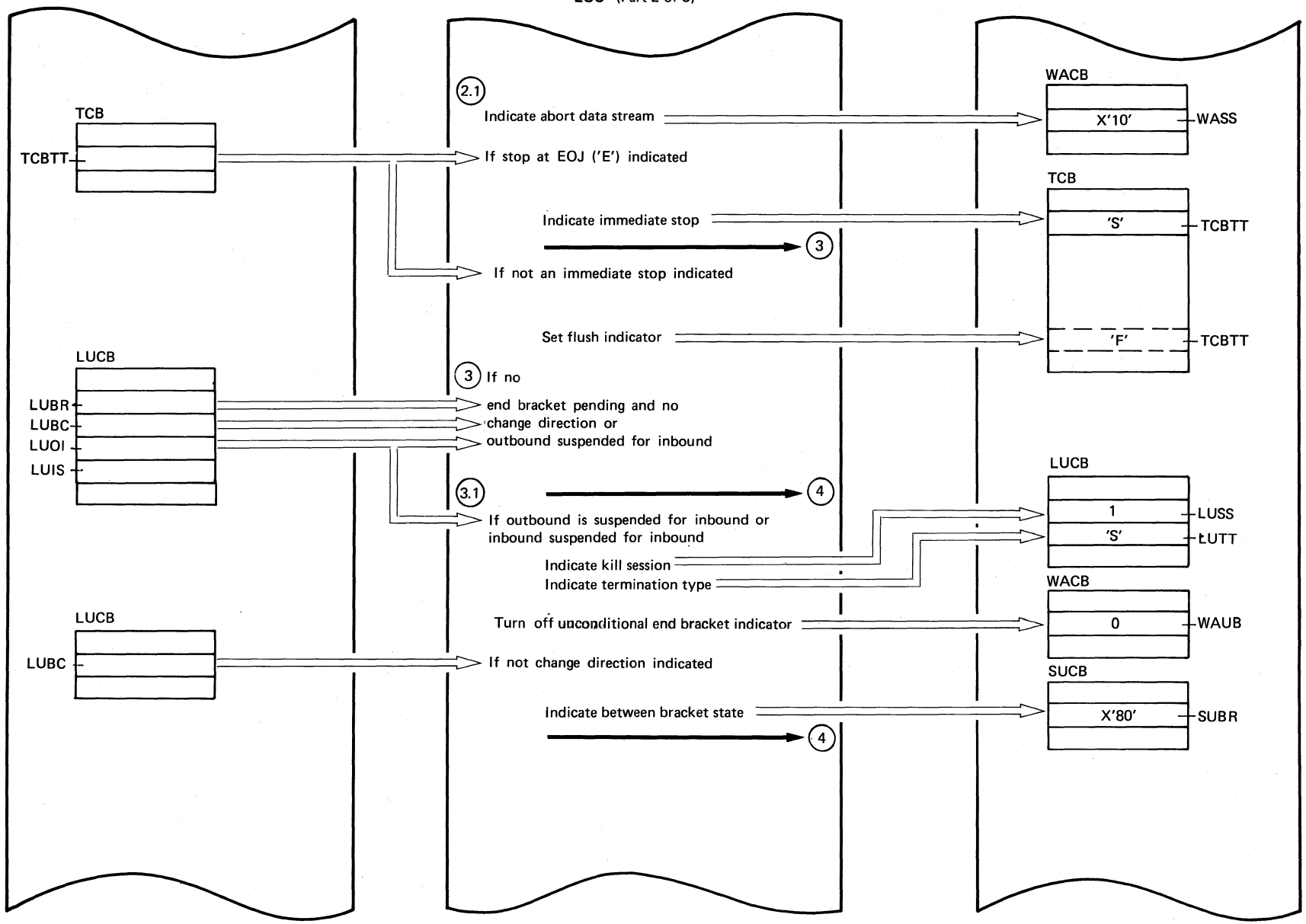
FMH - FIELD	CONDITIONS	DATA STREAM STATE (WASS) B	B	B	B	I	I	I	
FMH - DSS	BDS ('010'B)	ON	ON	ON	OFF	OFF	OFF	OFF	ANY OTHER COMBI-NATION OF CONDI-TIONS
	EDS ('001'B)	ON	ON	OFF	OFF	ON	OFF	OFF	
	ADS ('101'B)	OFF	OFF	OFF	OFF	OFF	ON	OFF	
	SDS ('100'B)	OFF	OFF	OFF	OFF	OFF	OFF	ON	
	RDS ('000'B)	OFF	OFF	OFF	ON	OFF	OFF	OFF	
FMH - MEDIA	RDR INPUT ('010'B)	--	NO	YES	YES	YES	YES	YES	
	CONS.INP. ('000'B)	--	YES	NO	NO	NO	NO	NO	
1)	LUIS	OFF	ON	OFF	ON	---	OFF	OFF	
ACTIONS									
ERROR		NO	NO	NO	NO	NO	NO	NO	YES
NEW DS STATE (WASS)		EP	EP	1	B/R	EP	A	S	= OLD STATE
LUIS - SEE NOTE 1)		OFF	ON	OFF	ON	---	OFF	ON	UNCHANGED
CASE NR. (REFER. IN CODE)		1	2	3	4	5	6	7	

NOTES:
1) LUIS=ON ...INDICATES $IB-RDR INTERRUPTED (SUSPENDED) FOR $IB-CONSOLE (IN LUCB)
2) EP.....END STATE PENDING
B/R ... BETWEEN DS STATE AND 'RESUME REQUEST' INDICATOR SET ON (WASR IN WACB)
B... BETWEEN DS STATE
I... IN DS STATE
A.. ABORT DS STATE
S... SUSPENDED (INTERRUPTED) DS STATE



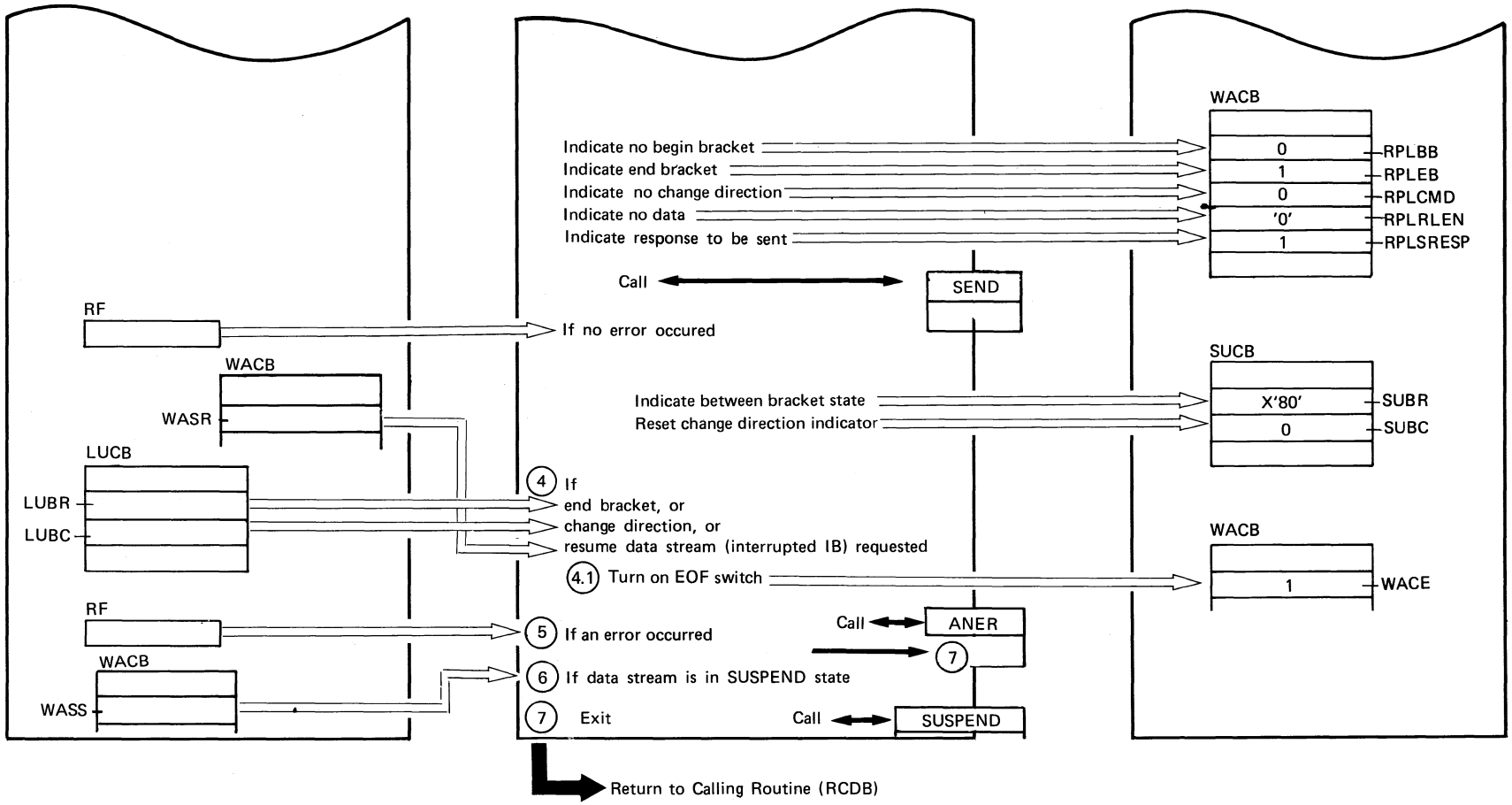
Continued on next page

EOC (Part 2 of 3)

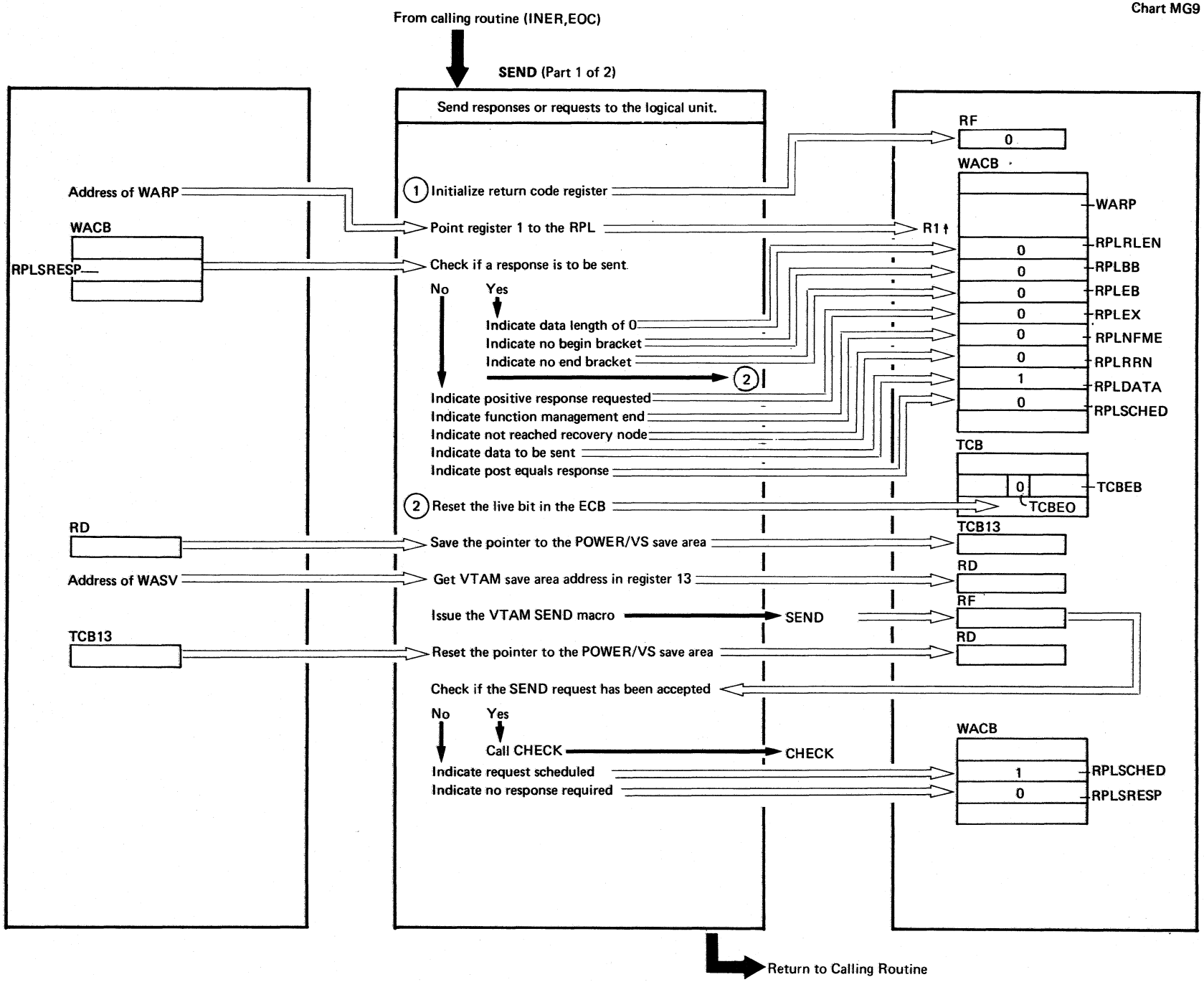


Continued on next page

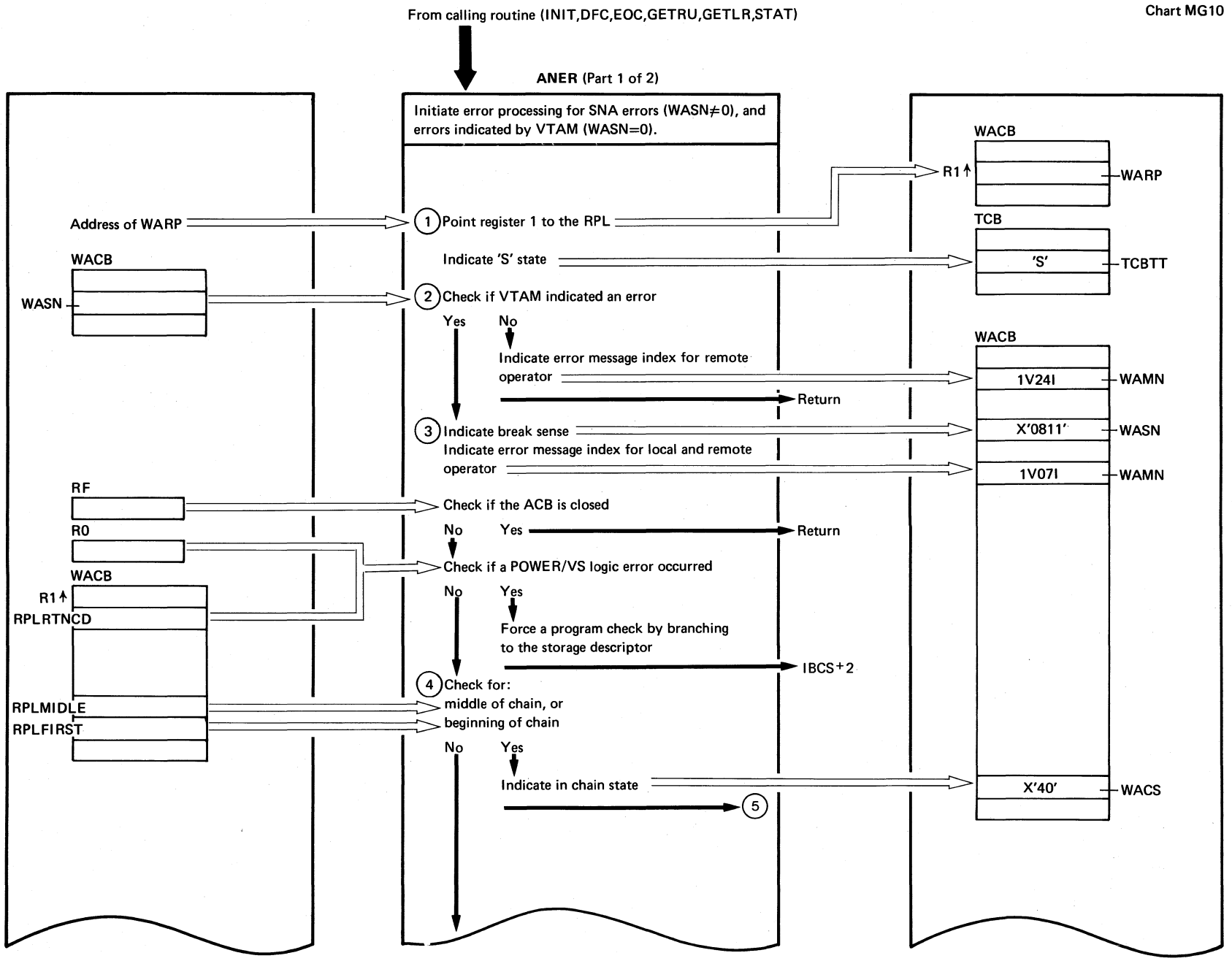
EOC (Part 3 of 3)



Extended Description	Include Segment	Call Subroutine/Macro	Chart
(2.1) If a data stream has to be aborted, no QUEUE record should be written. This is indicated with 'stop immediate' or 'flush' depending on the current condition of the task		SEND	MG9
(3.1) If an EB is pending and a reader or writer is interrupted the session must be kulled. The WST sent an EB and therefore did not follow the SNA-rules			
(4.1) The EOF switch is to come cut of the RCDB routine		SUSPEND ANER	MG18 MG10

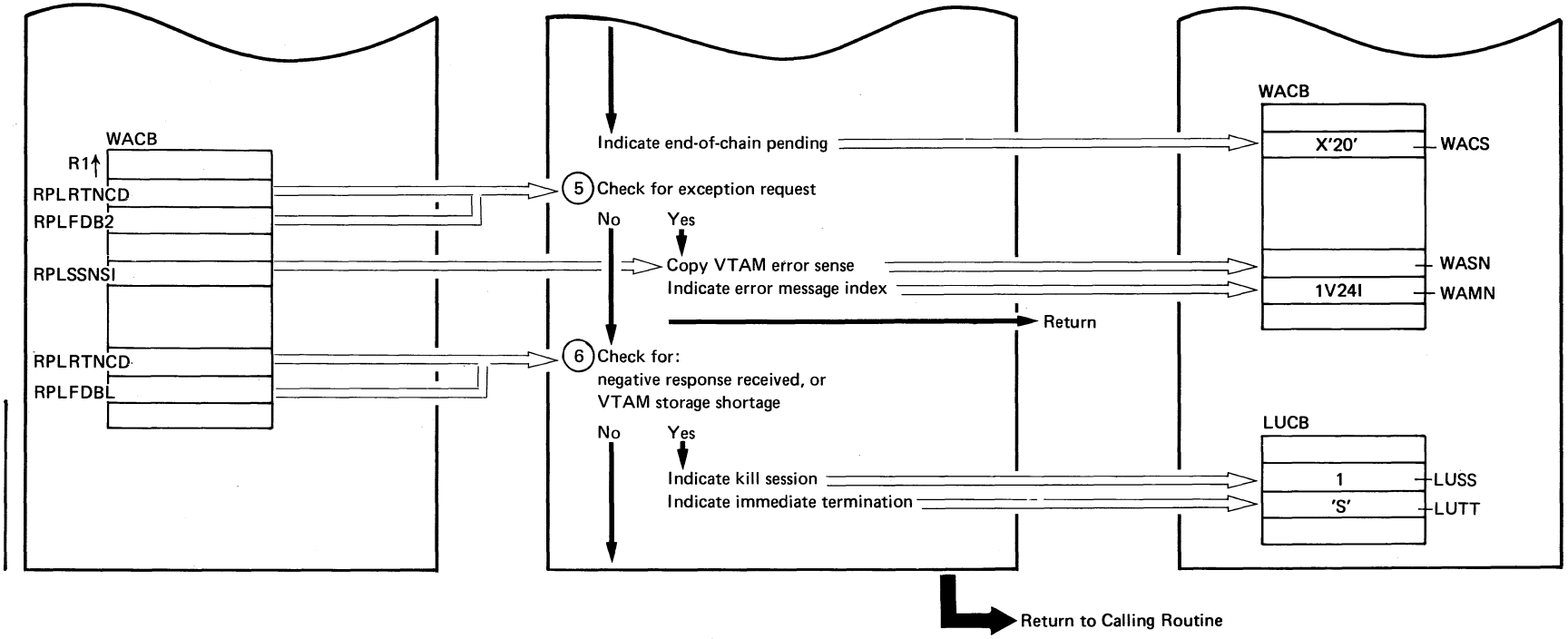

From calling routine (INER,EOC)
SEND (Part 1 of 2)
Send responses or requests to the logical unit.
Address of WARP
WACB
RPLSRESP
RD
Address of WASV
TCB13
1 Initialize return code register
Point register 1 to the RPL
Check if a response is to be sent
No
Yes
Indicate data length of 0
Indicate no begin bracket
Indicate no end bracket
2
Indicate positive response requested
Indicate function management end
Indicate not reached recovery node
Indicate data to be sent
Indicate post equals response
2 Reset the live bit in the ECB
Save the pointer to the POWER/VS save area
Get VTAM save area address in register 13
Issue the VTAM SEND macro
SEND
Reset the pointer to the POWER/VS save area
Check if the SEND request has been accepted
No
Yes
Call CHECK
CHECK
Indicate request scheduled
Indicate no response required
RF
0
WACB
WARP
R1
RPLRLEN
0
RPLBB
0
RPLEB
0
RPLEX
0
RPLNFME
0
RPLRRN
0
RPLDATA
1
RPLSCHED
0
TCB
TCBEB
0
TCBEO
TCB13
RD
RF
RD
WACB
RPLSCHED
1
RPLSRESP
0
Return to Calling Routine

Extended Description	Include Segment	Call Subroutine/Macro	Chart
(2)		SEND (VTAM) CHECK	MG16

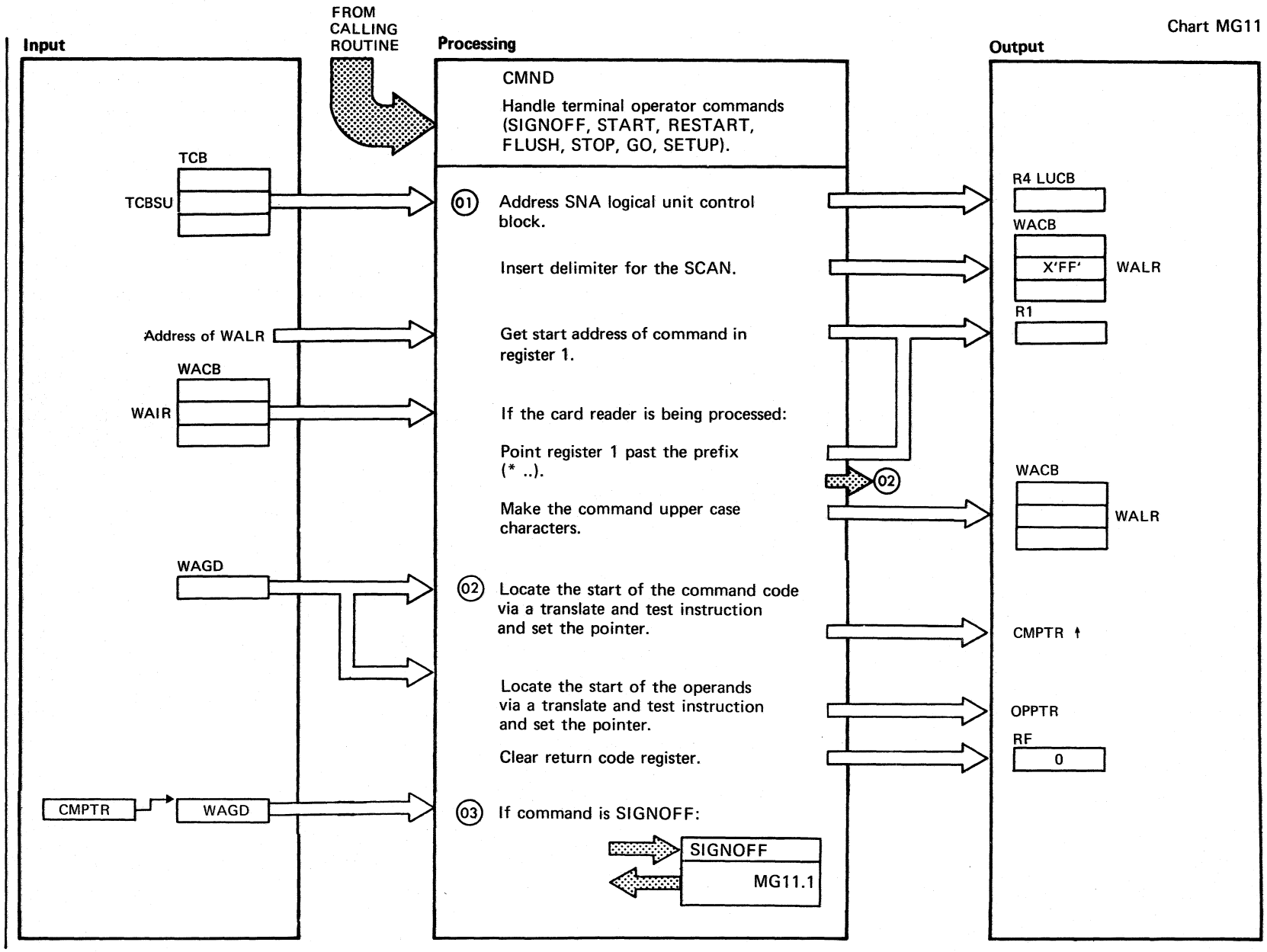


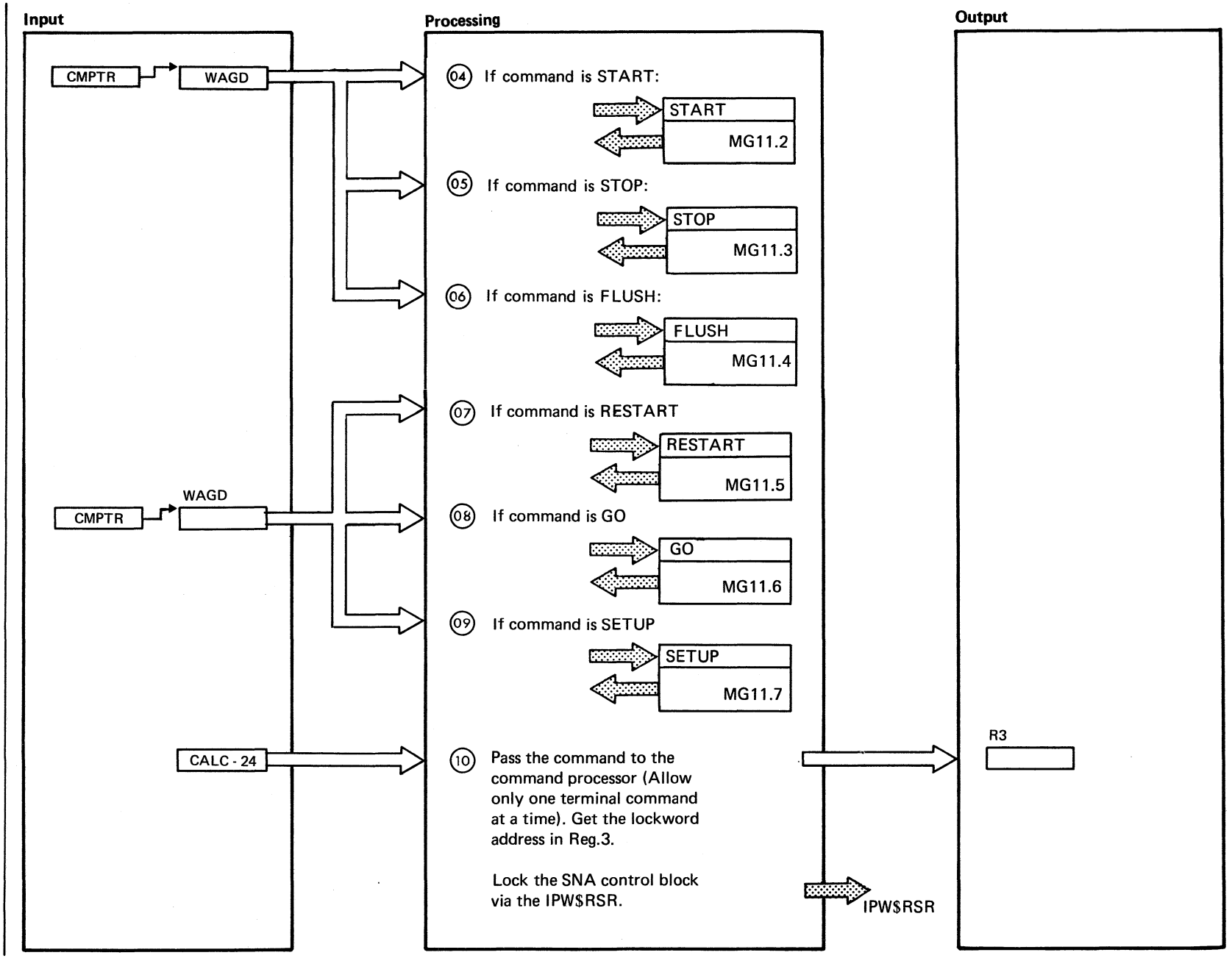
(Continued on next page)

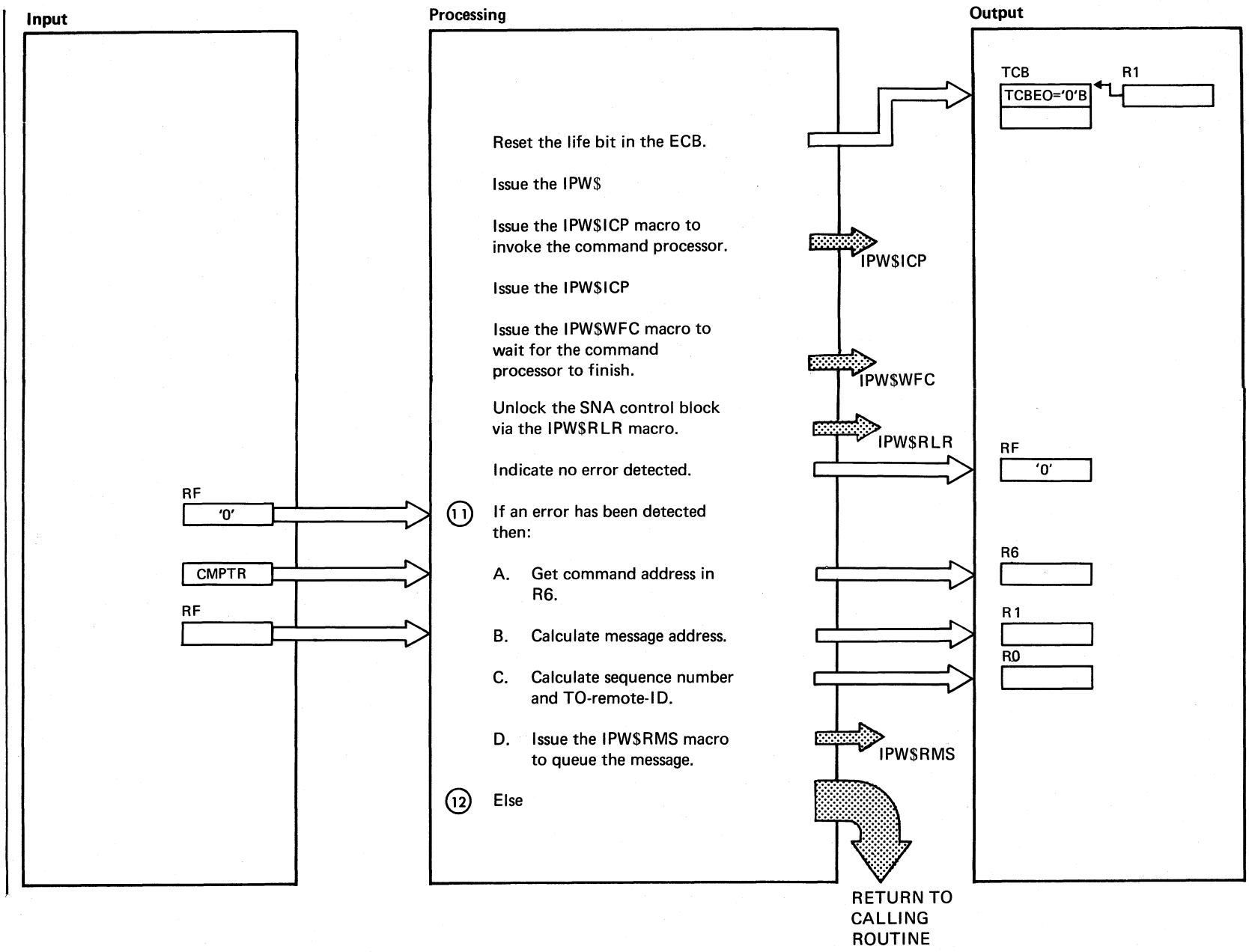
ANER (Part 2 of 2)



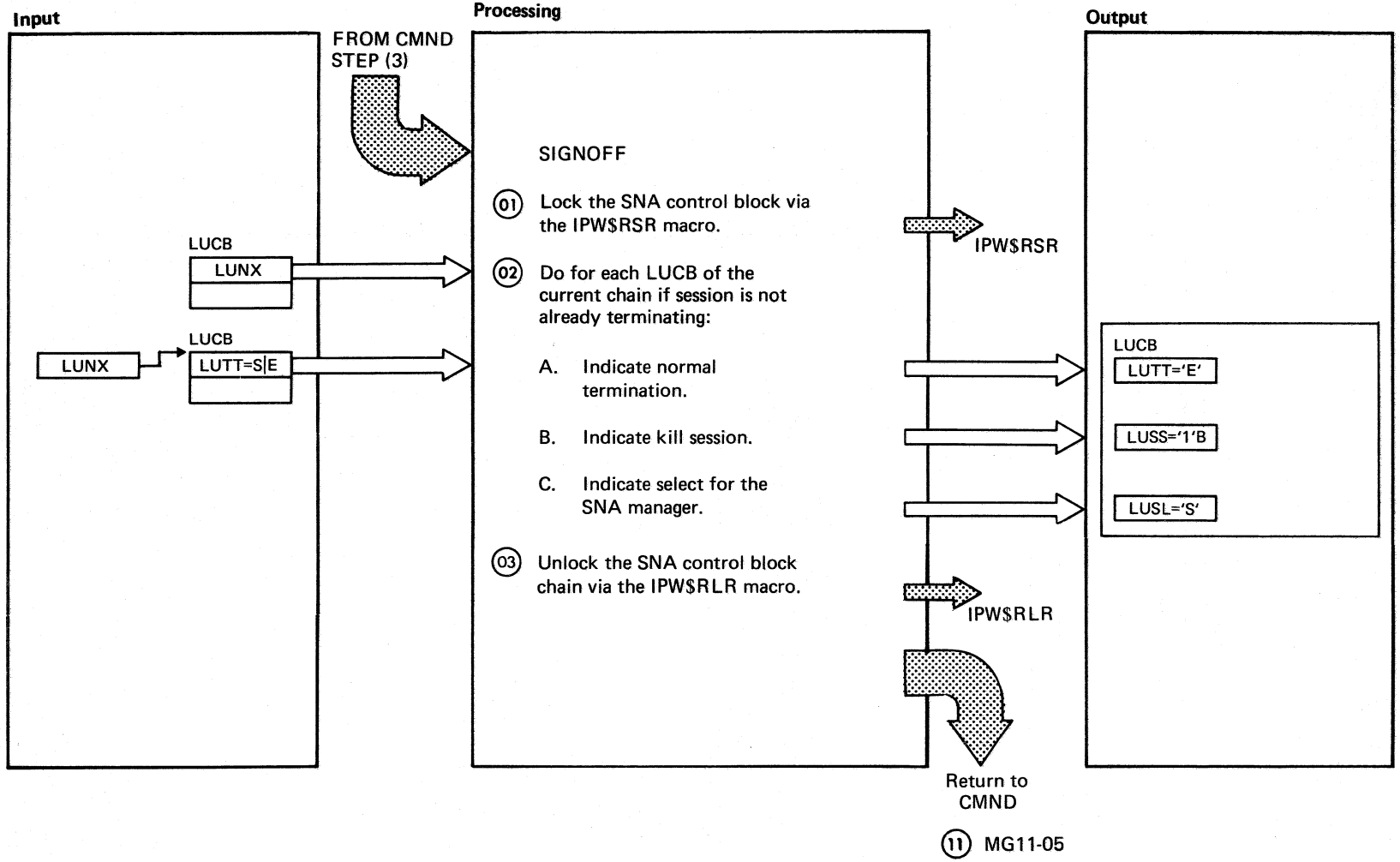
Extended Description	Include Segment	Call Subroutine/Macro	Chart
(2) 1V24I ttt TERMINATED REASON=xxxx			
(3) 1V07I ERROR ON request RTNCD,FDB2=xx,xx SENSE=xxxx ON luname			
(5) 1V24I ttt TERMINATED REASON=xxxx			


Input
FROM CALLING ROUTINE
Processing
Output
CMND
Handle terminal operator commands (SIGNOFF, START, RESTART, FLUSH, STOP, GO, SETUP).
TCB
TCBSU
01
Address SNA logical unit control block.
R4 LUCB
WACB
Insert delimiter for the SCAN.
X'FF'
WALR
R1
Address of WALR
Get start address of command in register 1.
WACB
WAIR
If the card reader is being processed:
Point register 1 past the prefix (* ..).
02
WACB
Make the command upper case characters.
WALR
WAGD
02
Locate the start of the command code via a translate and test instruction and set the pointer.
CMPTR
Locate the start of the operands via a translate and test instruction and set the pointer.
OPPTR
RF
Clear return code register.
0
CMPTR
WAGD
03
If command is SIGNOFF:
SIGNOFF
MG11.1

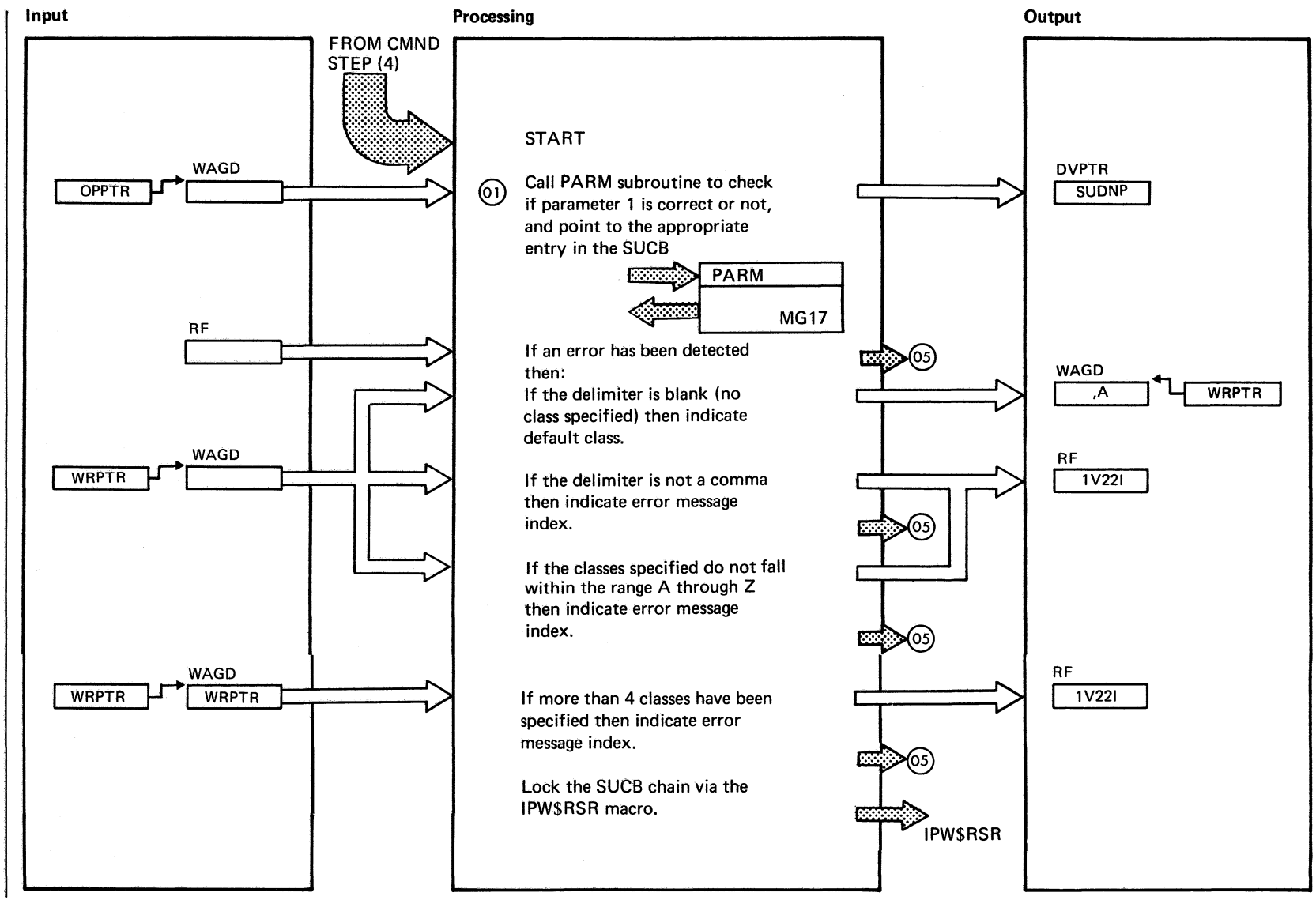

Input
Processing
Output
CMPTR
WAGD
04 If command is START:
START
MG11.2
05 If command is STOP:
STOP
MG11.3
06 If command is FLUSH:
FLUSH
MG11.4
07 If command is RESTART
RESTART
MG11.5
WAGD
CMPTR
08 If command is GO
GO
MG11.6
09 If command is SETUP
SETUP
MG11.7
CALC - 24
10 Pass the command to the command processor (Allow only one terminal command at a time). Get the lockword address in Reg.3.
Lock the SNA control block via the IPW$RSR.
IPW$RSR
R3

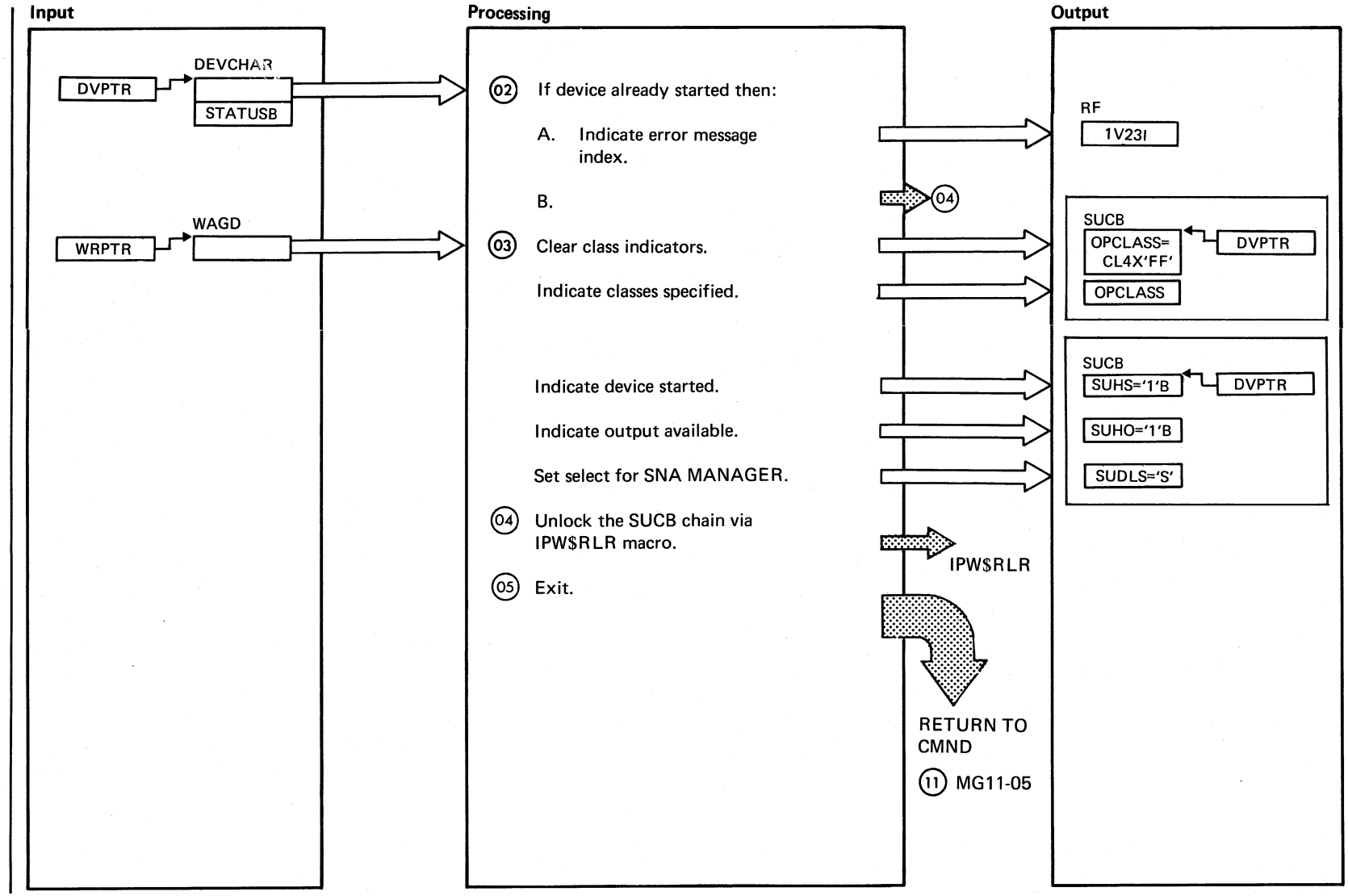

Input
Processing
Output
TCB
R1
TCBEO='0'B
Reset the life bit in the ECB.
Issue the IPW$
Issue the IPW$ICP macro to invoke the command processor.
IPW$ICP
Issue the IPW$ICP
Issue the IPW$WFC macro to wait for the command processor to finish.
IPW$WFC
Unlock the SNA control block via the IPW$RLR macro.
IPW$RLR
RF
'0'
Indicate no error detected.
RF
'0'
(11)
If an error has been detected then:
CMPTR
R6
A. Get command address in R6.
RF
R1
B. Calculate message address.
R0
C. Calculate sequence number and TO-remote-ID.
D. Issue the IPW$RMS macro to queue the message.
IPW$RMS
(12)
Else
RETURN TO CALLING ROUTINE

Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(03) Signoff			MG 11.1
(04) START			MG 11.2
(05) STOP			MG 11.3
(06) FLUSH			MG 11.4
(07) RESTART			MG 11.5
(08) GO			MG 11.6
(09) SETUP			MG 11.7
(10)	IPW$RSR (POWER/ VS) IPW$ICP (POWER/ VS) IPW$WFC (POWER/ VS) IPW$RLR (POWER/ VS) IPW$RMS (POWER/ VS)		



Extended Description	Include Segment	Call Subroutine/ Macro	Chart
01	IPW$RSR (POWER/VS)		
03	IPW$RLR (POWER/VS)		


Input
Processing
Output
FROM CMND STEP (4)
START
OPPTR
WAGD
01
Call PARM subroutine to check if parameter 1 is correct or not, and point to the appropriate entry in the SUCB
PARM
MG17
DVPTR
SUDNP
RF
If an error has been detected then:
If the delimiter is blank (no class specified) then indicate default class.
05
WAGD
,A
WRPTR
WRPTR
WAGD
If the delimiter is not a comma then indicate error message index.
RF
1V22I
If the classes specified do not fall within the range A through Z then indicate error message index.
WRPTR
WAGD
WRPTR
If more than 4 classes have been specified then indicate error message index.
RF
1V22I
Lock the SUCB chain via the IPW$RSR macro.
IPW$RSR


Input
Processing
Output
DEVCHAR
DVPTR
STATUSB
WAGD
WRPTR
02 If device already started then:
A. Indicate error message index.
B.
03 Clear class indicators.
Indicate classes specified.
Indicate device started.
Indicate output available.
Set select for SNA MANAGER.
04 Unlock the SUCB chain via IPW$RLR macro.
05 Exit.
04
IPW$RLR
RETURN TO CMND
11 MG11-05
RF
1V231
SUCB
OPCLASS= CL4X'FF'
DVPTR
OPCLASS
SUCB
SUHS='1'B
DVPTR
SUHO='1'B
SUDLS='S'

Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(01) The device ptr (DVPTR) points to the appropriate entry in the SUCB. It is calculated by the subrtn PARM. SUDNP d. . . device (LST/PUN) n. . . device number LST: n between 1 and 3 PUN: n = 1 e.g. SUL2P is first byte of LST2 characteristic field in SUCB. 1V22I INVALID COMMAND 1V23I COMMAND OUT OF SEQUENCE	IPW$RSR (POWER/ VS)		
(02) STATUSBYTE (STATUSB) based on DVPTR appropriates to SUDNS. d. . . device (LST/PUN) n. . . device number			
(03) The classes and the indicators are set in the device characteristics area in the SUCB. This area is pointed to by DVPTR. OPCLASS is set according to SUDNC.			
(04)	IPW$RLR (POWER/ VS)		

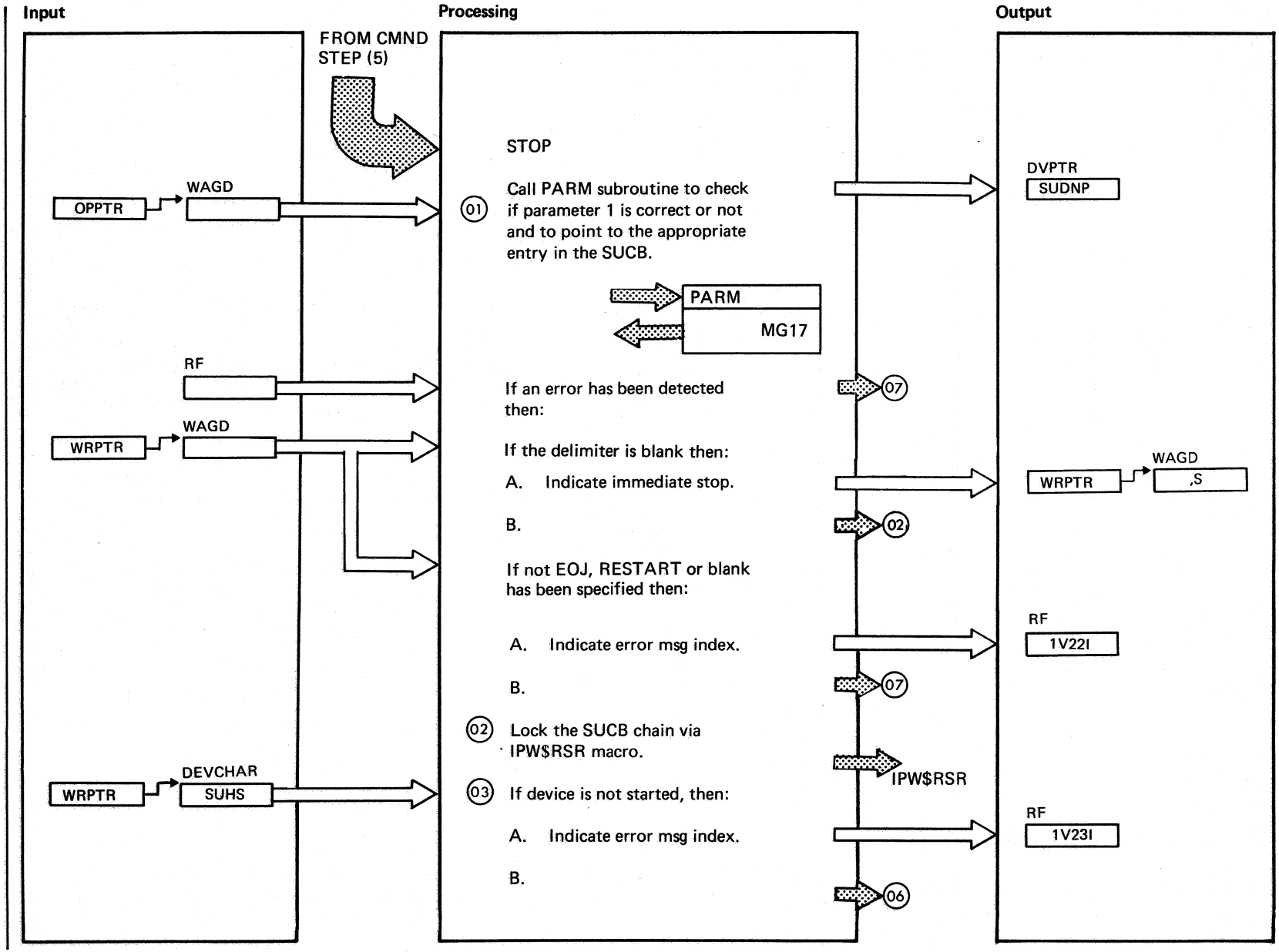
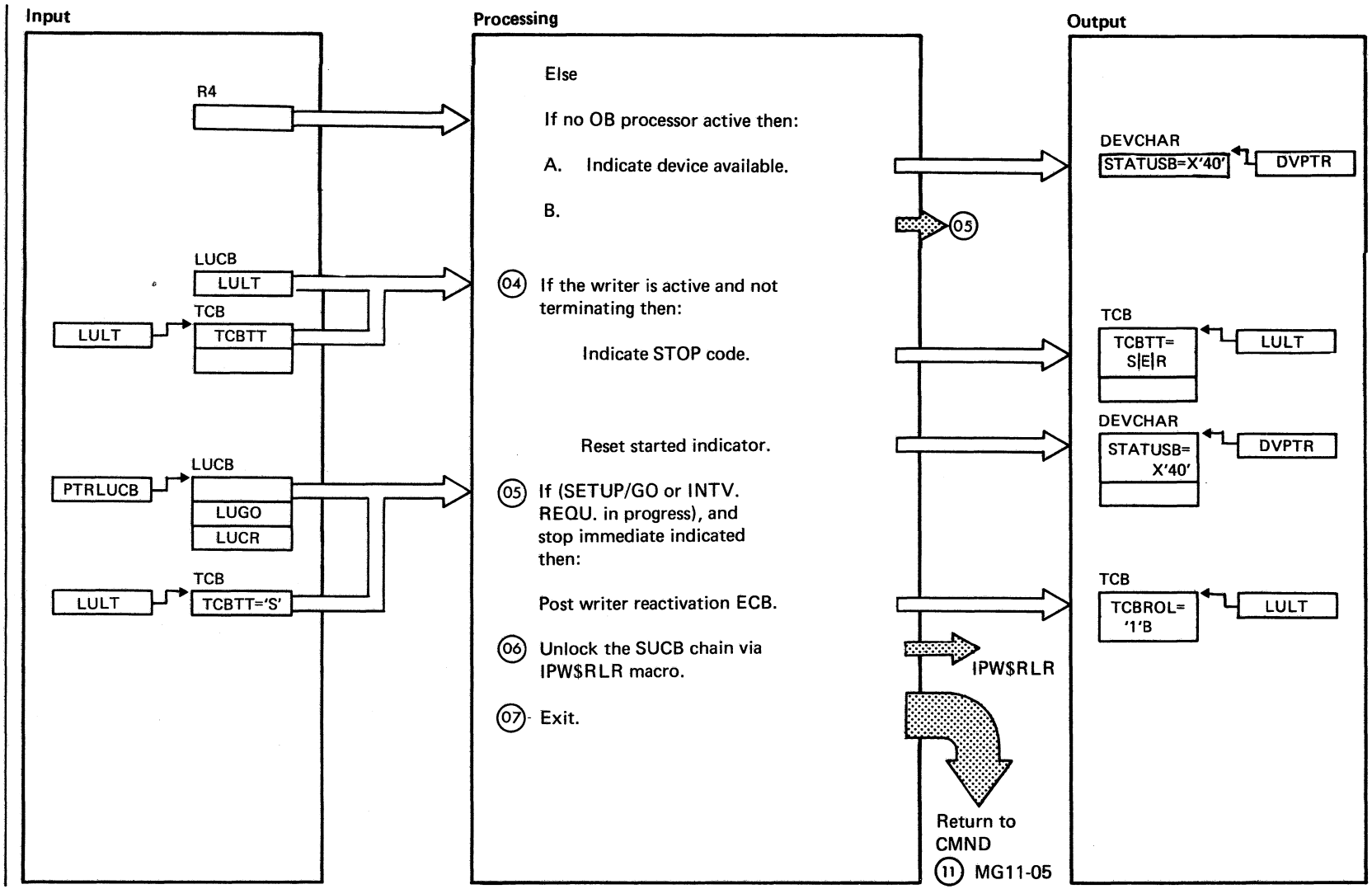
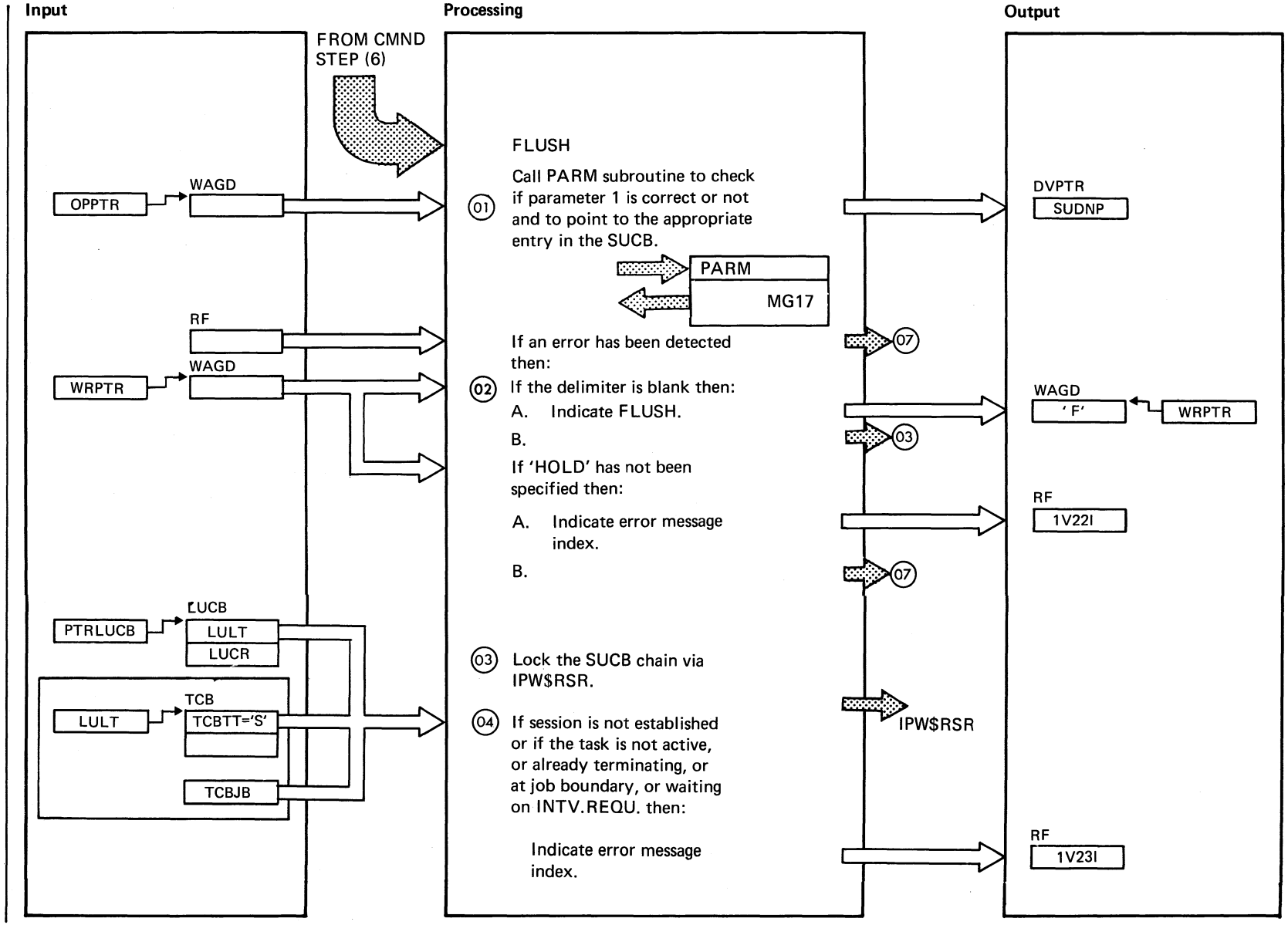
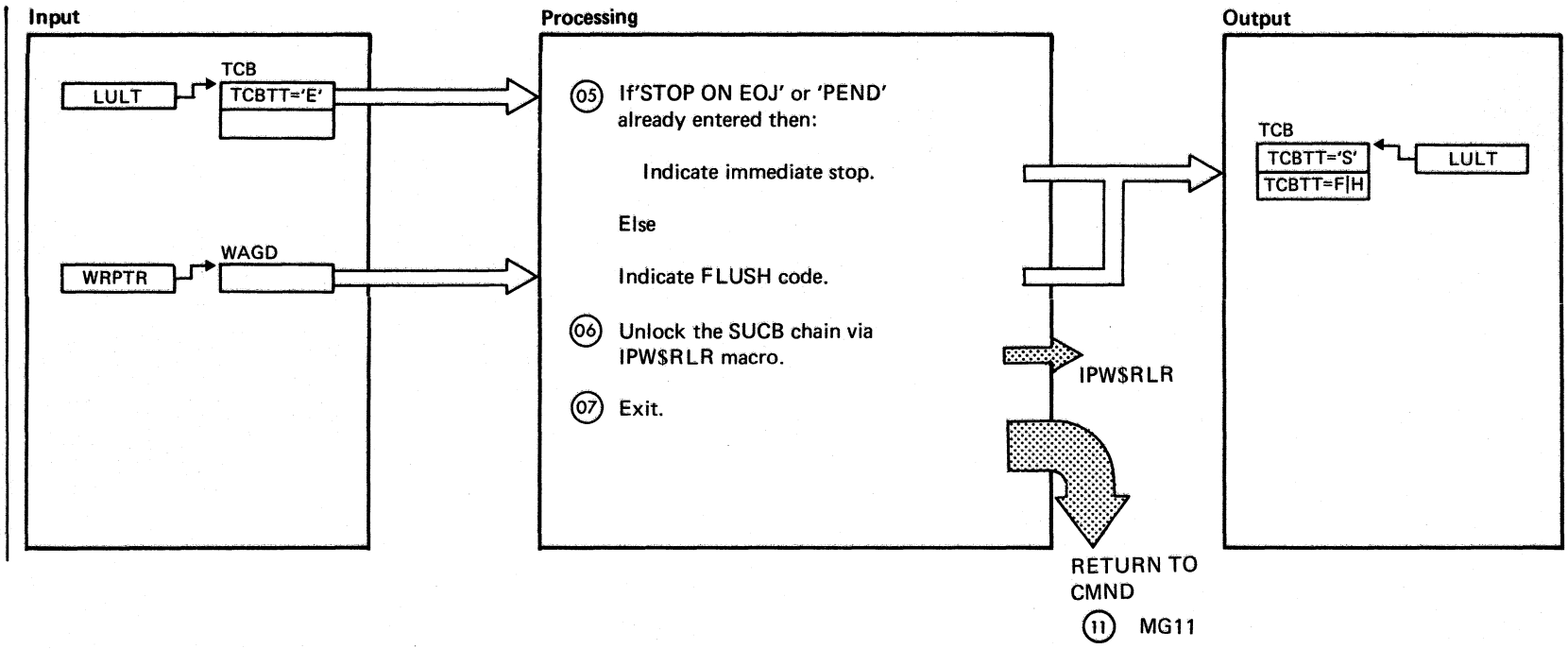


Chart MG11.3
Part 1 of 3


Input
Processing
Output
R4
LUCB
LULT
TCB
LULT
TCBTT
LUCB
PTRLUCB
LUGO
LUCR
TCB
LULT
TCBTT='S'
Else
If no OB processor active then:
A. Indicate device available.
B.
05
04 If the writer is active and not terminating then:
Indicate STOP code.
Reset started indicator.
05 If (SETUP/GO or INTV. REQU. in progress), and stop immediate indicated then:
Post writer reactivation ECB.
06 Unlock the SUCB chain via IPW$RLR macro.
07 Exit.
IPW$RLR
Return to CMND
11 MG11-05
DEVCHAR
STATUSB=X'40'
DVPTR
TCB
TCBTT= S|E|R
LULT
DEVCHAR
STATUSB= X'40'
DVPTR
TCB
TCBROL= '1'B
LULT

Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(01) The device ptr (DVPRT) points to the appropriate entry in the SUCB. It is calculated by the subroutine PARM. SUDNP d . . . device (LST/PUN) n . . . device number LST: n between 1 and 3 PUN: n = 1 e.g. SUL2P is first byte of LST2 characteristic field in SUCB.			MG 17
(02)	IPW$RSR (POWER/ VS)		
(03) STATUSBYTE (STATUSB) based on DVPTR appropriates to SUDNS. d . . . device (LST/PUN) n . . . device number			
(04) The writer task is only active, when LULT not equal '0' 1V22I INVALID COMMAND. 1V23I COMMAND OUT OF SEQUENCE.			
(05) Only when immediate STOP is indicated, the OB-processor must be posted. Otherwise the $OB processor goes immediate in the wait state without any action taken.			
(06)	IPW$RLR (POWER/ VS)		


Input
Processing
Output
FROM CMND STEP (6)
OPPTR
WAGD
RF
WRPTR
WAGD
PTRLUCB
LUCB
LULT
LUCR
LULT
TCB
TCBTT='S'
TCBJB
FLUSH
01 Call PARM subroutine to check if parameter 1 is correct or not and to point to the appropriate entry in the SUCB.
PARM
MG17
If an error has been detected then:
02 If the delimiter is blank then:
A. Indicate FLUSH.
B.
If 'HOLD' has not been specified then:
A. Indicate error message index.
B.
03 Lock the SUCB chain via IPW$RSR.
04 If session is not established or if the task is not active, or already terminating, or at job boundary, or waiting on INTV.REQU. then:
Indicate error message index.
07
03
07
IPW$RSR
DVPTR
SUDNP
WAGD
'F'
WRPTR
RF
1V22I
RF
1V23I


Input
Processing
Output
LULT
TCB
TCBTT='E'
WRPTR
WAGD
05 If 'STOP ON EOJ' or 'PEND' already entered then:
Indicate immediate stop.
Else
Indicate FLUSH code.
06 Unlock the SUCB chain via IPW$RLR macro.
07 Exit.
IPW$RLR
RETURN TO CMND
11 MG11
TCB
TCBTT='S'
TCBTT=F|H
LULT

Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(01) The device ptr (DVPTR) points to the appropriate entry in the SUCB. It is calculated by the subroutine PARM. SUDNP d . . . device (LST/PUN) n . . . device number LST: n between 1 and 3 PUN: n = 1 e.g. SUL2P is first byte of LST2 characteristic field in SUCB. (02) Valid command forms are e.g. F LST, Hold F LST2 - - - > F LST, F will be initialized. (03) (04) A writer is active, when a TCB is existing. That is also true, when the task is suspended. 1V22I INVALID COMMAND. 1V23I COMMAND OUT OF SEQUENCE. (05) If the PROCESSOR is in a STOP at EOJ condition, the task must be set into 'S' - state to simulate a flush. (06)	IPW$RSR (POWER/ VS) IPW$RLR (POWER/ VS)		

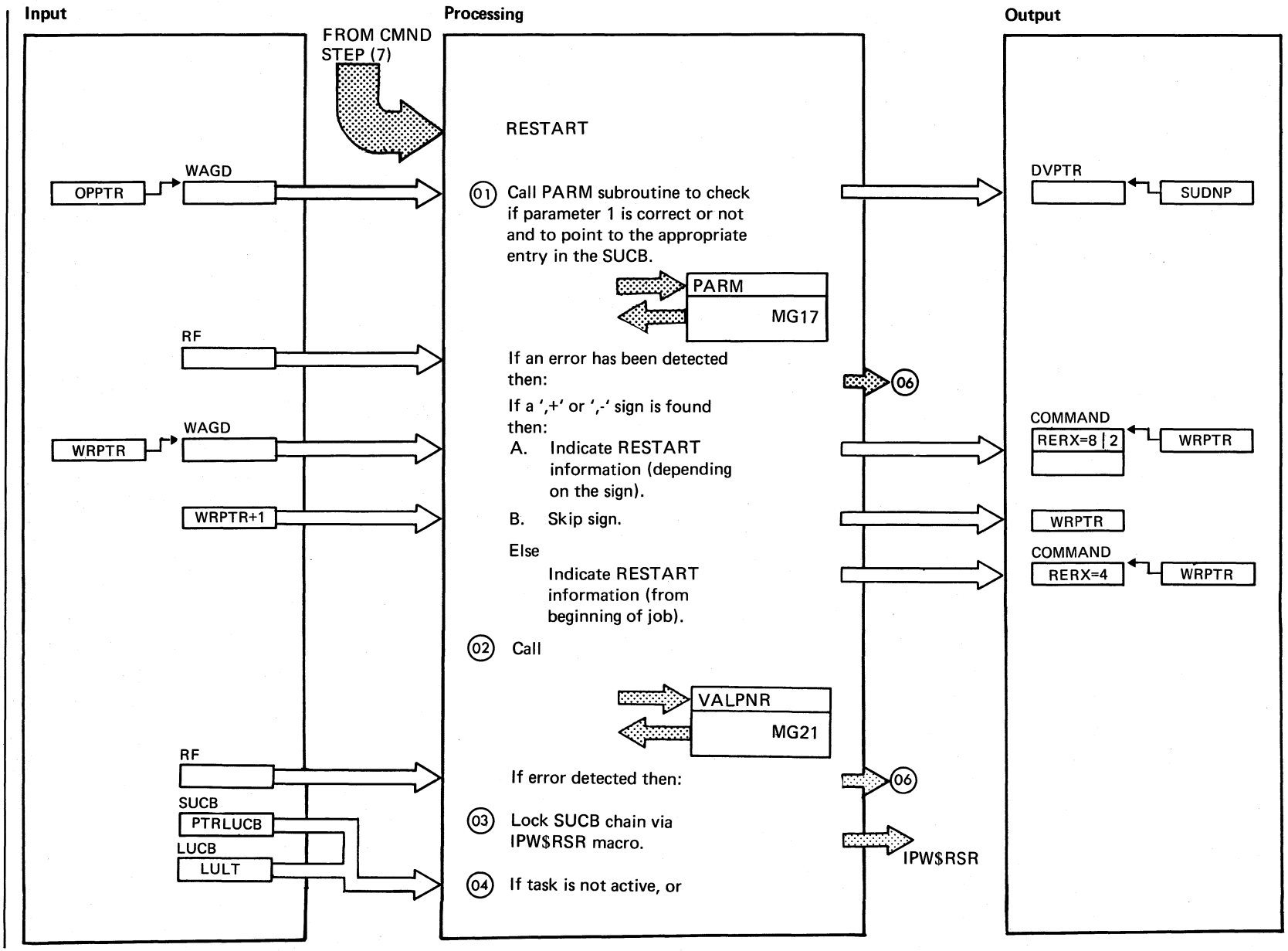
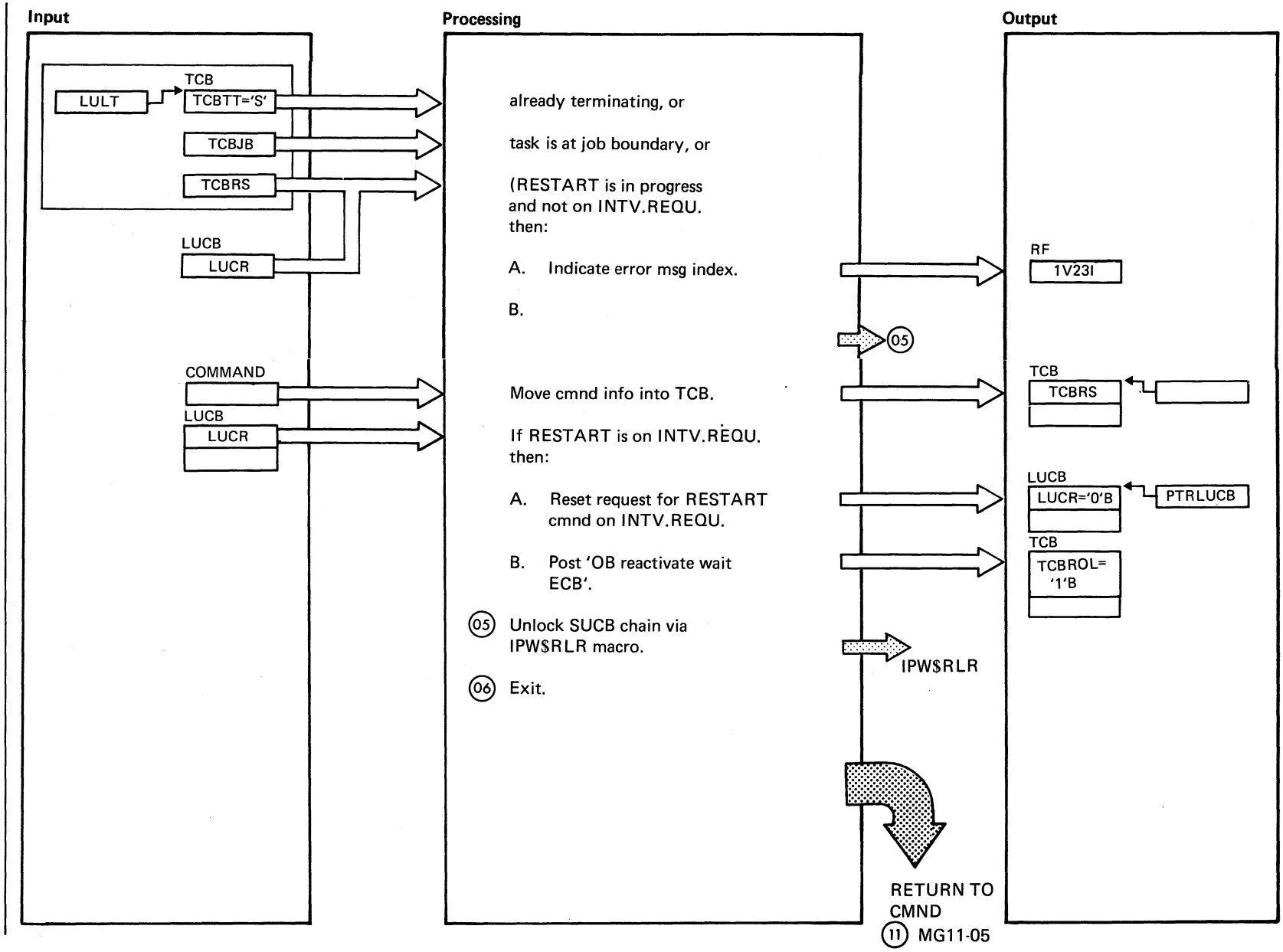
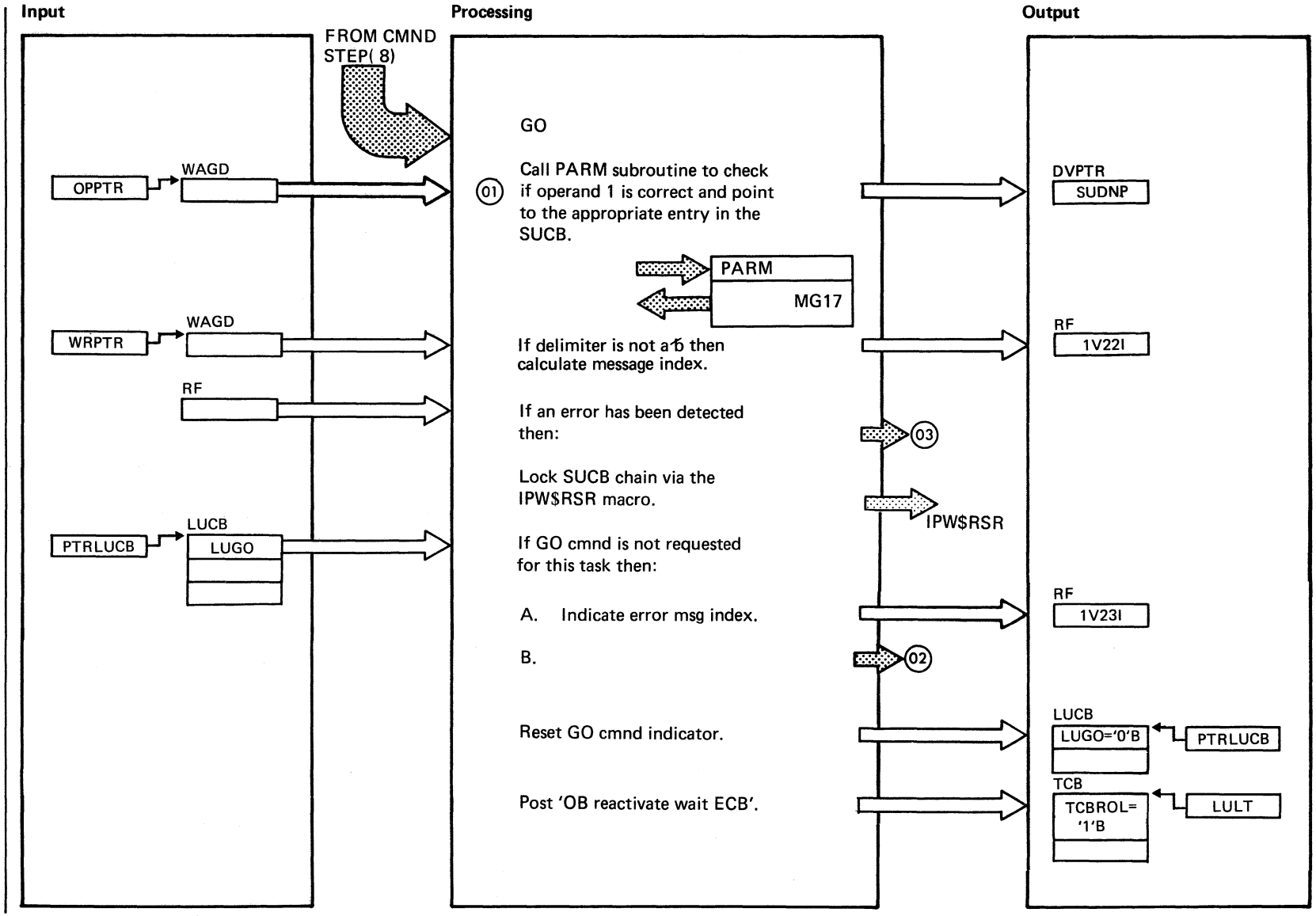
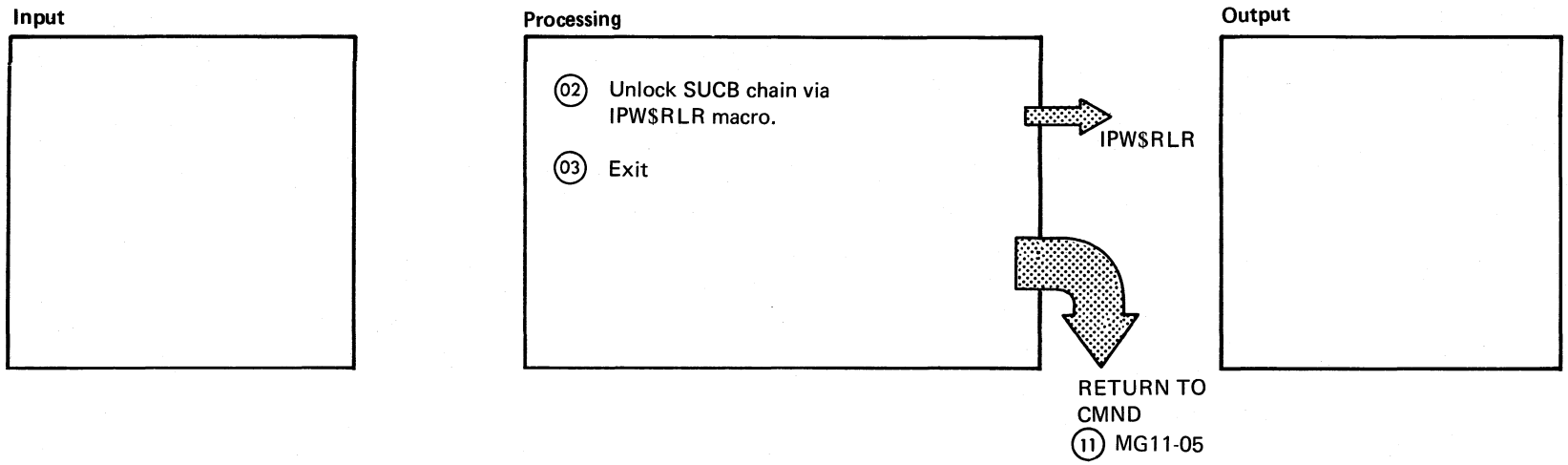


Chart MG11.5
Part 1 of 3


Input
Processing
Output
TCB
LULT
TCBTT='S'
TCBJB
TCBRS
LUCB
LUCR
COMMAND
LUCB
LUCR
already terminating, or
task is at job boundary, or
(RESTART is in progress
and not on INTV.REQU.
then:
A. Indicate error msg index.
B.
05
Move cmnd info into TCB.
If RESTART is on INTV.REQU.
then:
A. Reset request for RESTART
cmnd on INTV.REQU.
B. Post 'OB reactivate wait
ECB'.
05 Unlock SUCB chain via
IPW$RLR macro.
06 Exit.
RF
1V23I
TCB
TCBRS
LUCB
LUCR='0'B
PTRLUCB
TCB
TCBROL=
'1'B
IPW$RLR
RETURN TO
CMND
11 MG11-05

Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(01) The device pointer (DVPTR) points to the appropriate entry in the SUCB. It is calculated by the subroutine PARM. SUDNP d . . . device (LST/PUN) n . . . device number LST: n between 1 and 3 PUN: n = 1 e.g. SUL2P is first byte of LST2 characteristic field in SUCB. (02) Page number must be between 0 and 9999.			MG 17


Input
Processing
Output
FROM CMND STEP(8)
GO
OPPTR
WAGD
01
Call PARM subroutine to check if operand 1 is correct and point to the appropriate entry in the SUCB.
PARM
MG17
DVPTR
SUDNP
WRPTR
WAGD
If delimiter is not a) then calculate message index.
RF
1V221
RF
If an error has been detected then:
03
Lock SUCB chain via the IPW$RSR macro.
IPW$RSR
PTRLUCB
LUCB
LUGO
If GO cmnd is not requested for this task then:
A. Indicate error msg index.
RF
1V231
B.
02
LUCB
Reset GO cmnd indicator.
LUGO='0'B
PTRLUCB
TCB
Post 'OB reactivate wait ECB'.
TCBROL= '1'B
LULT



Extended Description	Include Segment	Call Subroutine/ Macro	Chart
01 The device ptr (DVPTR) points to the appropriate entry in the SUCB. It is calculated by the subroutine PARM. SUDNP d . . . device (LST/PUN) n . . . device number LST: n between 1 and 3 PUN: n = 1 e.g. SUL2P is first byte of LST2 characteristic field in SUSB. 1V23I COMMAND OUT OF SEQUENCE. 1 V22I INVALID COMMAND.			MG 17

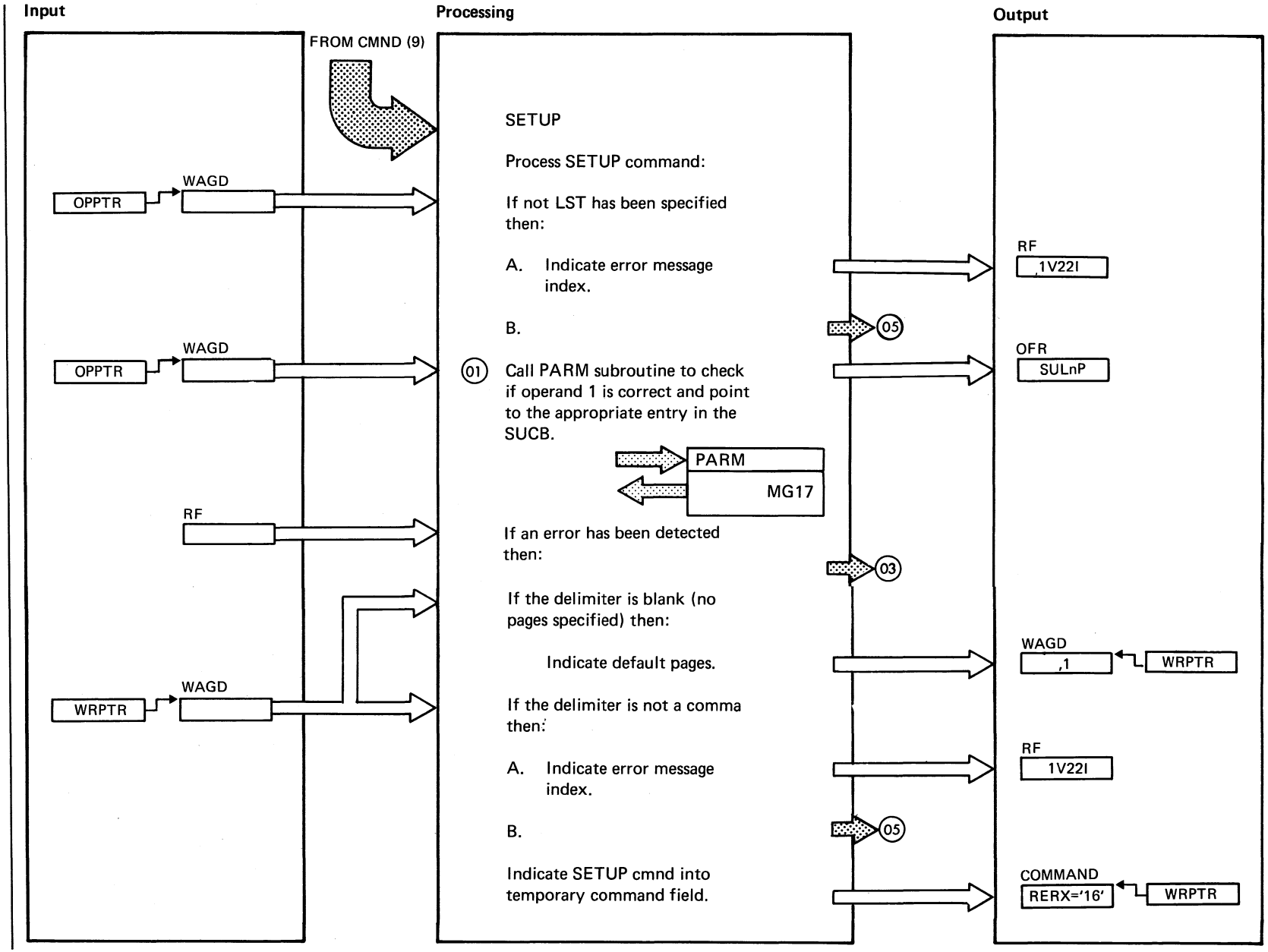
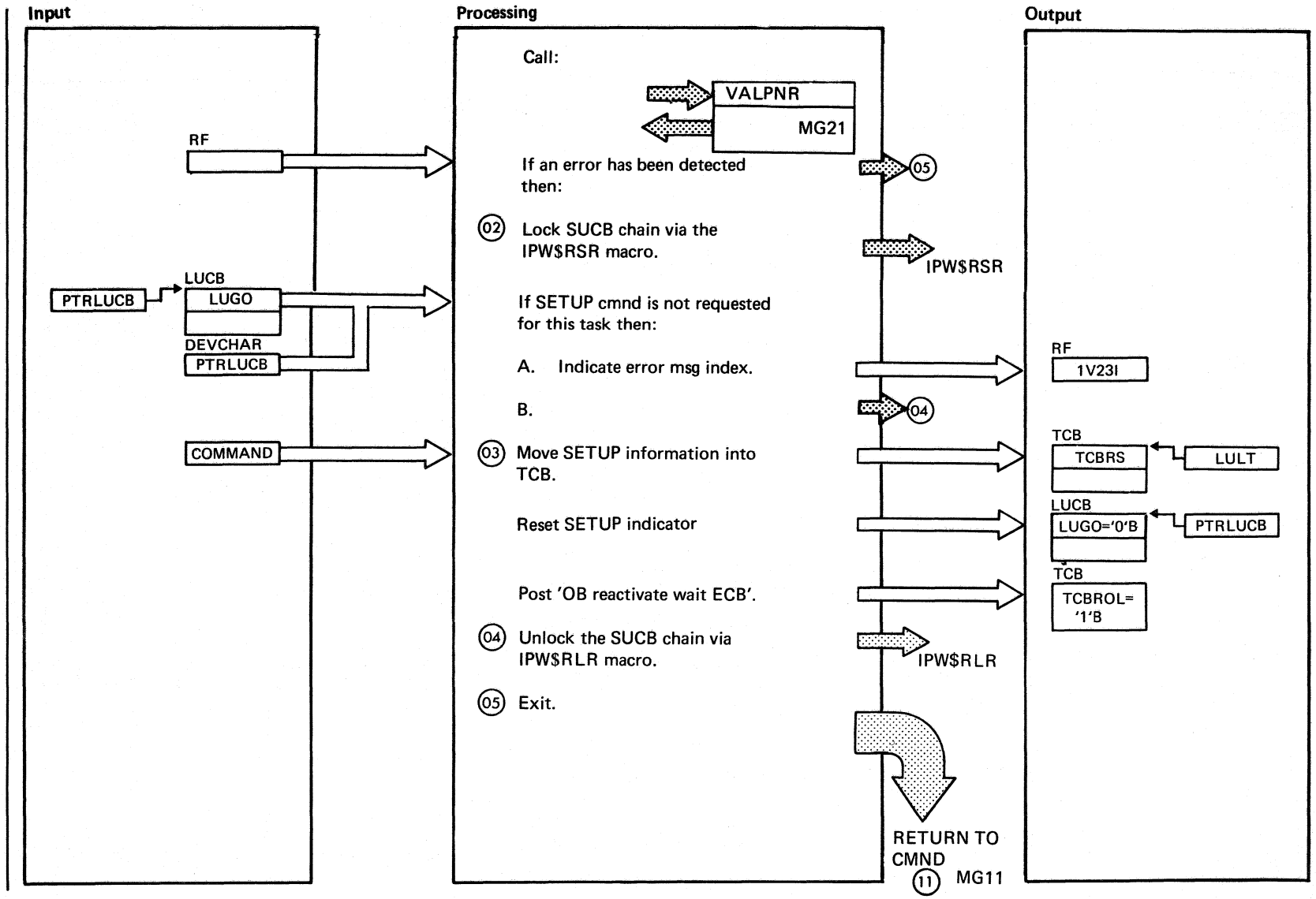
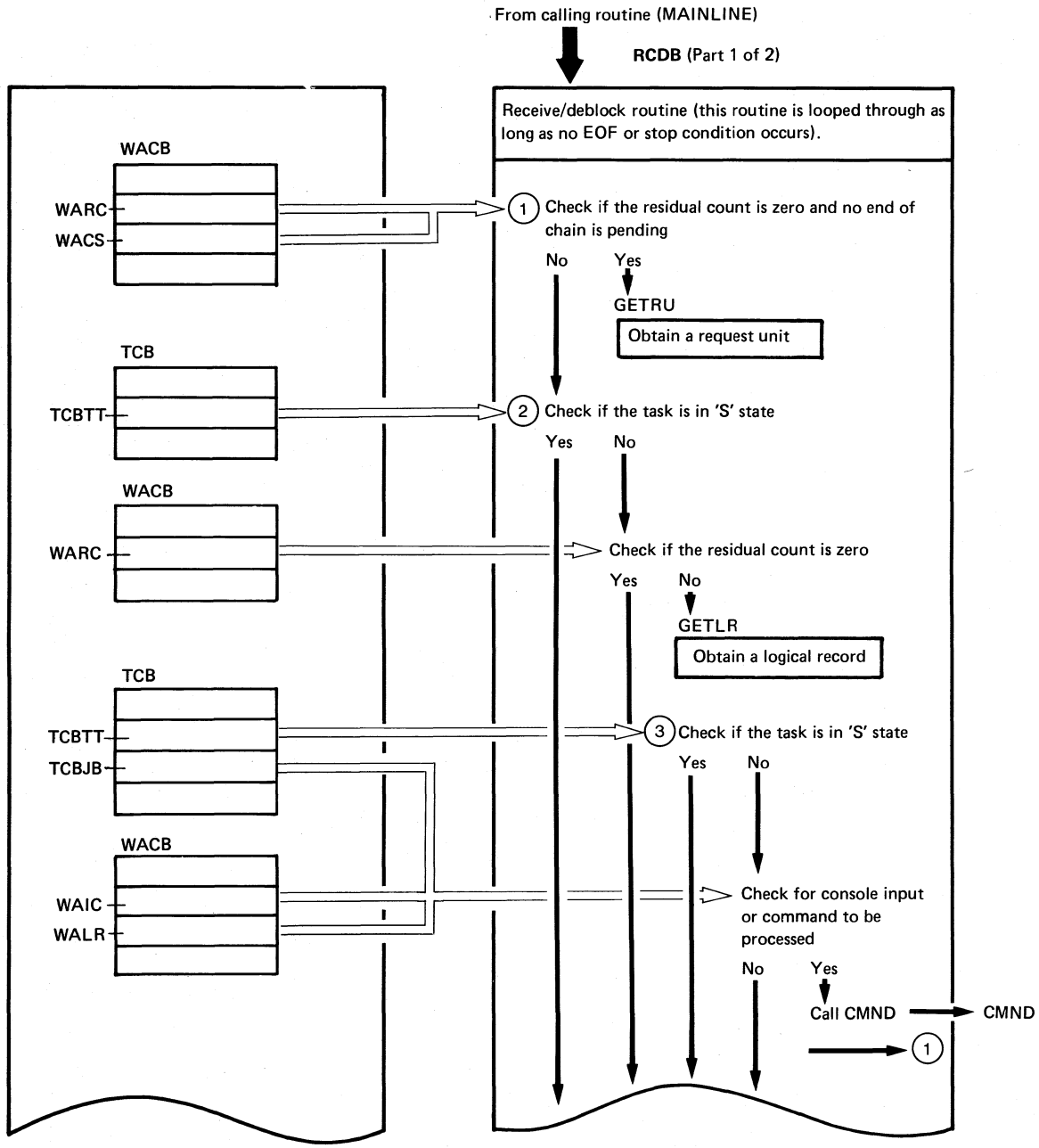


Chart 11.7
Part 1 of 3

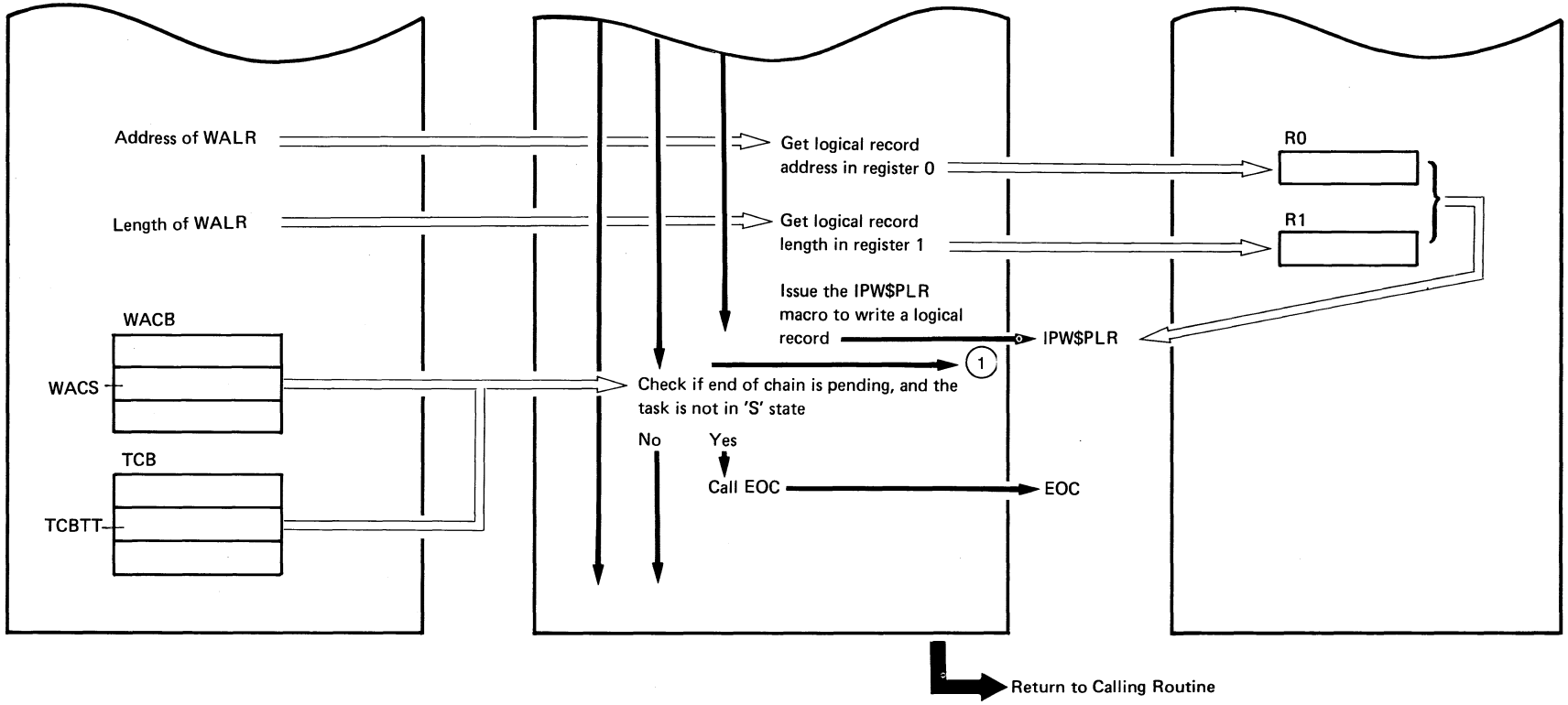

Input
Processing
Output
Call:
VALPNR
MG21
RF
If an error has been detected then:
05
02 Lock SUCB chain via the IPW$RSR macro.
IPW$RSR
PTRLUCB
LUCB
LUGO
DEVCHAR
PTRLUCB
If SETUP cmnd is not requested for this task then:
A. Indicate error msg index.
RF
1V23I
B.
04
COMMAND
03 Move SETUP information into TCB.
TCB
TCBRS
LULT
Reset SETUP indicator
LUCB
LUGO='0'B
PTRLUCB
Post 'OB reactivate wait ECB'.
TCB
TCBROL='1'B
04 Unlock the SUCB chain via IPW$RLR macro.
IPW$RLR
05 Exit.
RETURN TO CMND
11 MG11

Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(01) The device ptr (DVPTR) points to the appropriate entry in the SUCB. It is calculated by the subroutine PARM. SUDNP d . . . device (LST/PUN) n . . . device number LST: n between 1 and 3 PUN: n = 1 e.g. SUL2P is first byte of LST2 characteristic field in SUCB.			MG 17
(02) Chain must be locked because second IB processor can also enter commands on a different session during same time. 1V22I INVALID COMMAND 1V23I COMMAND OUT OF SEQUENCE			
(04)	IPW$RLR (POWER/ VS)		



(Continued on next page)

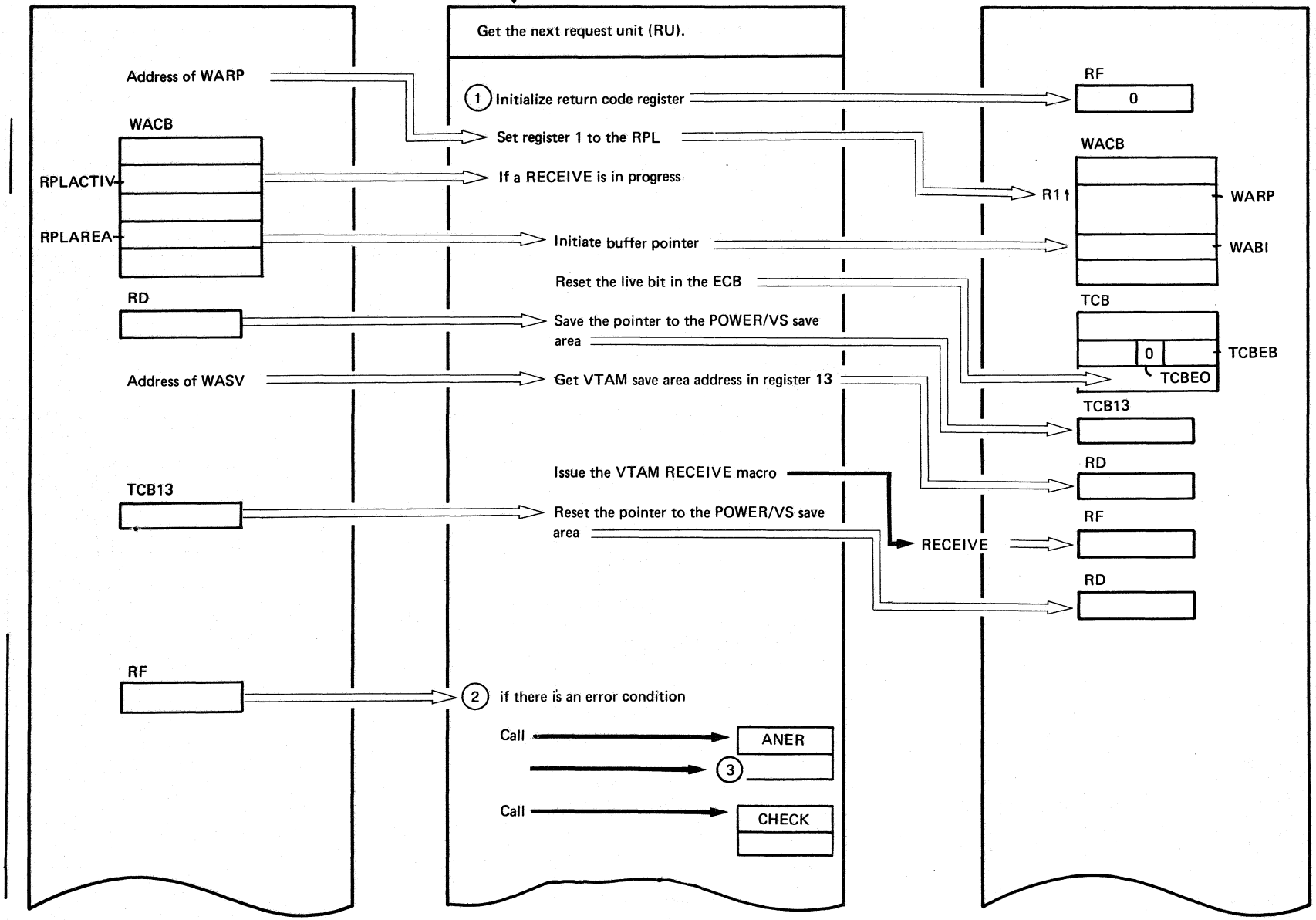
RCDB (Part 2 of 2)



Extended Description	Include Segment	Call Subroutine/Macro	Chart
(1)	GETRU		MG13
(2)	GETLR		MG14
(3)		CMND	MG11
		IPW$PLR(POWER/VS)	
		EOC	MG8

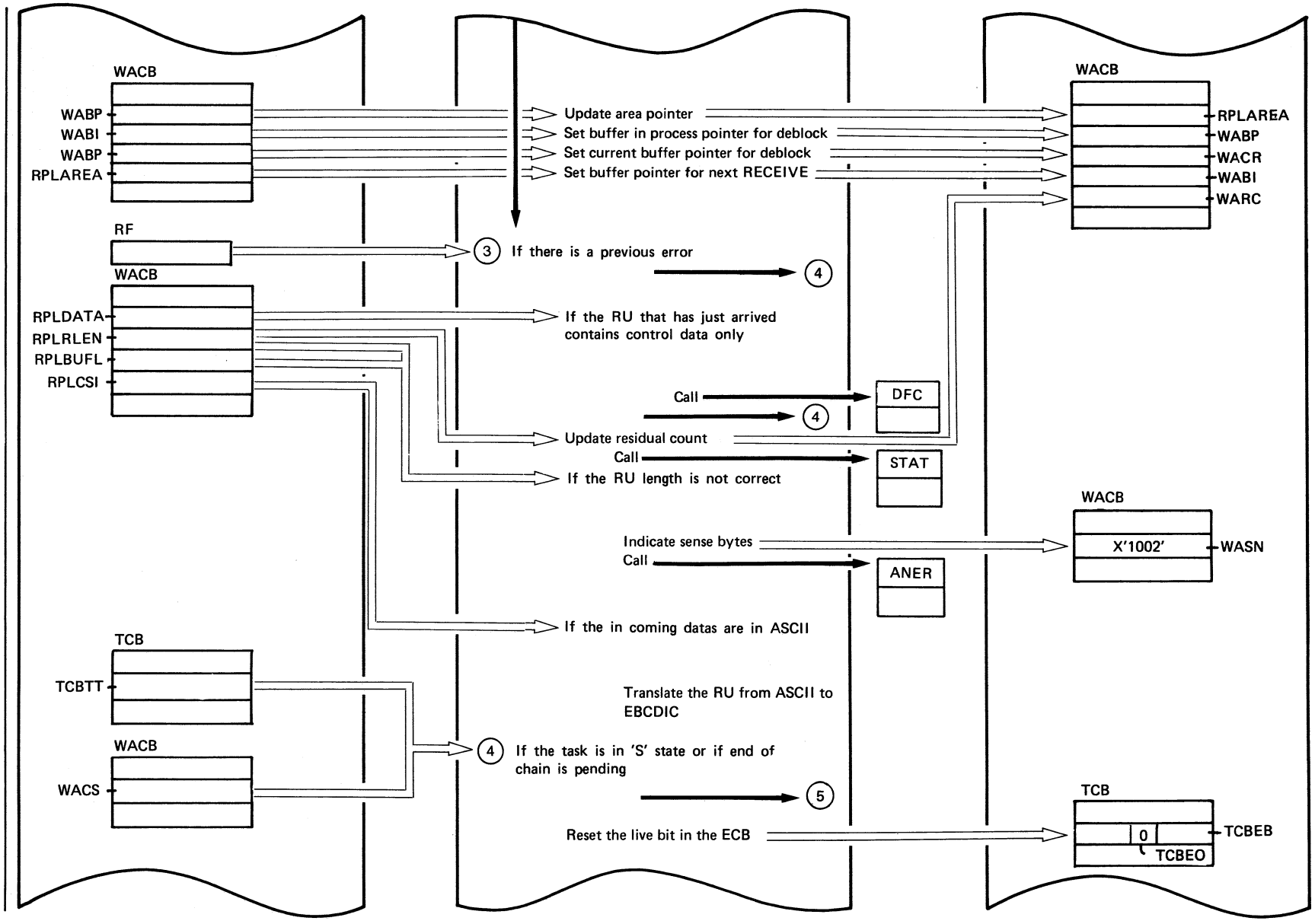
Included by RCDB, Chart MG12

GETRU (Part 1 of 3)



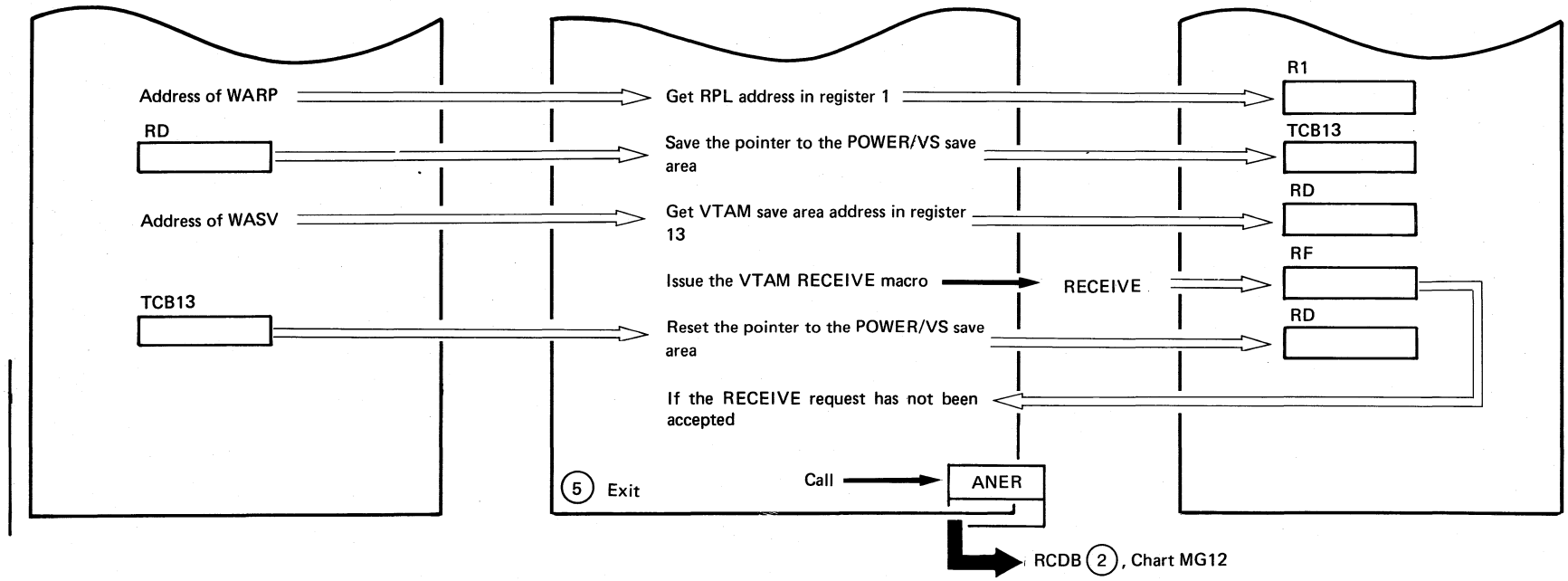
Continued on next page

GETRU (Part 2 of 3)



Continued on next page

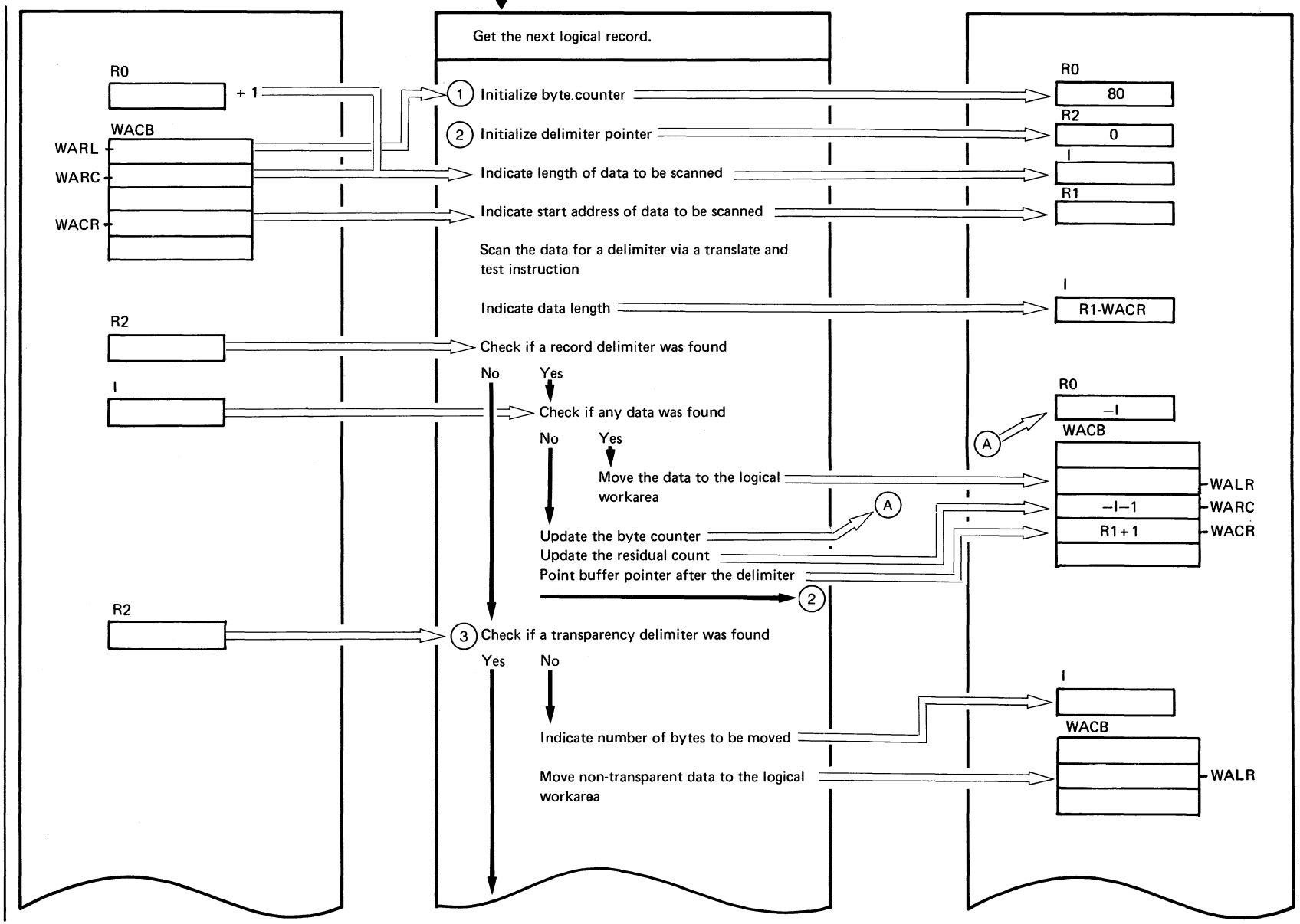
GETRU (Part 3 of 3)



Extended Description	Include Segment	Call Subroutine/Macro	Chart
1		RECEIVE(VTAM)	
		ANER	MG10
2		CHECK	MG16
		ANER	MG10
3		DFC	MG6
		STAT	MG15
		ANER	MG10
4		RECEIVE(VTAM)	
		ANER	MG10

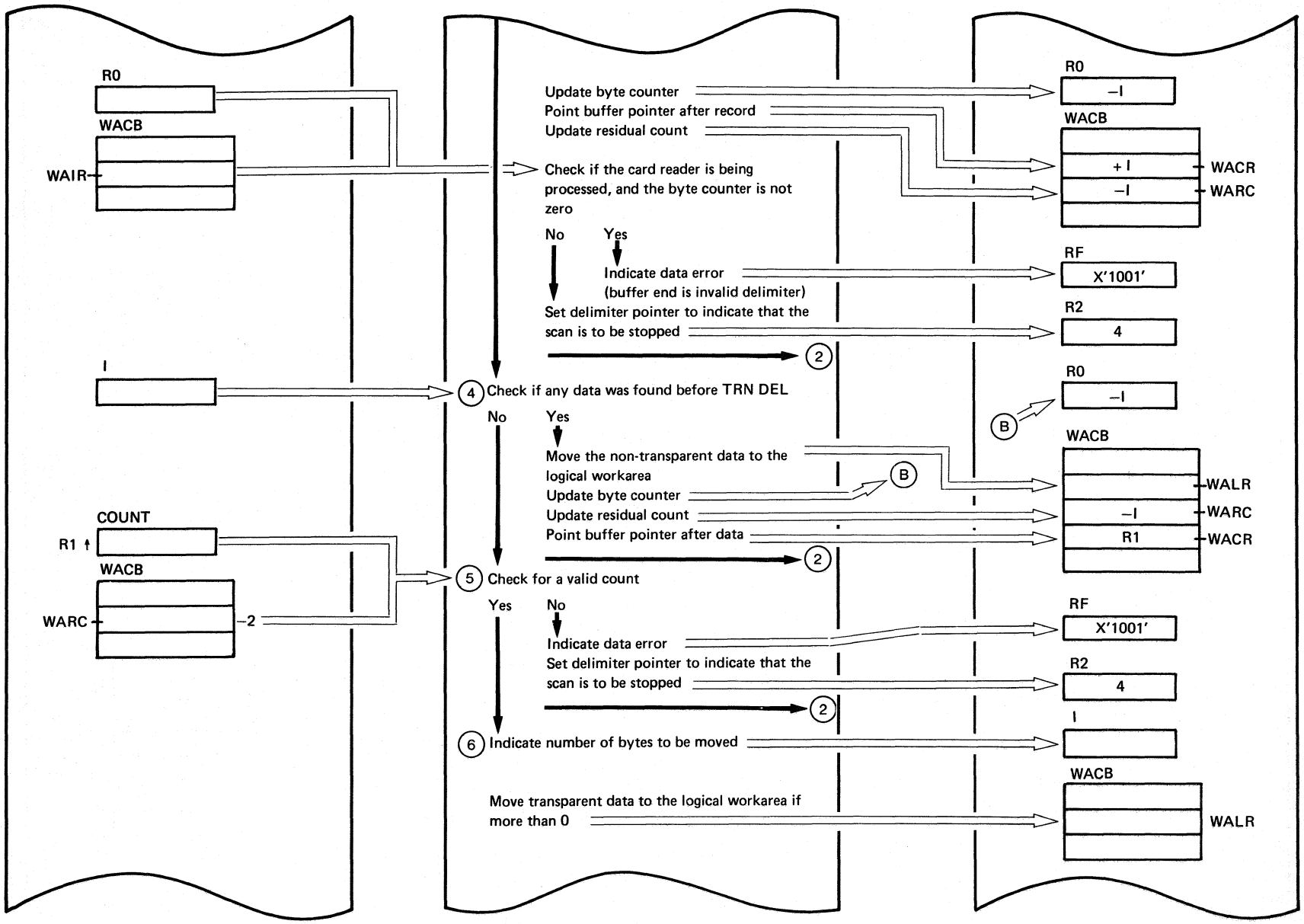
Included by RCDB, Chart MG12

GETLR (Part 1 of 5)



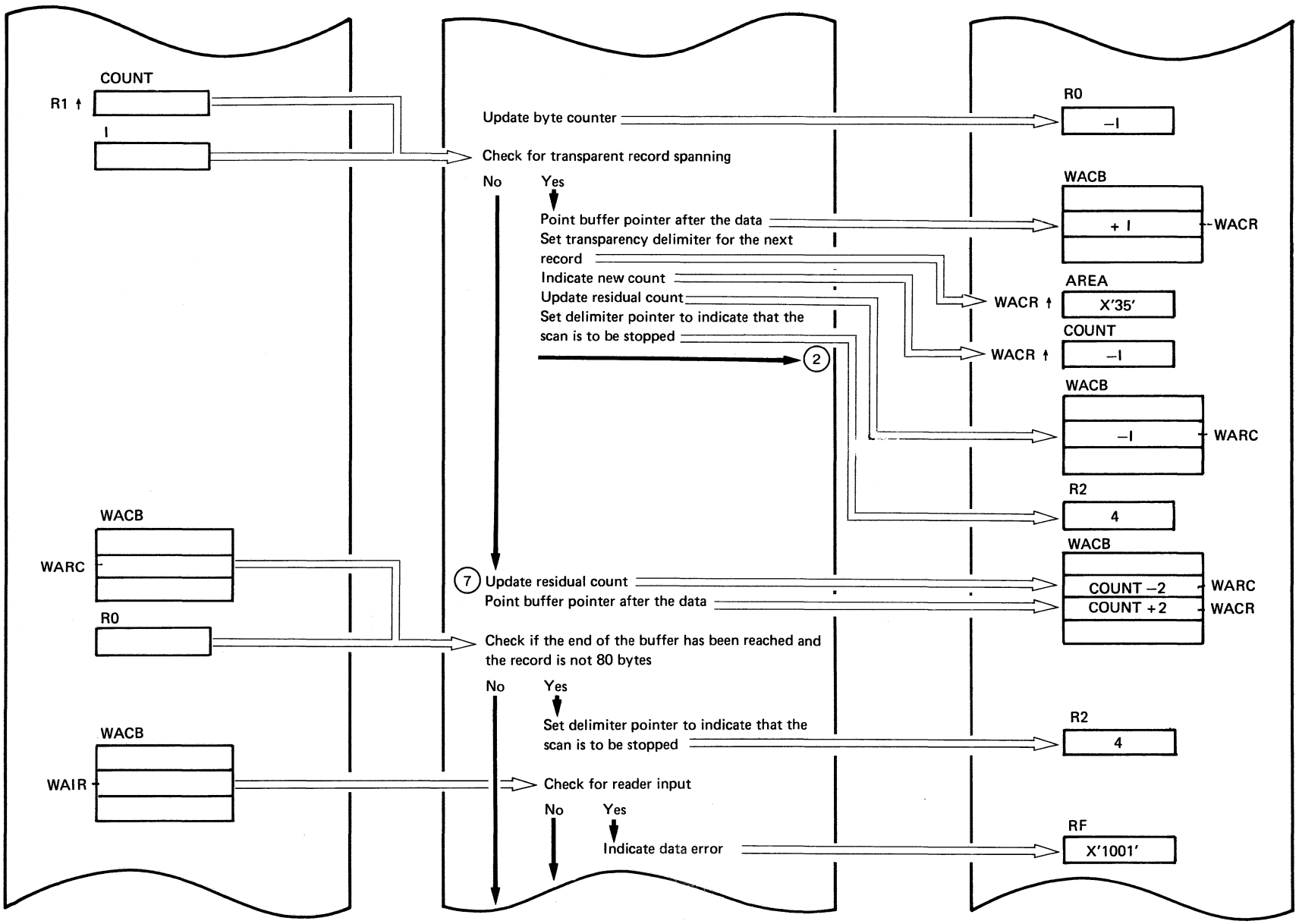
Continued on next page

GETLR (Part 2 of 5)



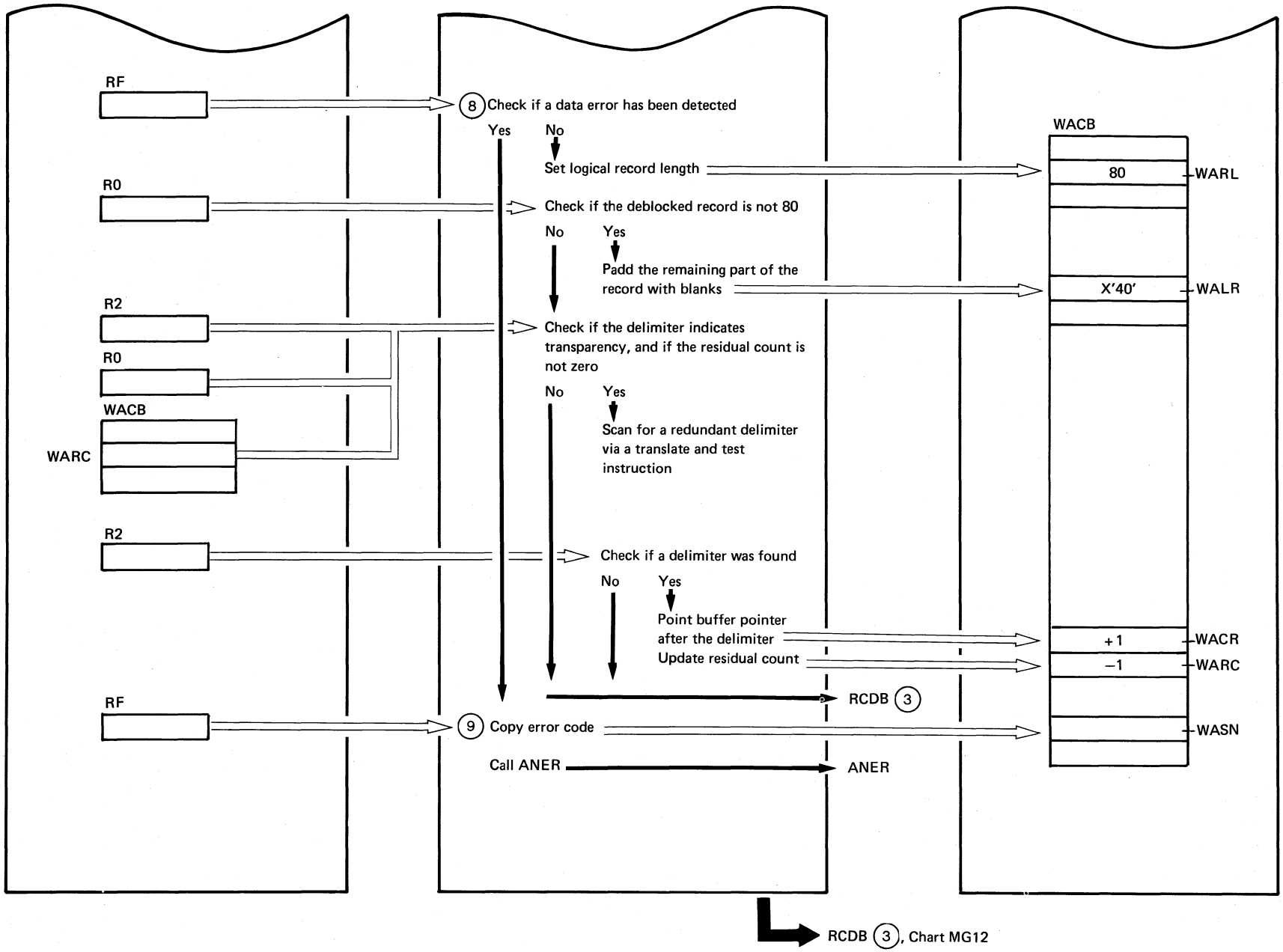
(Continued on next page)

GETLR (Part 3 of 5)



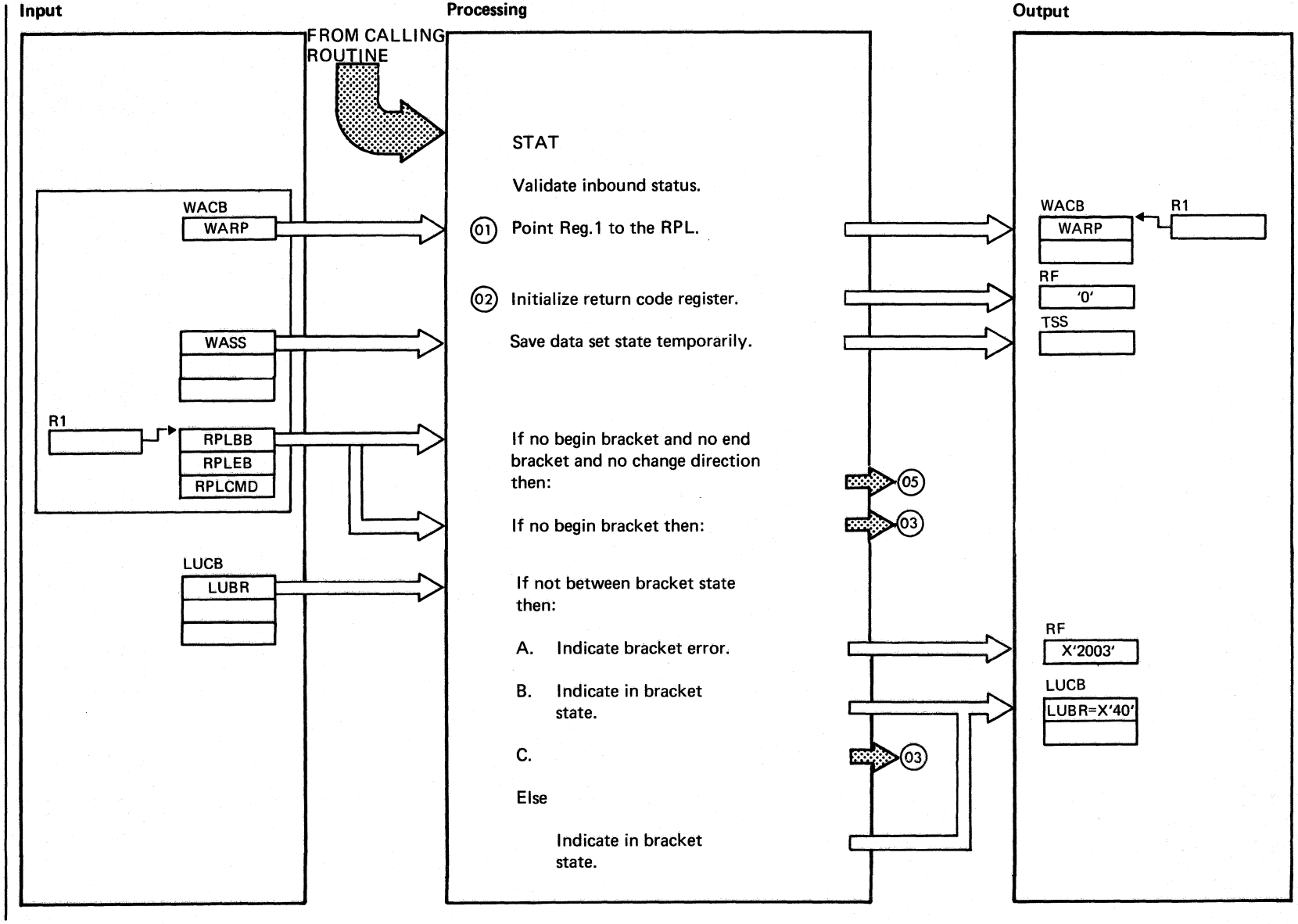
(Continued on next page)

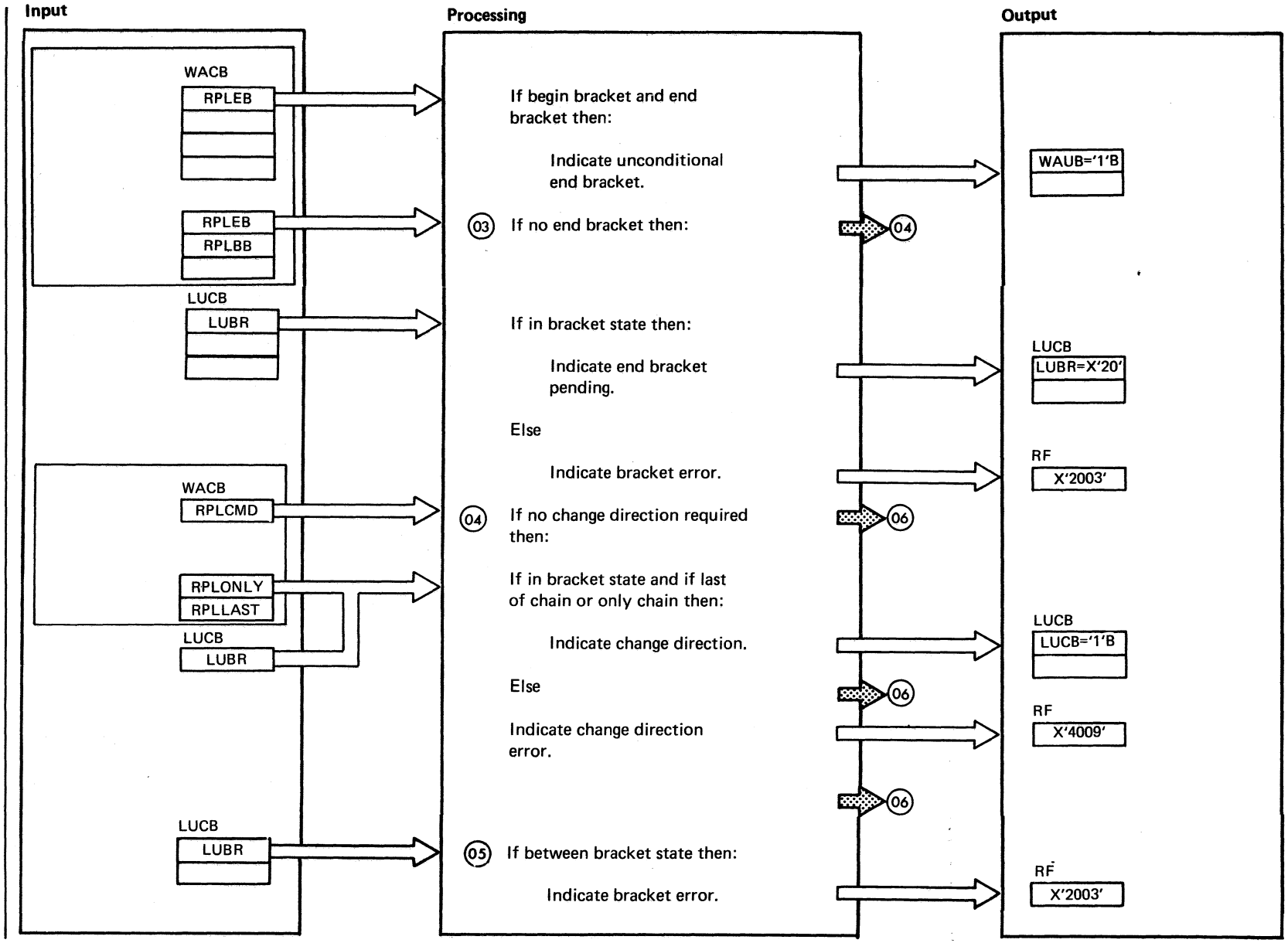
GETLR (Part 4 of 5)

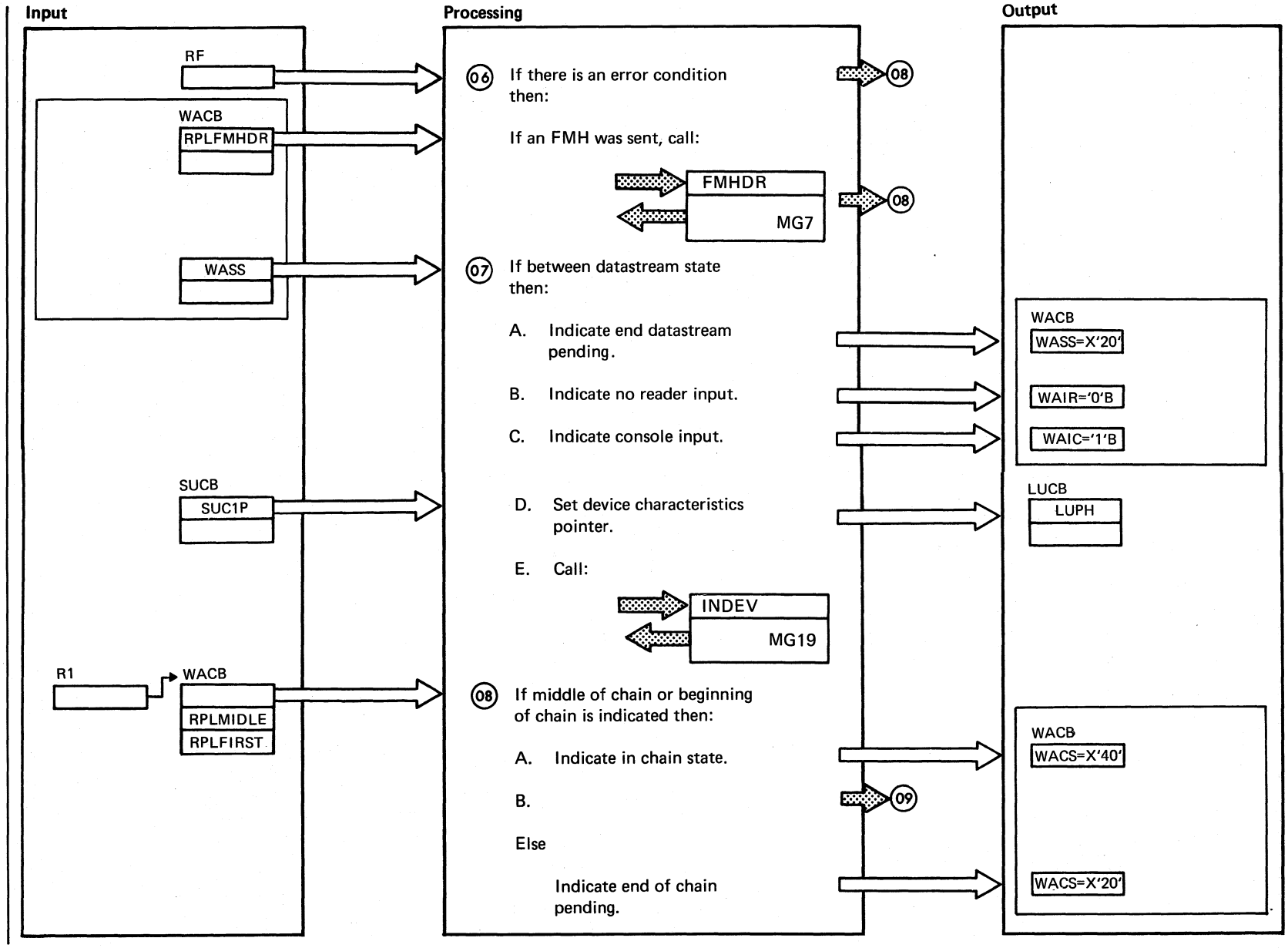


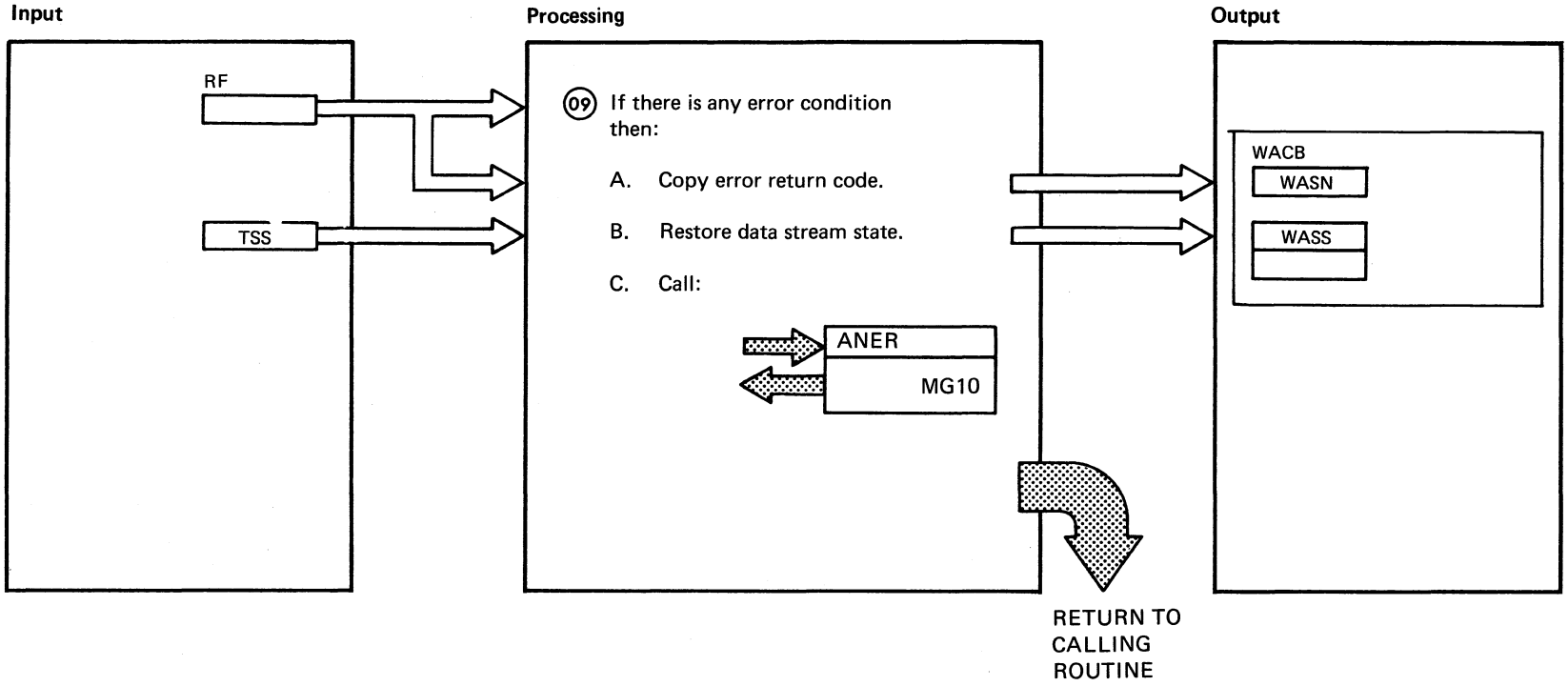
GETLR (Part 5 of 5)

Extended Description	Include Segment	Call Subroutine/Macro	Chart
(2) (7) This code is looped through until a delimiter is found, or an 80 byte logical record (9)		ANER	MG10

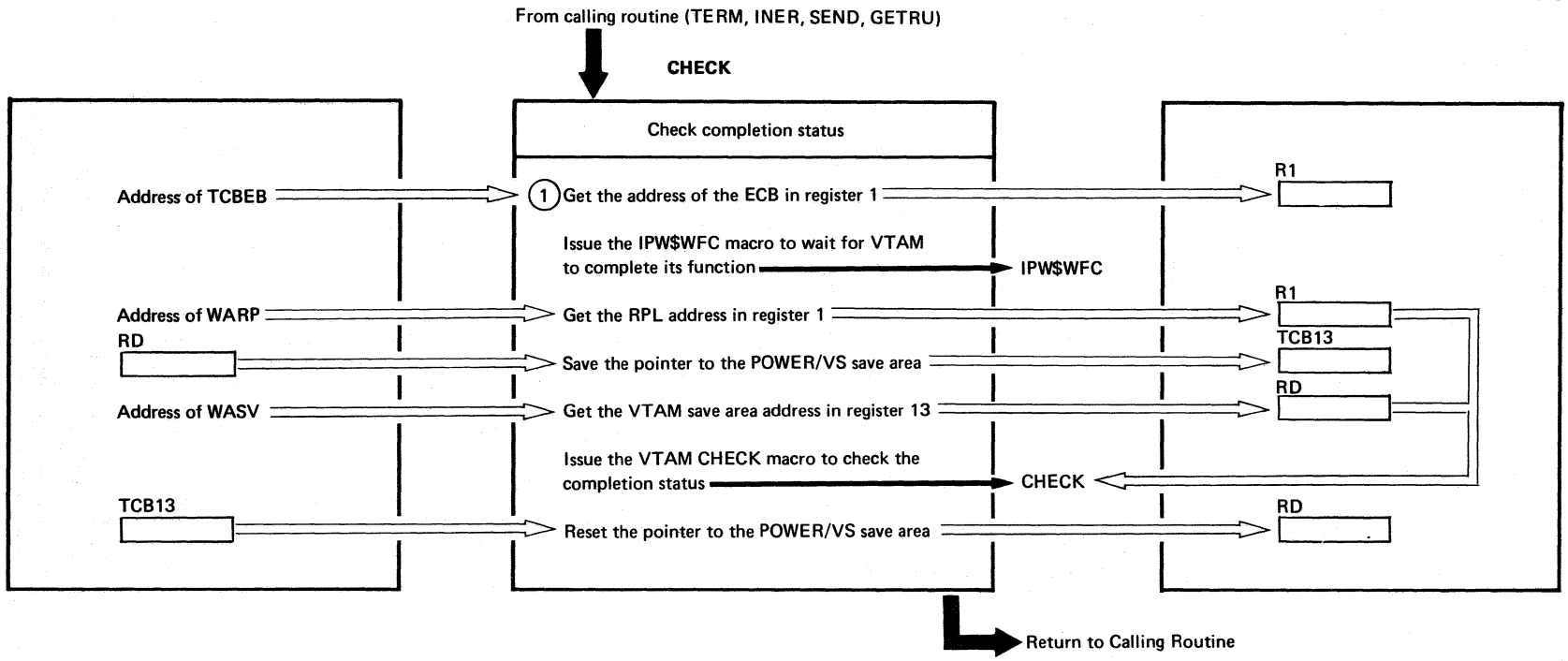

Input
Processing
Output
FROM CALLING ROUTINE
STAT
Validate inbound status.
WACB
WARP
01 Point Reg.1 to the RPL.
02 Initialize return code register.
WASS
Save data set state temporarily.
R1
RPLBB
RPLEB
RPLCMD
If no begin bracket and no end bracket and no change direction then:
05
If no begin bracket then:
03
LUCB
LUBR
If not between bracket state then:
A. Indicate bracket error.
B. Indicate in bracket state.
C.
03
Else
Indicate in bracket state.
WACB
WARP
R1
RF
'0'
TSS
RF
X'2003'
LUCB
LUBR=X'40'


Input
Processing
Output
WACB
RPLEB
RPLEB
RPLBB
LUCB
LUBR
WACB
RPLCMD
RPLONLY
RPLLAST
LUCB
LUBR
LUCB
LUBR
If begin bracket and end bracket then:
Indicate unconditional end bracket.
03 If no end bracket then:
04
If in bracket state then:
Indicate end bracket pending.
Else
Indicate bracket error.
04 If no change direction required then:
06
If in bracket state and if last of chain or only chain then:
Indicate change direction.
Else
06
Indicate change direction error.
06
05 If between bracket state then:
Indicate bracket error.
WAUB='1'B
LUCB
LUBR=X'20'
RF
X'2003'
LUCB
LUCB='1'B
RF
X'4009'
RF
X'2003'

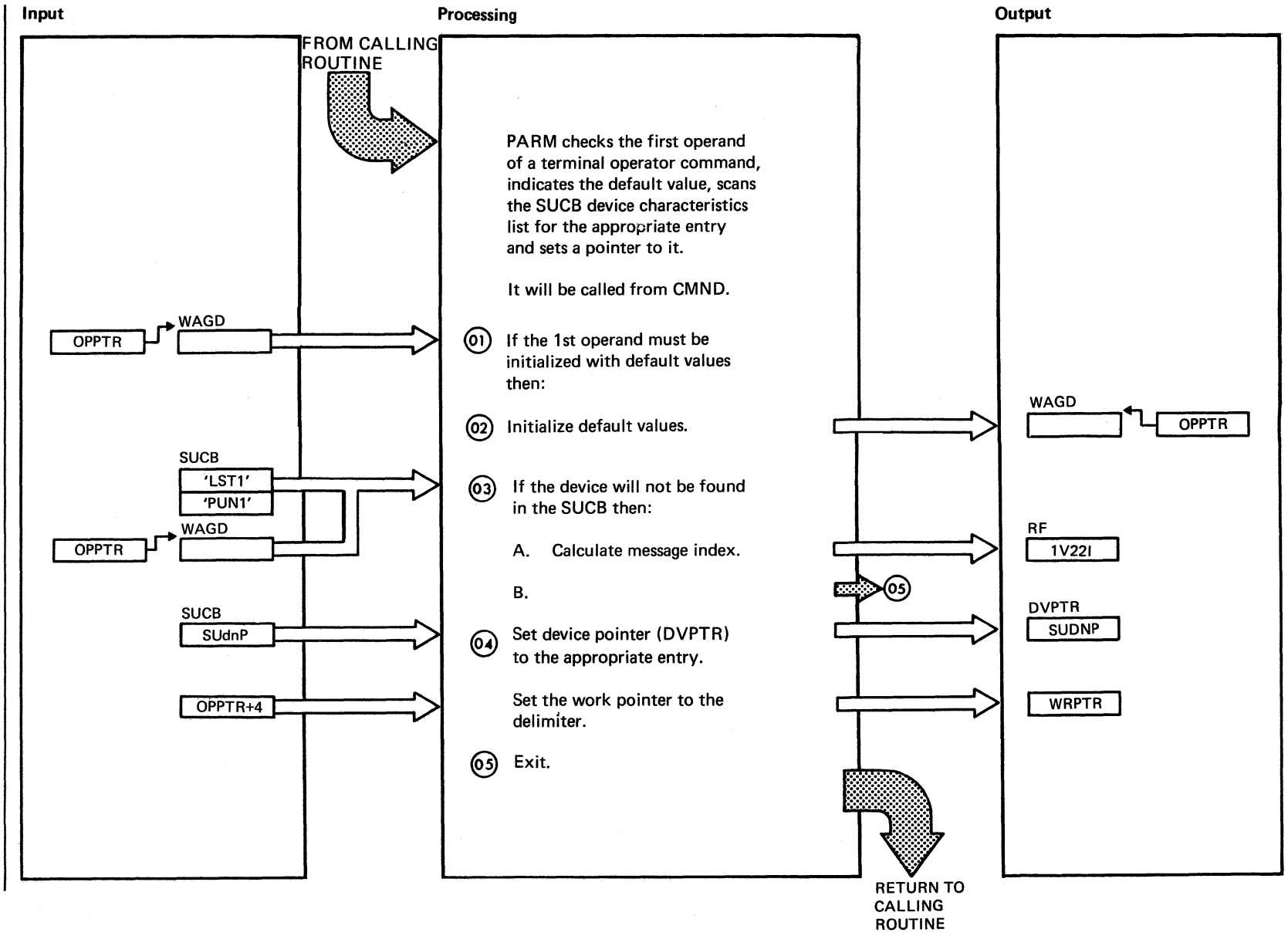

Input
Processing
Output
RF
WACB
RPLFMHDR
WASS
SUCB
SUC1P
R1
WACB
RPLMIDLE
RPLFIRST
06 If there is an error condition then:
If an FMH was sent, call:
FMHDR
MG7
08
08
07 If between datastream state then:
A. Indicate end datastream pending.
B. Indicate no reader input.
C. Indicate console input.
D. Set device characteristics pointer.
E. Call:
INDEV
MG19
08 If middle of chain or beginning of chain is indicated then:
A. Indicate in chain state.
B.
09
Else
Indicate end of chain pending.
WACB
WASS=X'20'
WAIR='0'B
WAIC='1'B
LUCB
LUPH
WACB
WACS=X'40'
WACS=X'20'



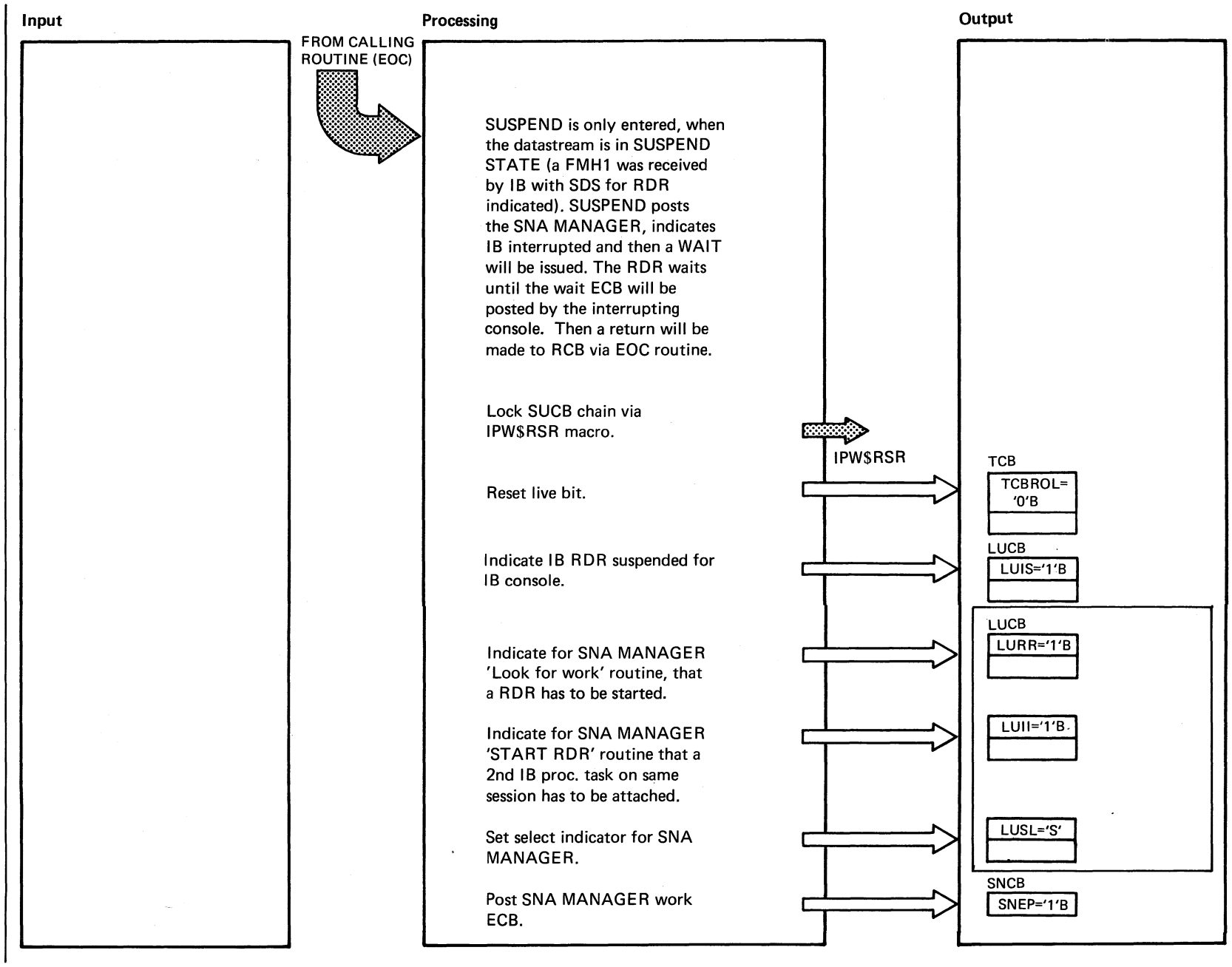
Extended Description	Include Segment	Call Subroutine/ Macro	Chart
06	FMHDR		MG 7
07 No FMH was sent. Default FMH with EDS, BDS, Console has to be taken. For further action see decision table chart MG 7.	INDEV		MG 19
08 The chain state must also be updated in error case (RF not equal ZERO). The processor has to know whether he has to go in purge state or not (subroutine INER).			
09	ANER		MG 10

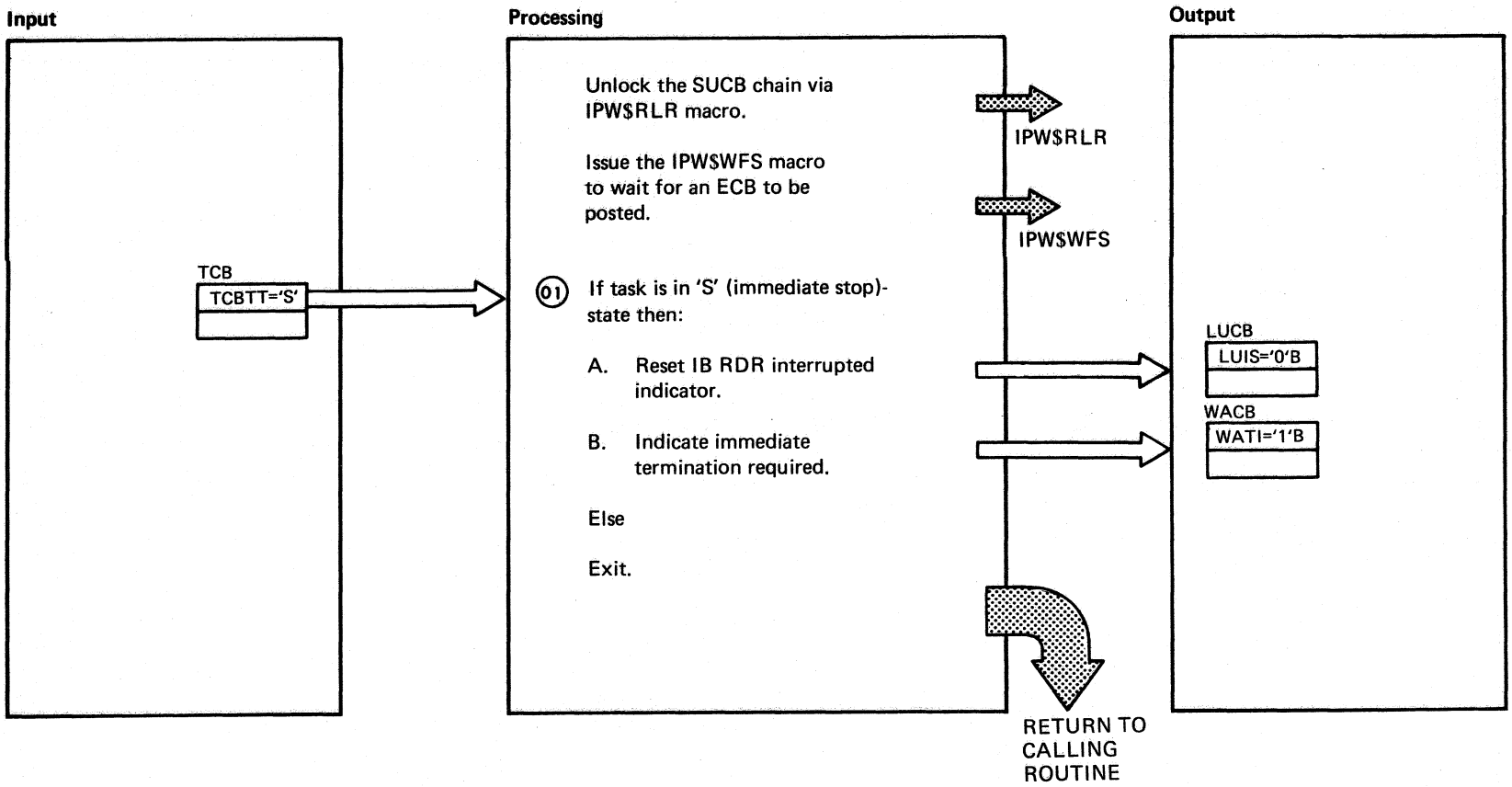


Extended Description	Include Segment	Call Subroutine/Macro	Chart
1		IPW$WFC (POWER/VS) CHECK (VTAM)	

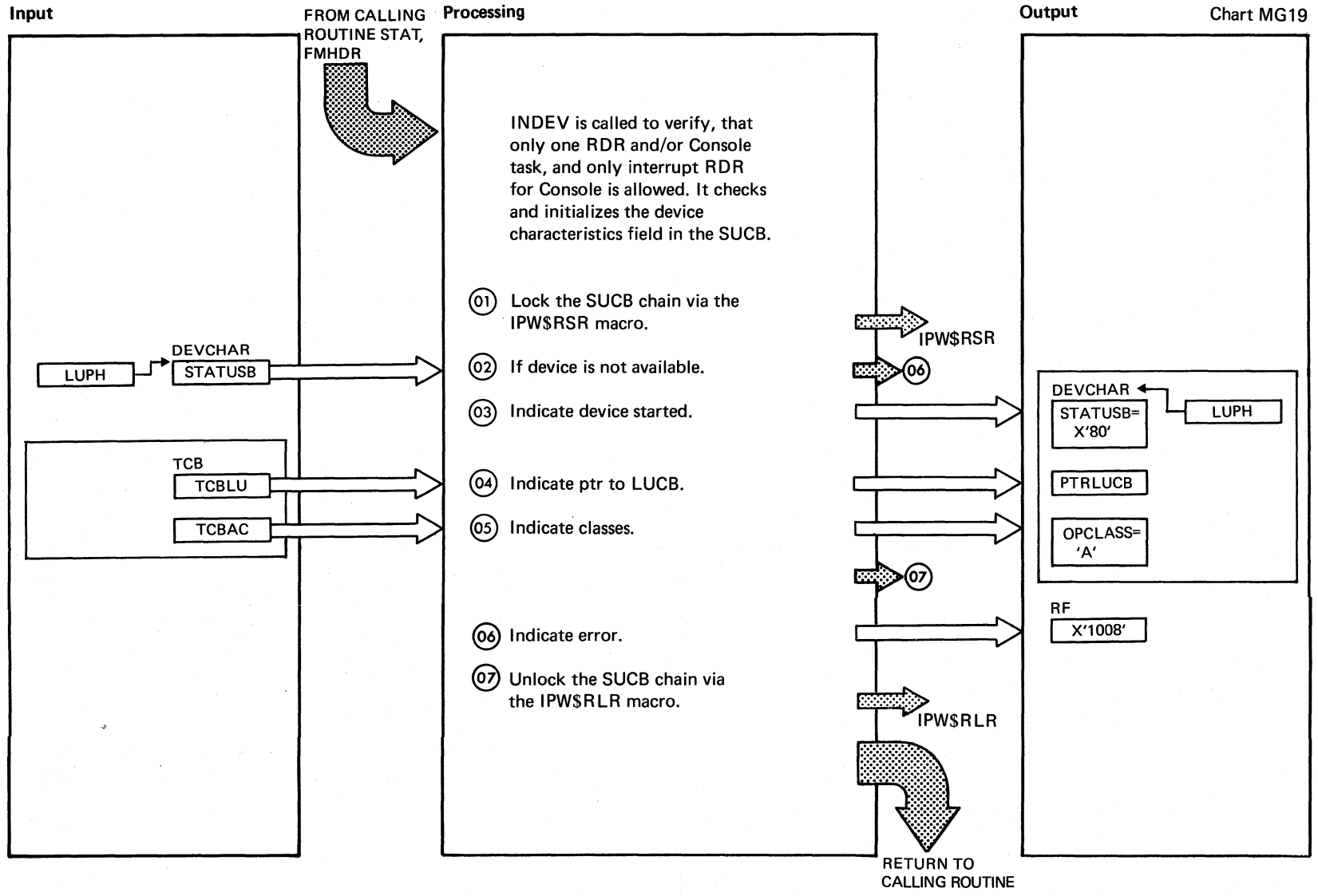

Input
Processing
Output
FROM CALLING ROUTINE
PARM checks the first operand of a terminal operator command, indicates the default value, scans the SUCB device characteristics list for the appropriate entry and sets a pointer to it.
It will be called from CMND.
OPPTR
WAGD
01 If the 1st operand must be initialized with default values then:
02 Initialize default values.
SUCB
'LST1'
'PUN1'
OPPTR
WAGD
03 If the device will not be found in the SUCB then:
A. Calculate message index.
B.
05
SUCB
SUdnP
04 Set device pointer (DVPTR) to the appropriate entry.
OPPTR+4
Set the work pointer to the delimiter.
05 Exit.
WAGD
OPPTR
RF
1V221
DVPTR
SUDNP
WRPTR
RETURN TO CALLING ROUTINE

Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(01) The first operand can be: A. LST or LST1 or LST2 or LST3 B. PUN or PUN1 (02) LST will be initialized to LST1 and PUN to PUN 1. (03) The field SUDNP in the SUCB will be compared with the operator entered value. (04) The device ptr (DVPTR) points to the appropriate entry in the SUCB, e.g. for LST2 to SUL2P. d ... device (LST/PUN) n . . . device number LST between 1 and 3 PUN = 1			SUDNP

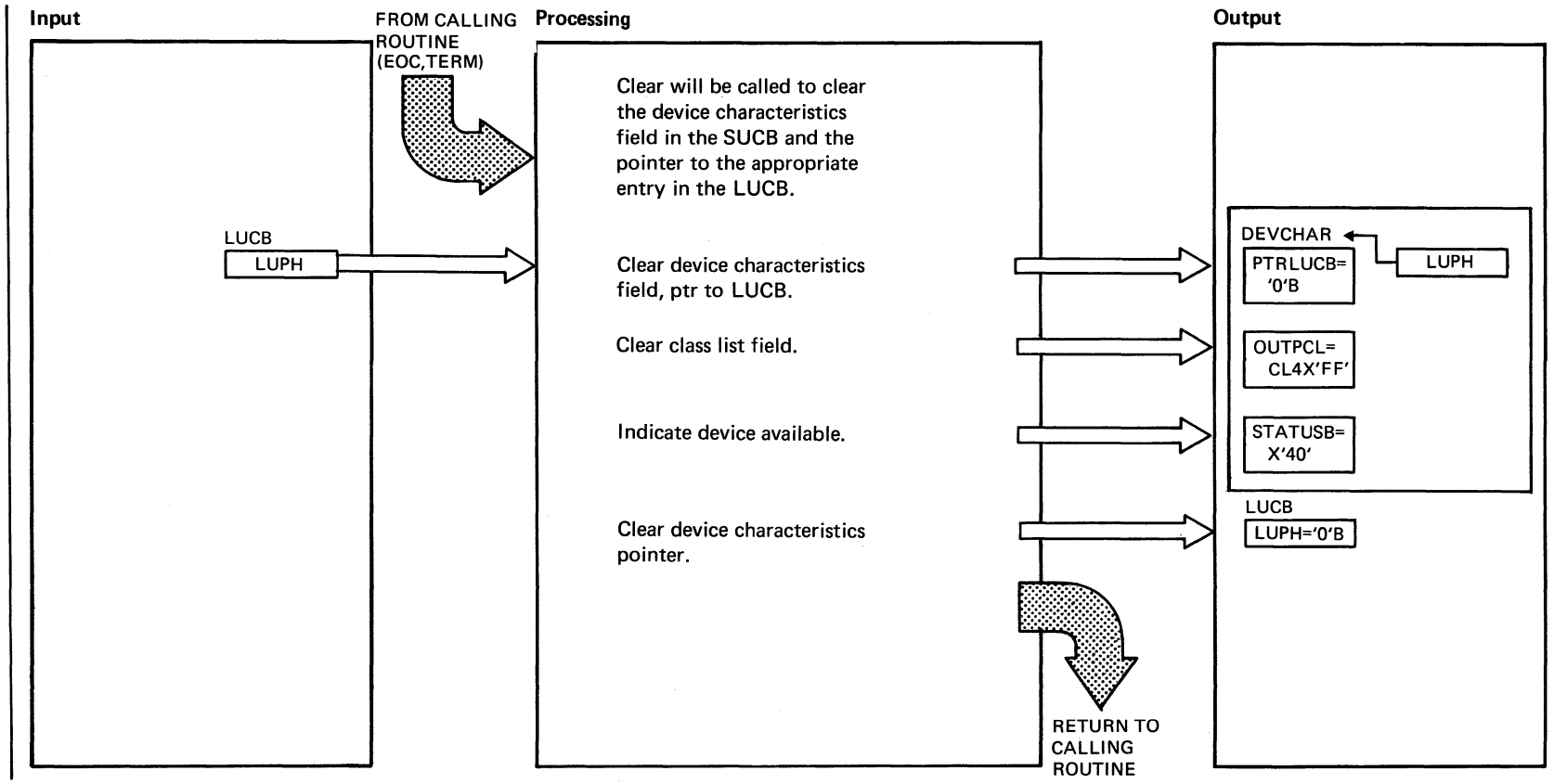

Input
Processing
Output
FROM CALLING ROUTINE (EOC)
SUSPEND is only entered, when the datastream is in SUSPEND STATE (a FMH1 was received by IB with SDS for RDR indicated). SUSPEND posts the SNA MANAGER, indicates IB interrupted and then a WAIT will be issued. The RDR waits until the wait ECB will be posted by the interrupting console. Then a return will be made to RCB via EOC routine.
Lock SUCB chain via IPW$RSR macro.
IPW$RSR
Reset live bit.
TCB
TCBROL='0'B
Indicate IB RDR suspended for IB console.
LUCB
LUIS='1'B
LUCB
Indicate for SNA MANAGER 'Look for work' routine, that a RDR has to be started.
LURR='1'B
Indicate for SNA MANAGER 'START RDR' routine that a 2nd IB proc. task on same session has to be attached.
LUII='1'B
Set select indicator for SNA MANAGER.
LUSL='S'
Post SNA MANAGER work ECB.
SNCB
SNEP='1'B

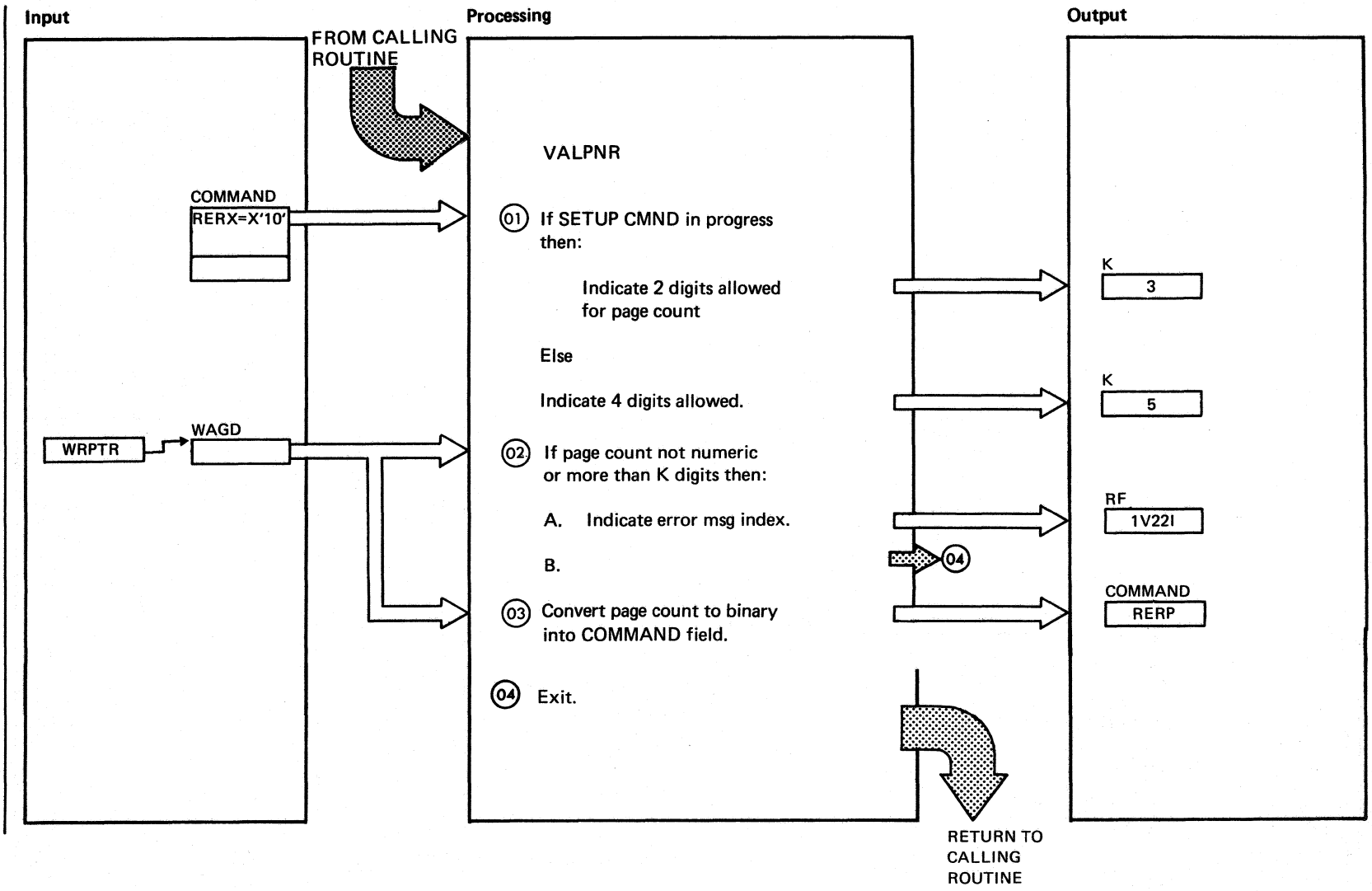

Input
Processing
Output
TCB
TCBTT='S'
Unlock the SUCB chain via IPW$RLR macro.
Issue the IPW$WFS macro to wait for an ECB to be posted.
01
If task is in 'S' (immediate stop)-state then:
A. Reset IB RDR interrupted indicator.
B. Indicate immediate termination required.
Else
Exit.
IPW$RLR
IPW$WFS
LUCB
LUIS='0'B
WACB
WATI='1'B
RETURN TO CALLING ROUTINE

Extended Description	Include Segment	Call Subroutine/ Macro	Chart
	IPW$RSR (POWER/ VS) IPW$RLR (POWER/ VS) IPW$WFS (POWER/ VS)		
(01) Normally LUIS will be reset by the interrupting IB console before detaching. In a shutdown period (terminating one session by SNA MANAGER) it can happen, that the SNA MANAGER (before the interrupting IB console got control) posts the interrupted IB reader. Then LUIS is not reset. When the interrupting console got an error, the processor will terminate all IB flows. That means the interrupted reader has to detach itself immediately after he was posted (indicator WATI). An end bracket was sent already depending if an OB was interrupted or not. If an OB processor also was interrupted, he will send the end bracket.			



Extended Description	Include Segment	Call Subroutine/ Macro	Chart
02	IPW$RSR (POWER/VS) IPW$RLR (POWER/VS)		

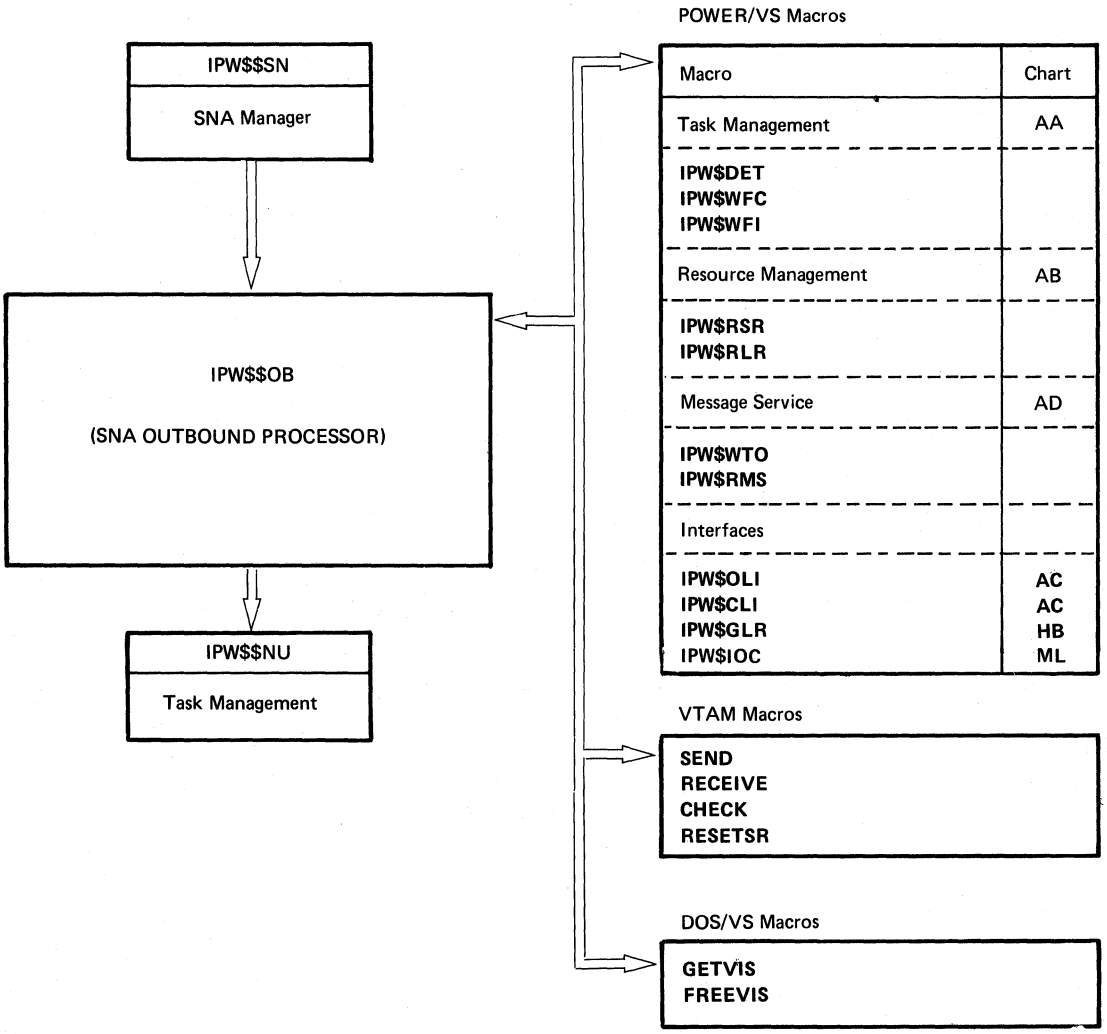

Input
FROM CALLING ROUTINE (EOC,TERM)
Processing
Output
Clear will be called to clear the device characteristics field in the SUCB and the pointer to the appropriate entry in the LUCB.
LUCB
LUPH
Clear device characteristics field, ptr to LUCB.
Clear class list field.
Indicate device available.
Clear device characteristics pointer.
DEVCHAR
PTRLUCB= '0'B
LUPH
OUTPCL= CL4X'FF'
STATUSB= X'40'
LUCB
LUPH='0'B
RETURN TO CALLING ROUTINE


Input
Processing
Output
FROM CALLING ROUTINE
VALPNR
COMMAND
RERX=X'10'
WRPTR
WAGD
01 If SETUP CMND in progress then:
Indicate 2 digits allowed for page count
Else
Indicate 4 digits allowed.
02 If page count not numeric or more than K digits then:
A. Indicate error msg index.
B.
03 Convert page count to binary into COMMAND field.
04 Exit.
K
3
K
5
RF
1V22I
04
COMMAND
RERP
RETURN TO CALLING ROUTINE

Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(01) VALPNR validates the page numbers (page count) entered as an operand of a SETUP or RESTART command. Therefore it will be called by the 'CMND' routine whenever a valid SETUP or RESTART command was entered. (02) The number is checked if all digits are numeric an then if the number has the allowed digits (for SETUP 2 digits are allowed, for RESTART 4 digits). (03) If the page number is valid, it will be packed, converted to binary and moved into the 'COMMAND' work field. This workfield will later on (when the whole command is valid) be moved into the appropriate TCB (will be done in the 'CMND' routine).			

CHART MH: IPW$$OB - SNA OUTBOUND PROCESSOR (65 PARTS)

Chart MH00: IPW$$OB - SNA Outbound Processor, General Flow and Macro Calls



SNA OUTBOUND PROCESSOR (IPW$$OB) ORGANIZATION (4Parts)

Chart MH1

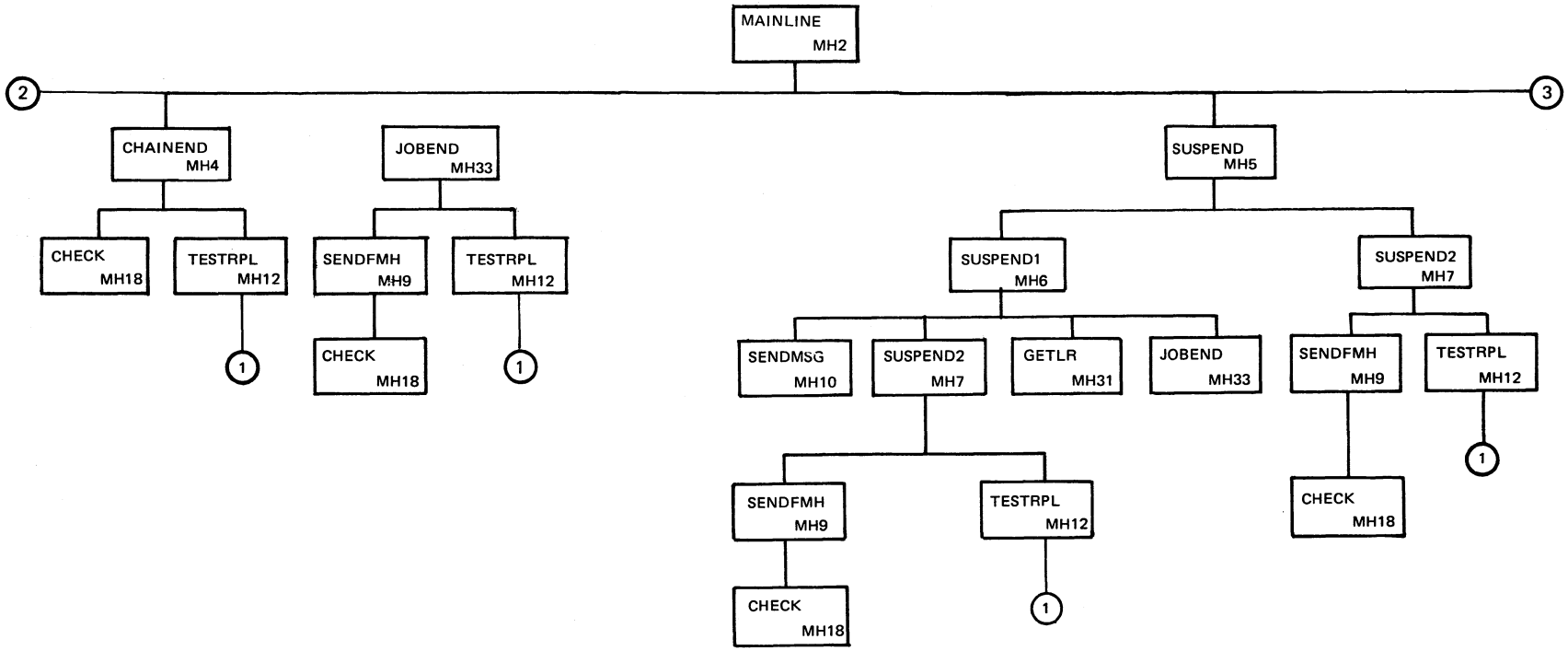


Chart MH1

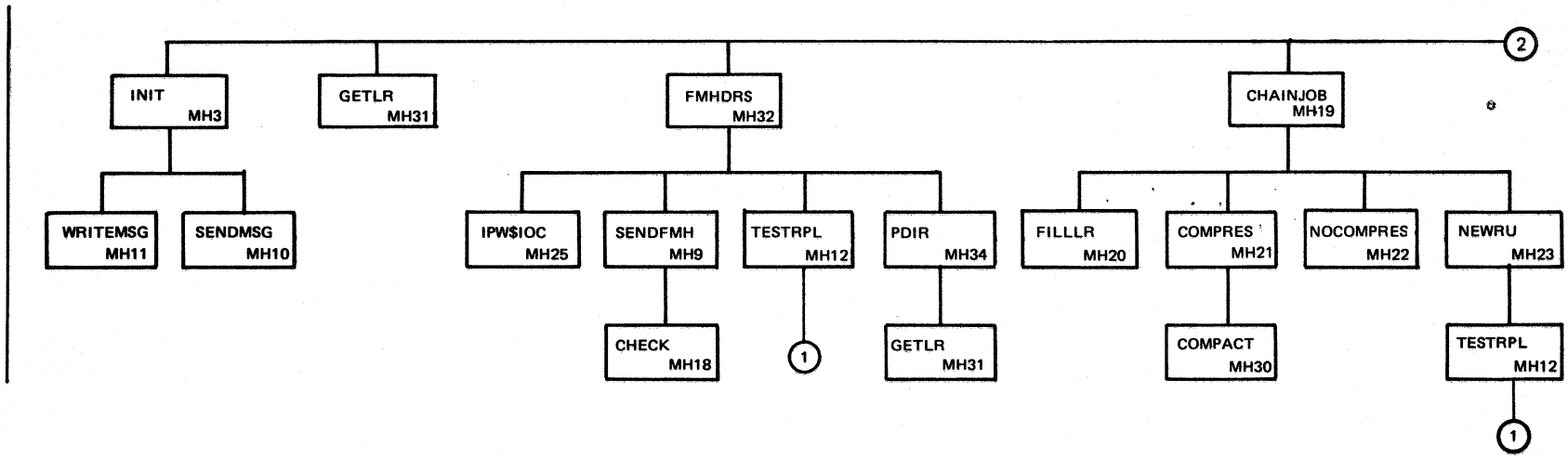

INIT
MH3
GETLR
MH31
FMHDRS
MH32
CHAINJOB
MH19
2
WRITEMSG
MH11
SENDMSG
MH10
IPW$IOC
MH25
SENDFMH
MH9
TESTRPL
MH12
PDIR
MH34
FILLLR
MH20
COMPRES
MH21
NOCOMPRES
MH22
NEWRU
MH23
CHECK
MH18
1
GETLR
MH31
COMPACT
MH30
TESTRPL
MH12
1

Chart MH1

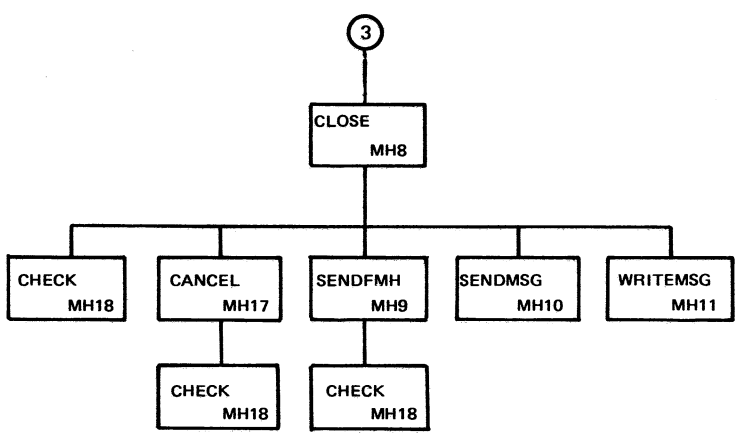
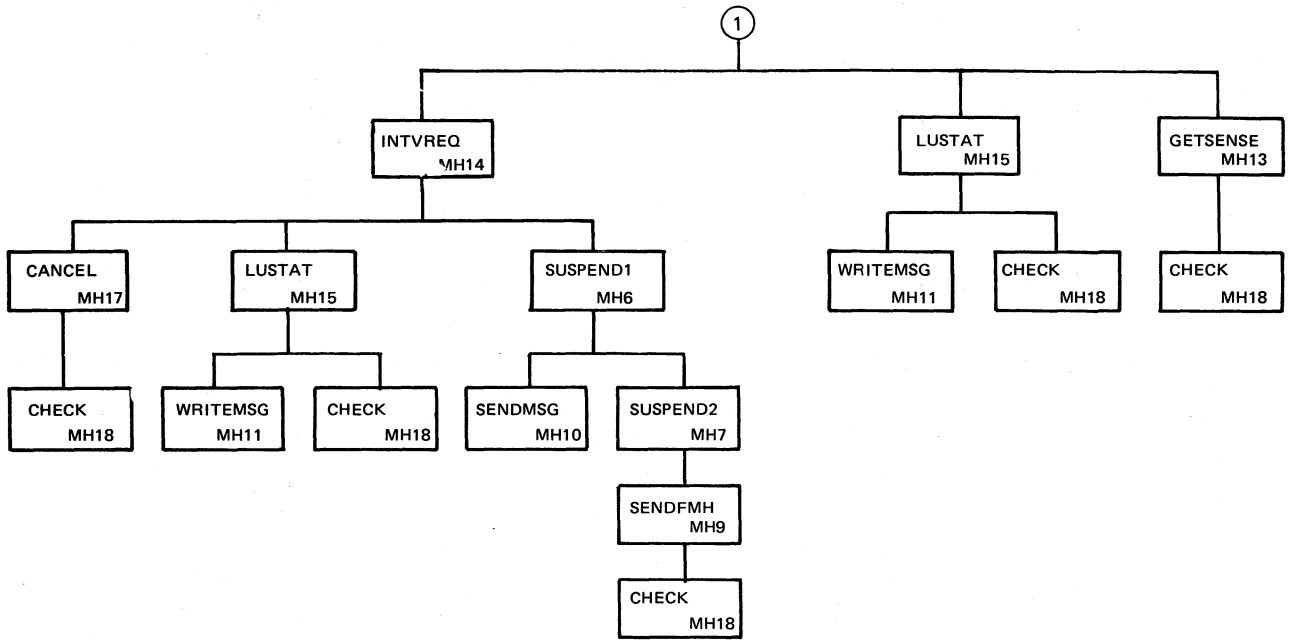
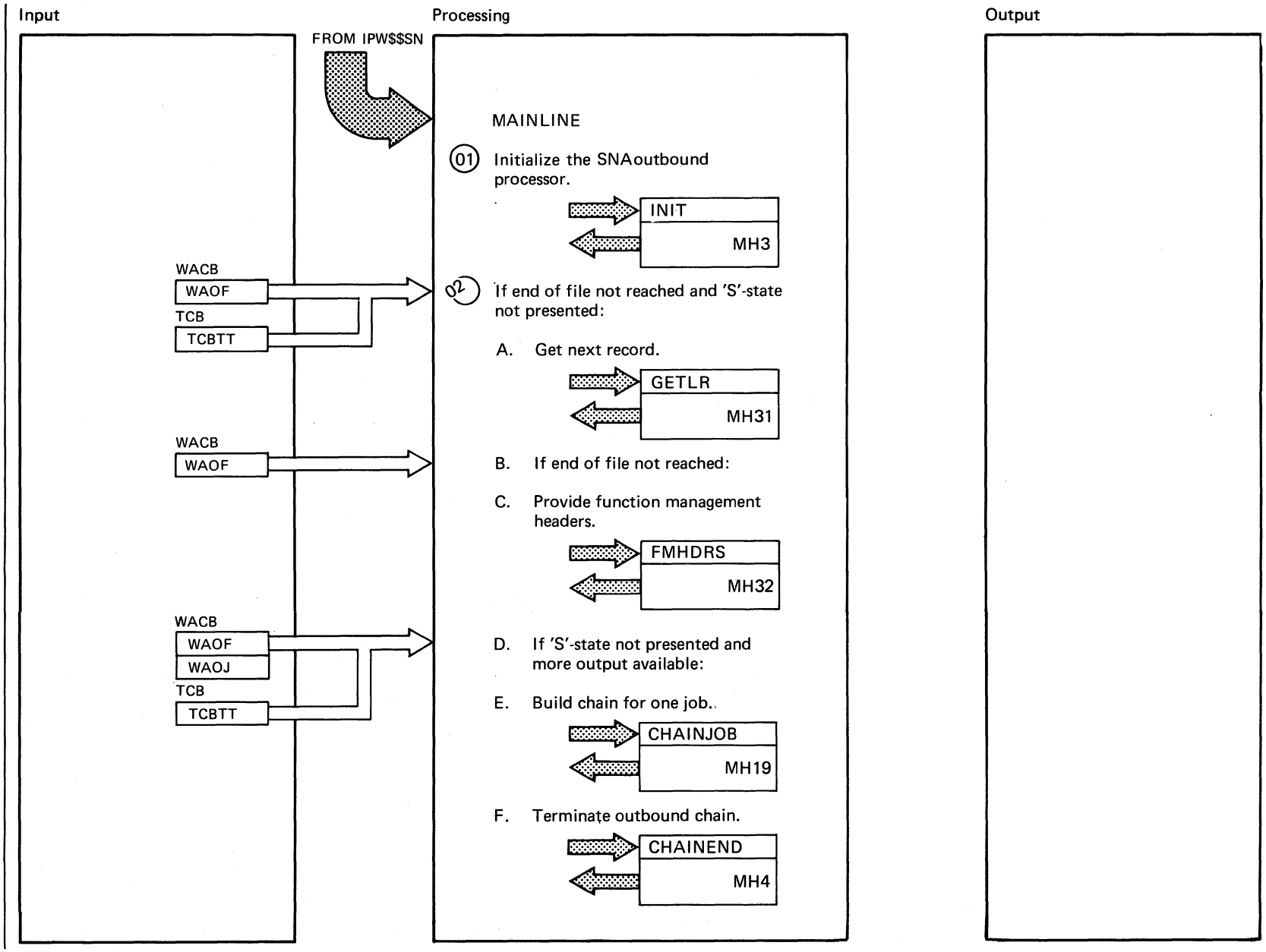
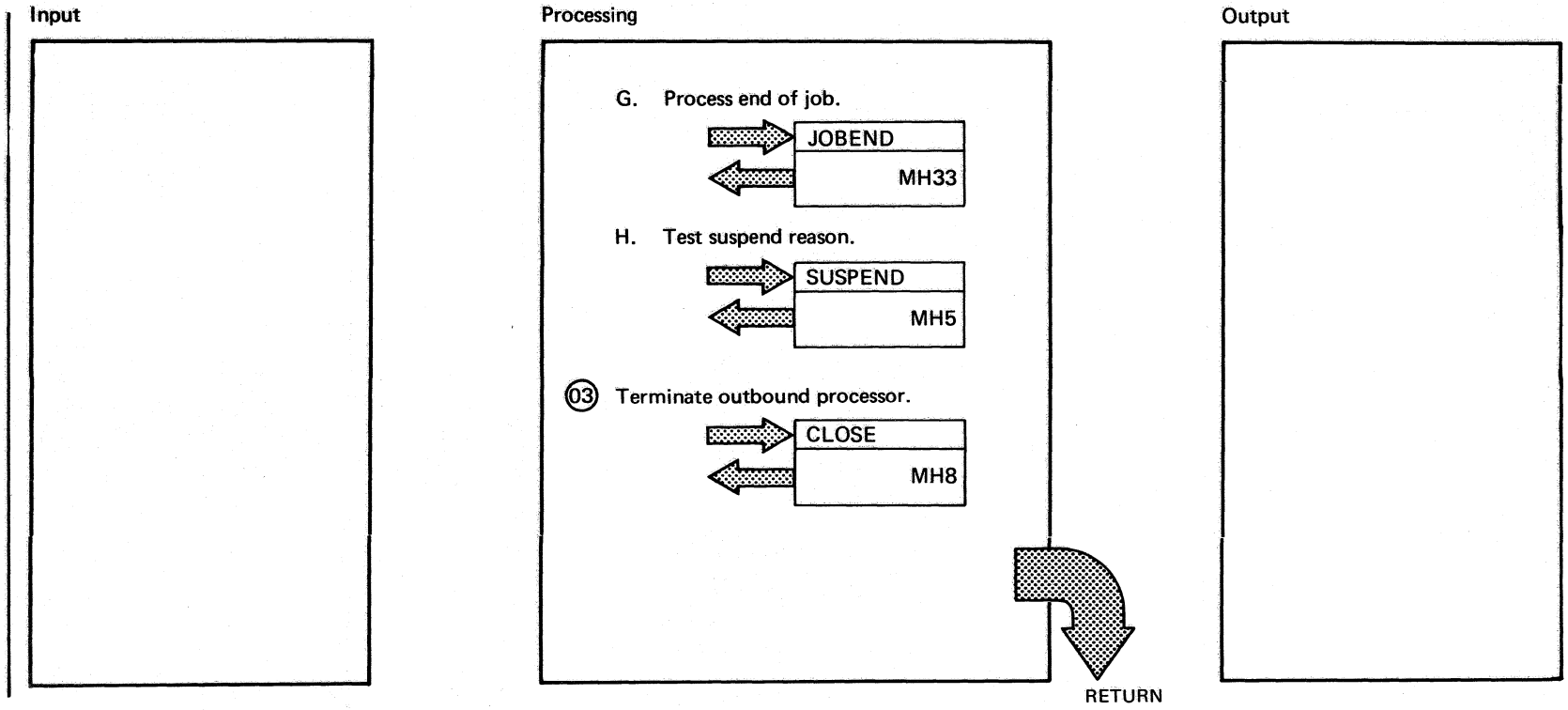
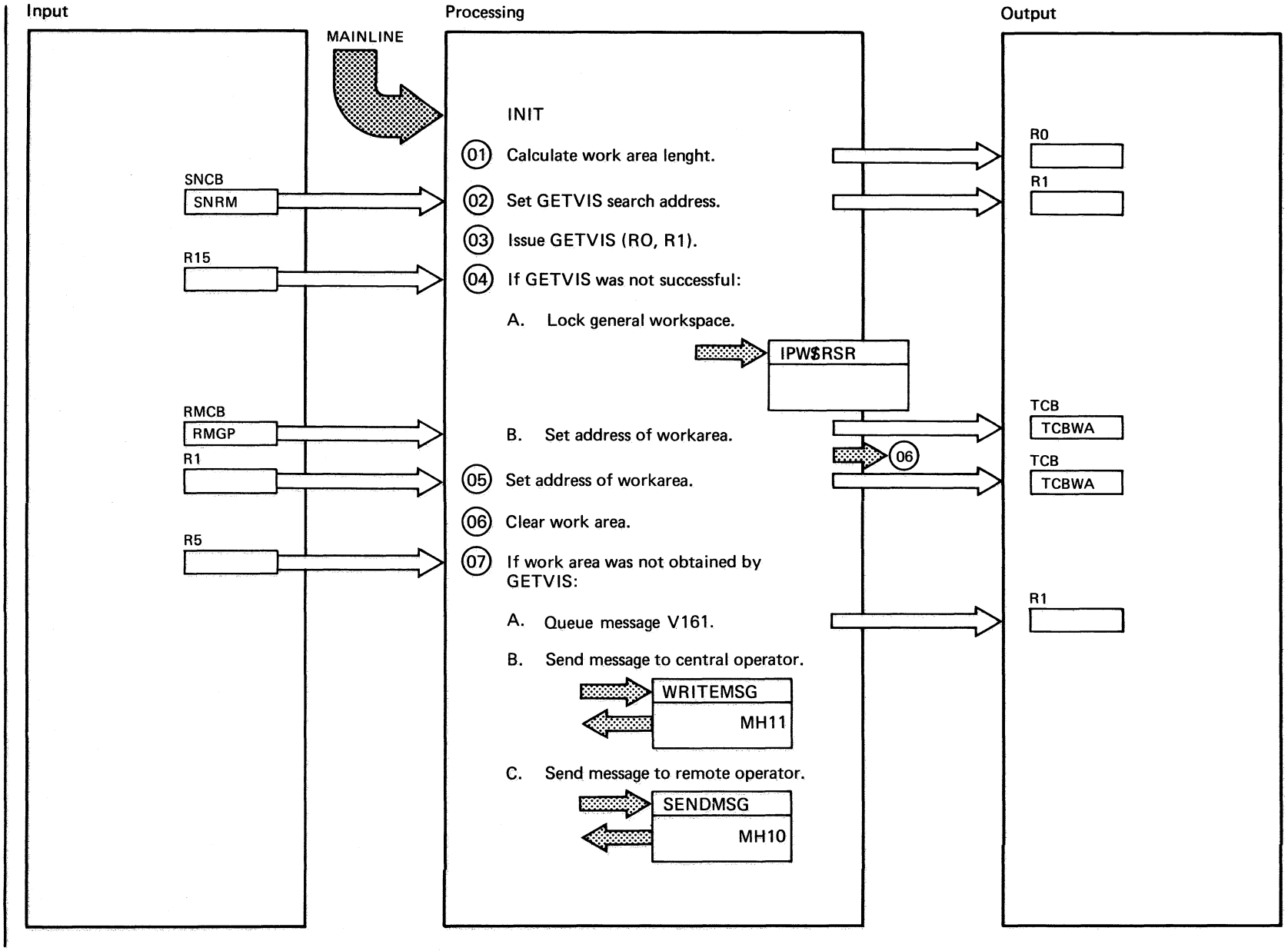

3
CLOSE
MH8
CHECK
MH18
CANCEL
MH17
SENDFMH
MH9
SENDMSG
MH10
WRITEMSG
MH11
CHECK
MH18
CHECK
MH18

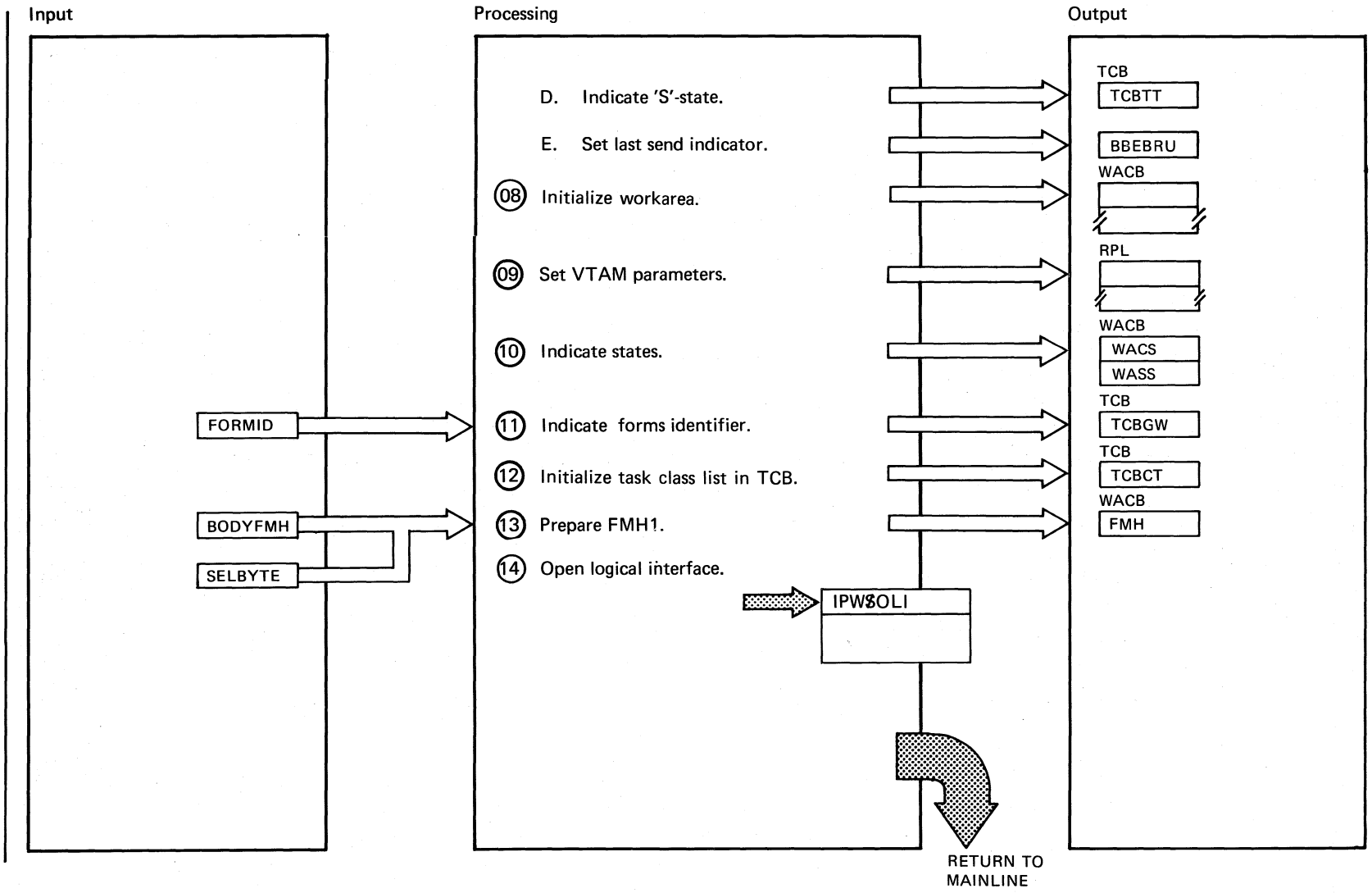
Chart MH1


1
INTVREQ
MH14
LUSTAT
MH15
GETSENSE
MH13
CANCEL
MH17
LUSTAT
MH15
SUSPEND1
MH6
WRITEMSG
MH11
CHECK
MH18
CHECK
MH18
CHECK
MH18
WRITEMSG
MH11
CHECK
MH18
SENDMSG
MH10
SUSPEND2
MH7
SENDFMH
MH9
CHECK
MH18


Input
Processing
Output
FROM IPW$$SN
MAINLINE
01 Initialize the SNAoutbound processor.
INIT
MH3
WACB
WAOF
TCB
TCBTT
02 If end of file not reached and 'S'-state not presented:
A. Get next record.
GETLR
MH31
WACB
WAOF
B. If end of file not reached:
C. Provide function management headers.
FMHDRS
MH32
WACB
WAOF
WAOJ
TCB
TCBTT
D. If 'S'-state not presented and more output available:
E. Build chain for one job.
CHAINJOB
MH19
F. Terminate outbound chain.
CHAINEND
MH4


Input
Processing
Output
G. Process end of job.
JOBEND
MH33
H. Test suspend reason.
SUSPEND
MH5
03 Terminate outbound processor.
CLOSE
MH8
RETURN


Input
Processing
Output
MAINLINE
INIT
01 Calculate work area lenght.
02 Set GETVIS search address.
03 Issue GETVIS (RO, R1).
04 If GETVIS was not successful:
A. Lock general workspace.
B. Set address of workarea.
05 Set address of workarea.
06 Clear work area.
07 If work area was not obtained by GETVIS:
A. Queue message V161.
B. Send message to central operator.
C. Send message to remote operator.
SNCB
SNRM
R15
RMCB
RMGP
R1
R5
IPW$RSR
WRITEMSG
MH11
SENDMSG
MH10
RO
R1
TCB
TCBWA
TCB
TCBWA
06
R1


Input
Processing
Output
D. Indicate 'S'-state.
E. Set last send indicator.
08 Initialize workarea.
09 Set VTAM parameters.
10 Indicate states.
11 Indicate forms identifier.
12 Initialize task class list in TCB.
13 Prepare FMH1.
14 Open logical interface.
FORMID
BODYFMH
SELBYTE
IPW$OLI
RETURN TO MAINLINE
TCB
TCBTT
BBEBRU
WACB
RPL
WACB
WACS
WASS
TCB
TCBGW
TCB
TCBCT
WACB
FMH

Included by MAINLINE, Chart MH2

CHAINEND (Part 1 of 3)

End the outbound chain and send the remaining buffer to the logical unit.

WACB
RPLACTIV
TCB
TCBTI
SUCB
SUCS
WACB
WAAS
WACR
WABP
WABI
RPLAREA
LUCB
LUCB

(1) Check if the RPL has to be checked
No Yes
Call CHECK → CHECK
Check if there is an error condition
RF
No Yes
Call TESTRPL → TESTRPL
(2)

(2) Check for interrupted outbound to console
No Yes
Check for ASCII
No Yes
Insert NL character
Update current record pointer
(3)
Insert SCB + NL character
Update current record pointer

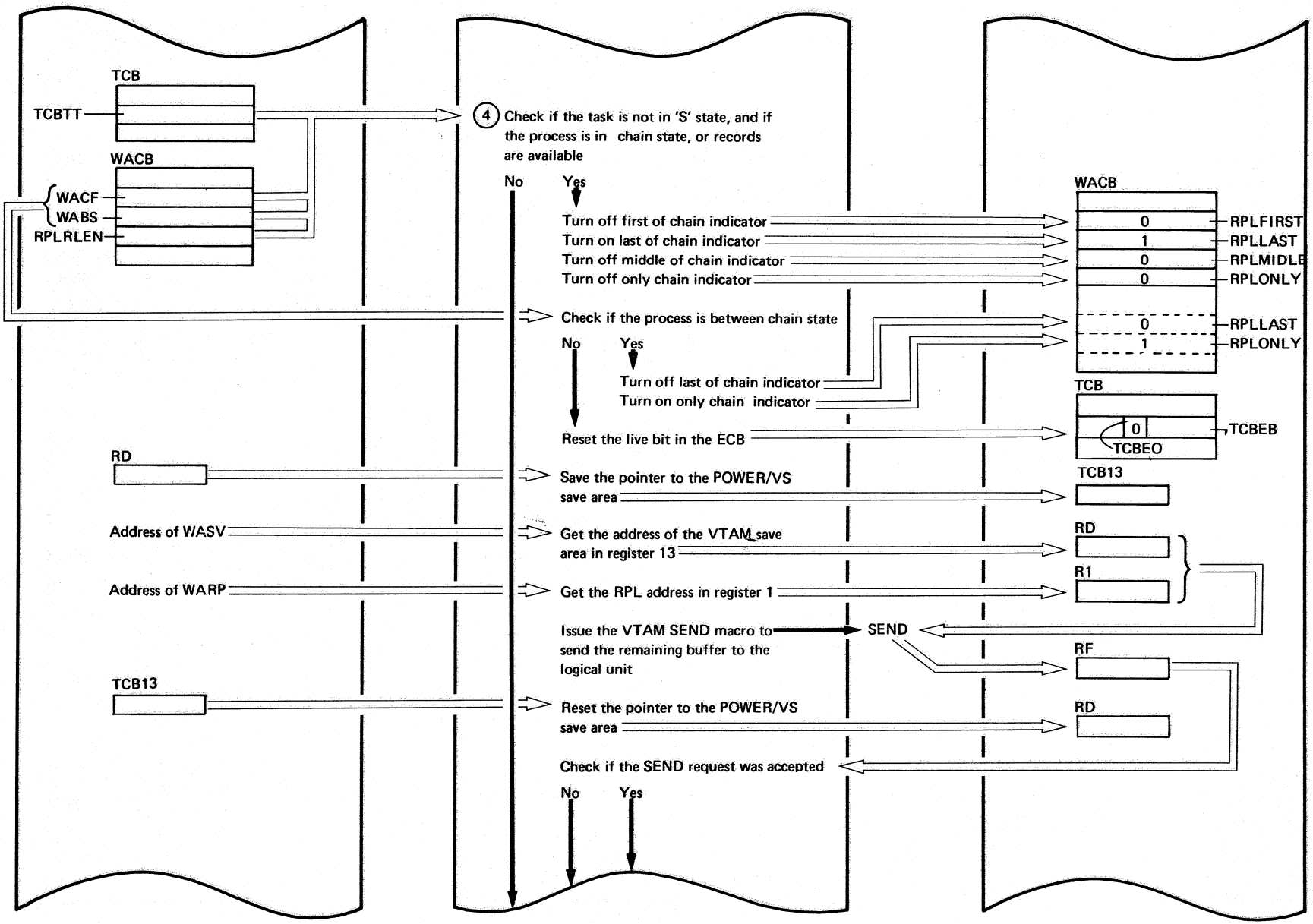
(3) Swap the buffers:
Indicate length of RU to be sent
Indicate buffer in process has to be sent
Buffer to send becomes buffer to process
Buffer in process becomes buffer to send
Set current pointer to beginning of RU
Indicate empty buffer

(A)
(A)

WACB
X'1A' WARU
+1 WACR
X'0115' WARU
+2 WACR
RPLRLEN
RPLAREA
WABP
WABI
WACR
WARC

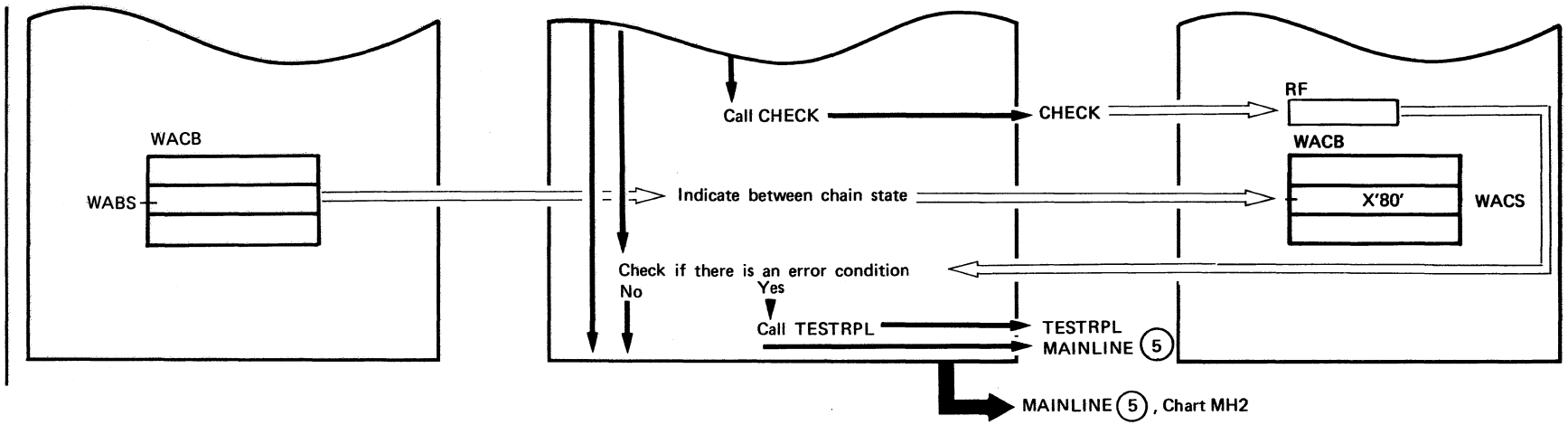
Continued on next page

CHAINEND (Part 2 of 3)

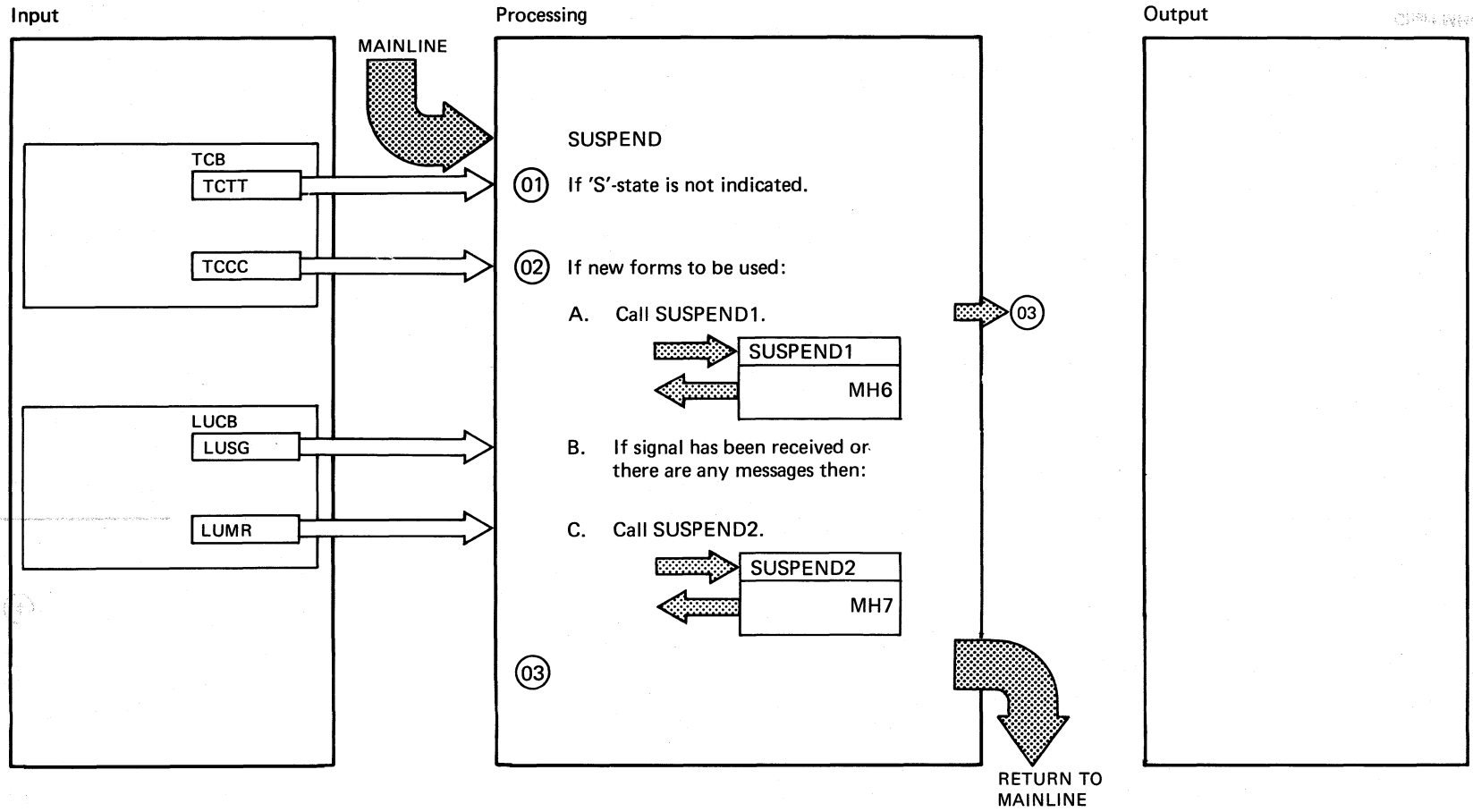


(Continued on next page)

CHAINEND (Part 3 of 3)

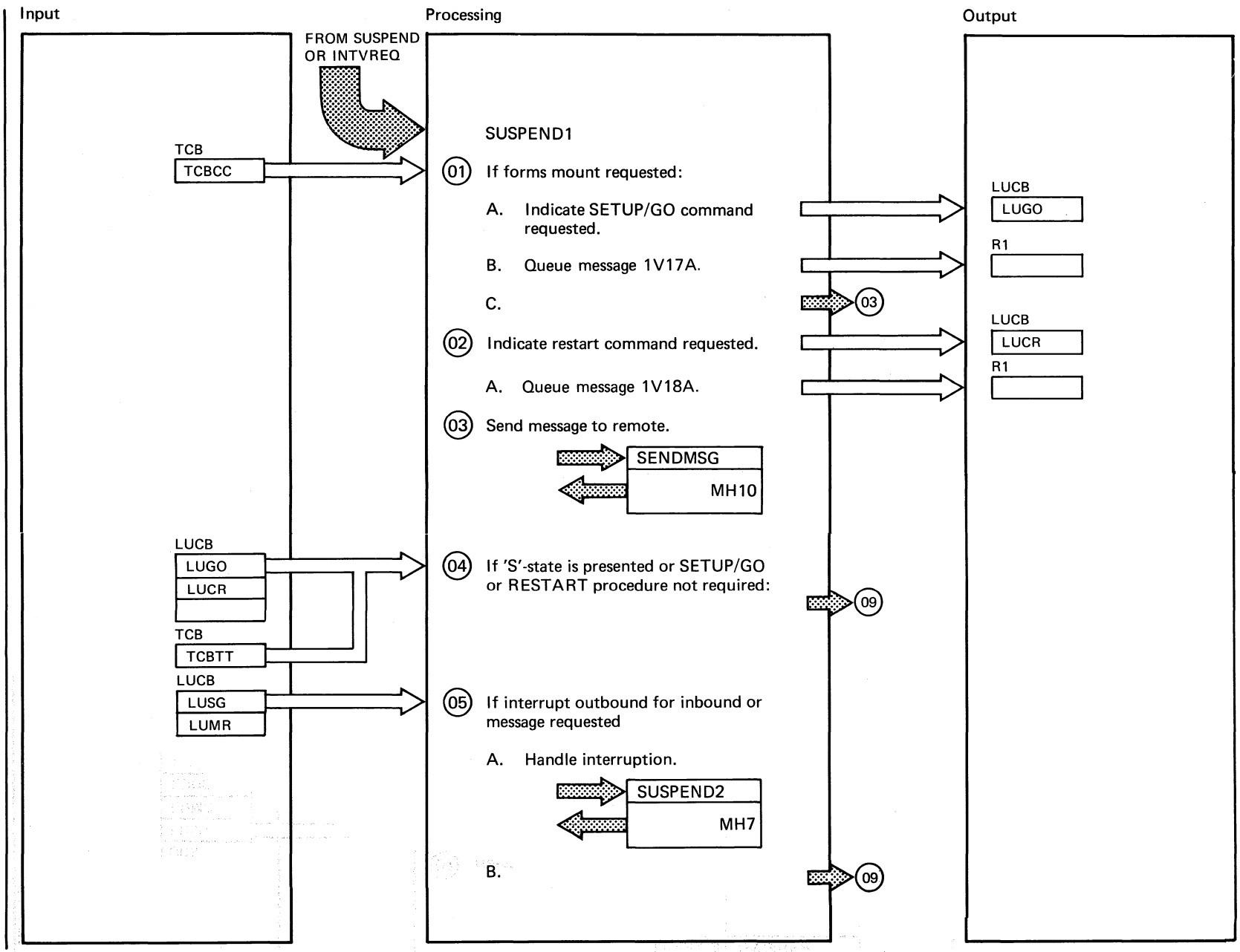


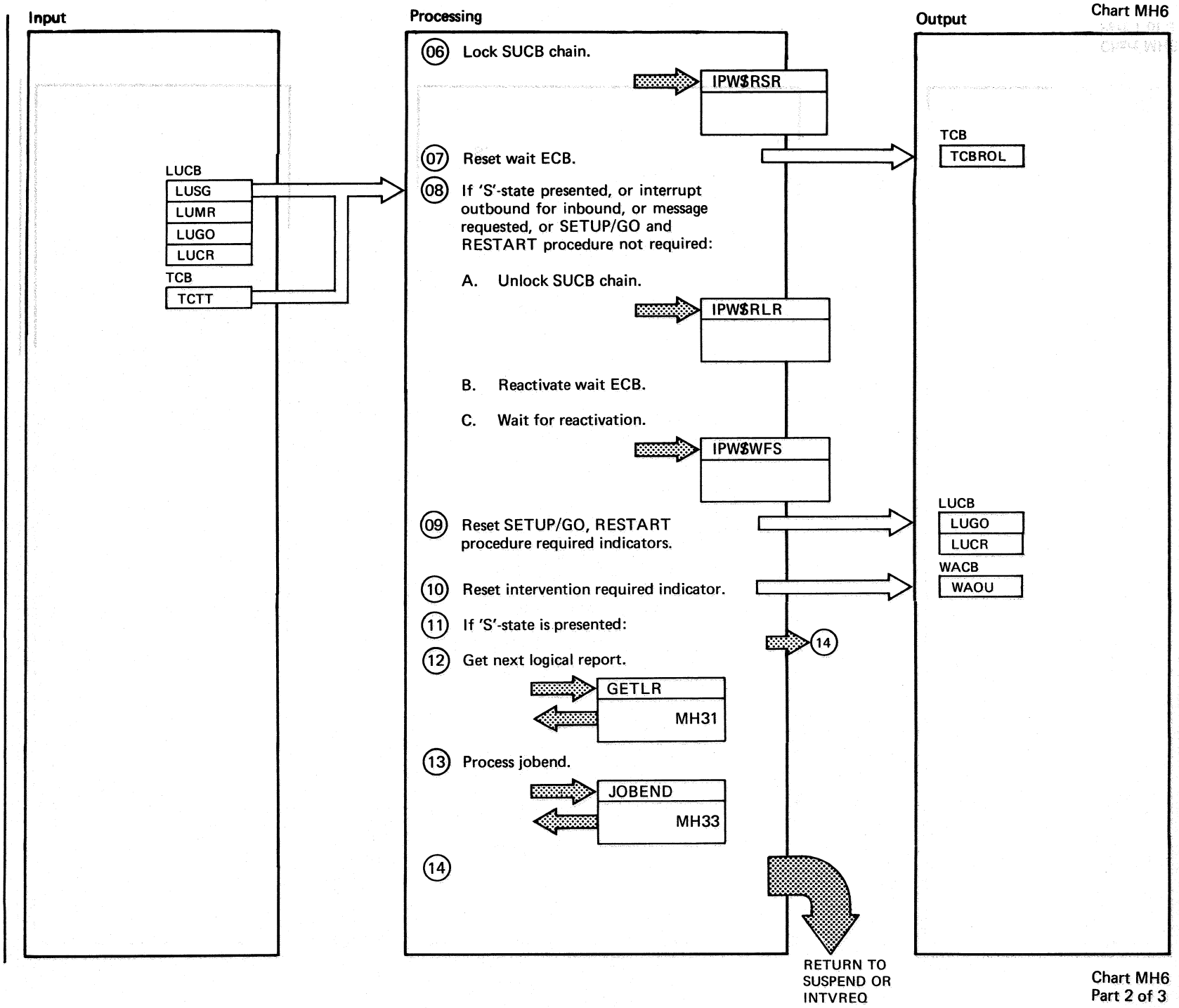
Extended Description	Include Segment	Call Subroutine/Macro	Chart
1		CHECK	MH18
		TESTRPL	MH12
4		SEND (VTAM)	
		CHECK	MH18
		TESTRPL	MH12



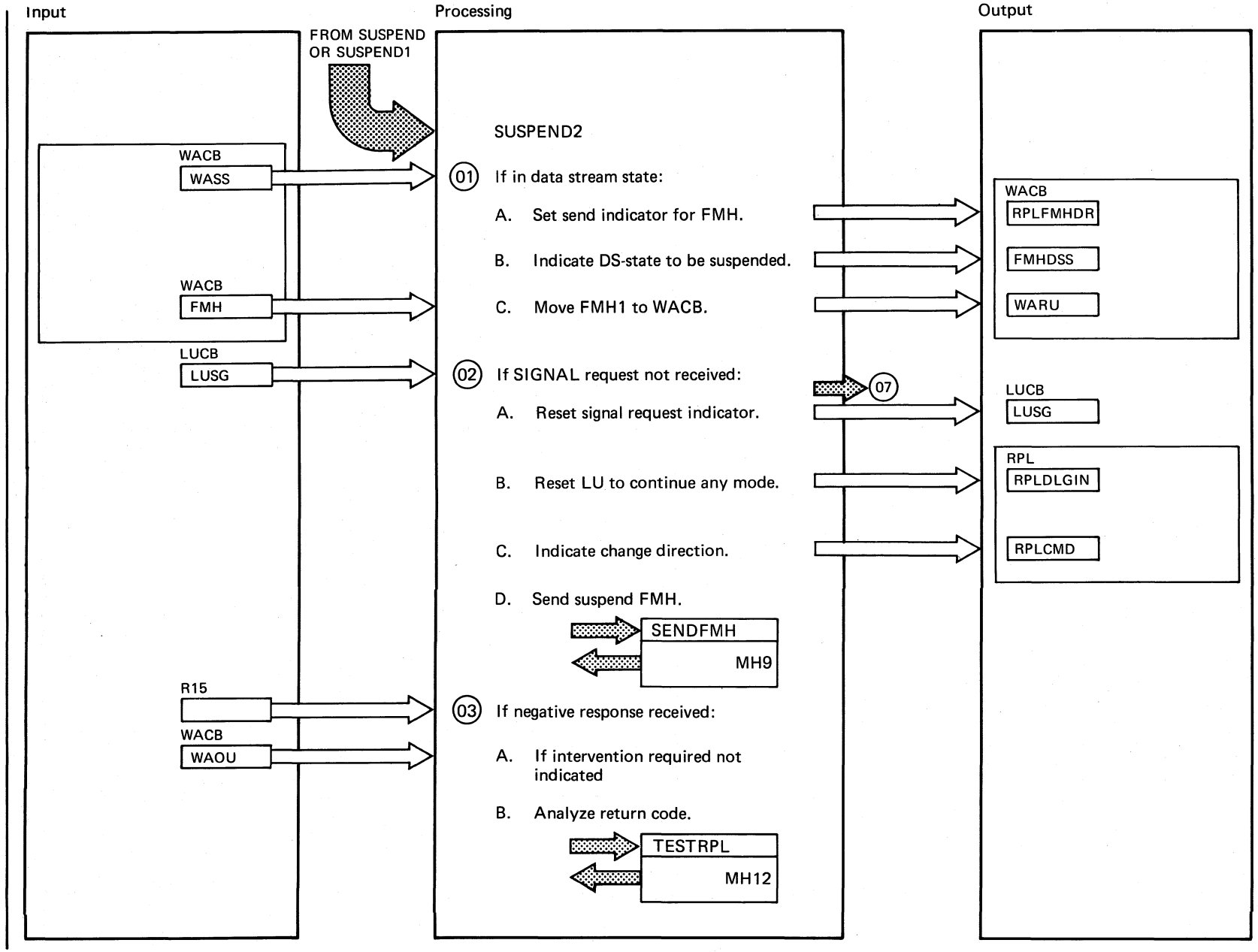
Extended Description

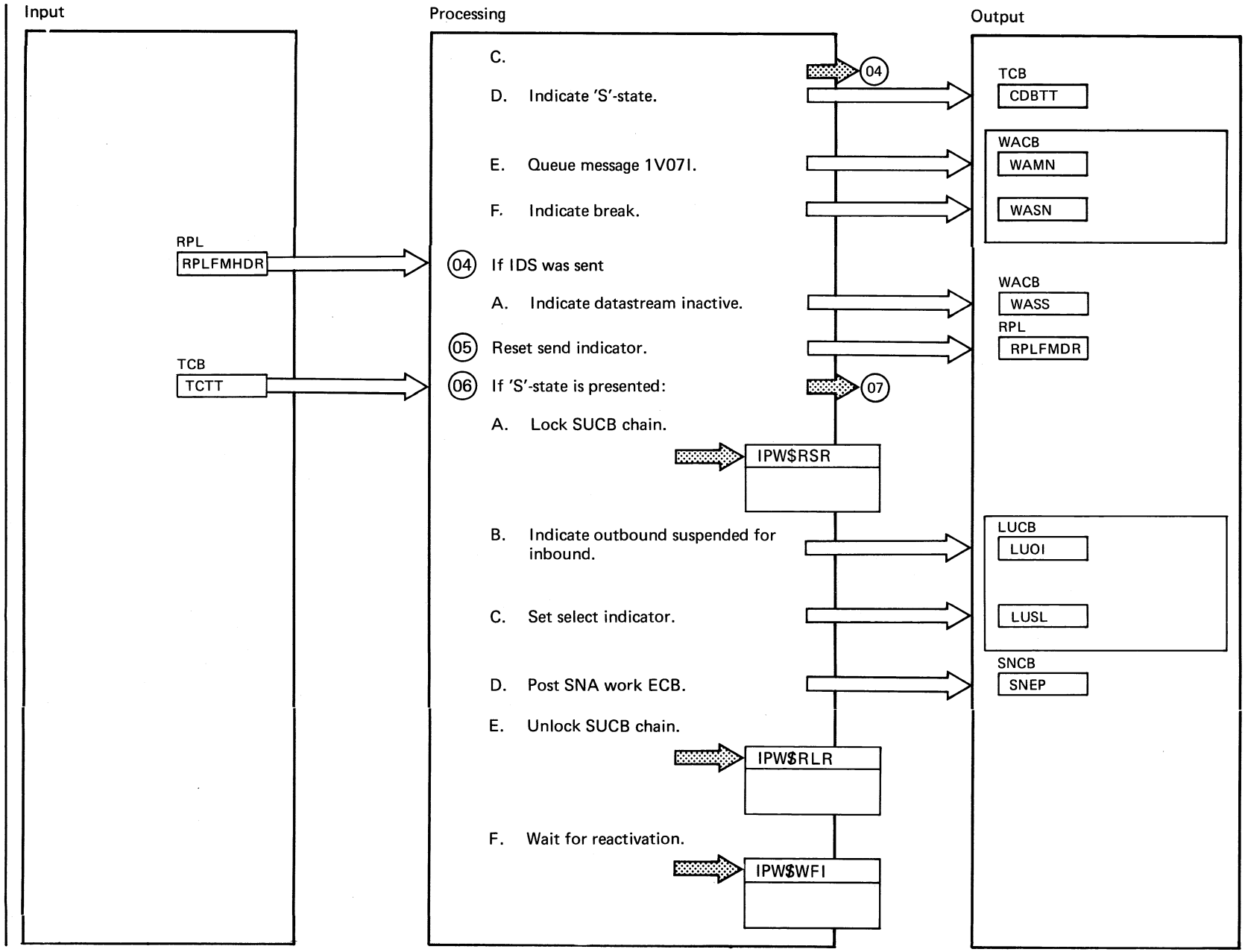
	Include Segment	Call Subroutine/ Macro	Chart
02A Handle SUSPEND for forms mount procedure.			
02B Allow interrupt for $IB or $MP.			
03 No suspend reason found.			

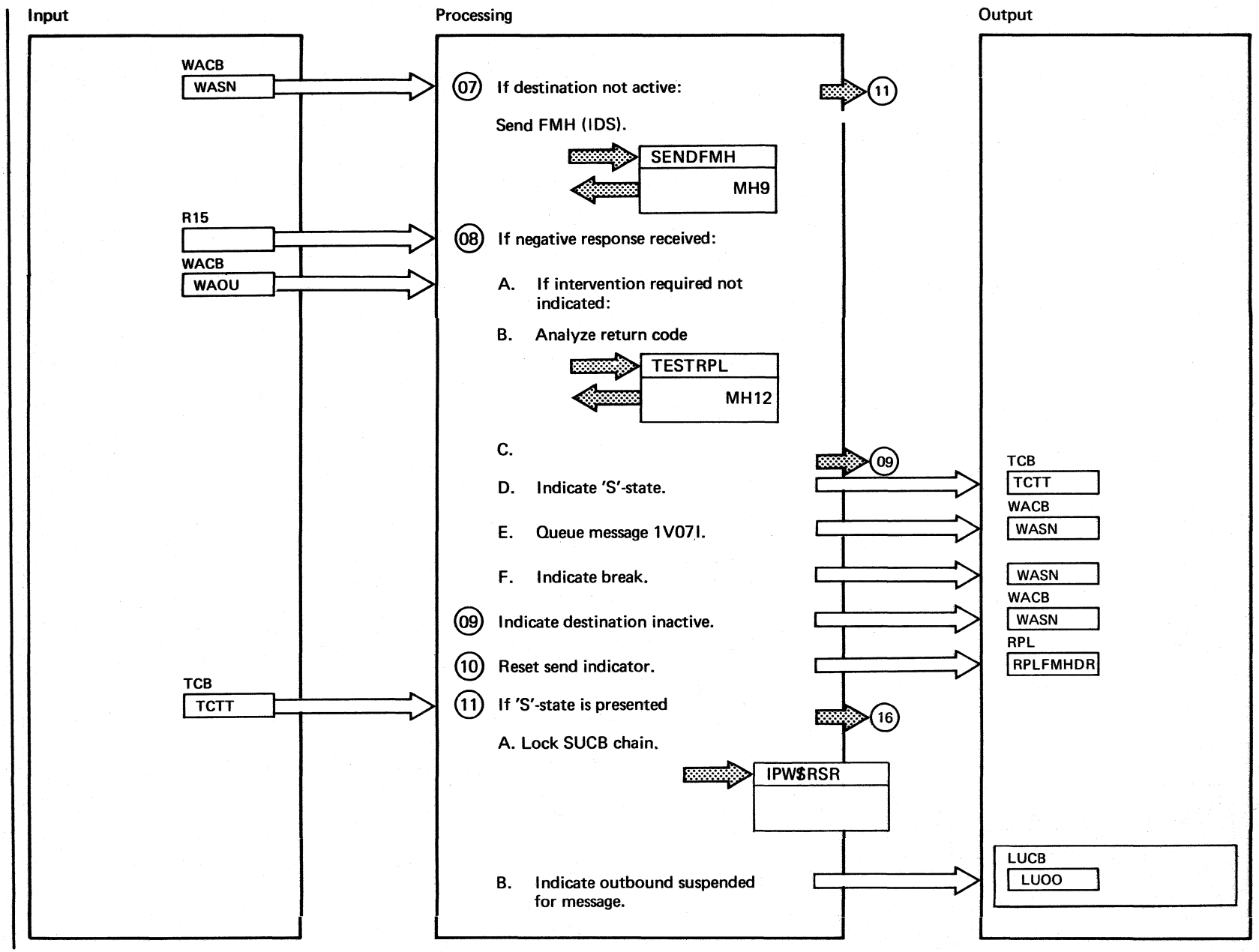

Input
Processing
Output
FROM SUSPEND
OR INTVREQ
SUSPEND1
TCB
TCBCC
01 If forms mount requested:
A. Indicate SETUP/GO command requested.
B. Queue message 1V17A.
C.
03
02 Indicate restart command requested.
A. Queue message 1V18A.
03 Send message to remote.
SENDMSG
MH10
LUCB
LUGO
R1
LUCB
LUCR
R1
LUCB
LUGO
LUCR
TCB
TCBTT
LUCB
LUSG
LUMR
04 If 'S'-state is presented or SETUP/GO or RESTART procedure not required:
09
05 If interrupt outbound for inbound or message requested
A. Handle interruption.
SUSPEND2
MH7
B.
09

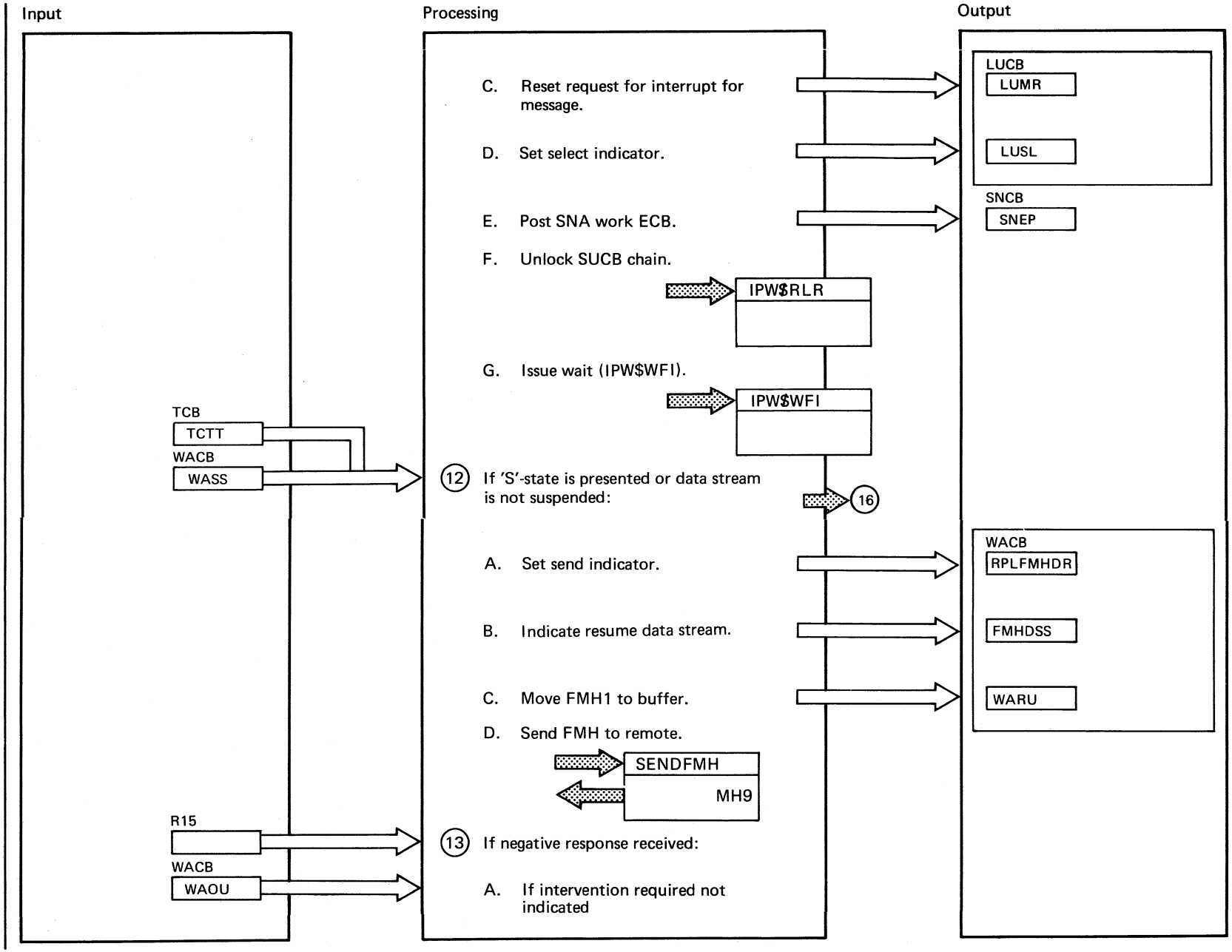

Input
Processing
Output
Chart MH6
LUCB
LUSG
LUMR
LUGO
LUCR
TCB
TCTT
06 Lock SUCB chain.
IPW$RSR
07 Reset wait ECB.
08 If 'S'-state presented, or interrupt outbound for inbound, or message requested, or SETUP/GO and RESTART procedure not required:
A. Unlock SUCB chain.
IPW$RLR
B. Reactivate wait ECB.
C. Wait for reactivation.
IPW$WFS
09 Reset SETUP/GO, RESTART procedure required indicators.
10 Reset intervention required indicator.
11 If 'S'-state is presented:
12 Get next logical report.
GETLR
MH31
13 Process jobend.
JOBEND
MH33
14
TCB
TCBROL
LUCB
LUGO
LUCR
WACB
WAOU
14
RETURN TO SUSPEND OR INTVREQ
Chart MH6
Part 2 of 3

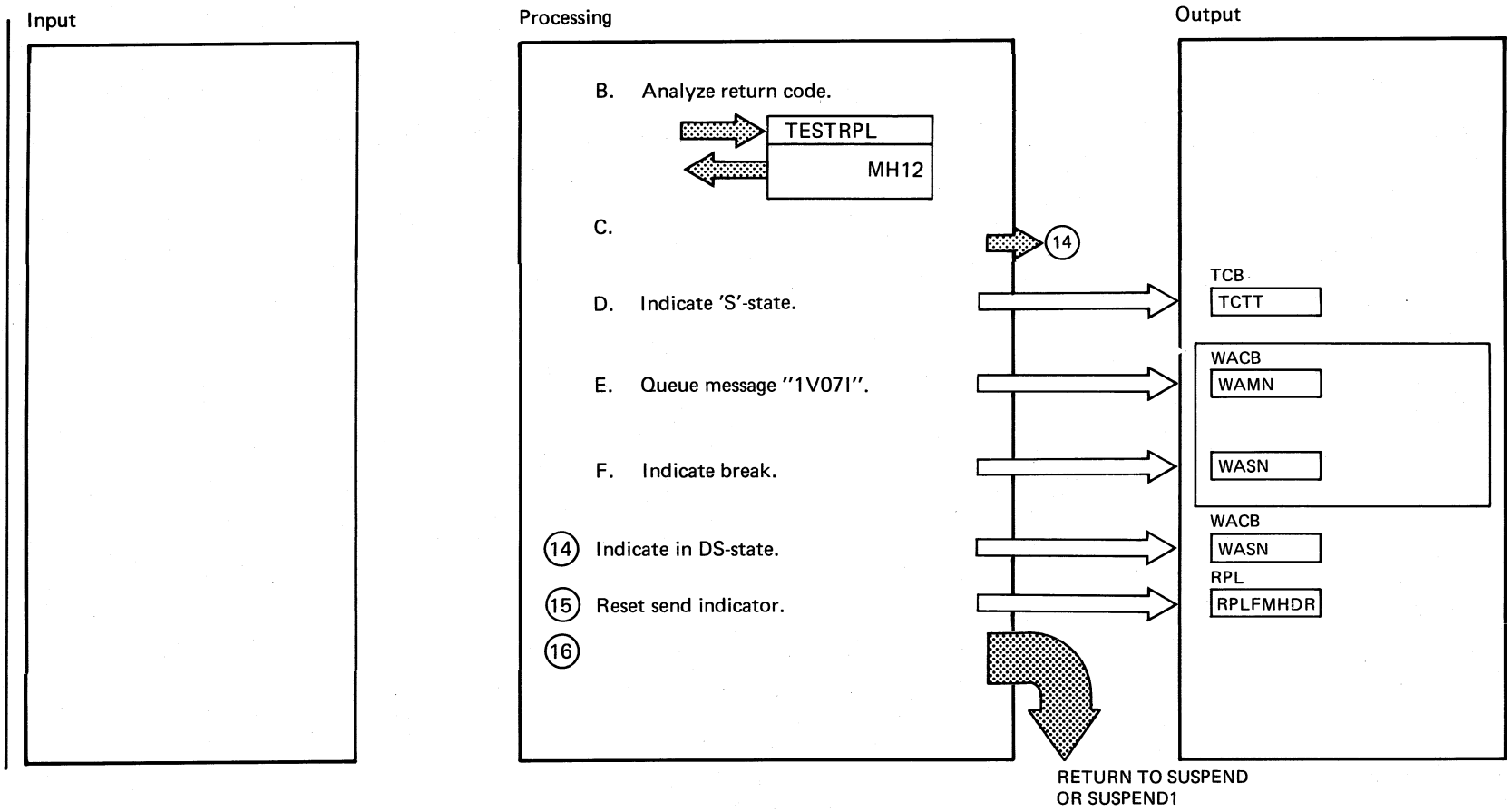
Extended Description	Include Segment	Call Subroutine/ Macro	Chart
SUSPEND1 may be entered because of: A. Activation of SETUP/GO procedure. B. Interrupt outbound destination for pending message (s) to be send by message processor on this session. C. Interrupt outbound destination and set secondary logical unit into send state to allow remote station to submit any inbound data. Initiation of SETUP/GO procedure is signaled by the logical writer by setting the command request word to blanks, when: D. The logical writer detects a new queue set that requires new forms on printer or punch. In this case message '1Q40A' has been queued by the remote station by logical writer. E. The end of the setup stream is reached. (01B) Message 1V17A: 1V17A "TTT" SUSPENDED FOR FORMS MOUNT. Message 1V18A: (02A) 1V18A "TTT" REPLY WITH RESTART ON INTERVENTION REQUIRED. (05B) SUSPEND2 may return with TCBTT set to STOP, FLUSH, HOLD. (08C) Communications with remote operator for SETUP/GO procedure may be performed on different sessions. In that case the outbound processor must wait for further notification and may be reactivated by either - SNA manager or - Inbound processor or - Message processor. After reactivation message processor will reset LUMR, outbound processor resets LUSG or inbound processor will reset LUCR. (12) This is to replace dummy record and make either SETUP/GO procedure or restart command effective.			


Input
Processing
Output
FROM SUSPEND
OR SUSPEND1
SUSPEND2
WACB
WASS
WACB
FMH
LUCB
LUSG
R15
WACB
WAOU
01 If in data stream state:
A. Set send indicator for FMH.
B. Indicate DS-state to be suspended.
C. Move FMH1 to WACB.
02 If SIGNAL request not received:
07
A. Reset signal request indicator.
B. Reset LU to continue any mode.
C. Indicate change direction.
D. Send suspend FMH.
SENDFMH
MH9
03 If negative response received:
A. If intervention required not indicated
B. Analyze return code.
TESTRPL
MH12
WACB
RPLFMHDR
FMHDSS
WARU
LUCB
LUSG
RPL
RPLDLGIN
RPLCMD

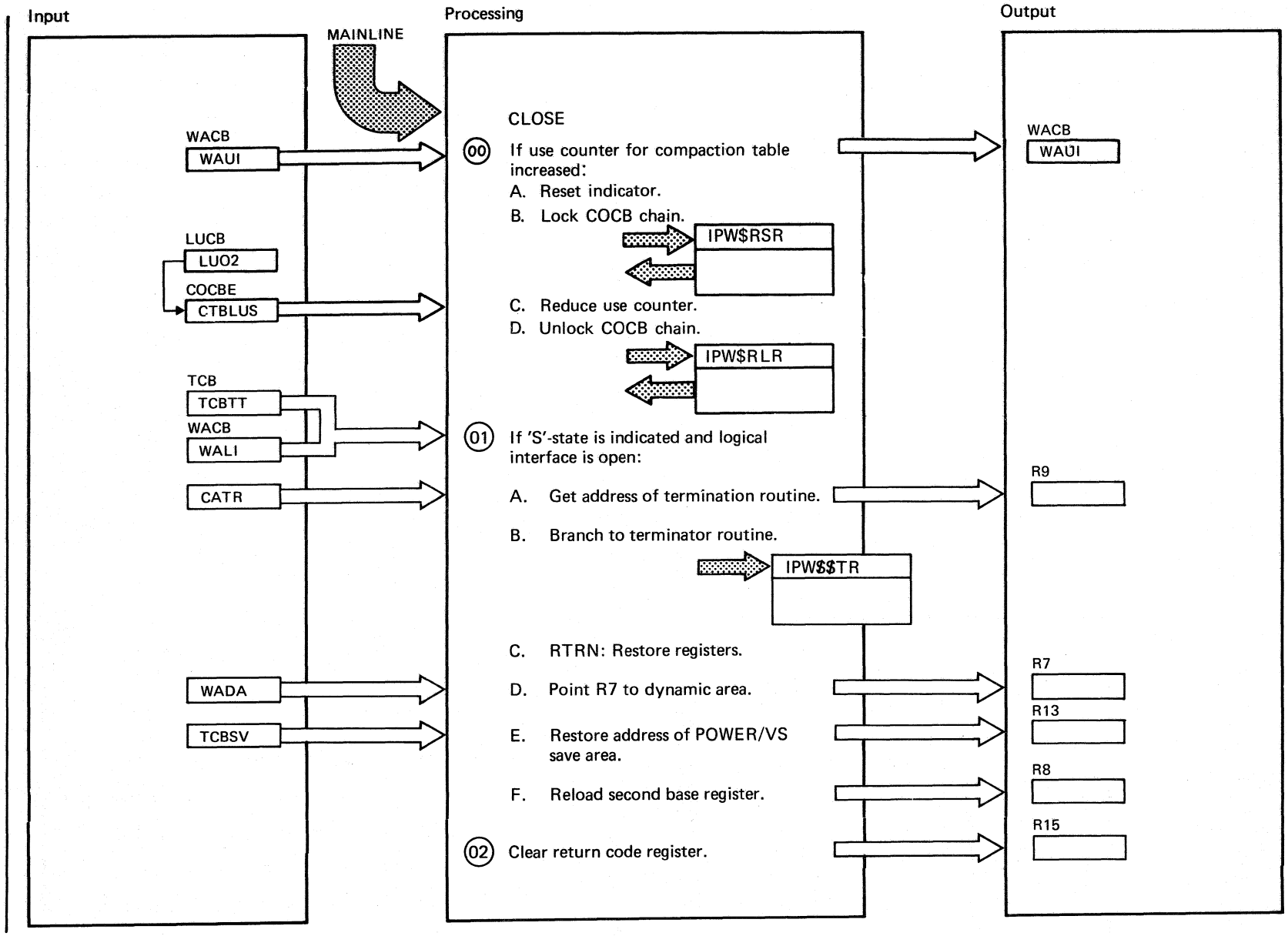

Input
Processing
Output
C.
04
D. Indicate 'S'-state.
TCB
CDBTT
WACB
WAMN
E. Queue message 1V07I.
WASN
F. Indicate break.
RPL
RPLFMHDR
04 If IDS was sent
WACB
WASS
A. Indicate datastream inactive.
RPL
RPLFMDR
05 Reset send indicator.
TCB
TCTT
06 If 'S'-state is presented:
07
A. Lock SUCB chain.
IPW$RSR
LUCB
LUOI
B. Indicate outbound suspended for inbound.
C. Set select indicator.
LUSL
SNCB
SNEP
D. Post SNA work ECB.
E. Unlock SUCB chain.
IPW$RLR
F. Wait for reactivation.
IPW$WFI

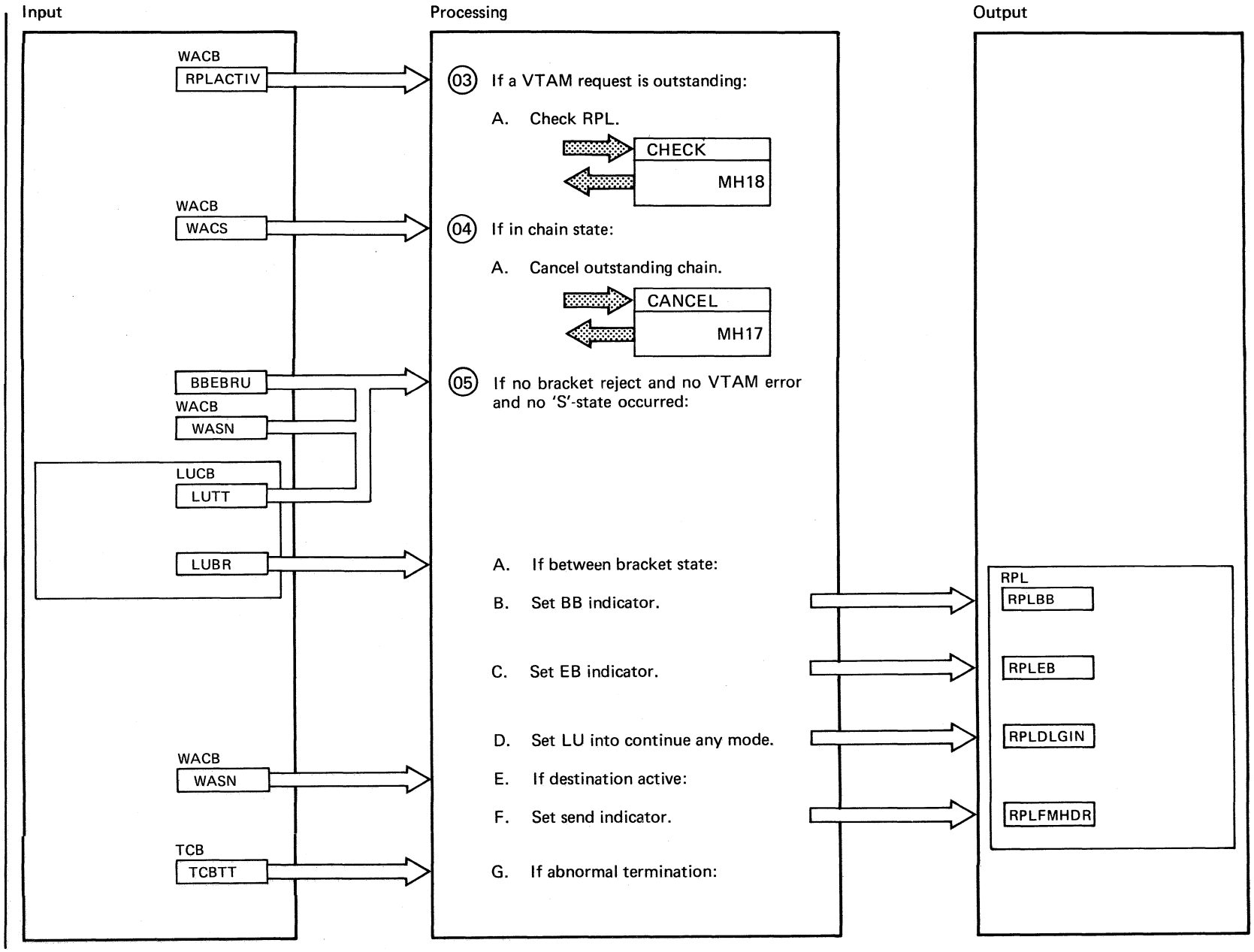

Input
Processing
Output
WACB
WASN
R15
WACB
WAOU
TCB
TCTT
07 If destination not active:
11
Send FMH (IDS).
SENDFMH
MH9
08 If negative response received:
A. If intervention required not indicated:
B. Analyze return code
TESTRPL
MH12
C.
09
D. Indicate 'S'-state.
E. Queue message 1V07I.
F. Indicate break.
09 Indicate destination inactive.
10 Reset send indicator.
11 If 'S'-state is presented
16
A. Lock SUCB chain.
IPW$RSR
B. Indicate outbound suspended for message.
TCB
TCTT
WACB
WASN
WASN
WACB
WASN
RPL
RPLFMHDR
LUCB
LUOO

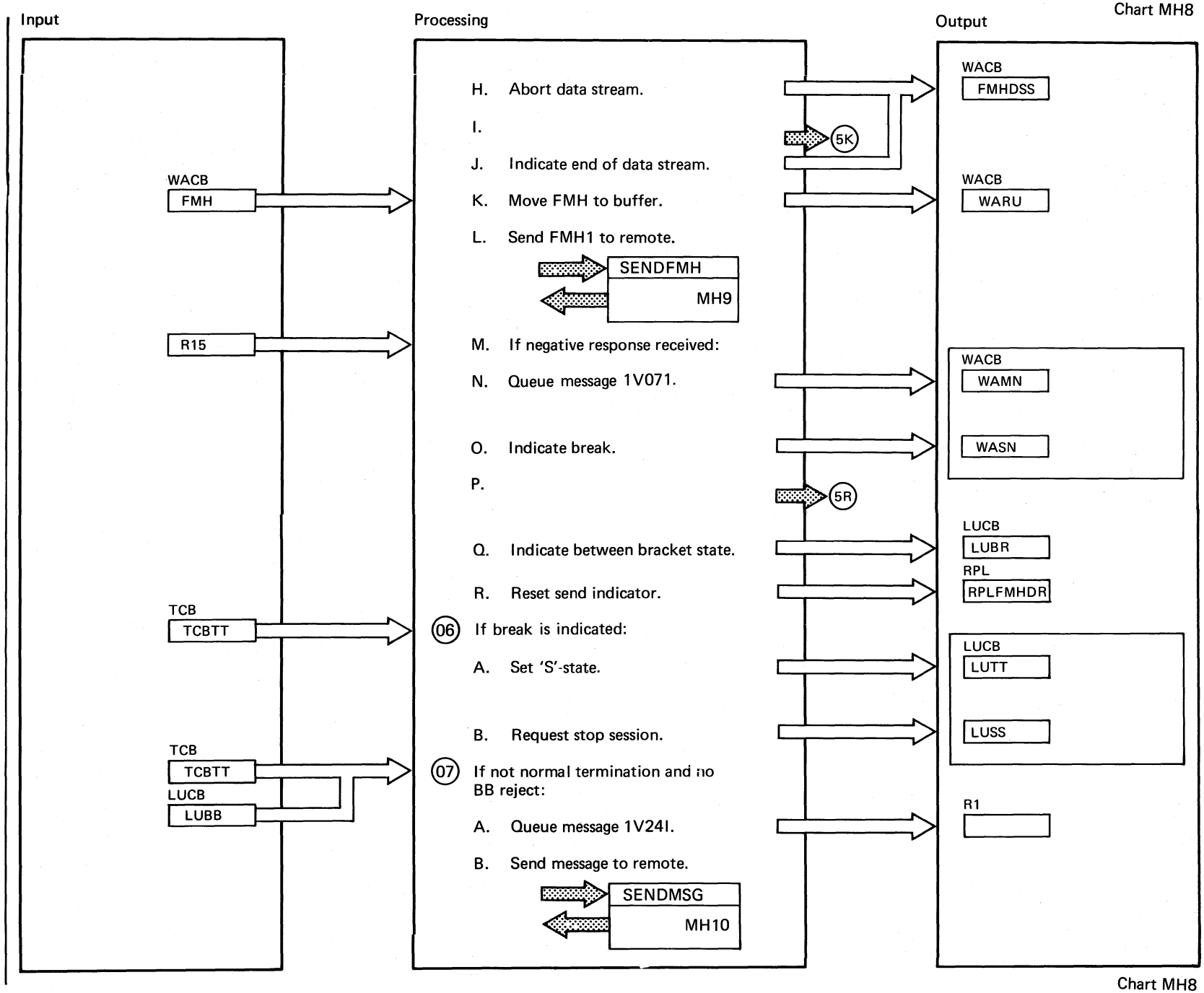

Input
Processing
Output
C. Reset request for interrupt for message.
D. Set select indicator.
E. Post SNA work ECB.
F. Unlock SUCB chain.
IPW$RLR
G. Issue wait (IPW$WFI).
IPW$WFI
LUCB
LUMR
LUSL
SNCB
SNEP
TCB
TCTT
WACB
WASS
(12) If 'S'-state is presented or data stream is not suspended:
(16)
A. Set send indicator.
B. Indicate resume data stream.
C. Move FMH1 to buffer.
D. Send FMH to remote.
SENDFMH
MH9
WACB
RPLFMHDR
FMHDSS
WARU
R15
WACB
WAOU
(13) If negative response received:
A. If intervention required not indicated

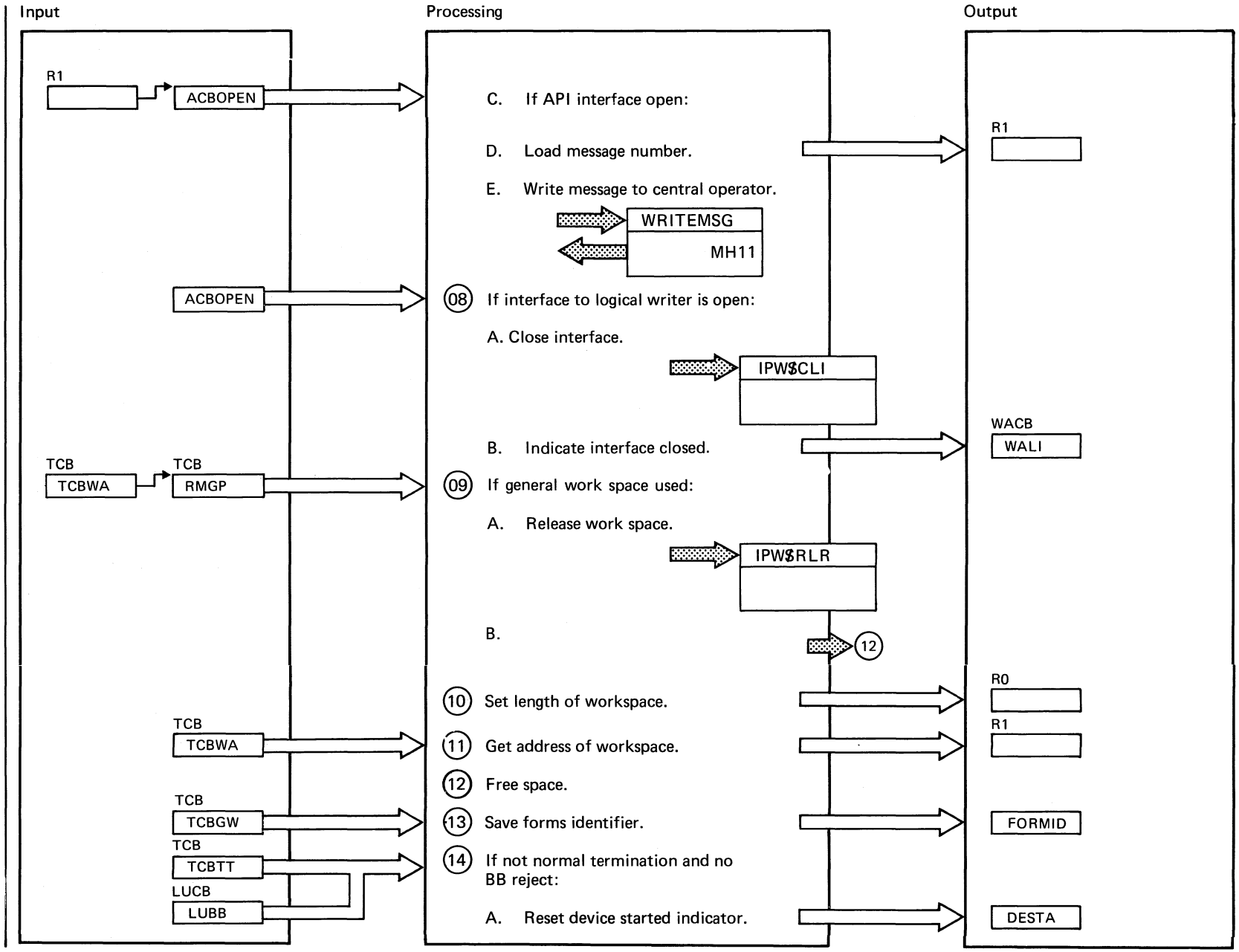

Input
Processing
Output
B. Analyze return code.
TESTRPL
MH12
C.
14
D. Indicate 'S'-state.
E. Queue message "1V07I".
F. Indicate break.
14 Indicate in DS-state.
15 Reset send indicator.
16
TCB
TCTT
WACB
WAMN
WASN
WACB
WASN
RPL
RPLFMHDR
RETURN TO SUSPEND
OR SUSPEND1

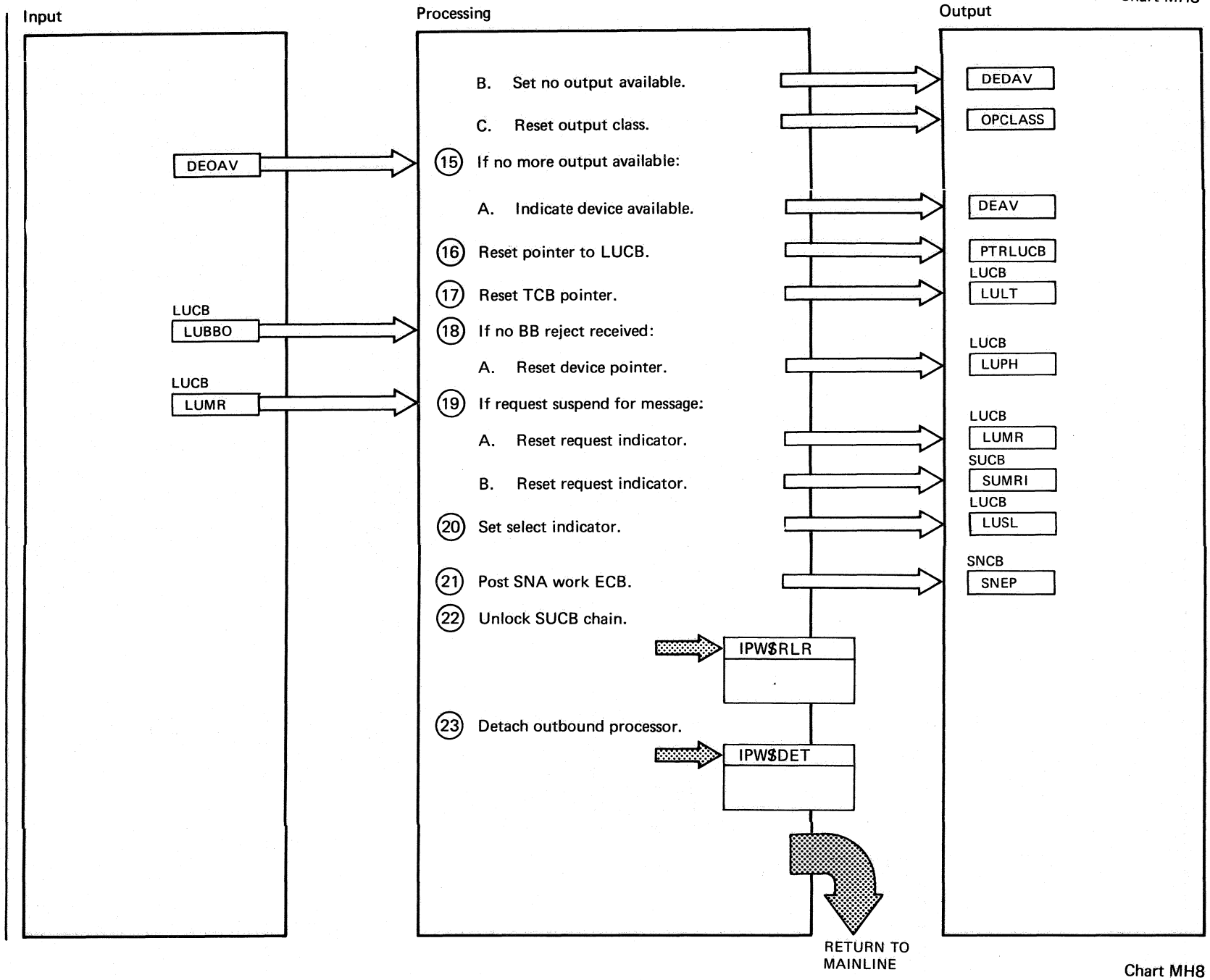
Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(01) When SUSPEND2 is entered the data stream state may be A. INDS-state (interruption while processing a job). B. BETDS-state (interruption between job boundaries). FMH1 (SDS) is only sent in case of INDS-state. FMH1 (RDS) is only sent in case of SUSPEND DS. (03E) Message 1V071: 1V071 Error on 'send' 'rtncd', 'fdbk2 = xx, yy' 'sense = xxxx' 'luname'. Message 1V071: 1V071 error on 'send' 'rtncd', 'fdbk2 = xx, yy' 'sense = xxxx' 'luname'.			

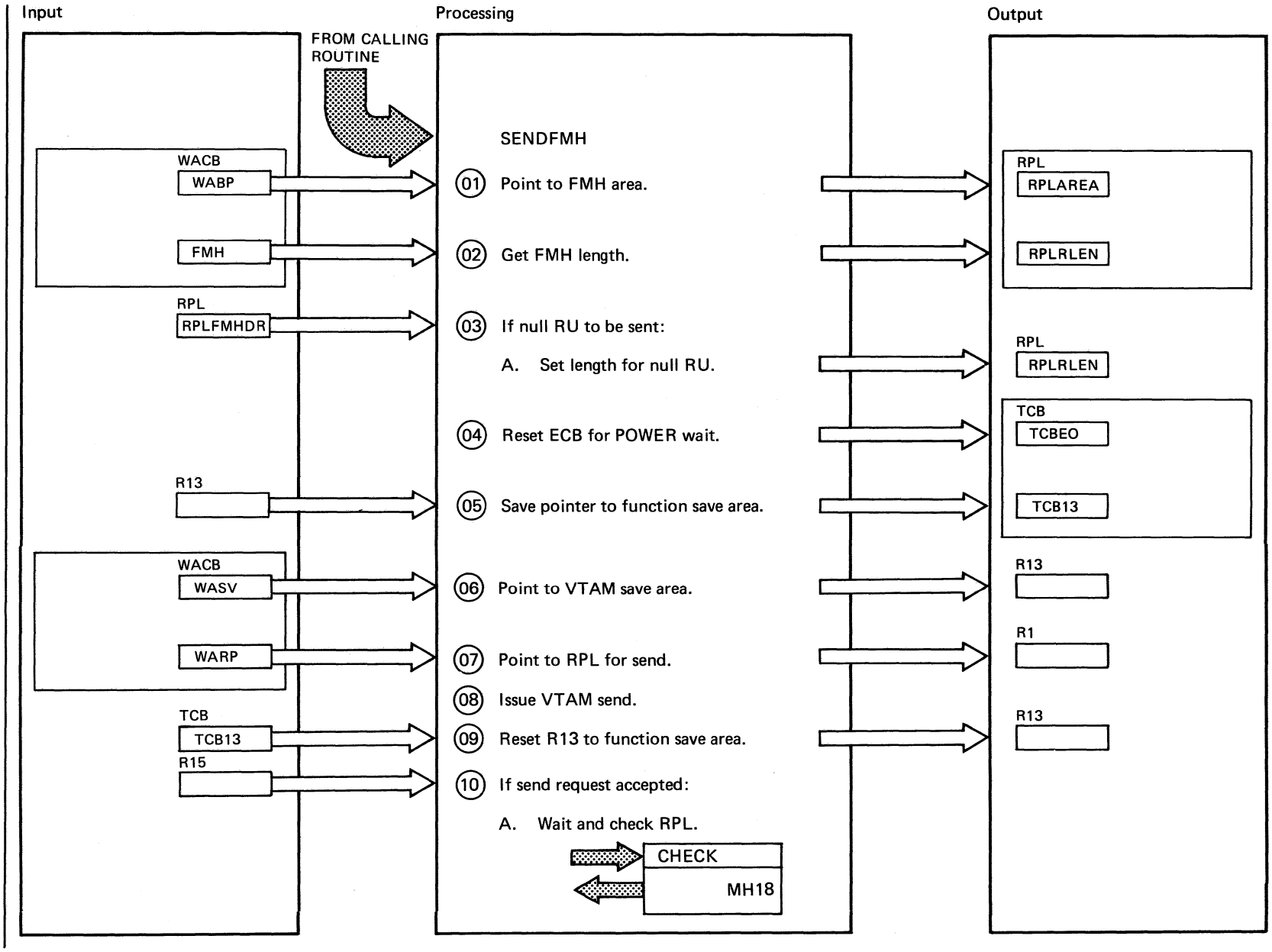

Input
Processing
Output
MAINLINE
WACB
WAUI
LUCB
LUO2
COCBE
CTBLUS
TCB
TCBTT
WACB
WALI
CATR
WADA
TCBSV
CLOSE
00 If use counter for compaction table increased:
A. Reset indicator.
B. Lock COCB chain.
IPW$RSR
C. Reduce use counter.
D. Unlock COCB chain.
IPW$RLR
01 If 'S'-state is indicated and logical interface is open:
A. Get address of termination routine.
B. Branch to terminator routine.
IPW$$TR
C. RTRN: Restore registers.
D. Point R7 to dynamic area.
E. Restore address of POWER/VS save area.
F. Reload second base register.
02 Clear return code register.
WACB
WAUI
R9
R7
R13
R8
R15

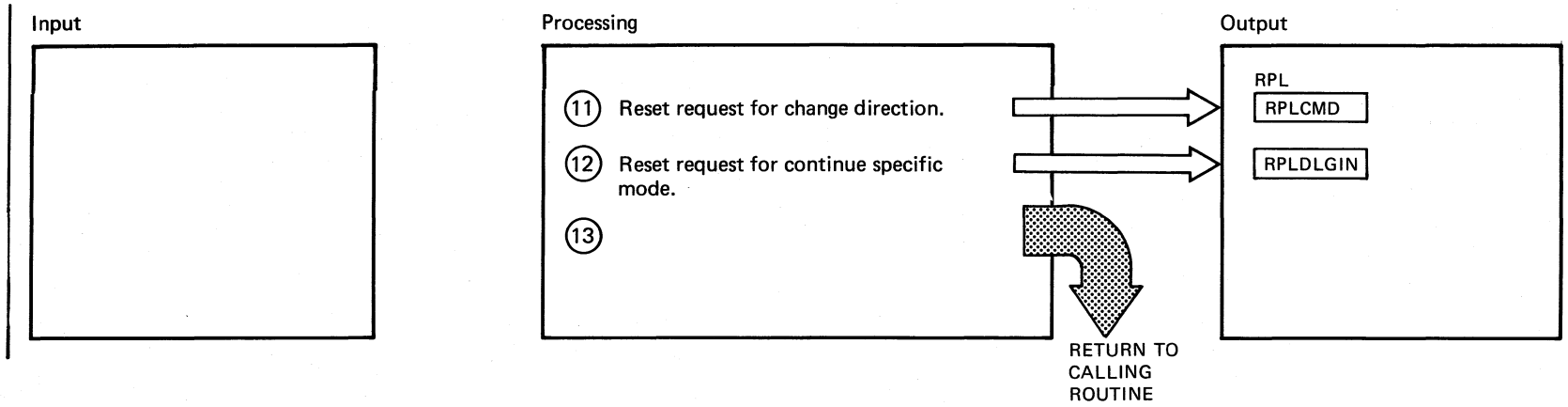

Input
Processing
Output
WACB
RPLACTIV
03 If a VTAM request is outstanding:
A. Check RPL.
CHECK
MH18
WACB
WACS
04 If in chain state:
A. Cancel outstanding chain.
CANCEL
MH17
BBEBRU
05 If no bracket reject and no VTAM error and no 'S'-state occurred:
WACB
WASN
LUCB
LUTT
LUBR
A. If between bracket state:
B. Set BB indicator.
C. Set EB indicator.
D. Set LU into continue any mode.
E. If destination active:
F. Set send indicator.
WACB
WASN
TCB
TCBTT
G. If abnormal termination:
RPL
RPLBB
RPLEB
RPLDLGIN
RPLFMHDR

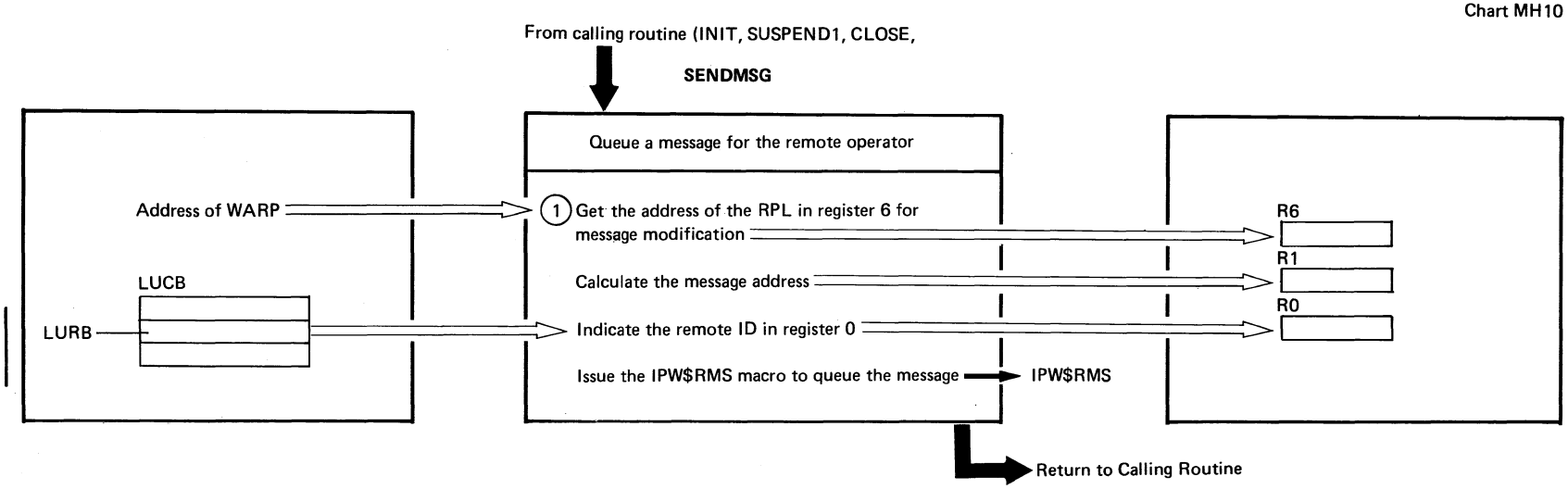

Input
Processing
Output
WACB
FMH
R15
TCB
TCBTT
TCB
TCBTT
LUCB
LUBB
H. Abort data stream.
I.
J. Indicate end of data stream.
K. Move FMH to buffer.
L. Send FMH1 to remote.
SENDFMH
MH9
M. If negative response received:
N. Queue message 1V071.
O. Indicate break.
P.
Q. Indicate between bracket state.
R. Reset send indicator.
06 If break is indicated:
A. Set 'S'-state.
B. Request stop session.
07 If not normal termination and no BB reject:
A. Queue message 1V24I.
B. Send message to remote.
SENDMSG
MH10
5K
5R
WACB
FMHDSS
WACB
WARU
WACB
WAMN
WASN
LUCB
LUBR
RPL
RPLFMHDR
LUCB
LUTT
LUSS
R1


Input
Processing
Output
R1
ACBOPEN
C. If API interface open:
D. Load message number.
E. Write message to central operator.
WRITEMSG
MH11
R1
ACBOPEN
08 If interface to logical writer is open:
A. Close interface.
IPW$CLI
WACB
WALI
B. Indicate interface closed.
TCB
TCBWA
TCB
RMGP
09 If general work space used:
A. Release work space.
IPW$RLR
B.
12
R0
10 Set length of workspace.
TCB
TCBWA
R1
11 Get address of workspace.
12 Free space.
TCB
TCBGW
13 Save forms identifier.
FORMID
TCB
TCBTT
14 If not normal termination and no BB reject:
LUCB
LUBB
A. Reset device started indicator.
DESTA

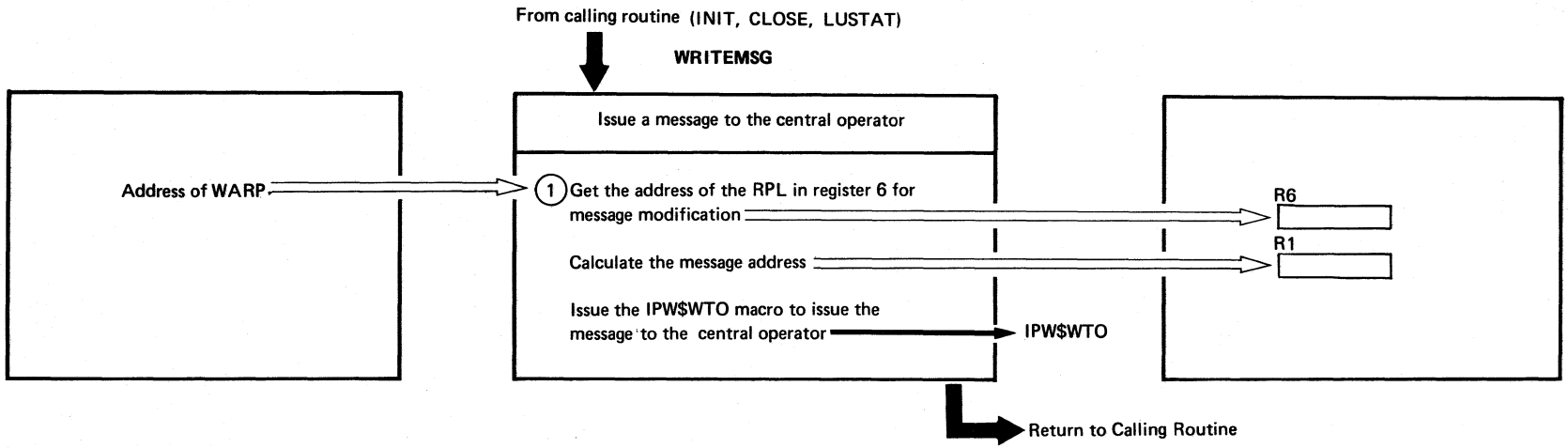

Input
Processing
Output
DEOAV
LUCB
LUBBO
LUCB
LUMR
B. Set no output available.
C. Reset output class.
15 If no more output available:
A. Indicate device available.
16 Reset pointer to LUCB.
17 Reset TCB pointer.
18 If no BB reject received:
A. Reset device pointer.
19 If request suspend for message:
A. Reset request indicator.
B. Reset request indicator.
20 Set select indicator.
21 Post SNA work ECB.
22 Unlock SUCB chain.
IPW$RLR
23 Detach outbound processor.
IPW$DET
RETURN TO MAINLINE
DEDAV
OPCLASS
DEAV
PTRLUCB
LUCB
LULT
LUCB
LUPH
LUCB
LUMR
SUCB
SUMRI
LUCB
LUSL
SNCB
SNEP


Input
Processing
Output
FROM CALLING ROUTINE
SENDFMH
WACB
WABP
FMH
RPL
RPLFMHDR
R13
WACB
WASV
WARP
TCB
TCB13
R15
01 Point to FMH area.
02 Get FMH length.
03 If null RU to be sent:
A. Set length for null RU.
04 Reset ECB for POWER wait.
05 Save pointer to function save area.
06 Point to VTAM save area.
07 Point to RPL for send.
08 Issue VTAM send.
09 Reset R13 to function save area.
10 If send request accepted:
A. Wait and check RPL.
CHECK
MH18
RPL
RPLAREA
RPLRLEN
RPL
RPLRLEN
TCB
TCBEO
TCB13
R13
R1
R13

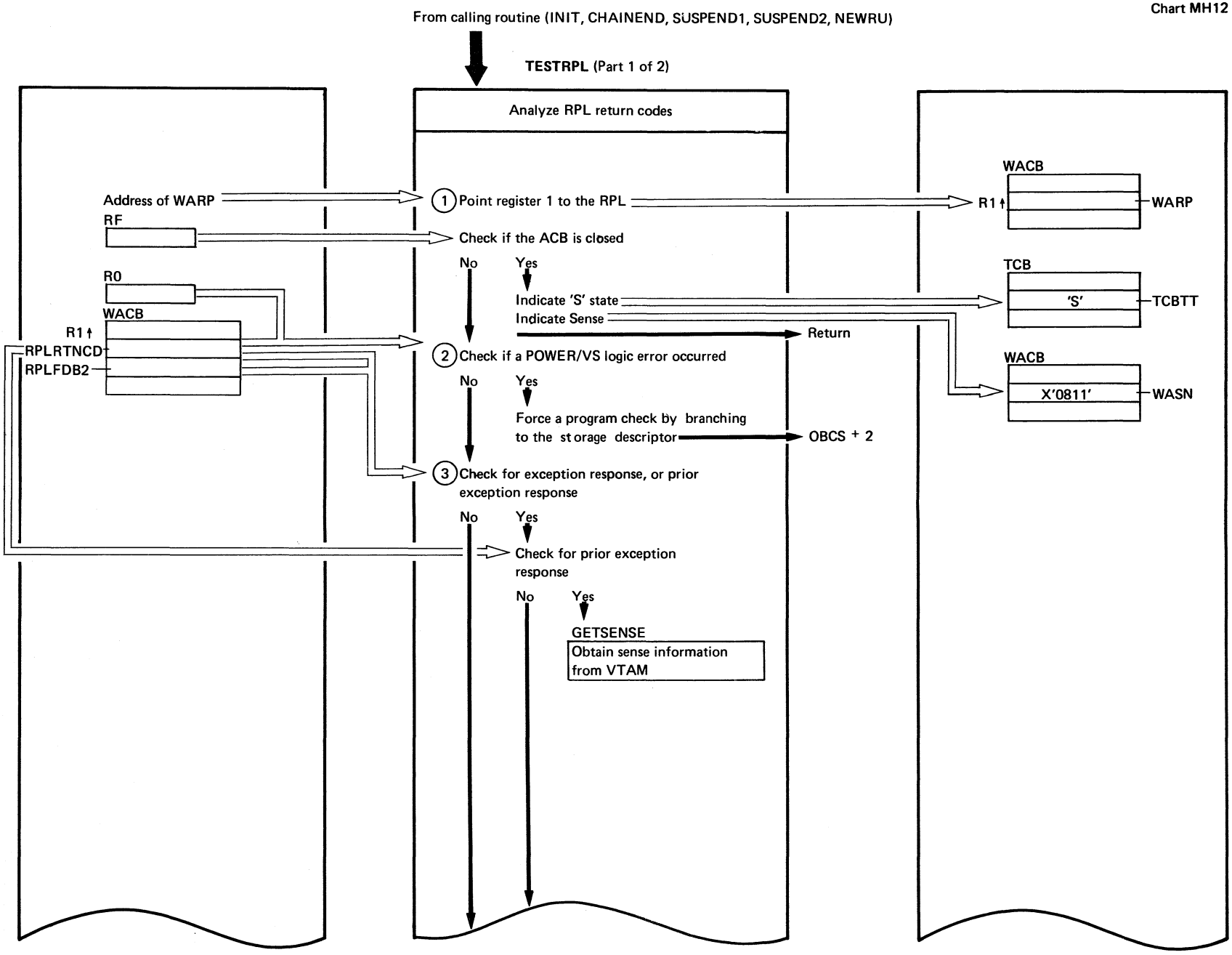

Input
Processing
Output
(11) Reset request for change direction.
(12) Reset request for continue specific mode.
(13)
RPL
RPLCMD
RPLDLGIN
RETURN TO CALLING ROUTINE



Extended Description	Include Segment	Call Subroutine/Macro	Chart
(1) The IPW$RMS macro uses registers 0, 1, 2 and 3		IPW$RMS (POWER/VS)	

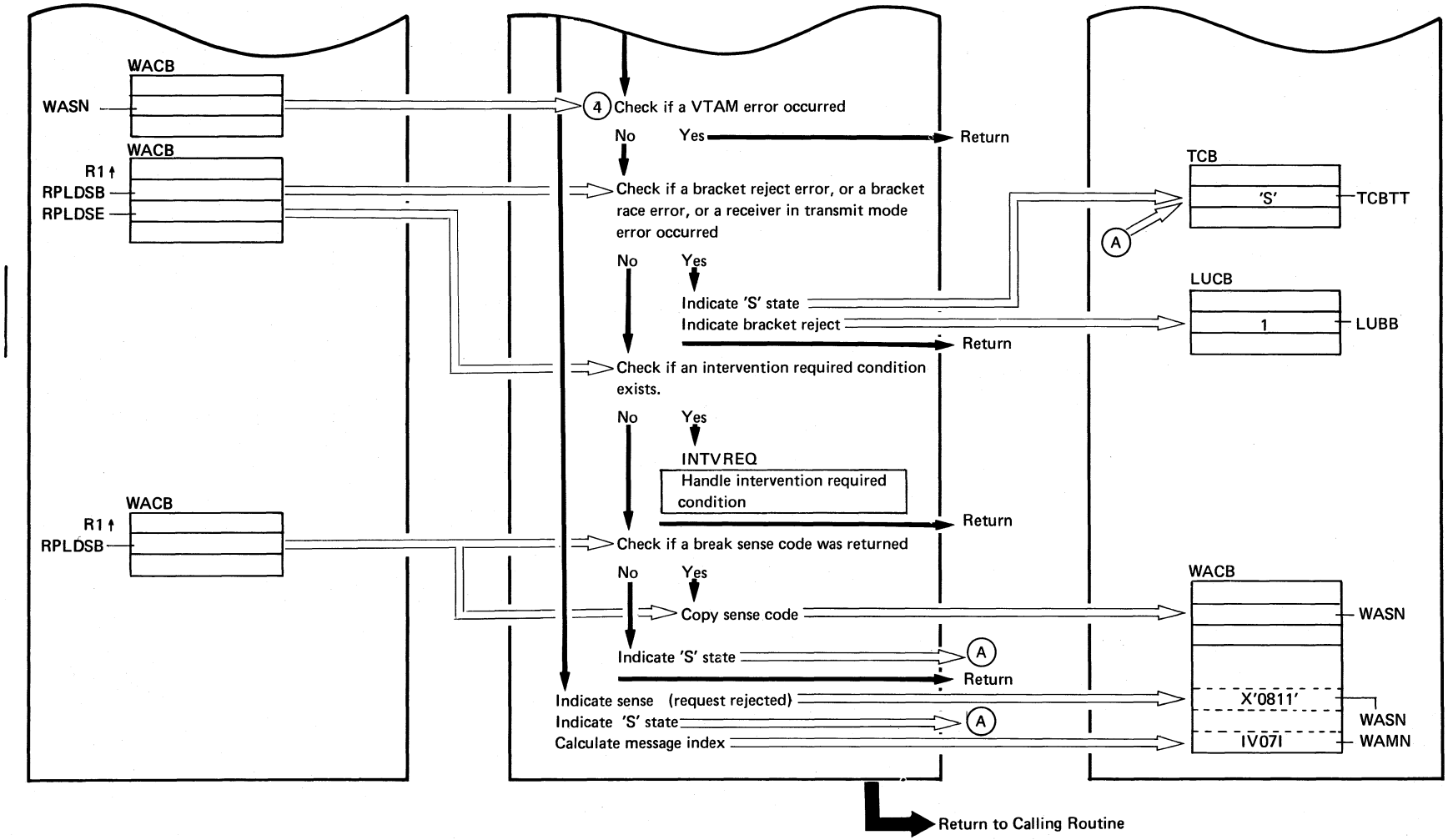


Extended Description	Include Segment	Call Subroutine/Macro	Chart
(1) The IPW$WTO macro uses registers 0, 1, 2, and 3.		IPW$WTO (POWER/VS)	

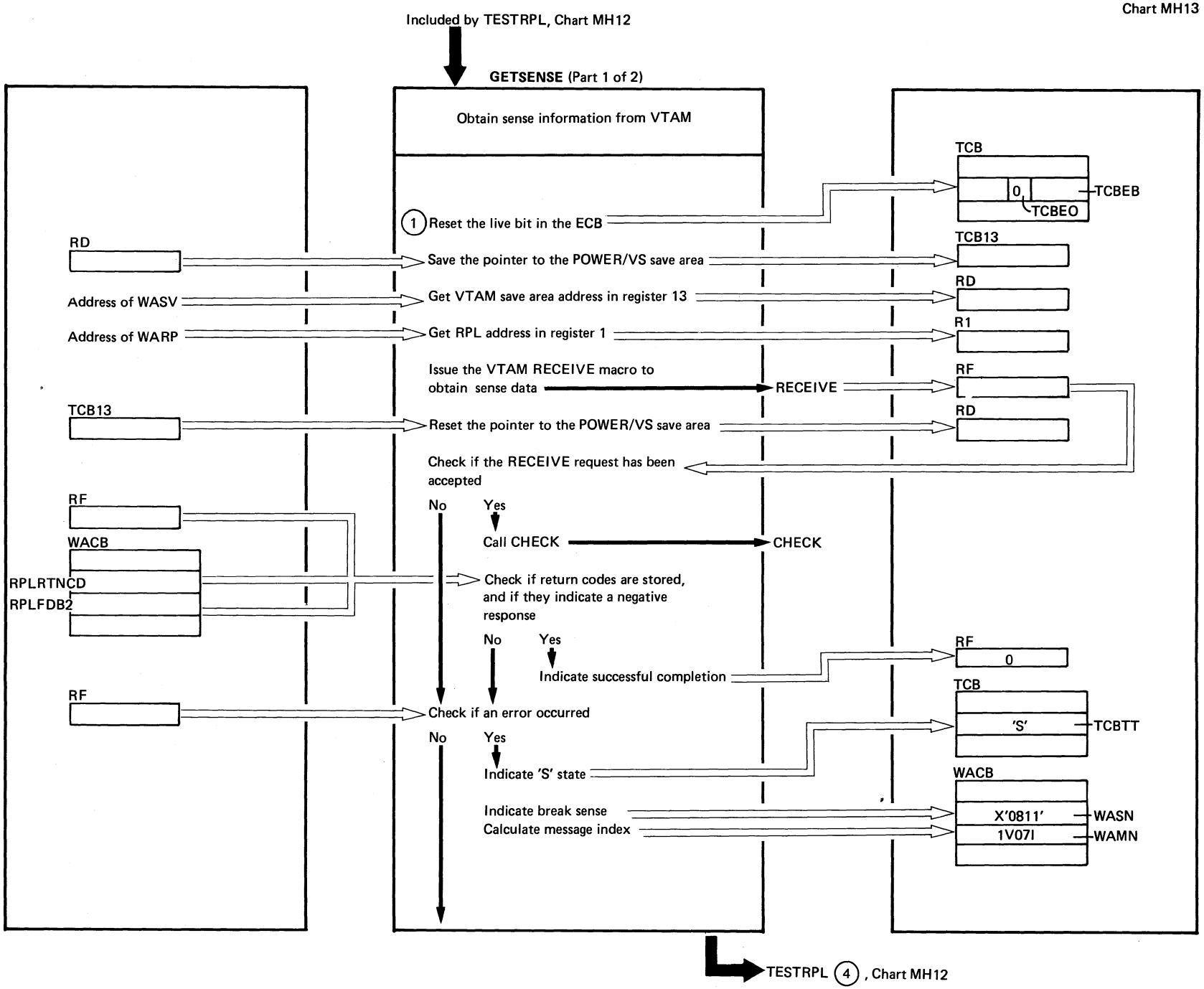


(Continued on next page)

TESTRPL (Part 2 of 2)



Extended Description	Include Segment	Call Subroutine/Macro	Chart
(3) 1V07I ERROR ON request RTNCD,FDBK2=xx,yy SENSE=xxxx ON luname			

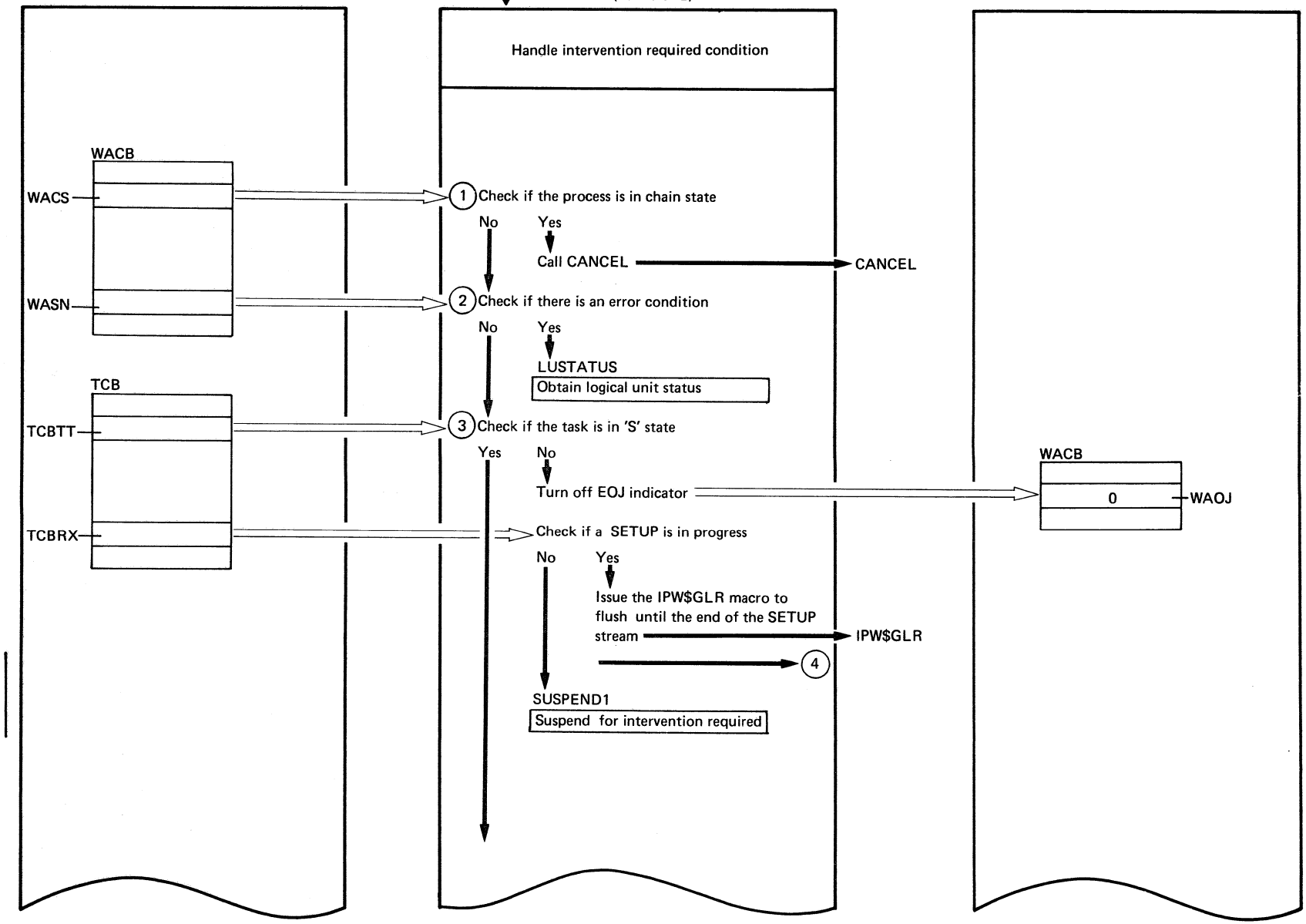

Included by TESTRPL, Chart MH12
GETSENSE (Part 1 of 2)
Obtain sense information from VTAM
TCB
0
TCBEB
TCBEO
1 Reset the live bit in the ECB
RD
Save the pointer to the POWER/VS save area
TCB13
Address of WASV
Get VTAM save area address in register 13
RD
Address of WARP
Get RPL address in register 1
R1
Issue the VTAM RECEIVE macro to obtain sense data
RECEIVE
RF
TCB13
Reset the pointer to the POWER/VS save area
RD
Check if the RECEIVE request has been accepted
RF
No
Yes
WACB
Call CHECK
CHECK
RPLRTNCD
RPLFDB2
Check if return codes are stored, and if they indicate a negative response
No
Yes
RF
0
Indicate successful completion
TCB
RF
Check if an error occurred
'S'
TCBTT
No
Yes
Indicate 'S' state
WACB
Indicate break sense
X'0811'
WASN
Calculate message index
1V07I
WAMN
TESTRPL 4 , Chart MH12

GETSENSE (Part 2 of 2)

Extended Description	Include Segment	Call Subroutine/Macro	Chart
(1) 1V07I ERROR ON request RTNCD,FDBK2=xx,yy SENSE=xxxx ON luname		RECEIVE (VTAM) CHECK	MH18

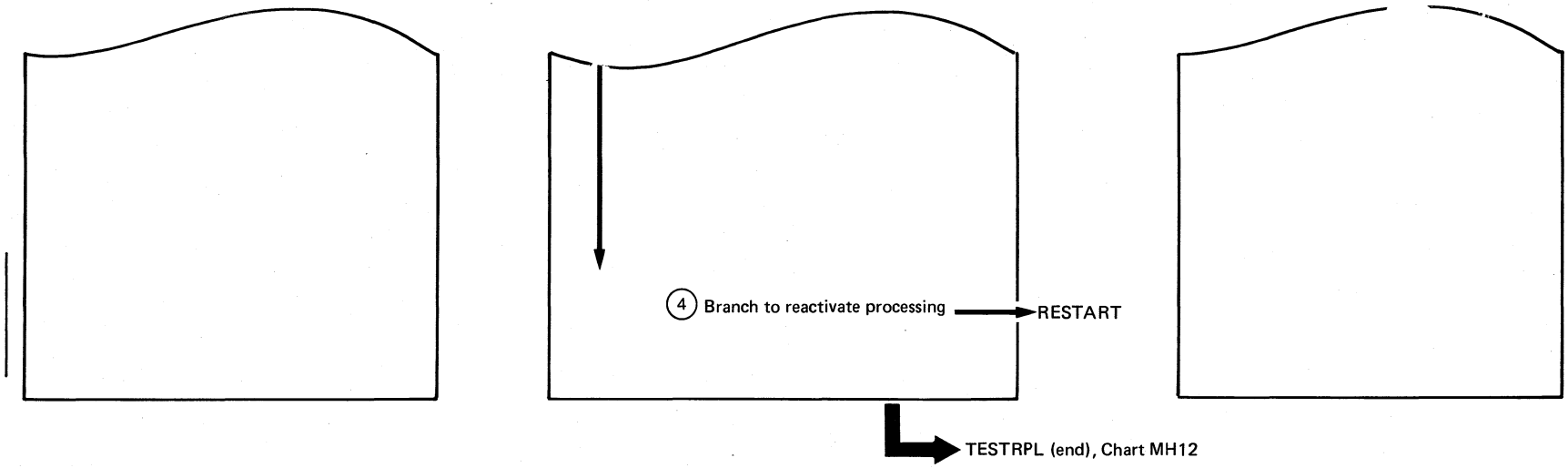
Included by TESTRPL, Chart MH12

INTVREQ (Part 1 of 2)



Continued on next page

INTVREQ (Part 2 of 2)



Extended Description	Include Segment	Call Subroutine/Macro	Chart
1		CANCEL	MH17
2	LUSTATUS		MH15
3		IPW$GLR(POWER/VS)	
	SUSPEND3		MH16
4		IPW$GLR(POWER/VS)	

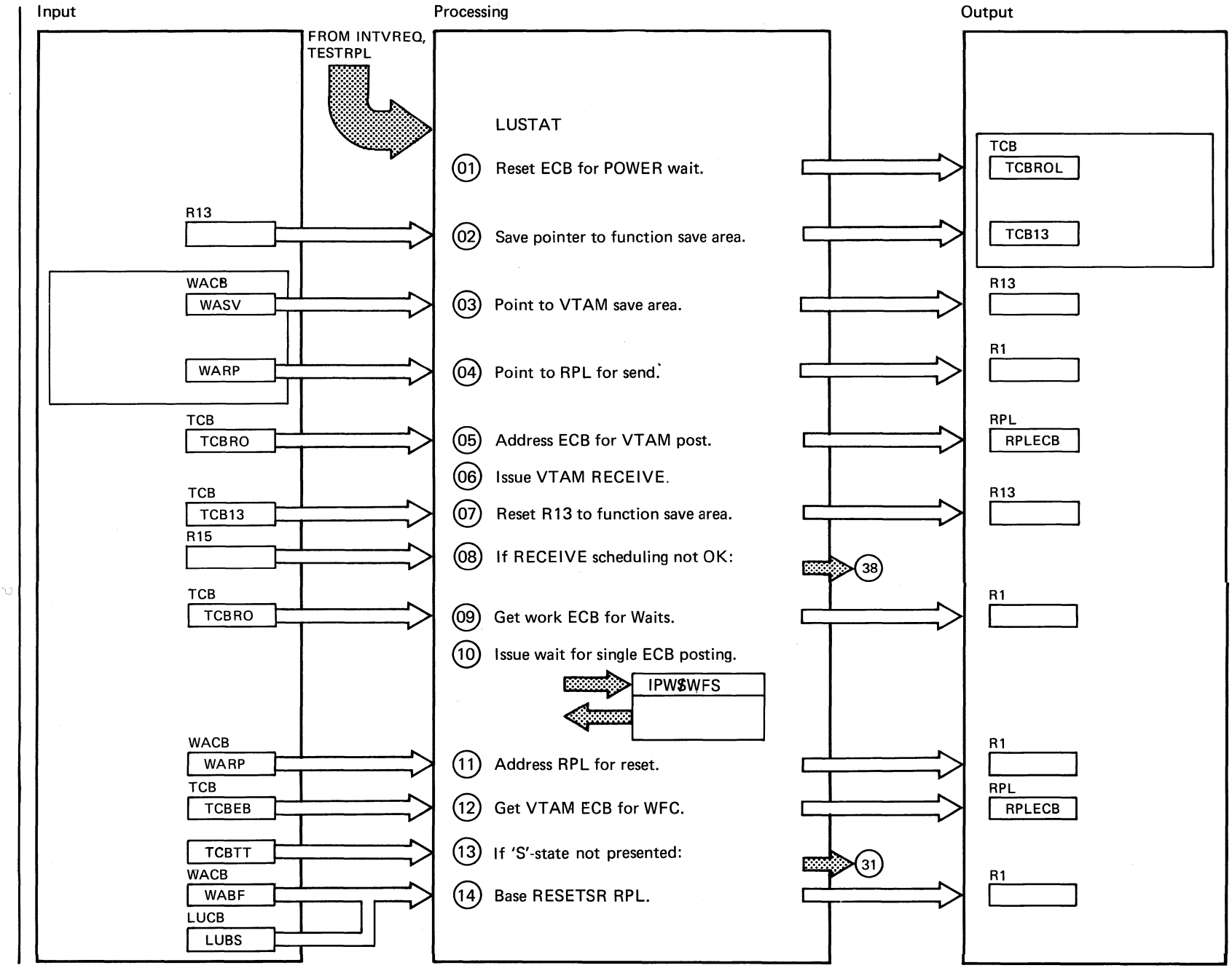
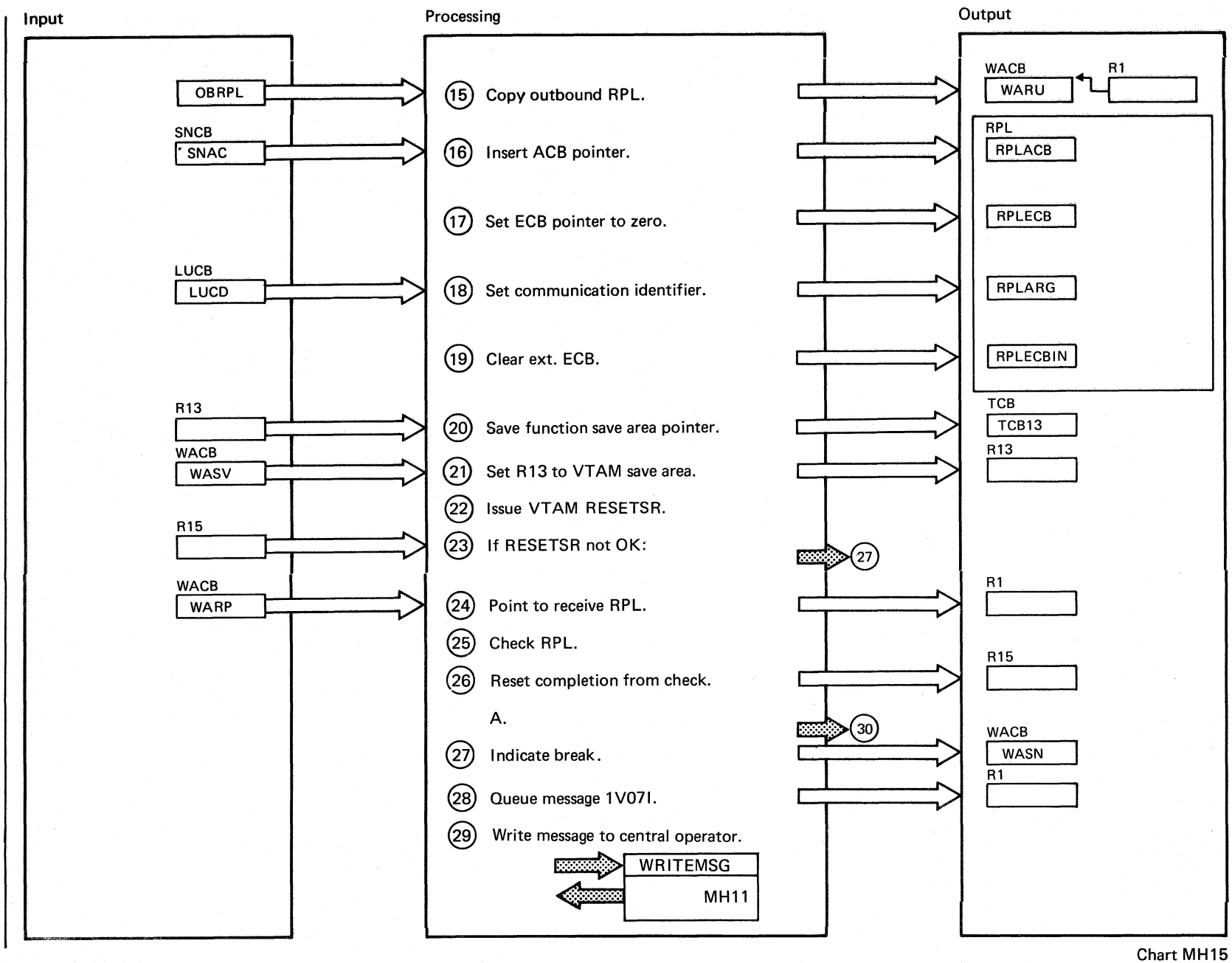
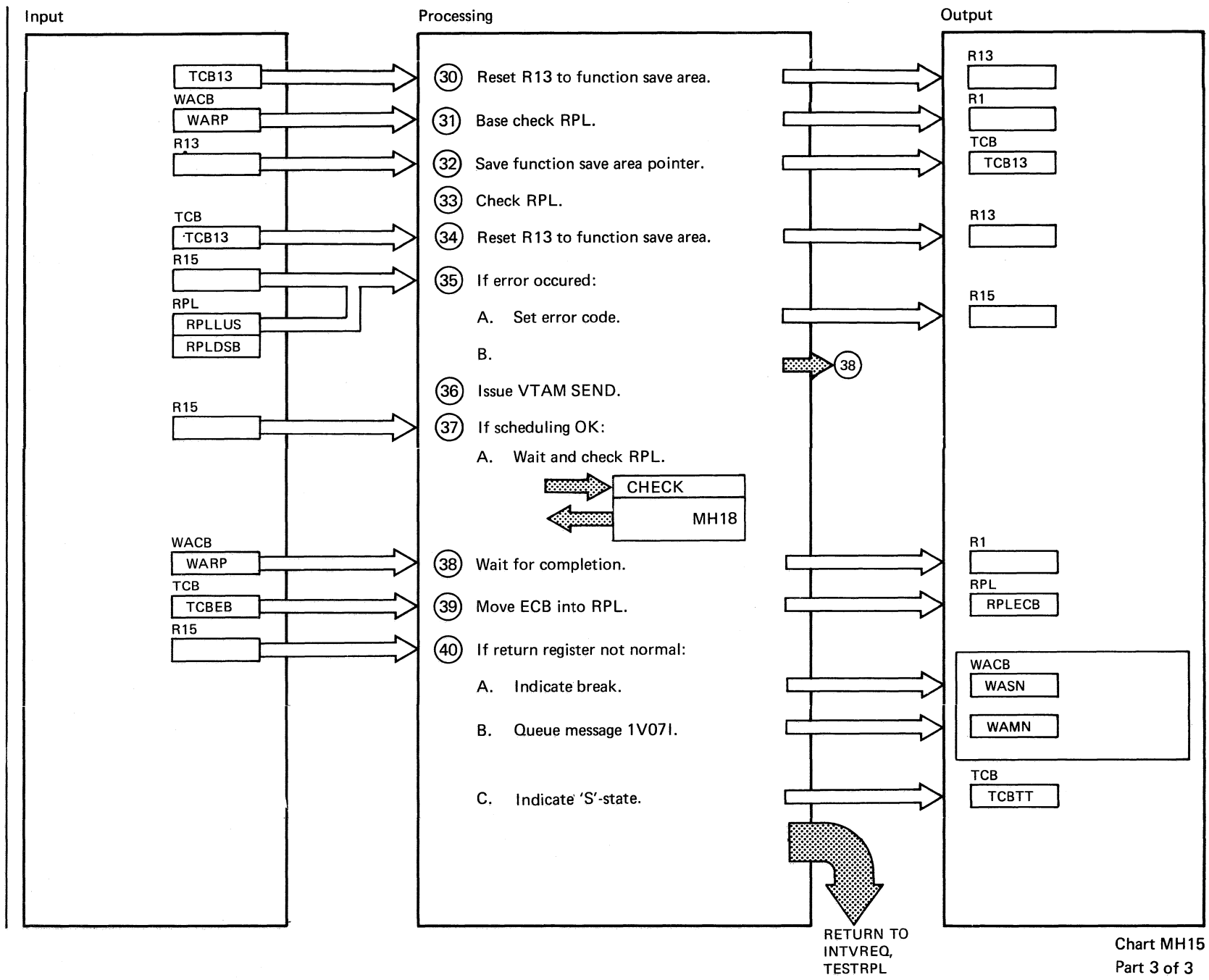


Chart MH15
Part 1 of 3


Input
Processing
Output
OBRPL
SNCB
SNAC
LUCB
LUCD
R13
WACB
WASV
R15
WACB
WARP
15 Copy outbound RPL.
16 Insert ACB pointer.
17 Set ECB pointer to zero.
18 Set communication identifier.
19 Clear ext. ECB.
20 Save function save area pointer.
21 Set R13 to VTAM save area.
22 Issue VTAM RESETSR.
23 If RESETSR not OK:
27
24 Point to receive RPL.
25 Check RPL.
26 Reset completion from check.
A.
30
27 Indicate break.
28 Queue message 1V07I.
29 Write message to central operator.
WRITEMSG
MH11
WACB
WARU
R1
RPL
RPLACB
RPLECB
RPLARG
RPLECBIN
TCB
TCB13
R13
R1
R15
WACB
WASN
R1


Input
Processing
Output
TCB13
WACB
WARP
R13
TCB
TCB13
R15
RPL
RPLLUS
RPLDSB
R15
WACB
WARP
TCB
TCBEB
R15
30 Reset R13 to function save area.
31 Base check RPL.
32 Save function save area pointer.
33 Check RPL.
34 Reset R13 to function save area.
35 If error occured:
A. Set error code.
B.
36 Issue VTAM SEND.
37 If scheduling OK:
A. Wait and check RPL.
CHECK
MH18
38 Wait for completion.
39 Move ECB into RPL.
40 If return register not normal:
A. Indicate break.
B. Queue message 1V07I.
C. Indicate 'S'-state.
38
R13
R1
TCB
TCB13
R13
R15
R1
RPL
RPLECB
WACB
WASN
WAMN
TCB
TCBTT
RETURN TO INTVREQ, TESTRPL

Included by INTVREQ, Chart MH14

SUSPEND3 (Part 1 of4)

Suspend the outbound process when an intervention required condition has occurred to allow the operator to restart the outbound flow

RF

(1) Calculate message index
Call SENDMSG to queue the message → SENDMSG

R1
1V18A

(2) Indicate that an FMH is to be sent
Indicate current data stream suspended
Call SENDFMH → SENDFMH

WACB
1 → RPLFMHDR
B'100' → FMHDSS

(3) Check if an error occurred

No

Yes
Indicate 'S' state

TCB
'S' → TCBTT

Calculate message index
Indicate break sense

WACB
IV071 → WAMN
X'0811' → WASN
X'08' → WASS

(5)

(4) Indicate data stream inactive

Reset the live bit in the ECB

TCB
0 → TCBEB
TCBEO

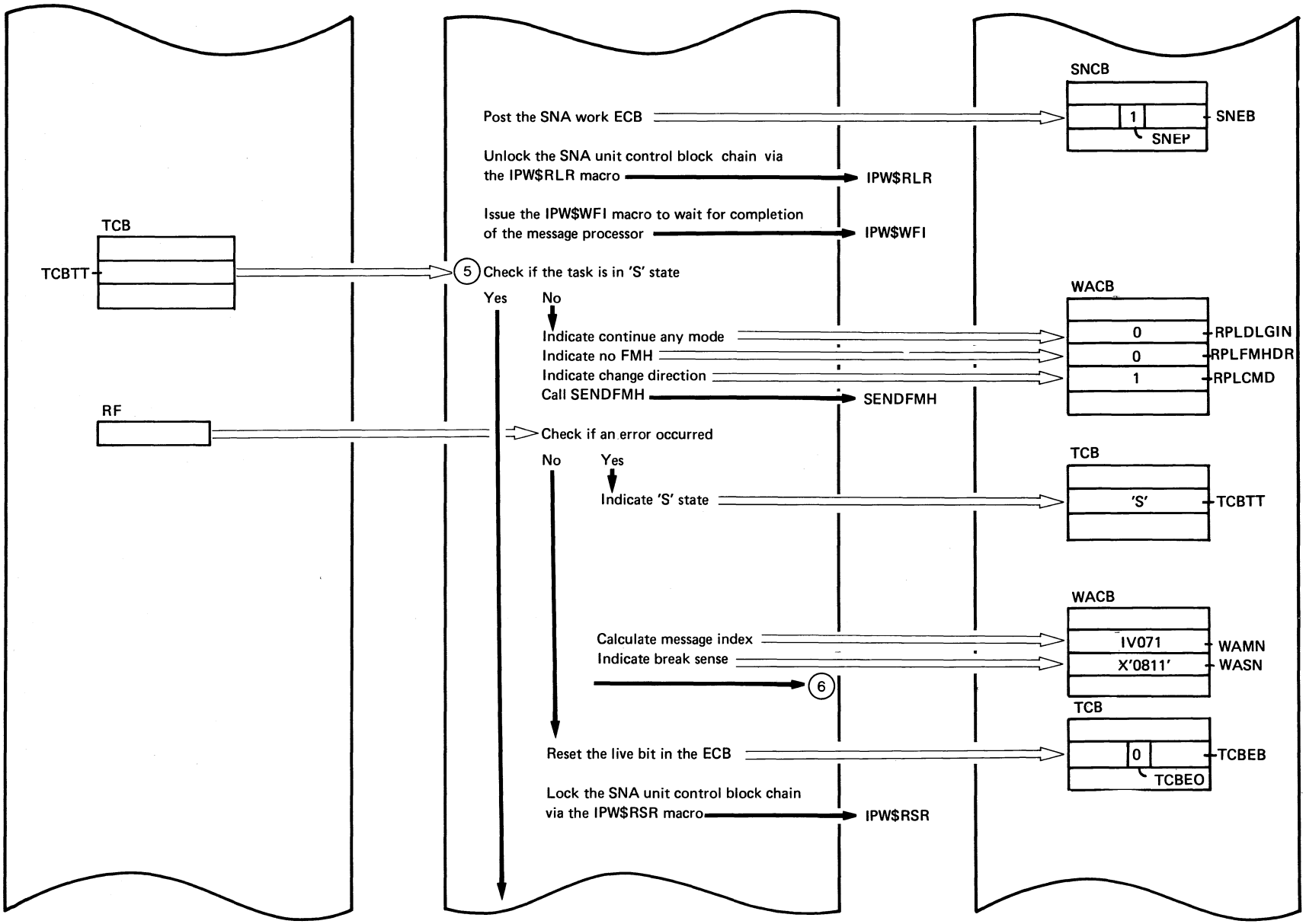
Lock the SNA unit control block chain
via the IPW$RSR macro → IPW$RSR

Indicate that the outbound process is suspended for the message processor
Set select indicator

SUCB
1 → SUOO
'S' → SUSL

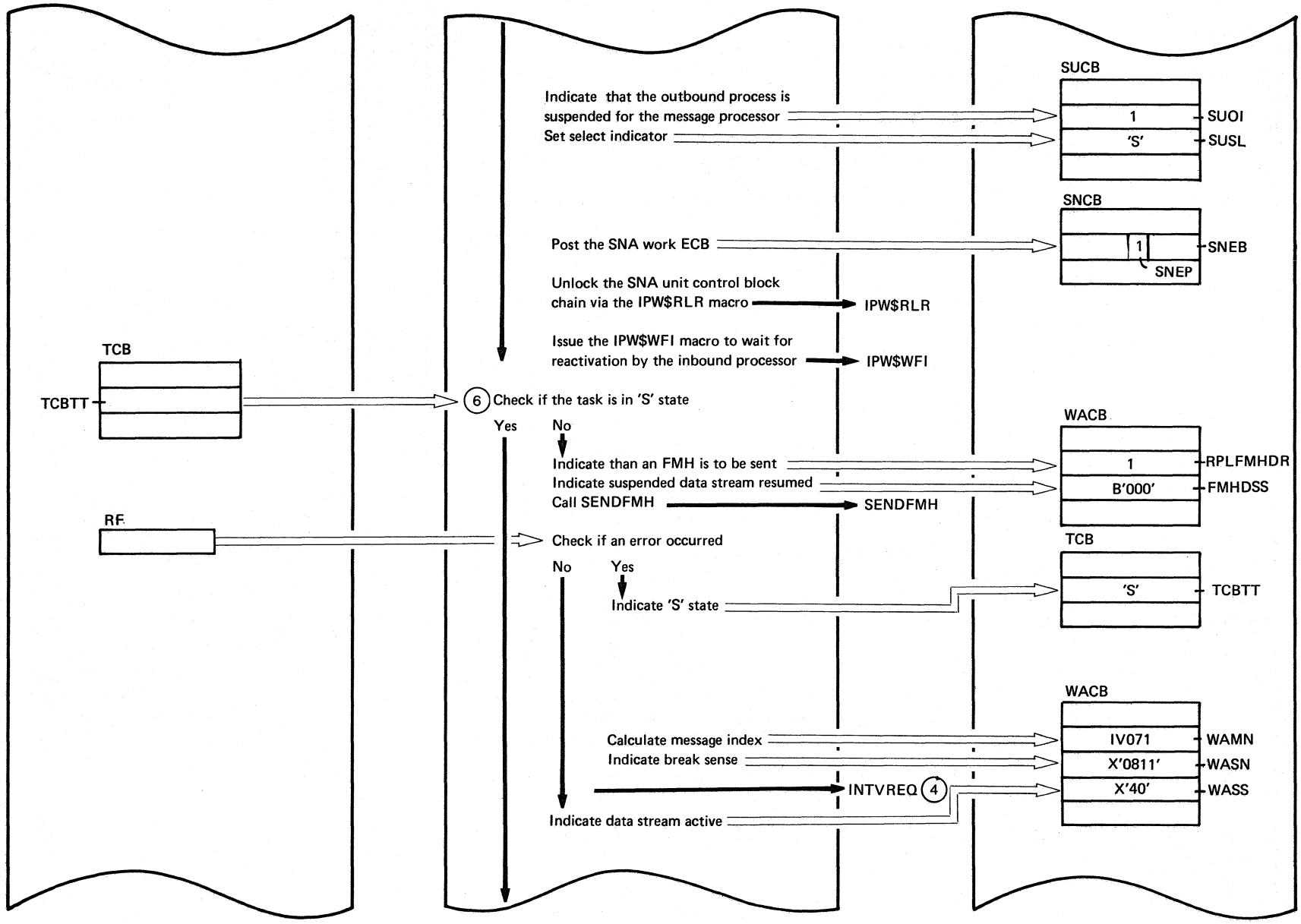
(Continued on next page)

SUSPEND3 (Part 2 of 4)



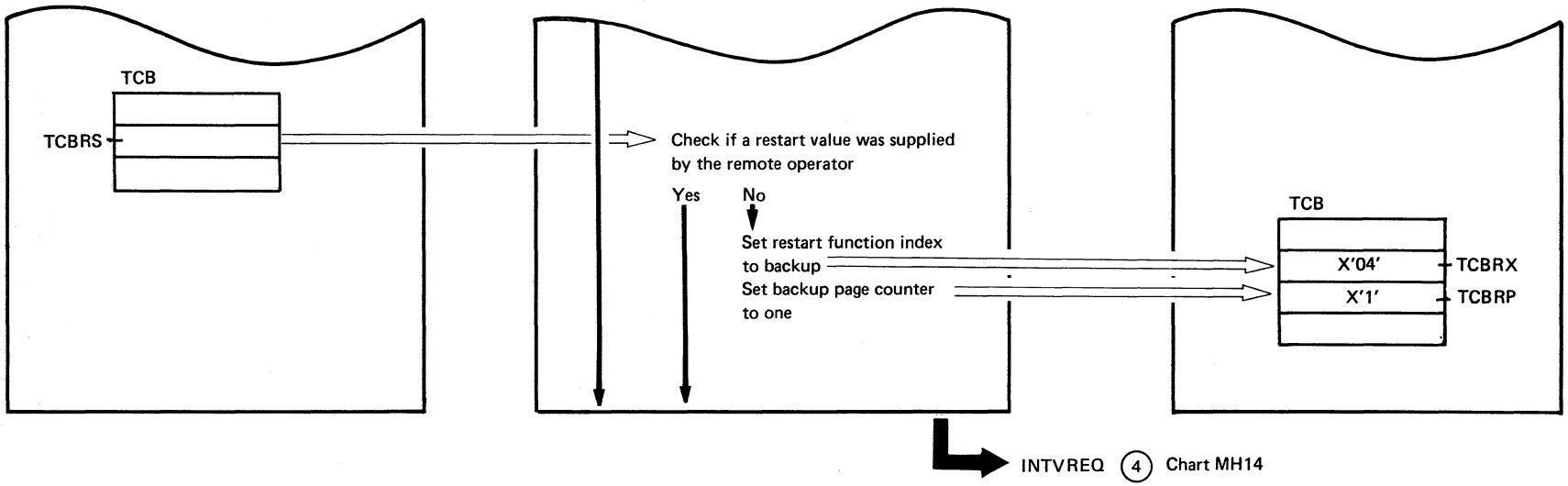
(Continued on next page)

SUSPEND3 (Part 3 of 4)

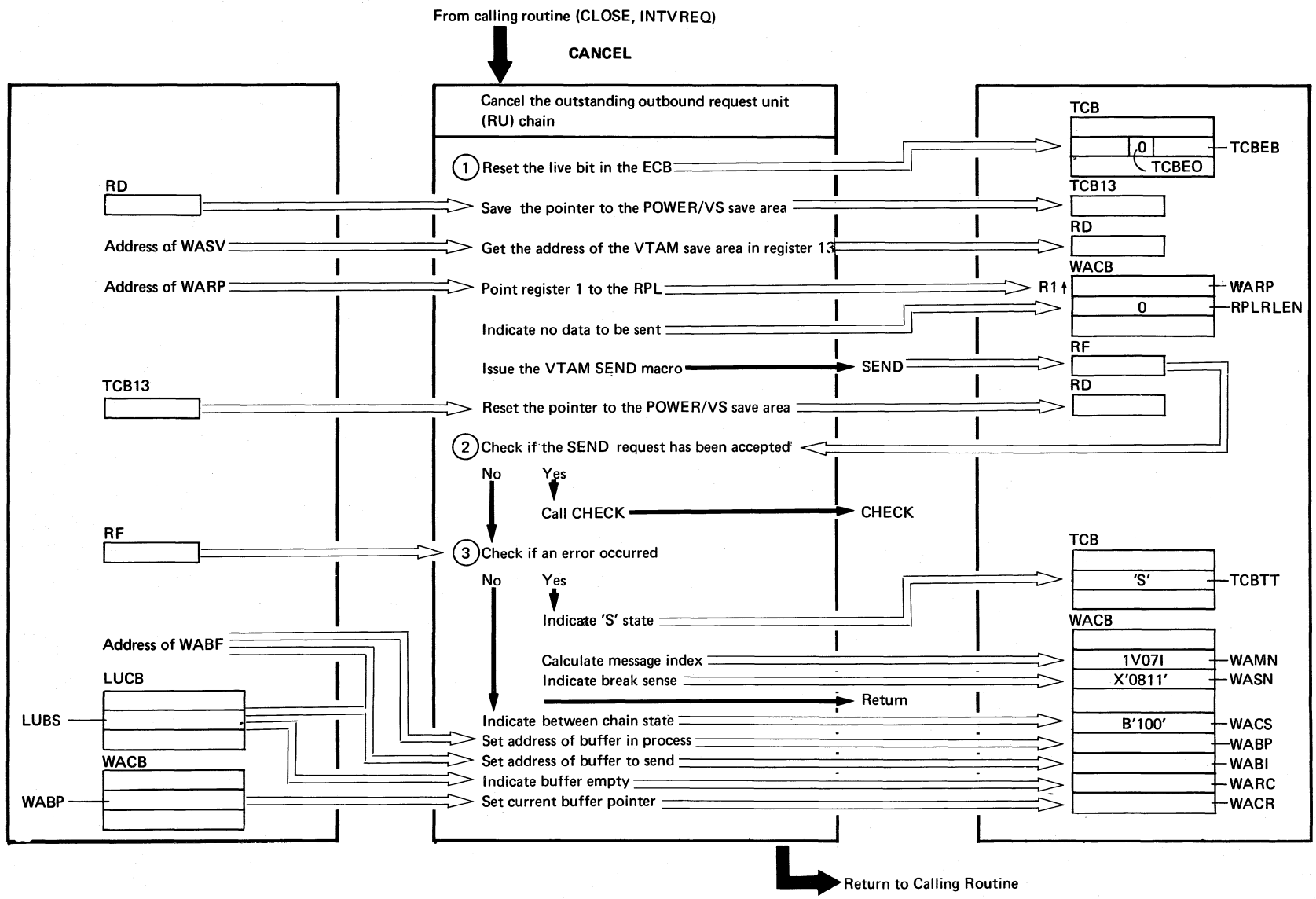


(Continued on next page)

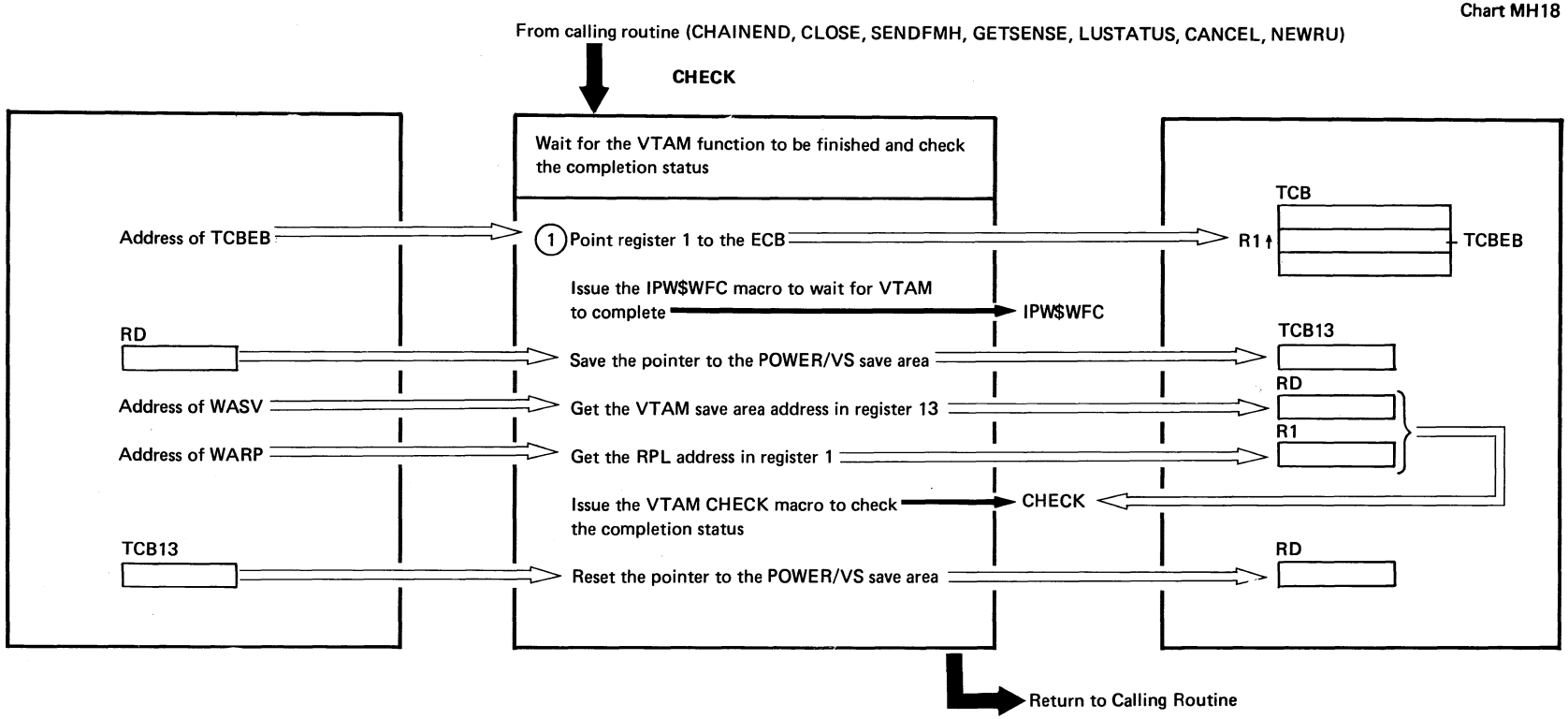
SUSPEND3 (Part 4 of 4)



Extended Description	Include Segment	Call Subroutine/Macro	Chart
(1) 1V18A REPLY WITH RESTART ON INTERVENTION REQUIRED		SENDMSG	MH10
(2)		SENDFMH	MH9
(4)		IPW$RSR (POWER/VS) IPW$RLR (POWER/VS) IPW$WFI (POWER/VS)	
(3) (5) (6) 1V07I ERROR ON request RTNCD,FDBK2=xx,yy SENSE=xxxx ON luname		SENDFMH	MH9
		IPW$RSR (POWER/VS) IPW$RLR (POWER/VS) IPW$WFI (POWER/VS)	
		SENDFMH	MH9



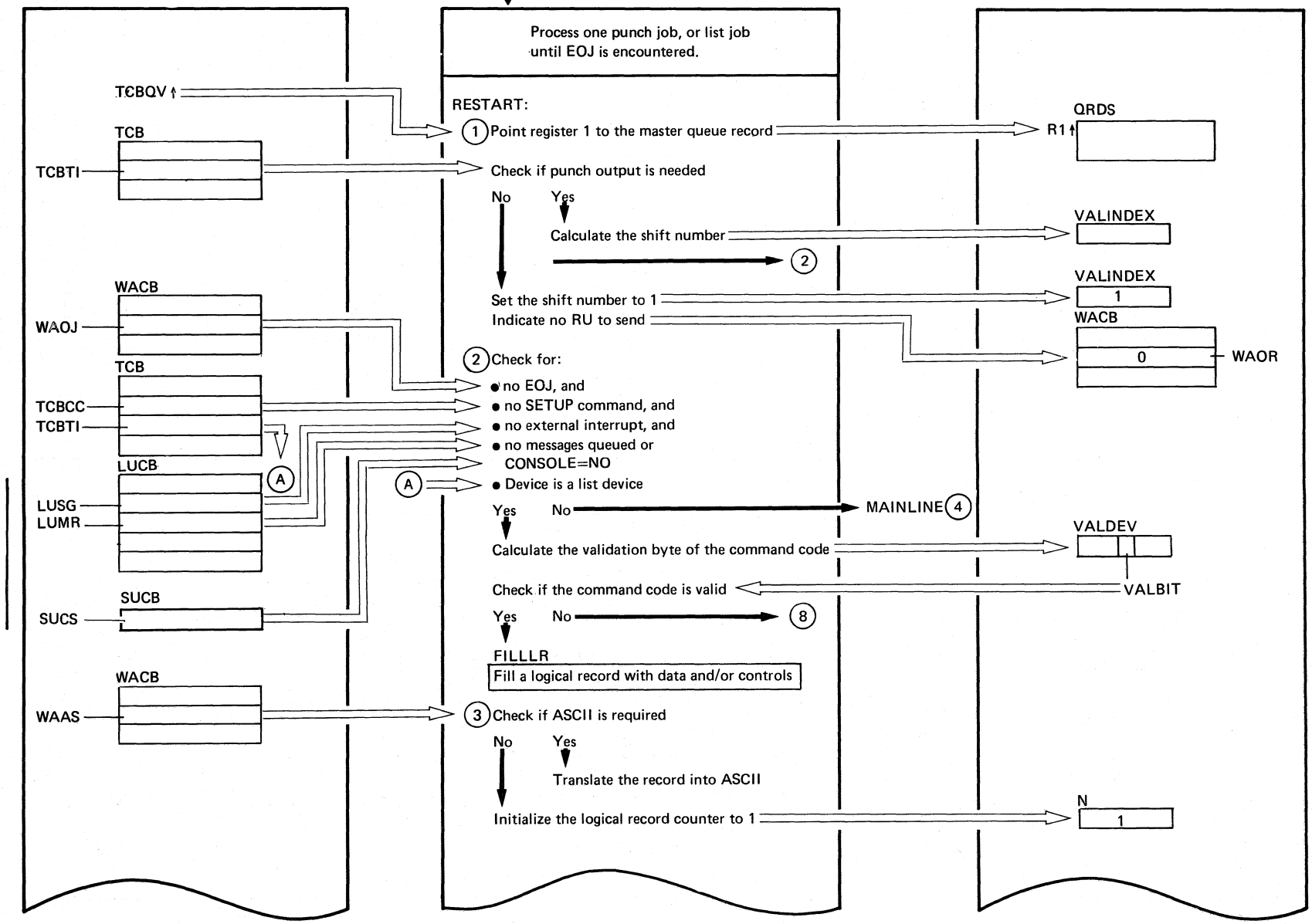
Extended Description	Include Segment	Call Subroutine/Macro	Chart
1 2 3 1V07I ERROR ON 'request' RTNCD, FDBK2 = xx,yy SENSE = xxxx 'luname'		SEND (VTAM) CHECK	MH18



Extended Description	Include Segment	Call Subroutine/Macro	Chart
1		IPW$WFC (POWER/VS) CHECK (VTAM)	

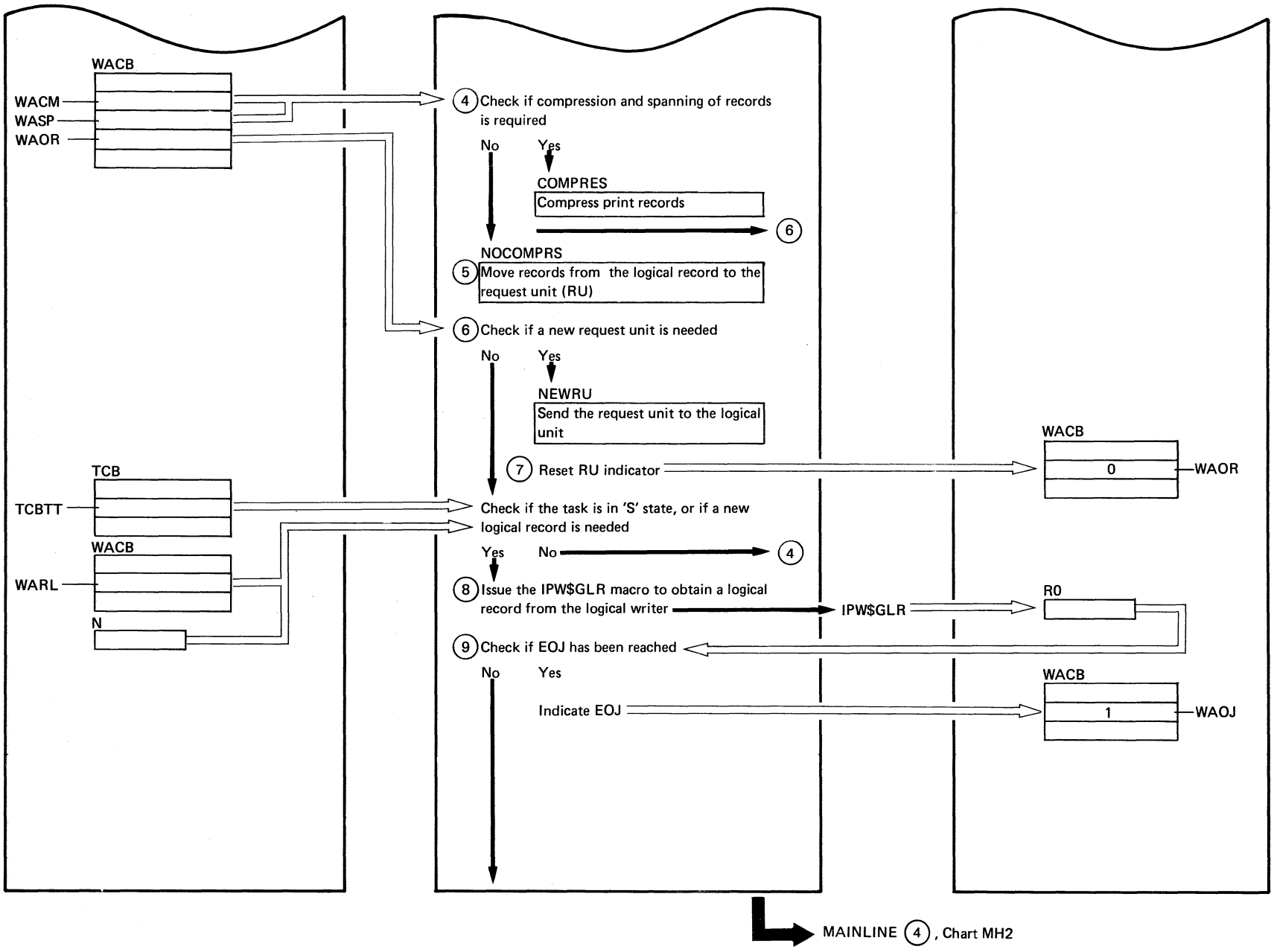
Called by MAINLINE, Chart MH2

CHAINJOB (Part 1 of 3)



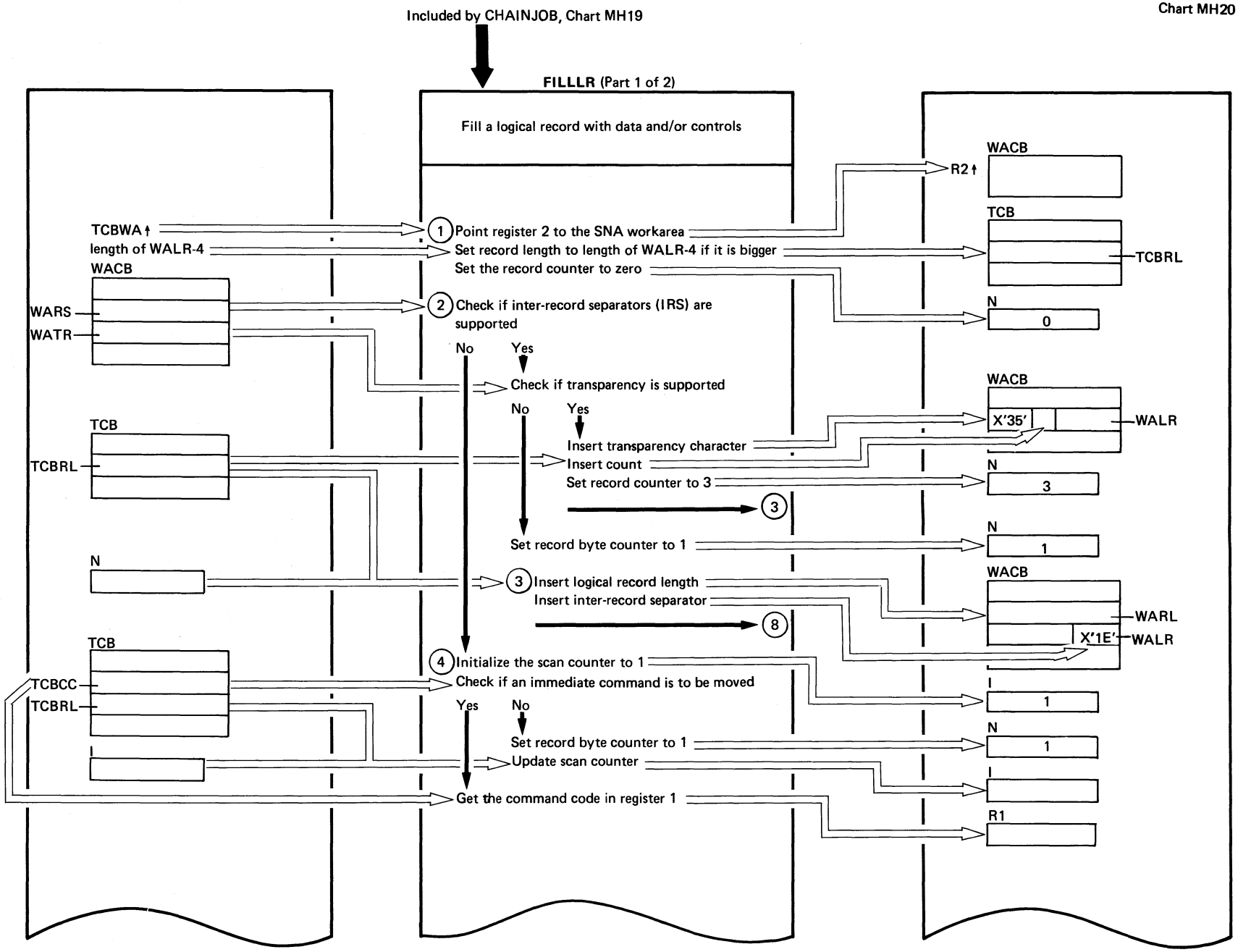
(Continued on next page)

CHAINJOB (Part 2 of 3)



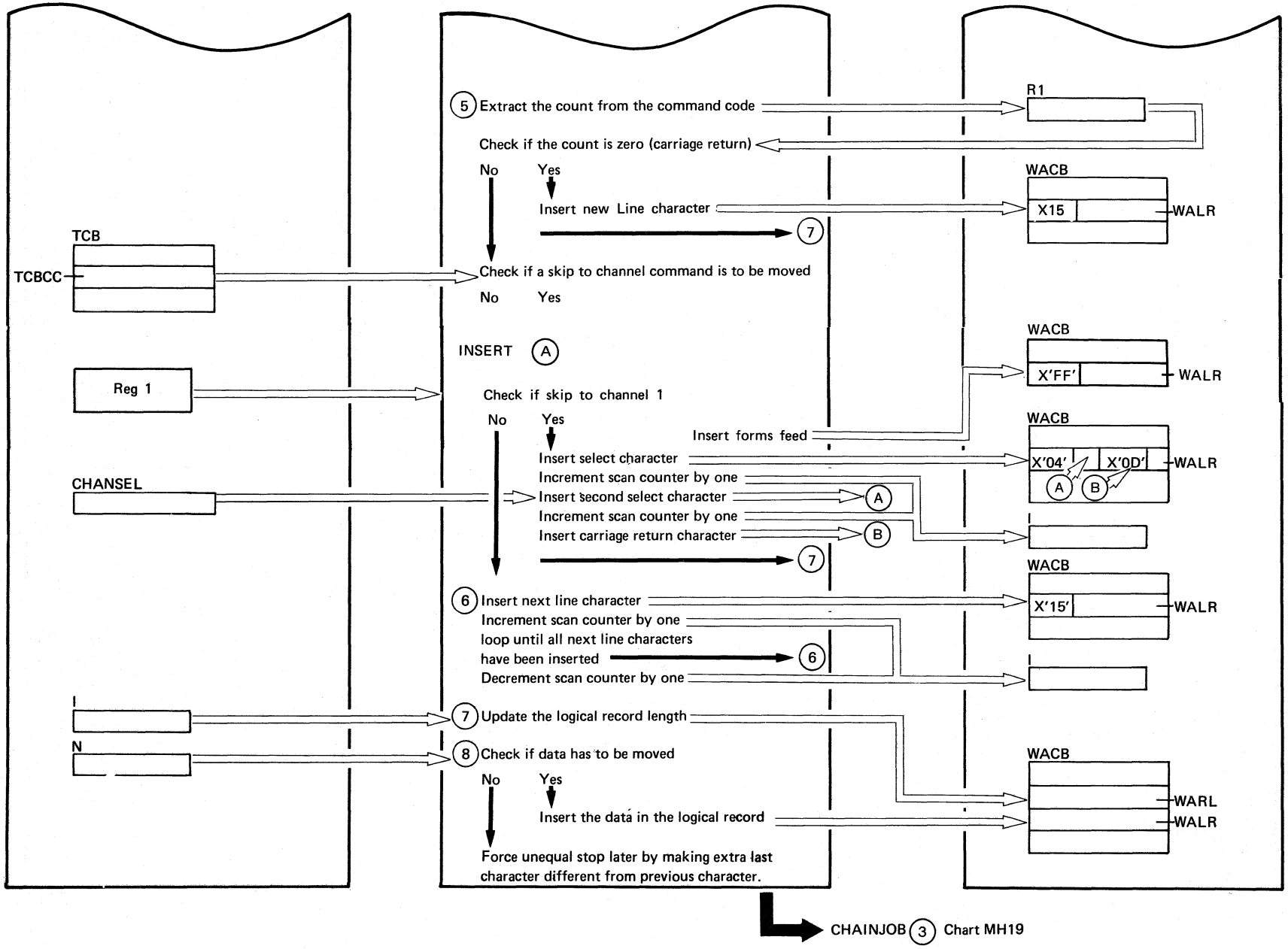
CHAINJOB (Part 3 of 3)

Extended Description	Include Segment	Call Subroutine/Macro	Chart
(1) The validation table (VALIDAT) is used to check whether a punch or print command for a particular device is valid. The table consists of 256 bytes, each of which correspond with a command code. Each bit position in a byte represents a punch or printer device. If the bit is on, the corresponding command for the device is valid. If the bit is off, the command will be ignored. To test the bit, it is shifted to the low order position. Therefore, the byte is devided by a number (VALINDEX), which is calculated with the help of the device type (QRDT) found in the queue record, and a device table (DEVTAB).			
(2)	FILLLR		MH20
(3)	COMPRES		MH21
(4)	NOCOMPRS		MH22
(5)	NEWRU		MH23
(6) The IPW$GLR macro uses registers 0, 1, 14 and 15.		IPW$GLR (POWER/VS)	



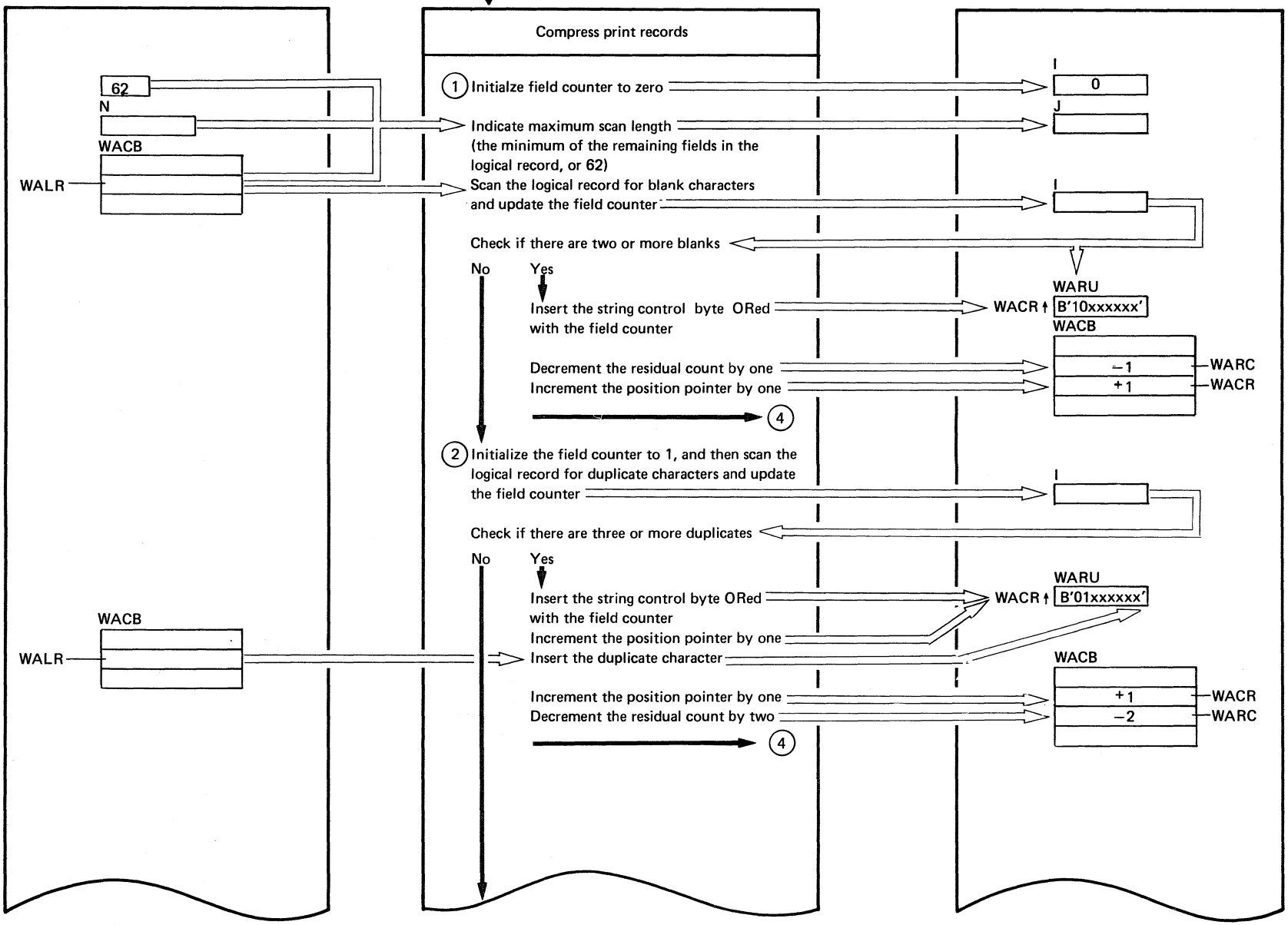
(Continued on next page)

FILLLR (Part 2 of 2)



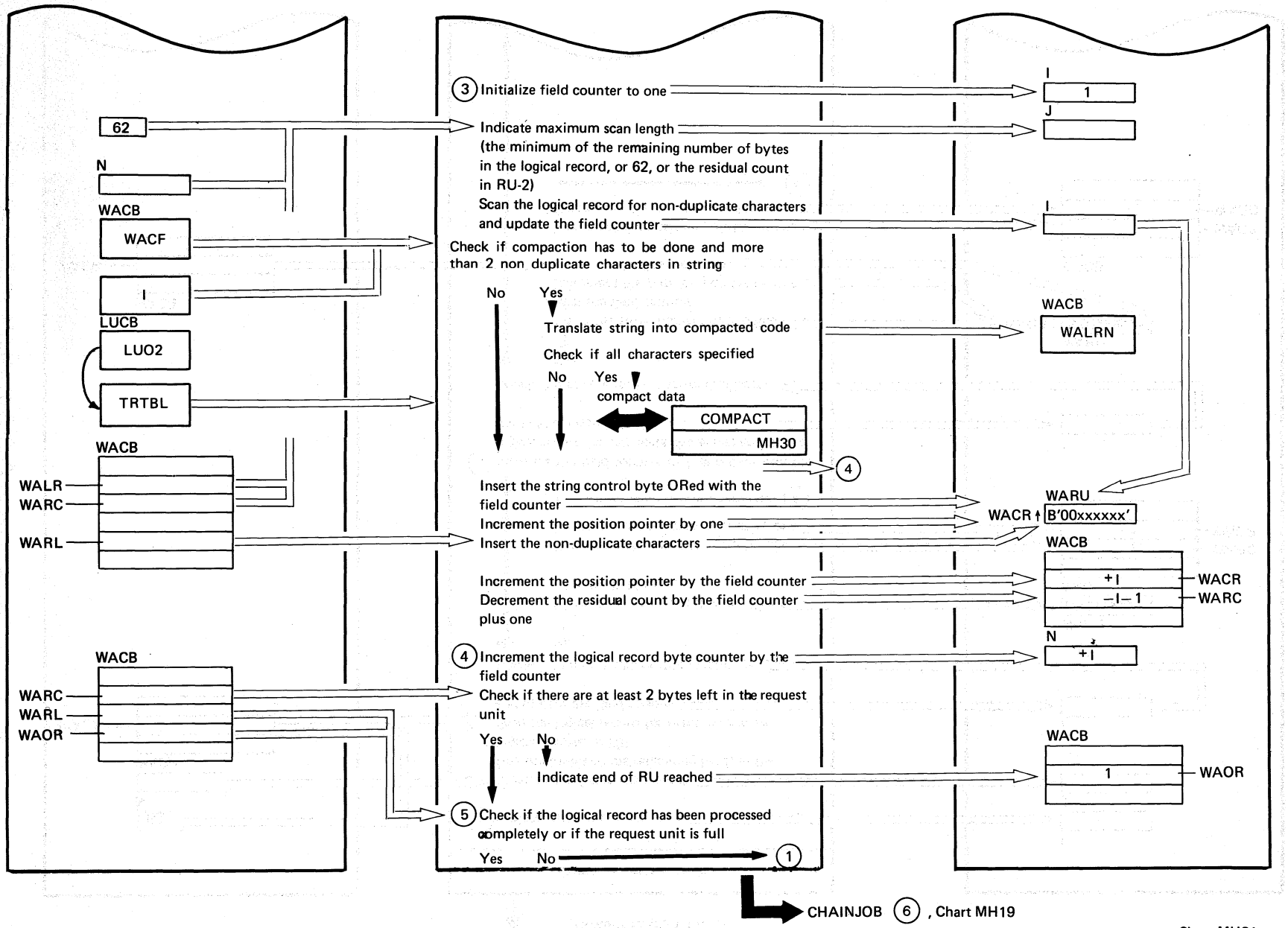
Included by CHAINJOB, Chart MH19

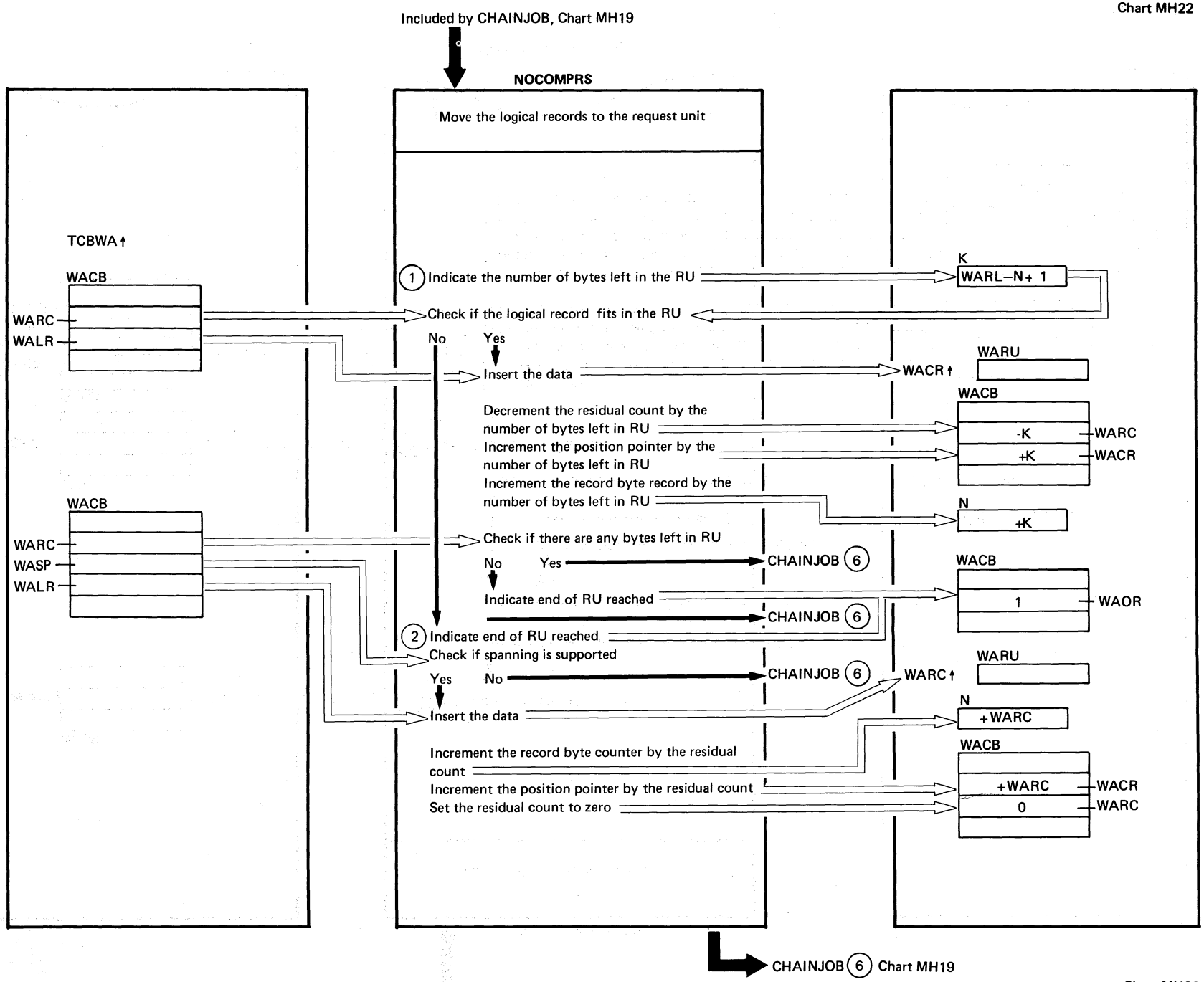
COMPRES (Part 1 of 2)



(Continued on next page)

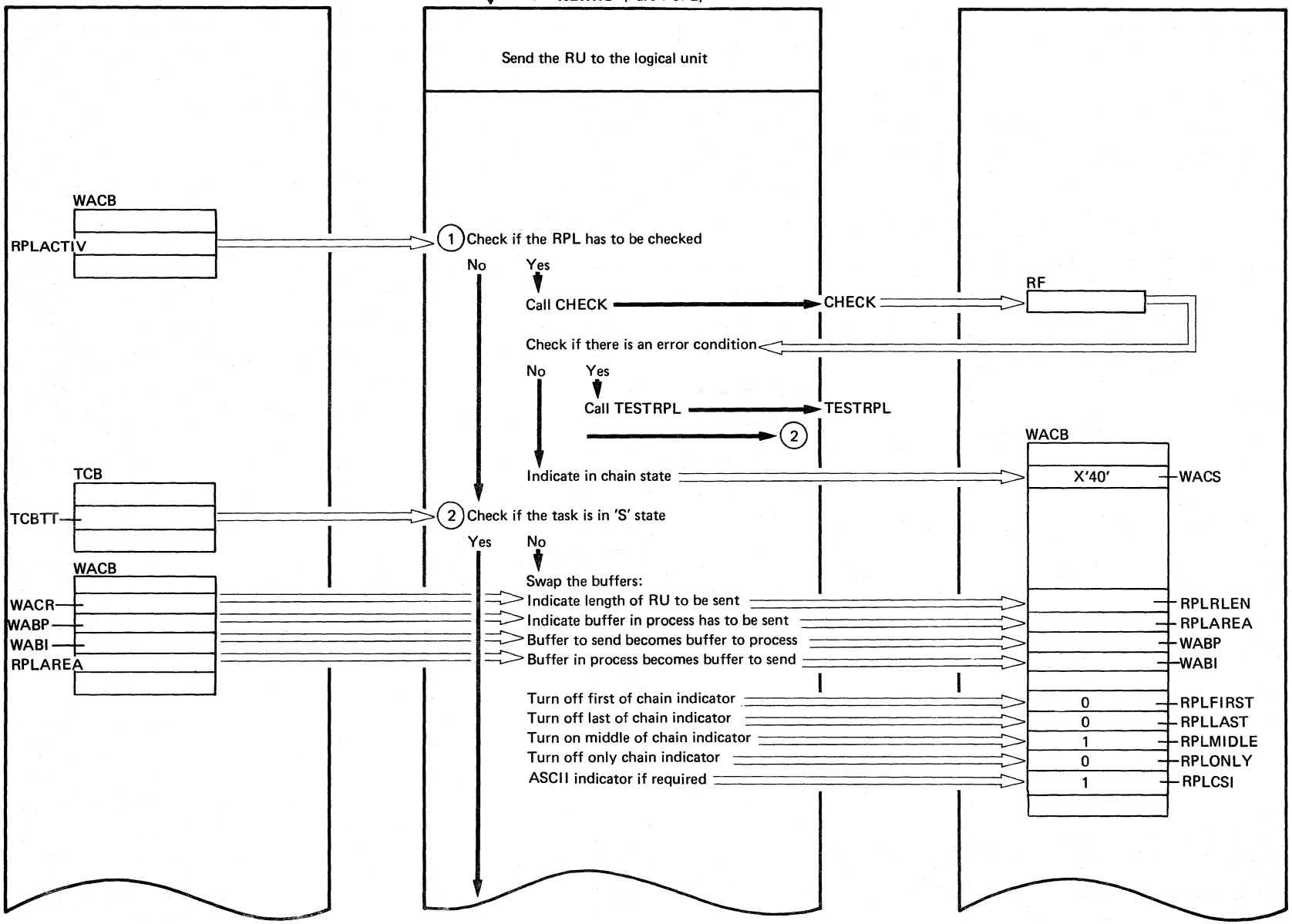
COMPRES (Part 2 of 3)




Included by CHAINJOB, Chart MH19
NOCOMPRS
Move the logical records to the request unit
TCBWA
WACB
WARC
WALR
WASP
1 Indicate the number of bytes left in the RU
K
WARL–N+ 1
Check if the logical record fits in the RU
No
Yes
Insert the data
WACR
WARU
Decrement the residual count by the number of bytes left in RU
Increment the position pointer by the number of bytes left in RU
-K
+K
Increment the record byte record by the number of bytes left in RU
N
Check if there are any bytes left in RU
CHAINJOB 6
Indicate end of RU reached
1
WAOR
2 Indicate end of RU reached
Check if spanning is supported
Insert the data
+WARC
Increment the record byte counter by the residual count
Increment the position pointer by the residual count
Set the residual count to zero
0
CHAINJOB 6 Chart MH19
Chart MH22

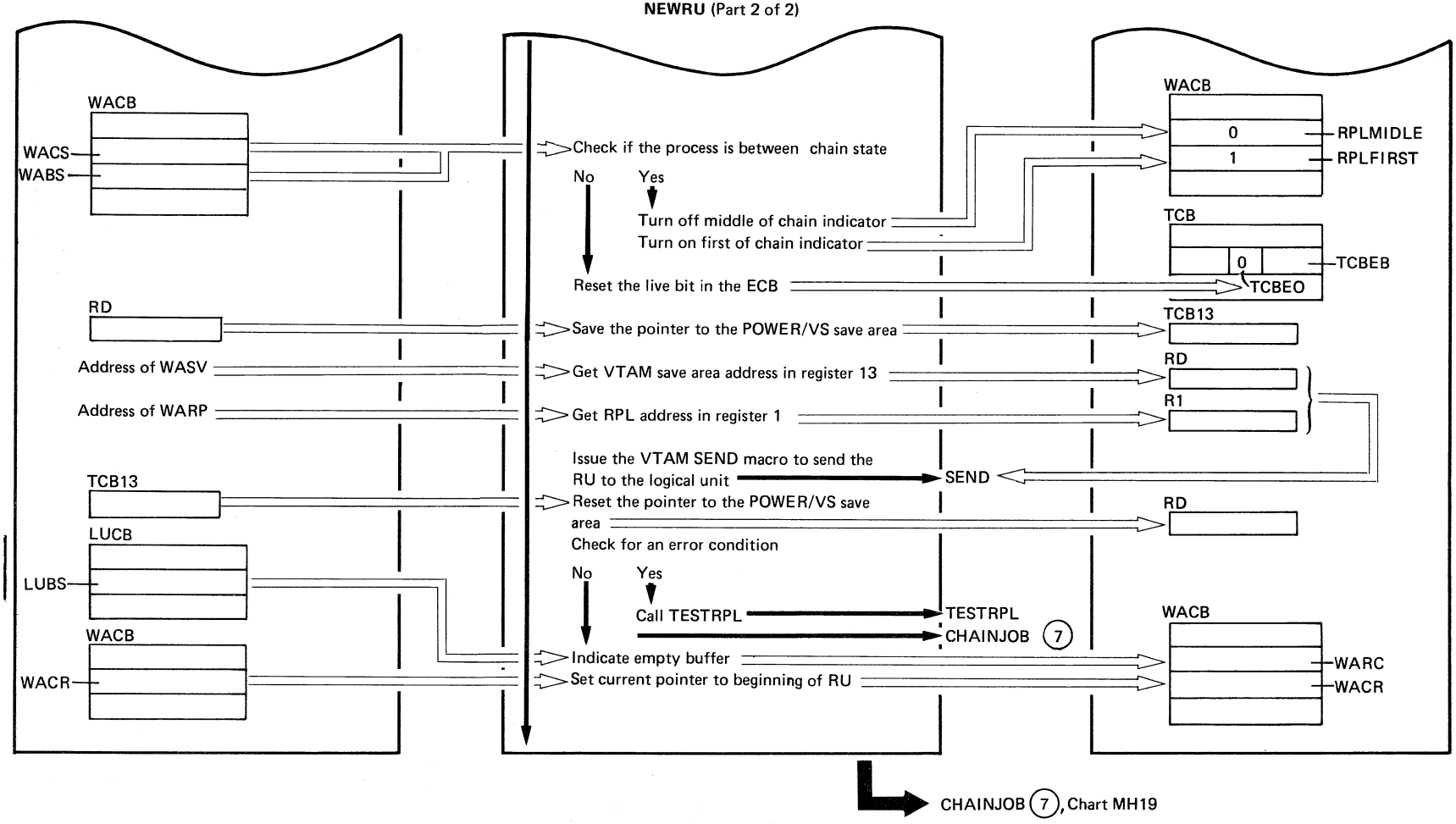
Included by CHAINJOB, Chart MH19

NEWRU (Part 1 of 2)

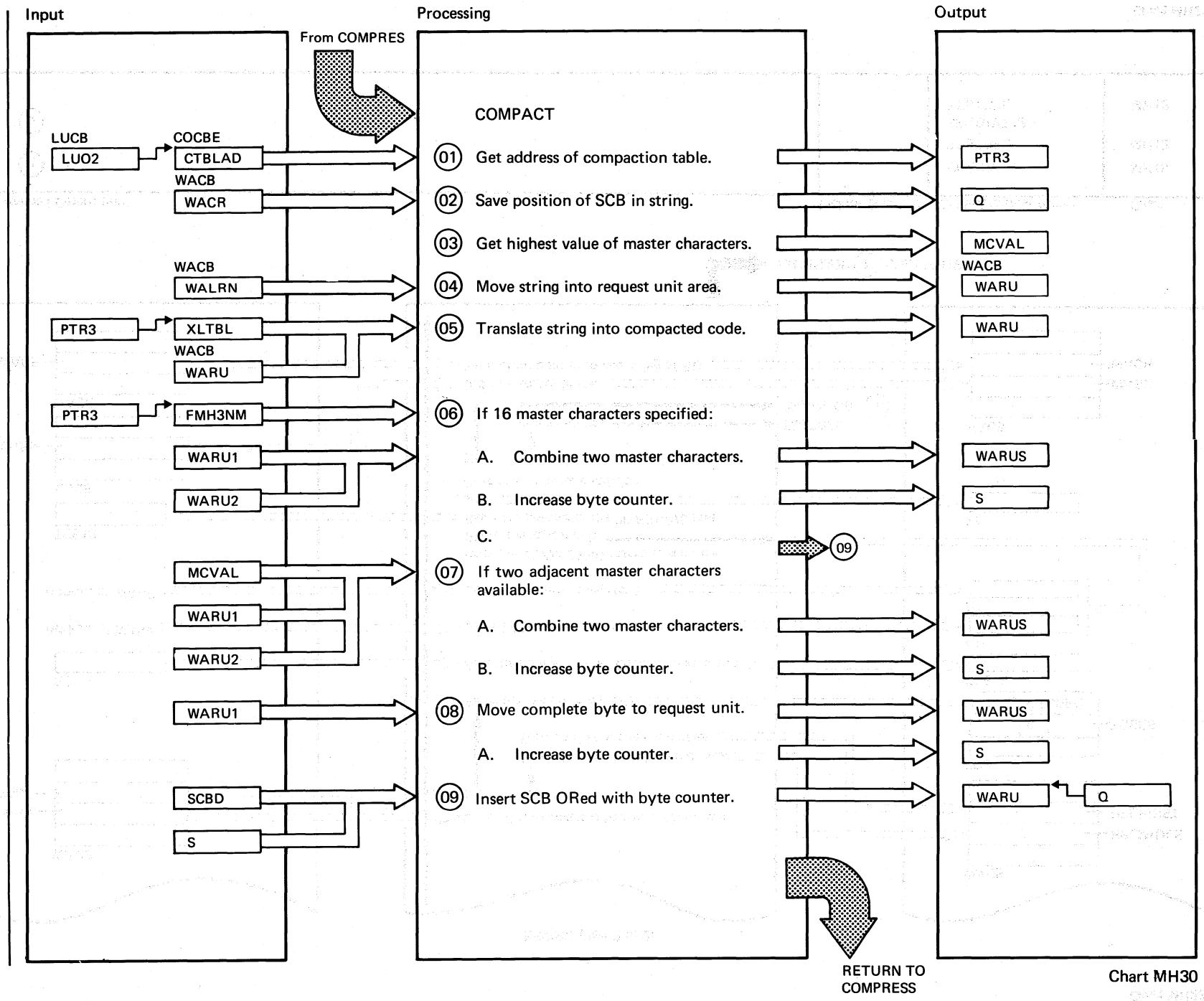


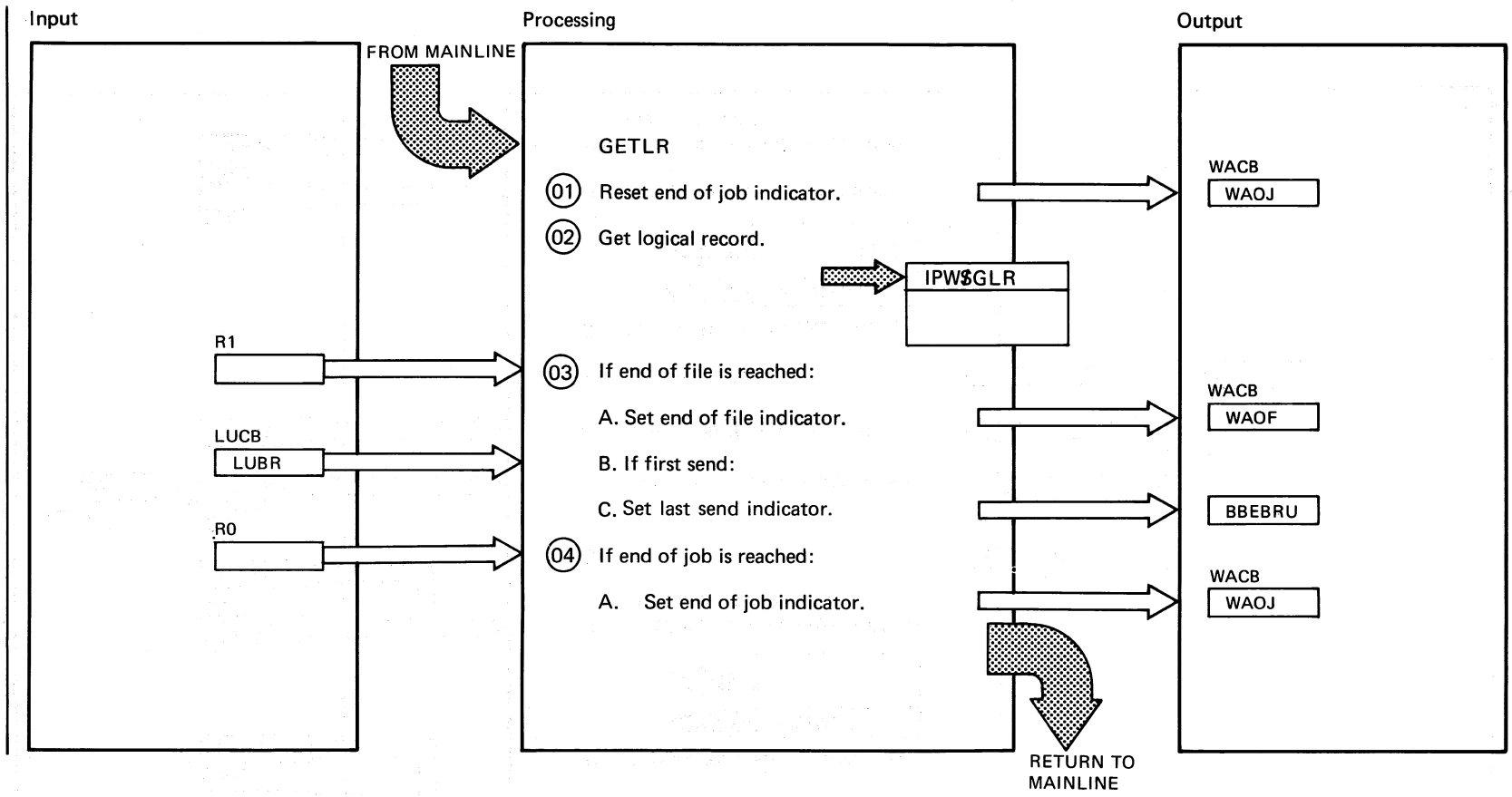
(Continued on next page)

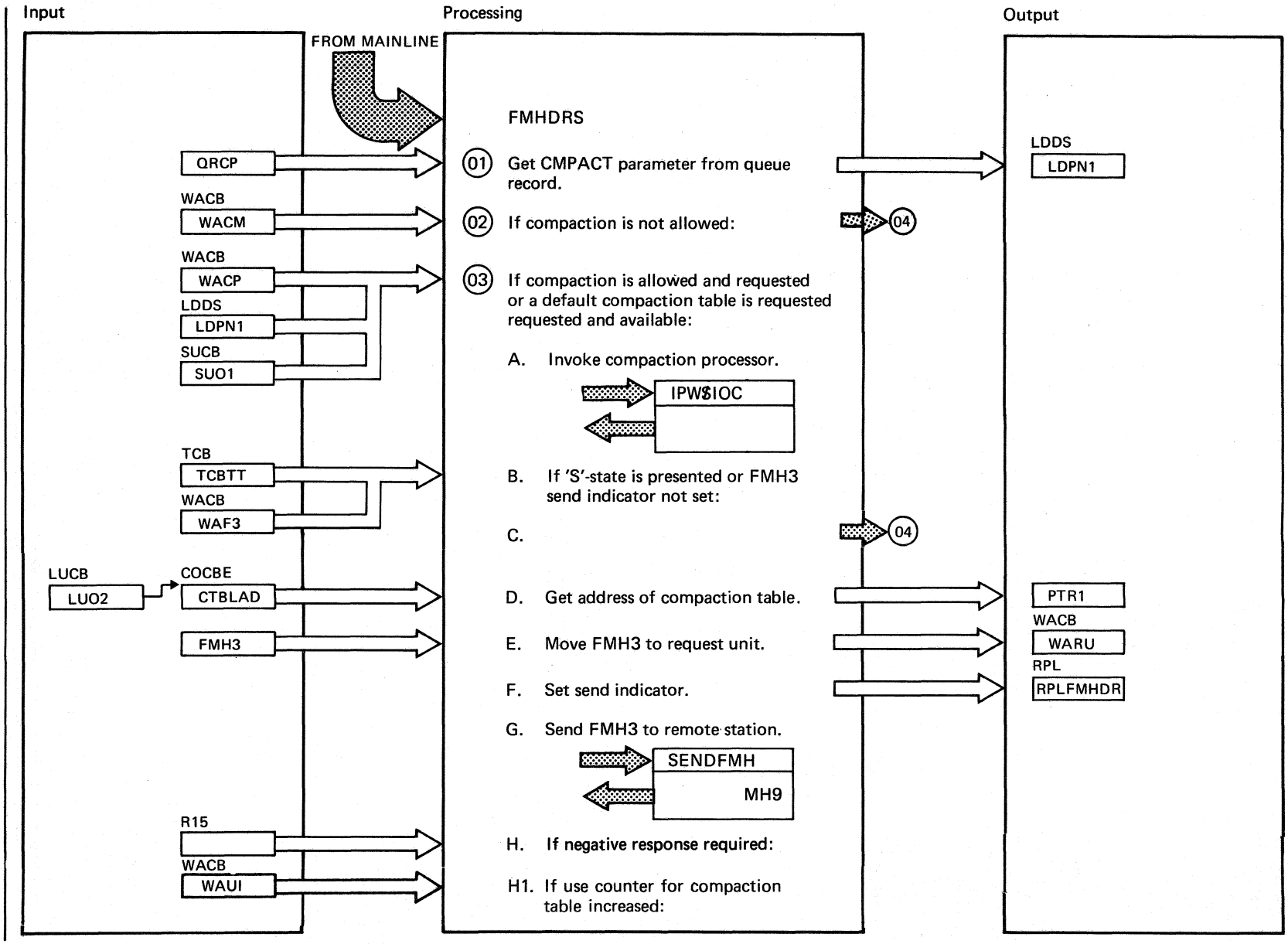
NEWRU (Part 2 of 2)

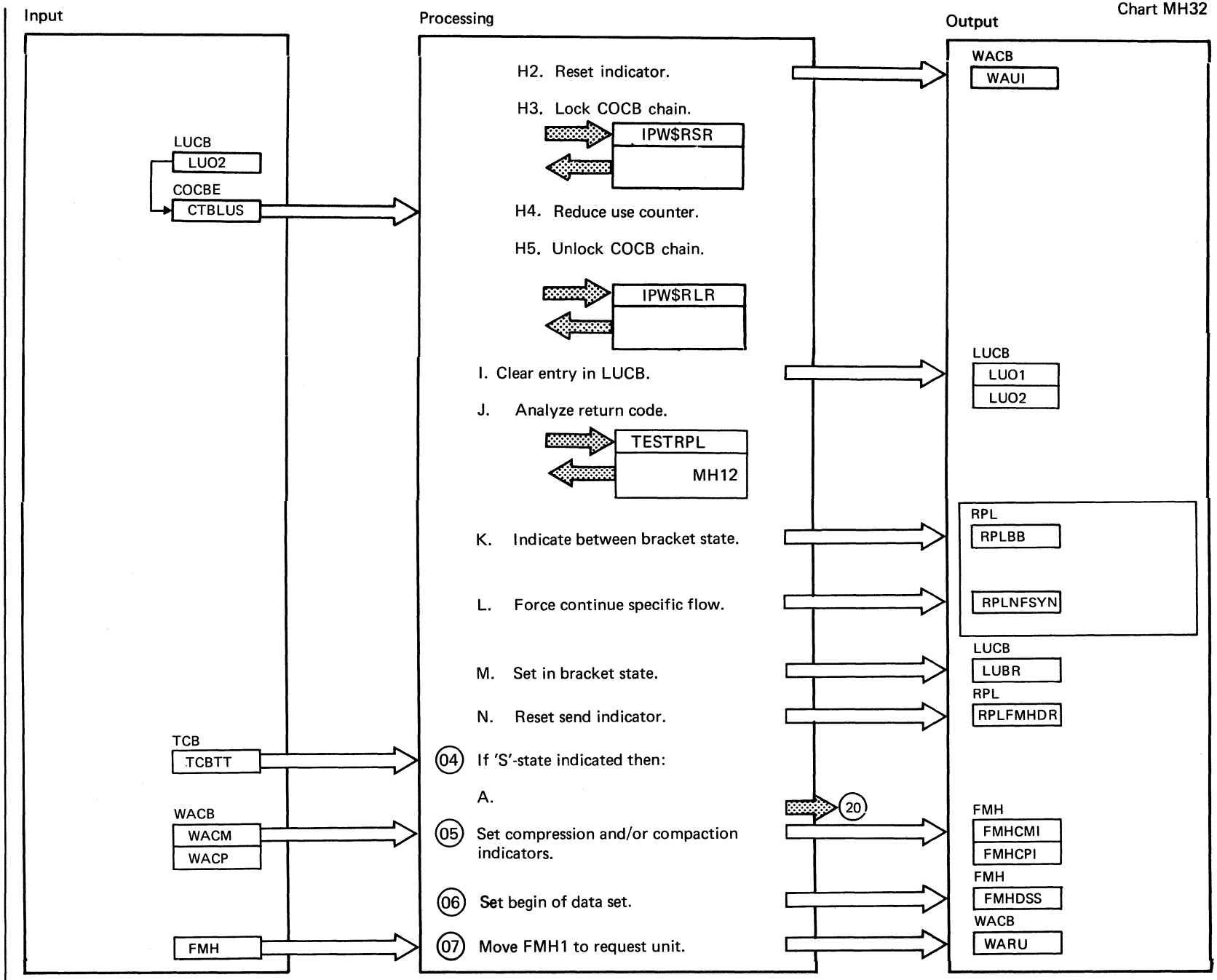


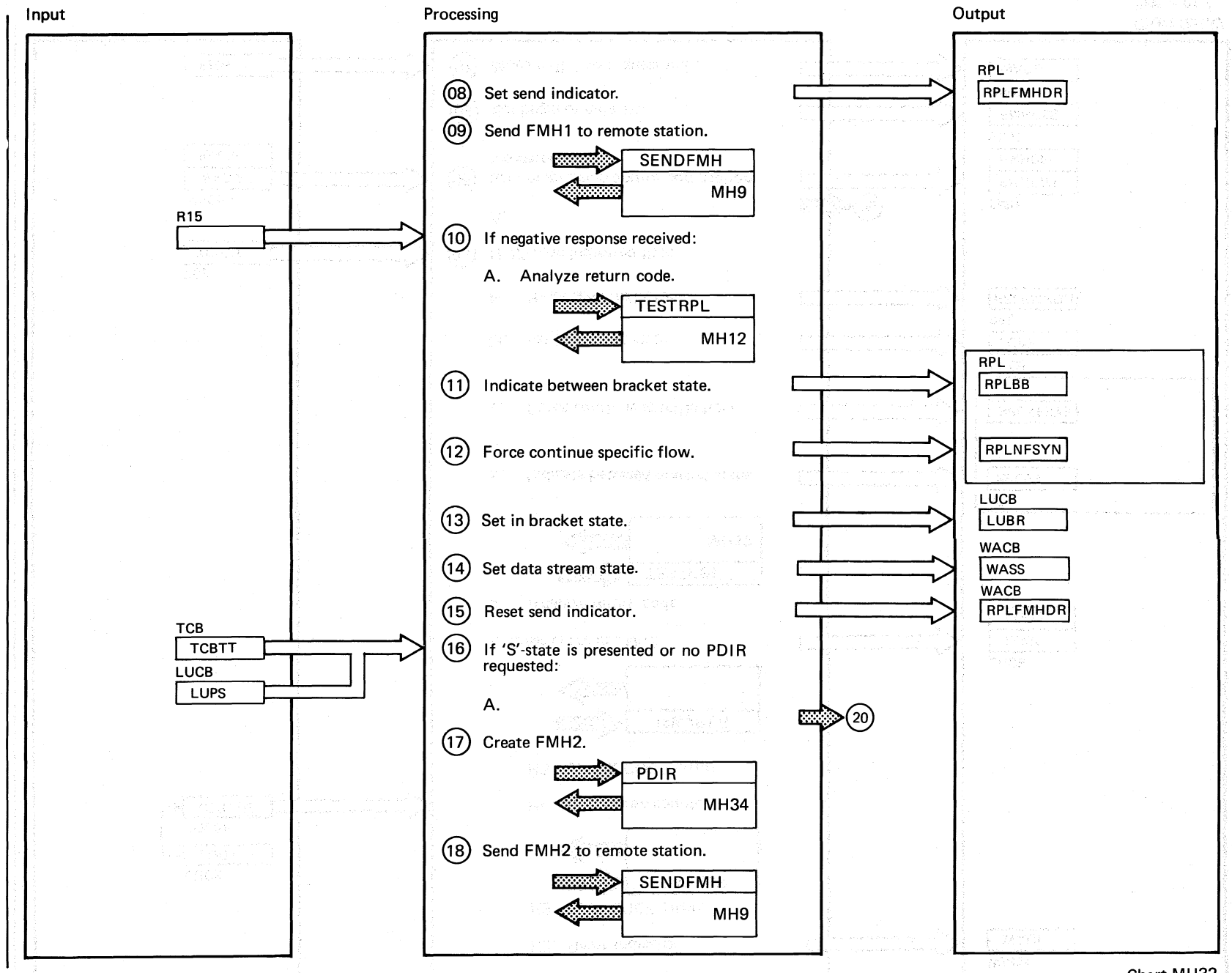
Extended Description	Include Segment	Call Subroutine/Macro	Chart
(1)		CHECK	MH18
		TESTRPL	MH12
(2)		SEND(VTAM)	
		TESTRPL	MH12

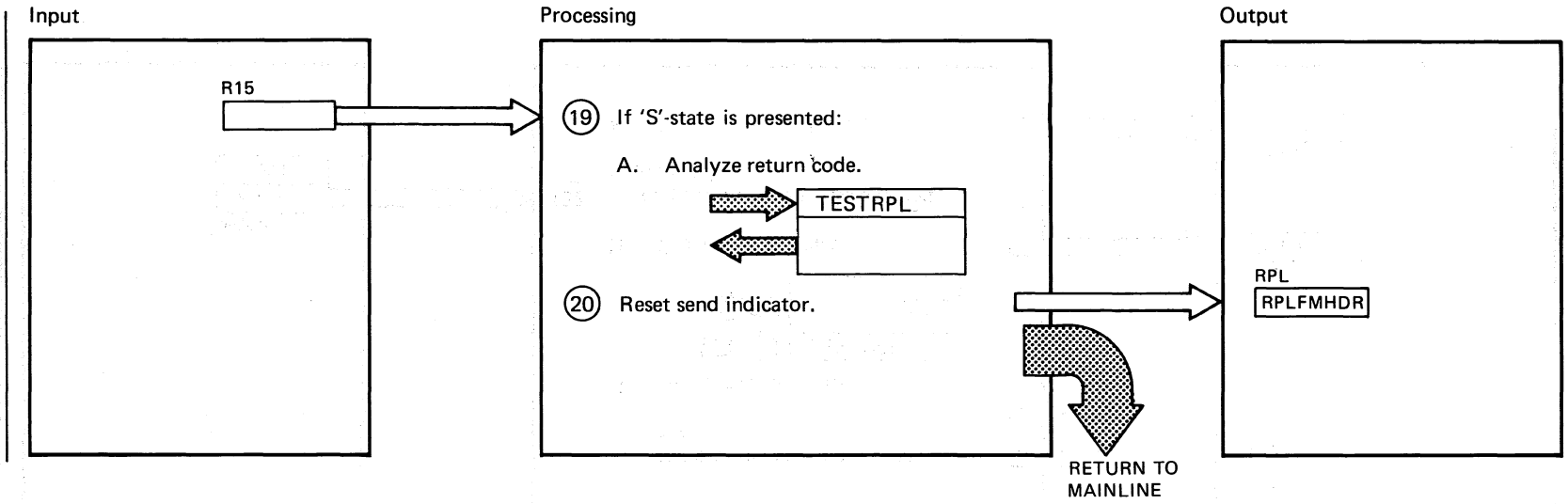

Input
Processing
Output
From COMPRES
COMPACT
LUCB
LUO2
COCBE
CTBLAD
WACB
WACR
WACB
WALRN
PTR3
XLTBL
WACB
WARU
PTR3
FMH3NM
WARU1
WARU2
MCVAL
WARU1
WARU2
WARU1
SCBD
S
01 Get address of compaction table.
02 Save position of SCB in string.
03 Get highest value of master characters.
04 Move string into request unit area.
05 Translate string into compacted code.
06 If 16 master characters specified:
A. Combine two master characters.
B. Increase byte counter.
C.
09
07 If two adjacent master characters available:
A. Combine two master characters.
B. Increase byte counter.
08 Move complete byte to request unit.
A. Increase byte counter.
09 Insert SCB ORed with byte counter.
PTR3
Q
MCVAL
WACB
WARU
WARU
WARUS
S
WARUS
S
WARUS
S
WARU
Q
RETURN TO COMPRESS


Input
Processing
Output
FROM MAINLINE
GETLR
01 Reset end of job indicator.
02 Get logical record.
IPW$GLR
R1
03 If end of file is reached:
A. Set end of file indicator.
B. If first send:
C. Set last send indicator.
LUCB
LUBR
R0
04 If end of job is reached:
A. Set end of job indicator.
WACB
WAOJ
WACB
WAOF
BBEBRU
WACB
WAOJ
RETURN TO MAINLINE


Input
Processing
Output
FROM MAINLINE
FMHDRS
QRCP
WACB
WACM
WACB
WACP
LDDS
LDPN1
SUCB
SUO1
TCB
TCBTT
WACB
WAF3
LUCB
LUO2
COCBE
CTBLAD
FMH3
R15
WACB
WAUI
01 Get CMPACT parameter from queue record.
02 If compaction is not allowed:
04
03 If compaction is allowed and requested or a default compaction table is requested requested and available:
A. Invoke compaction processor.
IPW$IOC
B. If 'S'-state is presented or FMH3 send indicator not set:
C.
04
D. Get address of compaction table.
E. Move FMH3 to request unit.
F. Set send indicator.
G. Send FMH3 to remote station.
SENDFMH
MH9
H. If negative response required:
H1. If use counter for compaction table increased:
LDDS
LDPN1
PTR1
WACB
WARU
RPL
RPLFMHDR


Input
Processing
Output
LUCB
LUO2
COCBE
CTBLUS
H2. Reset indicator.
H3. Lock COCB chain.
IPW$RSR
H4. Reduce use counter.
H5. Unlock COCB chain.
IPW$RLR
I. Clear entry in LUCB.
J. Analyze return code.
TESTRPL
MH12
K. Indicate between bracket state.
L. Force continue specific flow.
M. Set in bracket state.
N. Reset send indicator.
TCB
TCBTT
04 If 'S'-state indicated then:
A.
WACB
WACM
WACP
05 Set compression and/or compaction indicators.
06 Set begin of data set.
FMH
07 Move FMH1 to request unit.
20
WACB
WAUI
LUCB
LUO1
LUO2
RPL
RPLBB
RPLNFSYN
LUCB
LUBR
RPL
RPLFMHDR
FMH
FMHCMI
FMHCPI
FMH
FMHDSS
WACB
WARU


Input
Processing
Output
R15
TCB
TCBTT
LUCB
LUPS
08 Set send indicator.
09 Send FMH1 to remote station.
SENDFMH
MH9
10 If negative response received:
A. Analyze return code.
TESTRPL
MH12
11 Indicate between bracket state.
12 Force continue specific flow.
13 Set in bracket state.
14 Set data stream state.
15 Reset send indicator.
16 If 'S'-state is presented or no PDIR requested:
A.
20
17 Create FMH2.
PDIR
MH34
18 Send FMH2 to remote station.
SENDFMH
MH9
RPL
RPLFMHDR
RPL
RPLBB
RPLNFSYN
LUCB
LUBR
WACB
WASS
WACB
RPLFMHDR


Input
Processing
Output
R15
19 If 'S'-state is presented:
A. Analyze return code.
TESTRPL
20 Reset send indicator.
RPL
RPLFMHDR
RETURN TO
MAINLINE

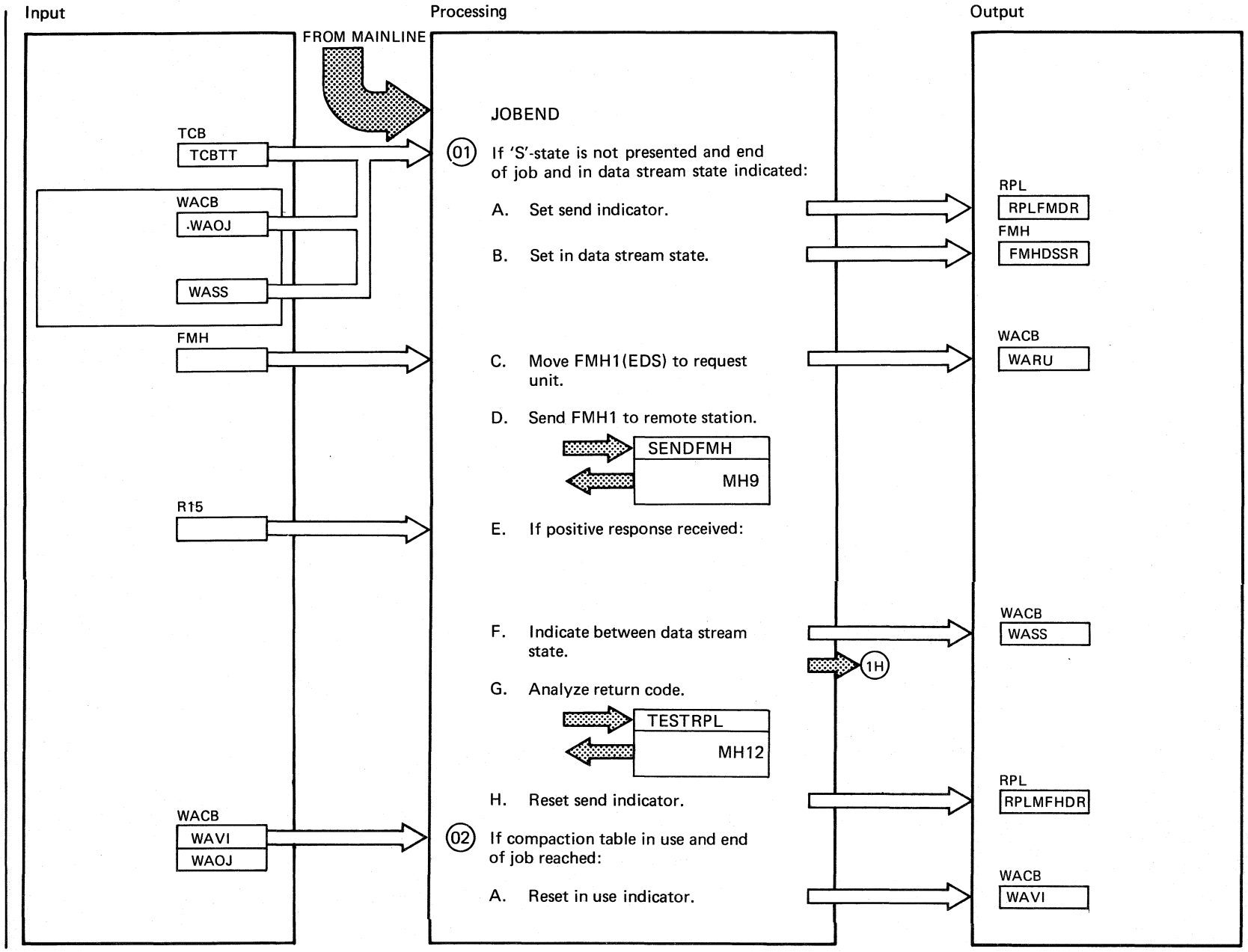
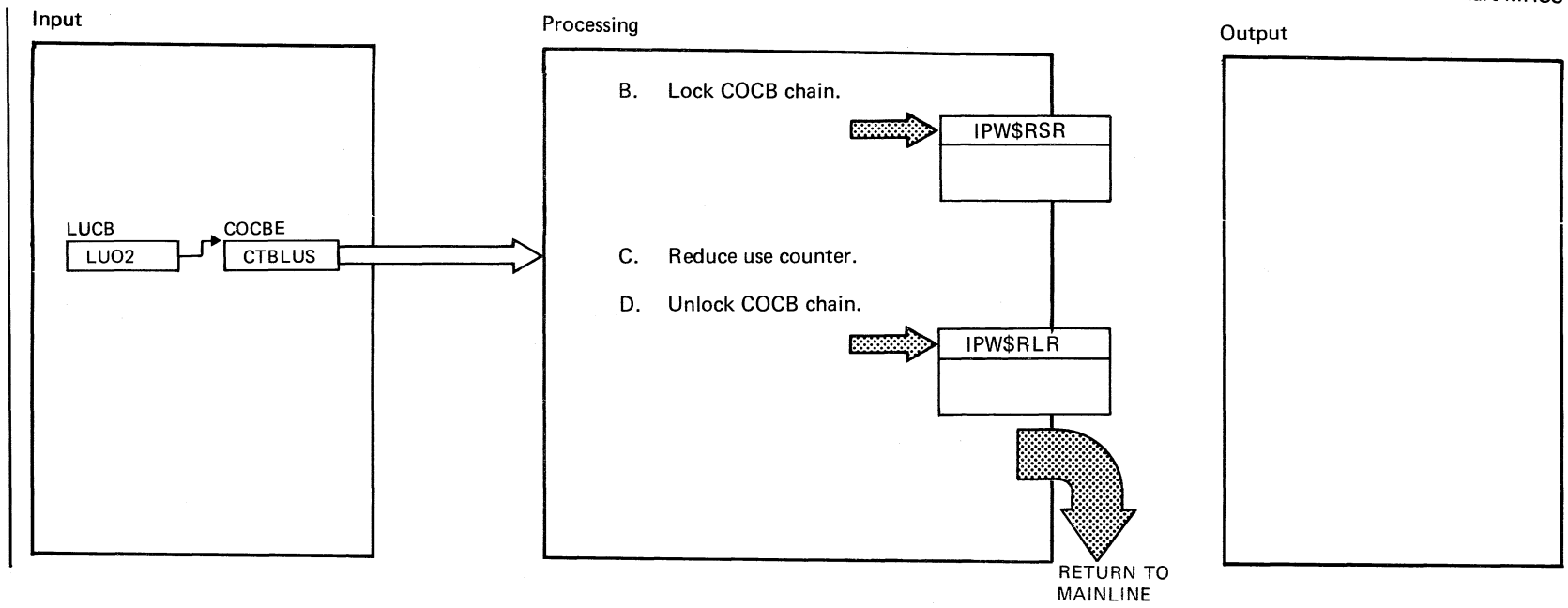


Chart MH33
Part 1 of 2


Input
Processing
Output
LUCB
LUO2
COCBE
CTBLUS
B. Lock COCB chain.
IPW$RSR
C. Reduce use counter.
D. Unlock COCB chain.
IPW$RLR
RETURN TO
MAINLINE

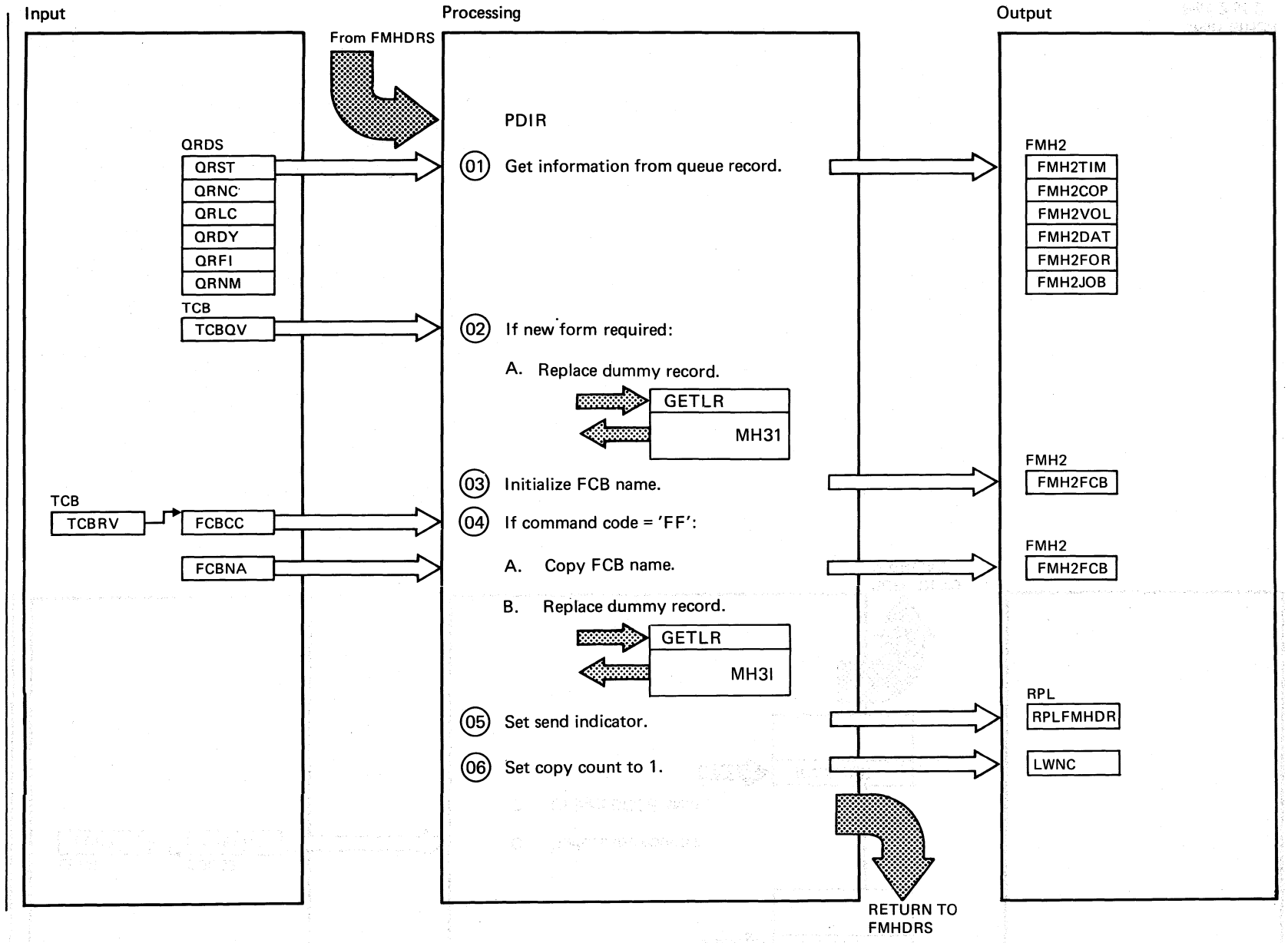
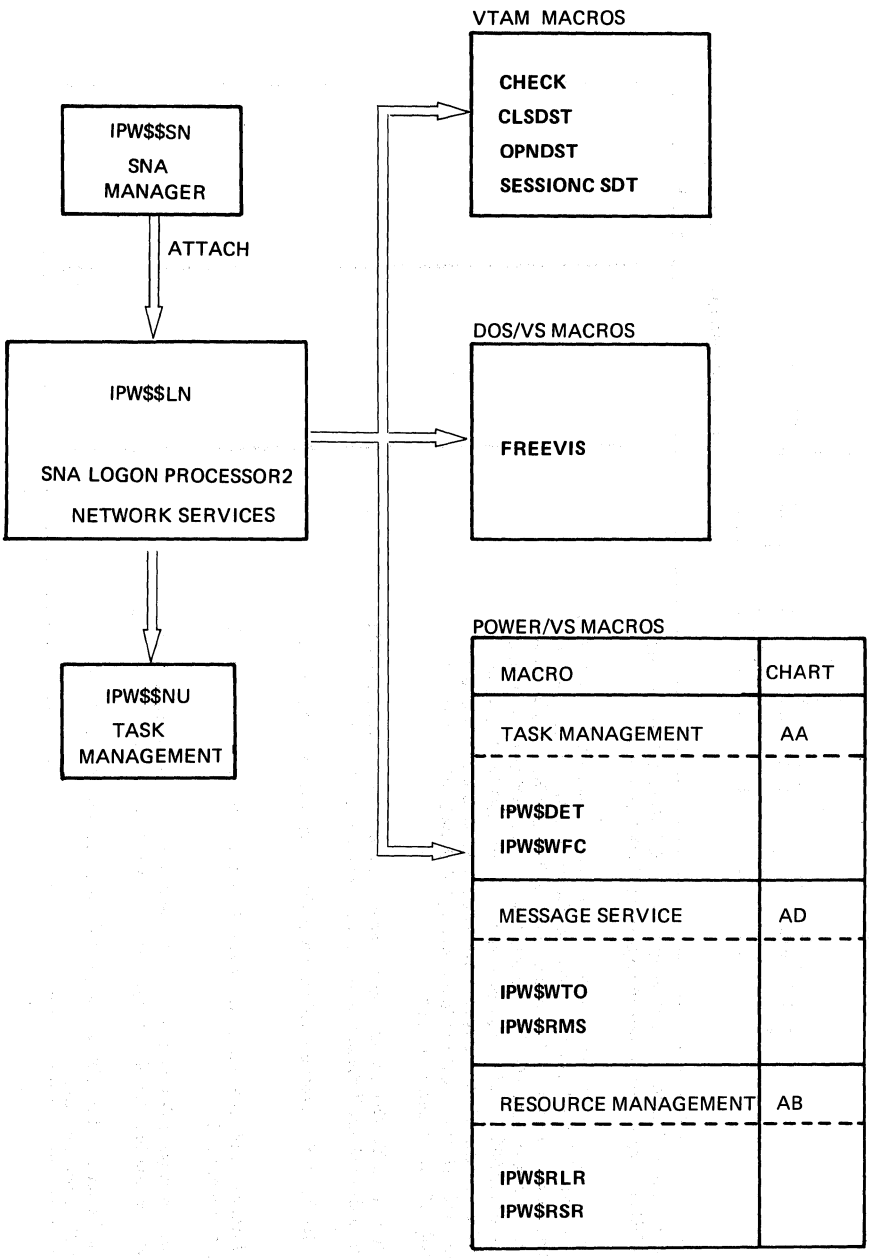

Input
Processing
Output
From FMHDRS
PDIR
QRDS
QRST
QRNC
QRLC
QRDY
QRFI
QRNM
TCB
TCBQV
01 Get information from queue record.
02 If new form required:
A. Replace dummy record.
GETLR
MH31
03 Initialize FCB name.
04 If command code = 'FF':
TCB
TCBRV
FCBCC
FCBNA
A. Copy FCB name.
B. Replace dummy record.
GETLR
MH3I
05 Set send indicator.
06 Set copy count to 1.
RETURN TO FMHDRS
FMH2
FMH2TIM
FMH2COP
FMH2VOL
FMH2DAT
FMH2FOR
FMH2JOB
FMH2
FMH2FCB
FMH2
FMH2FCB
RPL
RPLFMHDR
LWNC

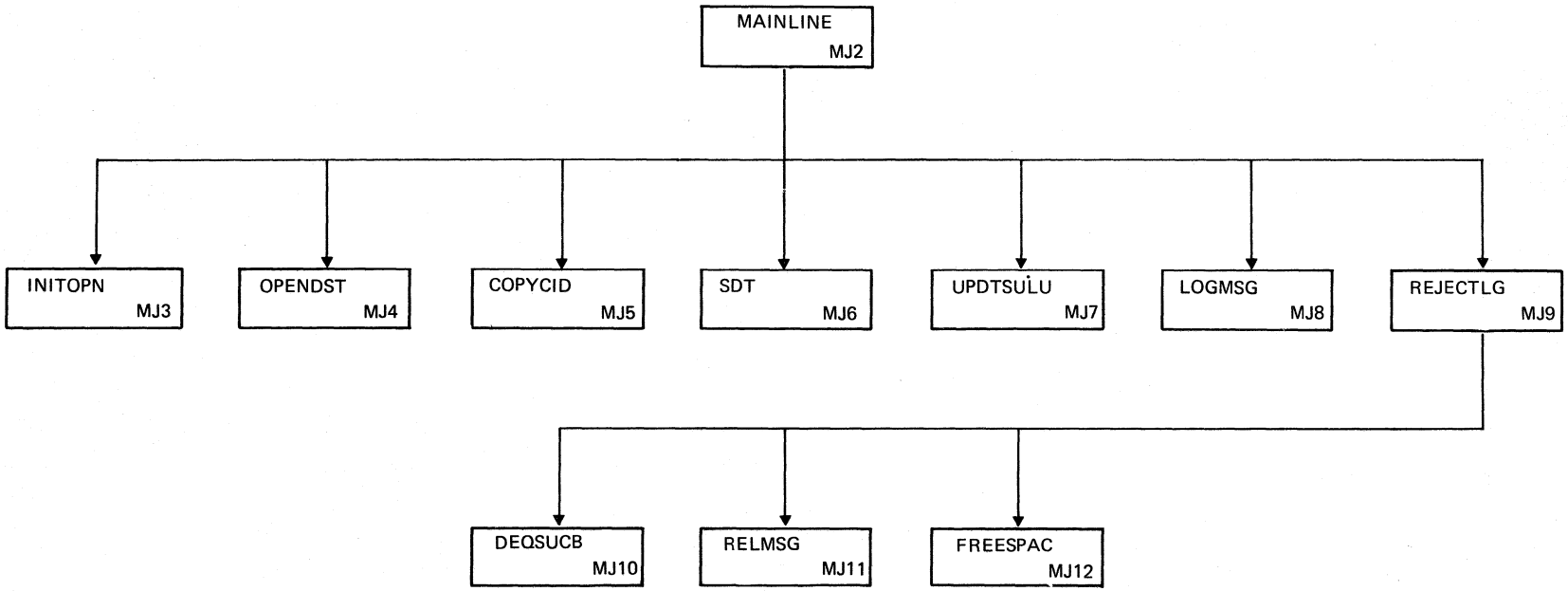
CHART MJ: IPW$$LN - SNA LOGON PROCESSOR2 NETWORK SERVICES (17 PARTS)

Chart MJ00: IPW$$LN - SNA Logon Processor2, General Flow and Macro Calls



LOGON ROUTINES (IPW$$LN) ORGANIZATION

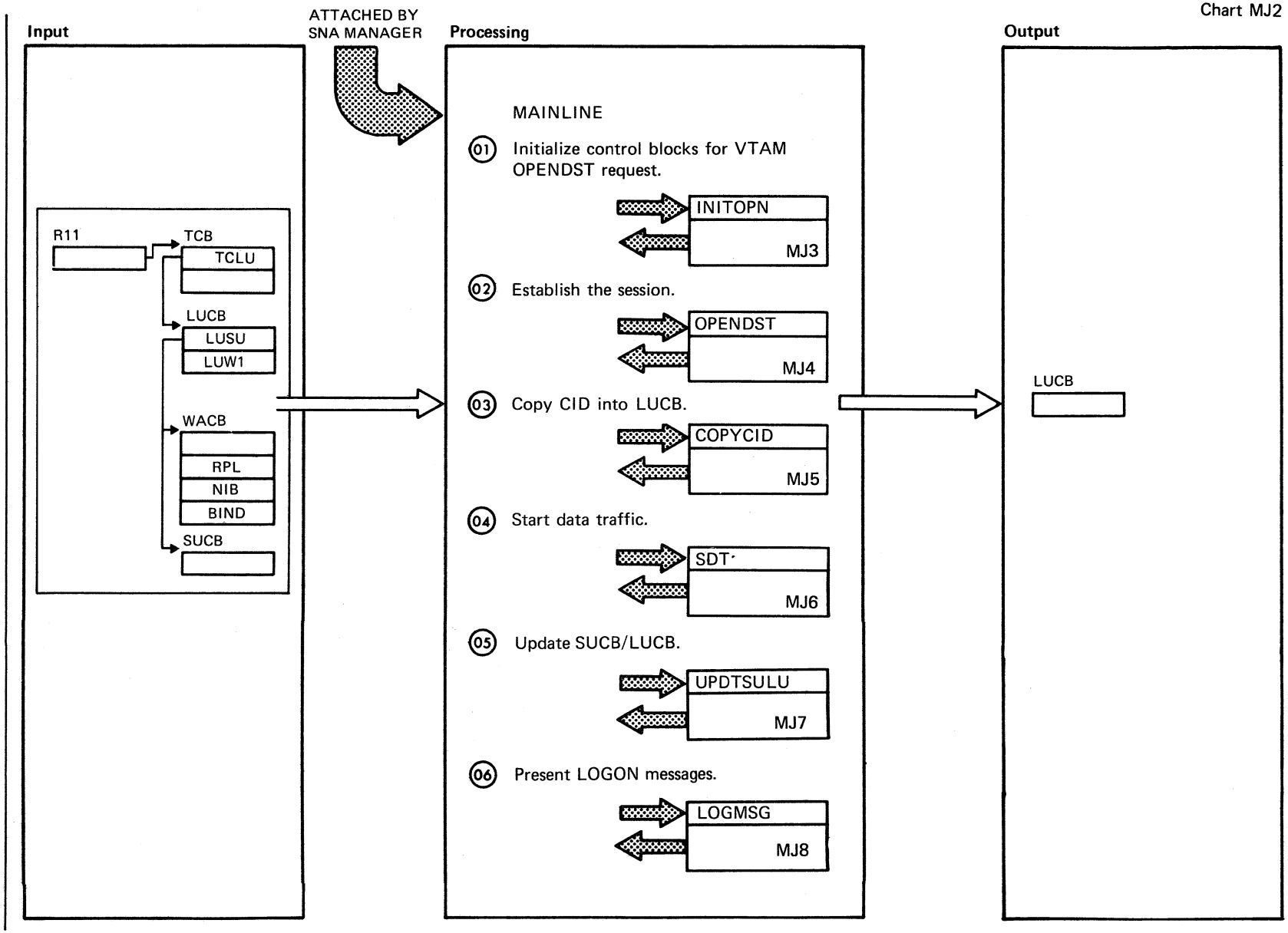
Chart MJ1

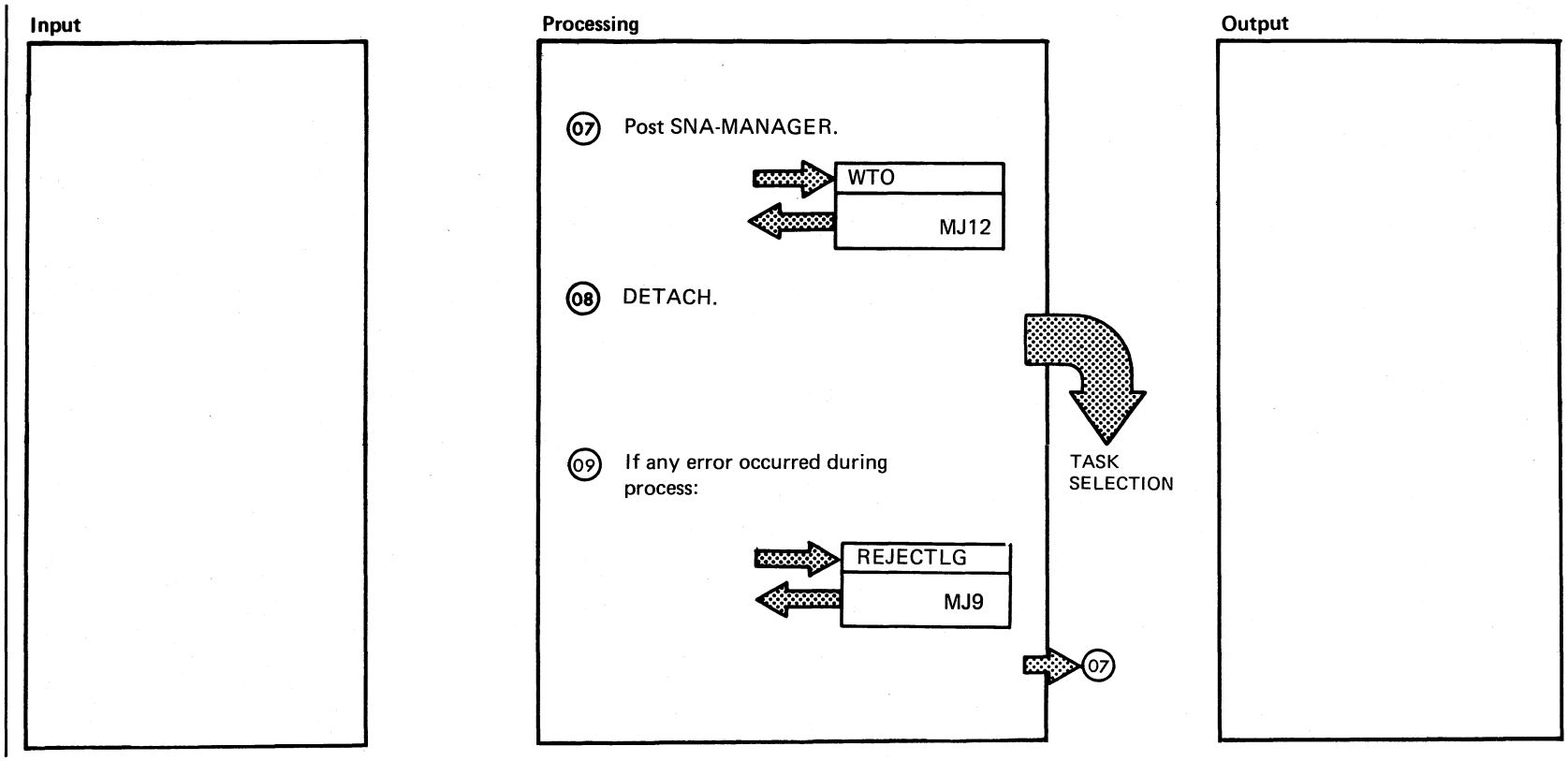


Extended Description

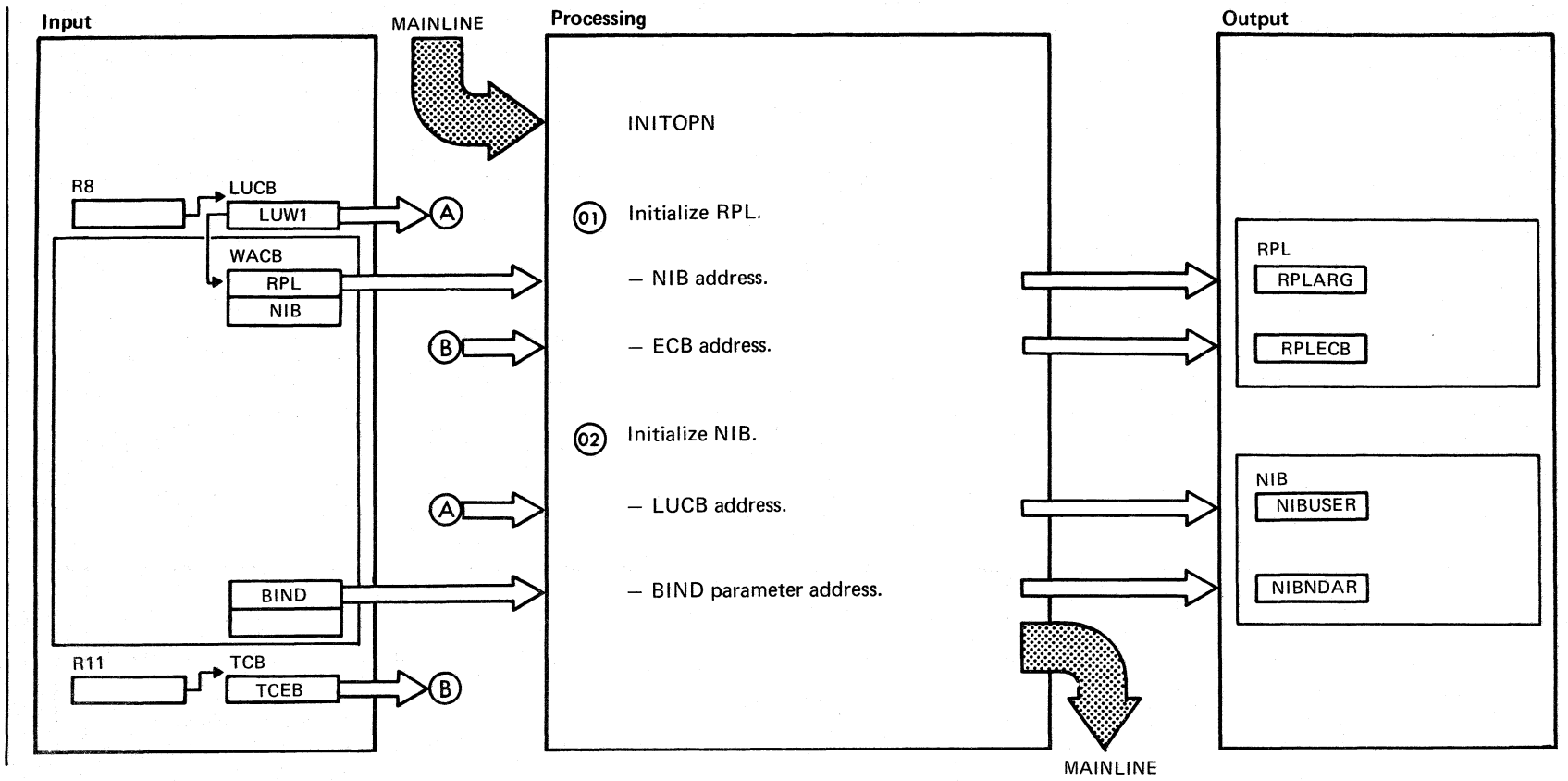
	Include Segment	Call Subroutine/ Macro	Chart
Each LOGON request is passed to the LOGON exit routine by VTAM. LOGON exit then allocates a LRCB, where it stores the LOGON request parameters. A request for IPW$$LH is set up. SNA-MANAGER attaches IPW$$LH, which retrieves 1 LOGON request and builds 1 SUCB, 1 LUCB, and 1 WACB to process the request. A VTAM inquire is issued to obtain the session parameters. The user data and bind parameters are checked for validity, and a session SUCB, LUCBs, and WACBs are obtained for the logical unit, which issued the LOGON request. A request for LOGON is set in the LUCB. SNA-MANAGER attaches IPW$$LN, which processes stage 2 of the LOGON. The session is established by using the OPNDST macro. Then messages to the central and remote operators are issued to inform on sucessful LOGON. The SUCB/LUCB are updated, and the LOGON complete is indicated in the LUCB. Finally, IPW$$LN detaches itself. In case of any error on OPNDST or SESSIONC, a message is issued to the central operator. The RPL is then analyzed for the error code to check if the session has to be terminated via CLSDST. If an error occurs on the CLSDST, no check request will be issued. Error testing on the check request is not relevant. If it was a LOGON request from a multi-logical unit, and it was not the first request from it, IPW$$LN takes no further action, and detaches itself. If it was a LOGON request from a single logical unit, or it was the first one from a multi-logical unit, any messages already queued for this logical unit (i.e., broadcast messages) are deleted from the queue. The SUCB is unchained from the active SUCB-LUCB chain, and the space for the WACB, SUCB, and LUCB is released. Then IPW$$LN detaches itself.			

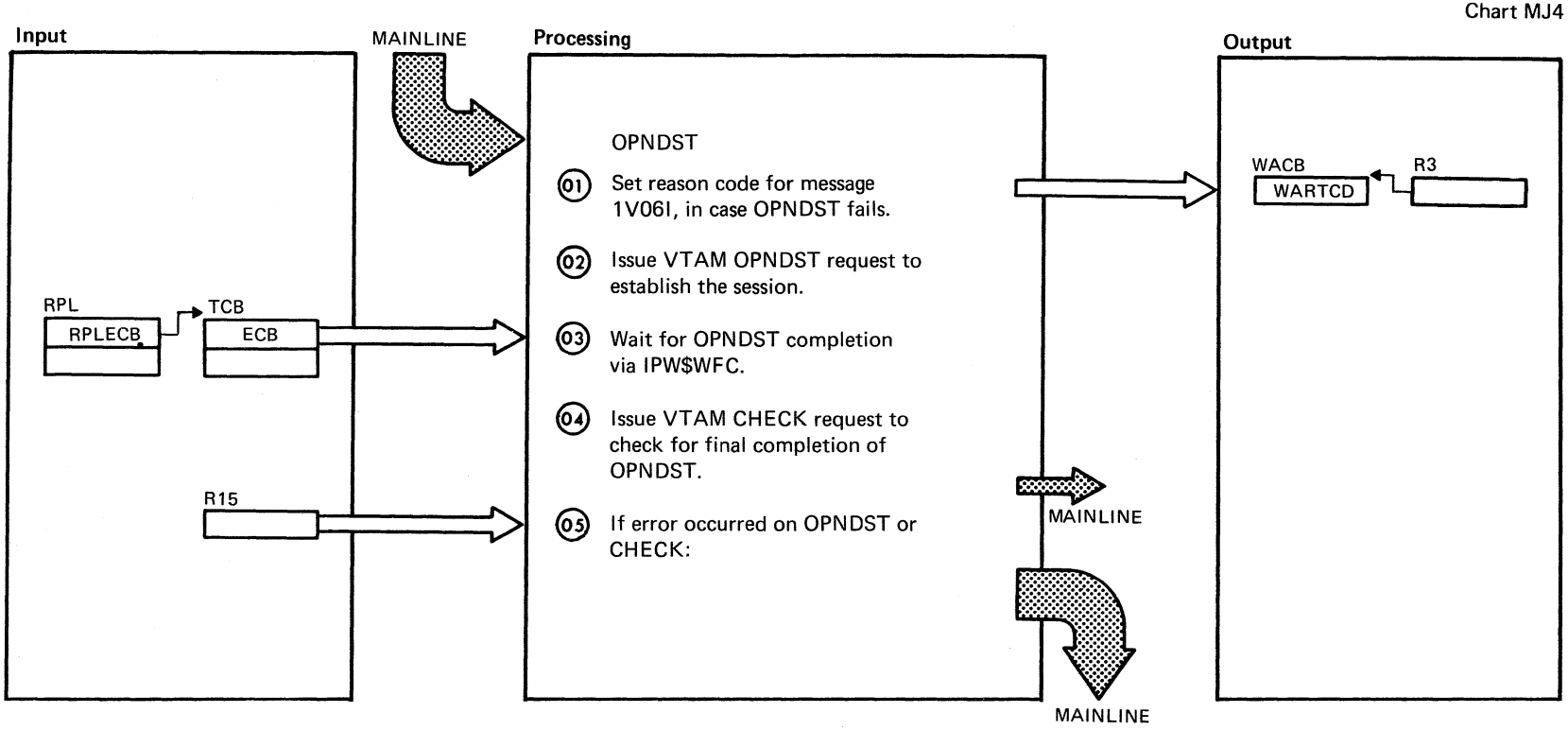
Chart MJ1


Input
ATTACHED BY
SNA MANAGER
Processing
Output
R11
TCB
TCLU
LUCB
LUSU
LUW1
WACB
RPL
NIB
BIND
SUCB
MAINLINE
01 Initialize control blocks for VTAM OPENDST request.
INITOPN
MJ3
02 Establish the session.
OPENDST
MJ4
03 Copy CID into LUCB.
COPYCID
MJ5
04 Start data traffic.
SDT
MJ6
05 Update SUCB/LUCB.
UPDTSULU
MJ7
06 Present LOGON messages.
LOGMSG
MJ8
LUCB

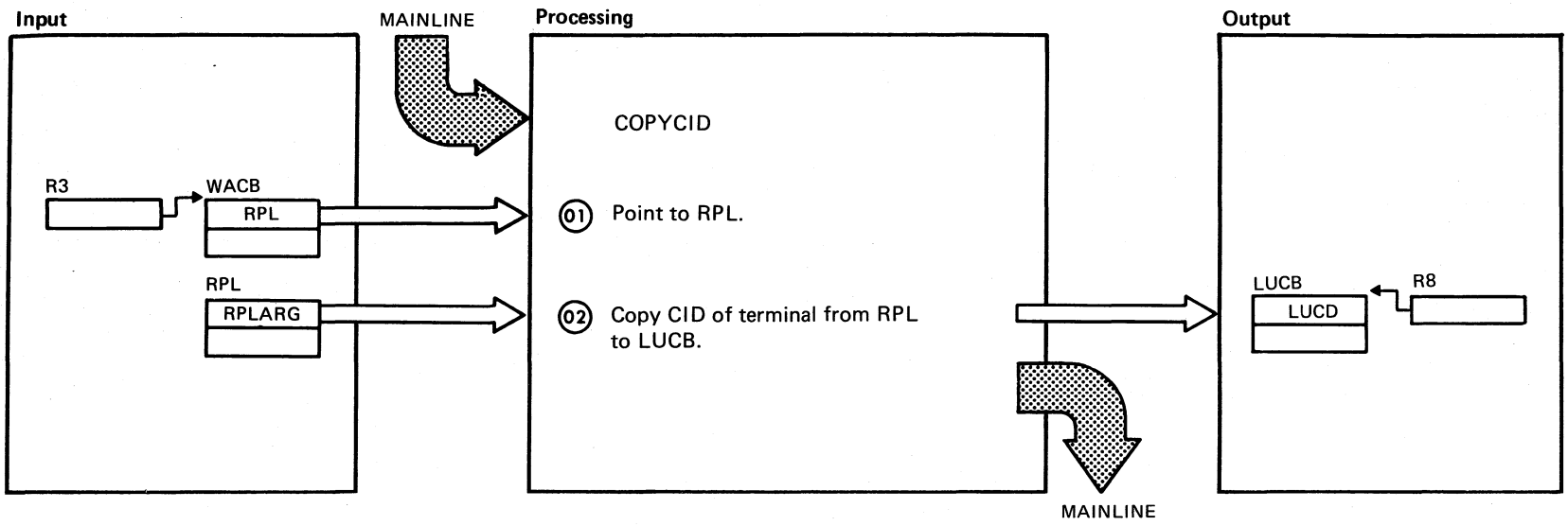

Input
Processing
Output
07 Post SNA-MANAGER.
WTO
MJ12
08 DETACH.
TASK SELECTION
09 If any error occurred during process:
REJECTLG
MJ9
07

Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(01) Set up addresses for VTAM save area and RPL, initialize ECB in TCB, set LUCB address into NIB. (02) The OPNDST is processed asynchronously, the initial acceptance is checked, and a wait for completion is issued. After final completion, the ECB is posted, and the RPL is checked. (03) If no error occurred, the CID of the terminal is copied from the RPL to the LUCB. (04) The SESSIONC SDT request is processed asynchronously. (05) Clear TCB address in LUCB to indicate LUCB is inactive, and set LOGON complete indicator. (06) Message "1V09I" REMOTE xxx LOGGED ON TO POWER ON LUNAME" is presented, both to the central and remote operator to inform them about successful LOGON. (08) A IPW$DET is issued to detach IPW$$LN and to pass control to the task selection in IPW$$NU. (09) Reject the LOGON request via a CLSDST request, free space, delete queued messages.			


Input
MAINLINE
Processing
Output
INITOPN
R8
LUCB
LUW1
A
WACB
RPL
NIB
B
01 Initialize RPL.
– NIB address.
– ECB address.
02 Initialize NIB.
A
– LUCB address.
BIND
– BIND parameter address.
R11
TCB
TCEB
B
RPL
RPLARG
RPLECB
NIB
NIBUSER
NIBNDAR
MAINLINE



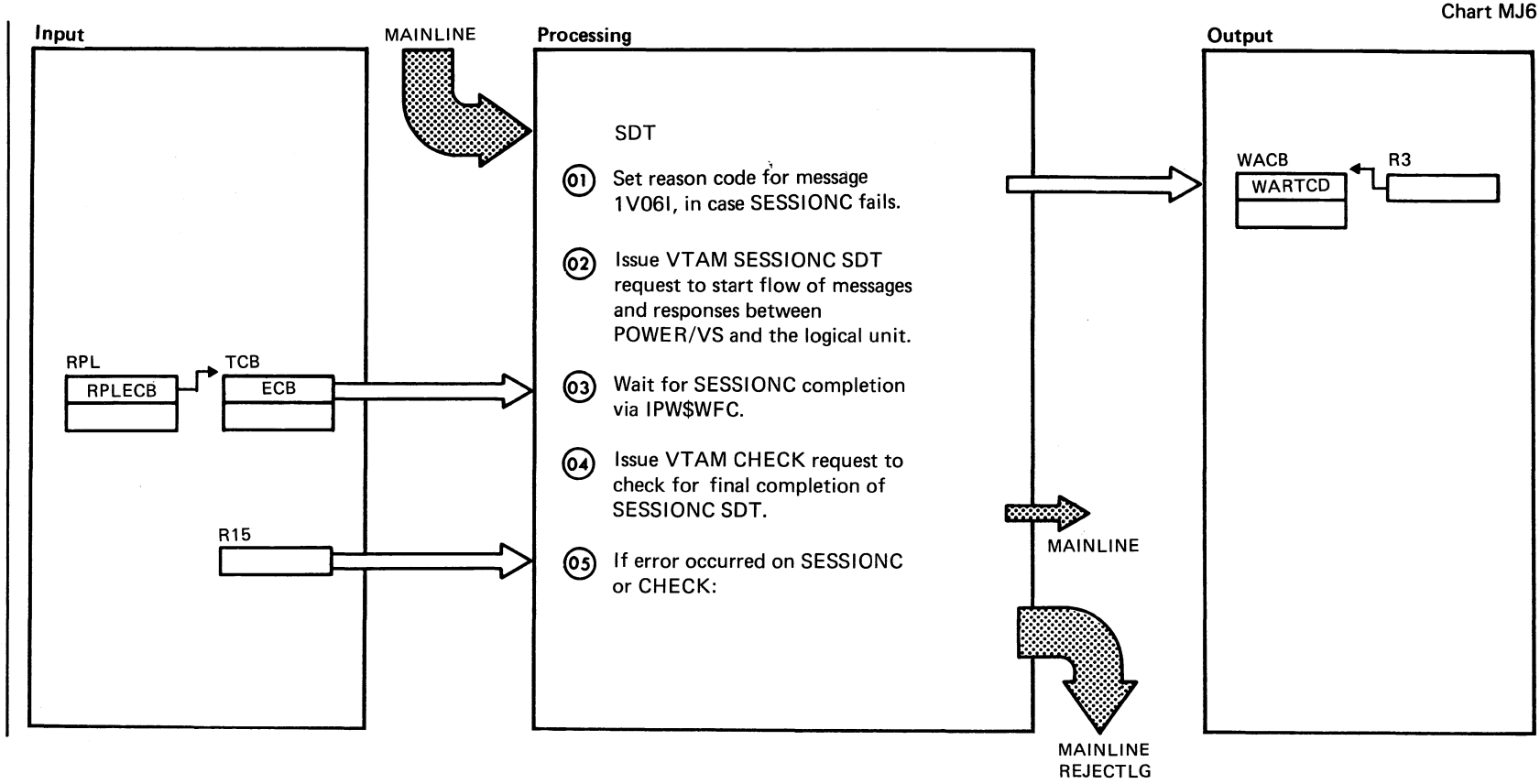
Extended Description	Include Segment	Call Subroutine/ Macro	Chart
02 The OPNDST is processed asynchronously. An ECB will be posted, when VTAM completes the OPNDST request. 03 A POWER/VS wait is issued to wait on the ECB to be posted.			

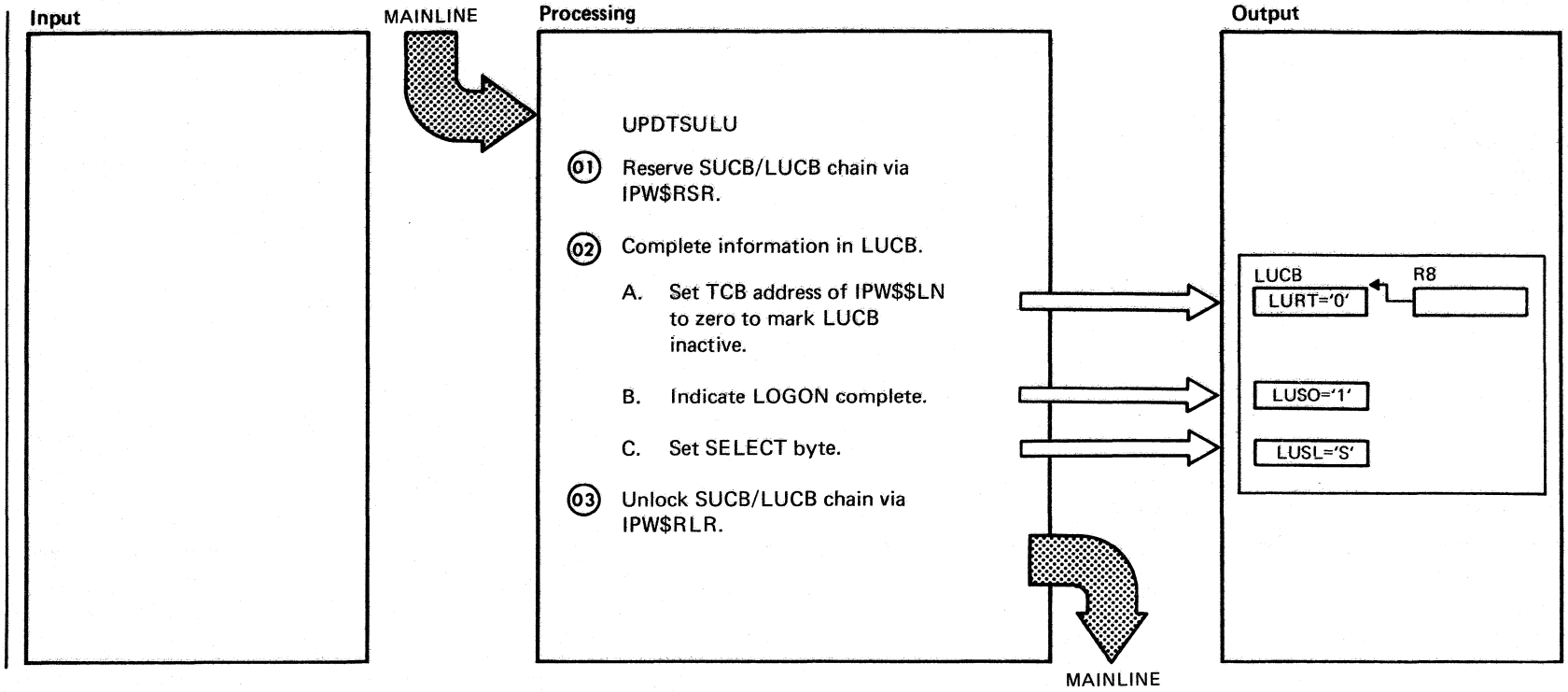


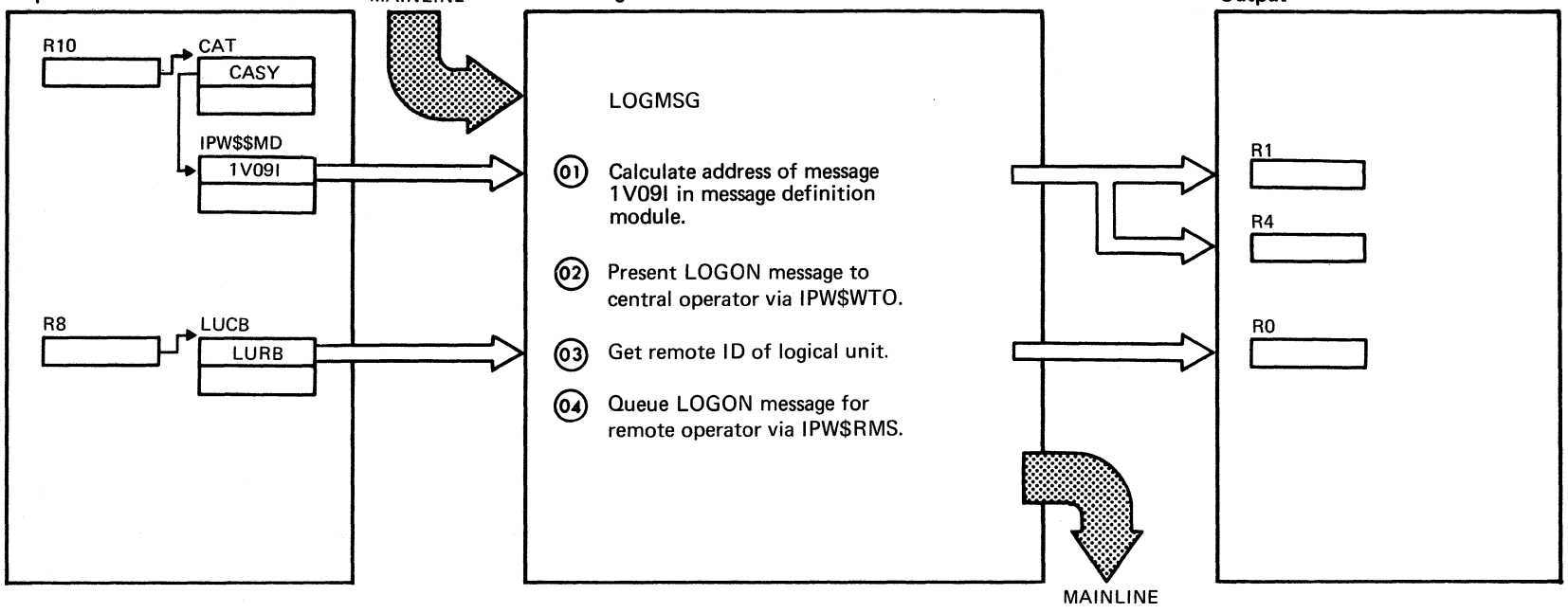
Extended Description

Include Segment	Call Subroutine/ Macro	Chart

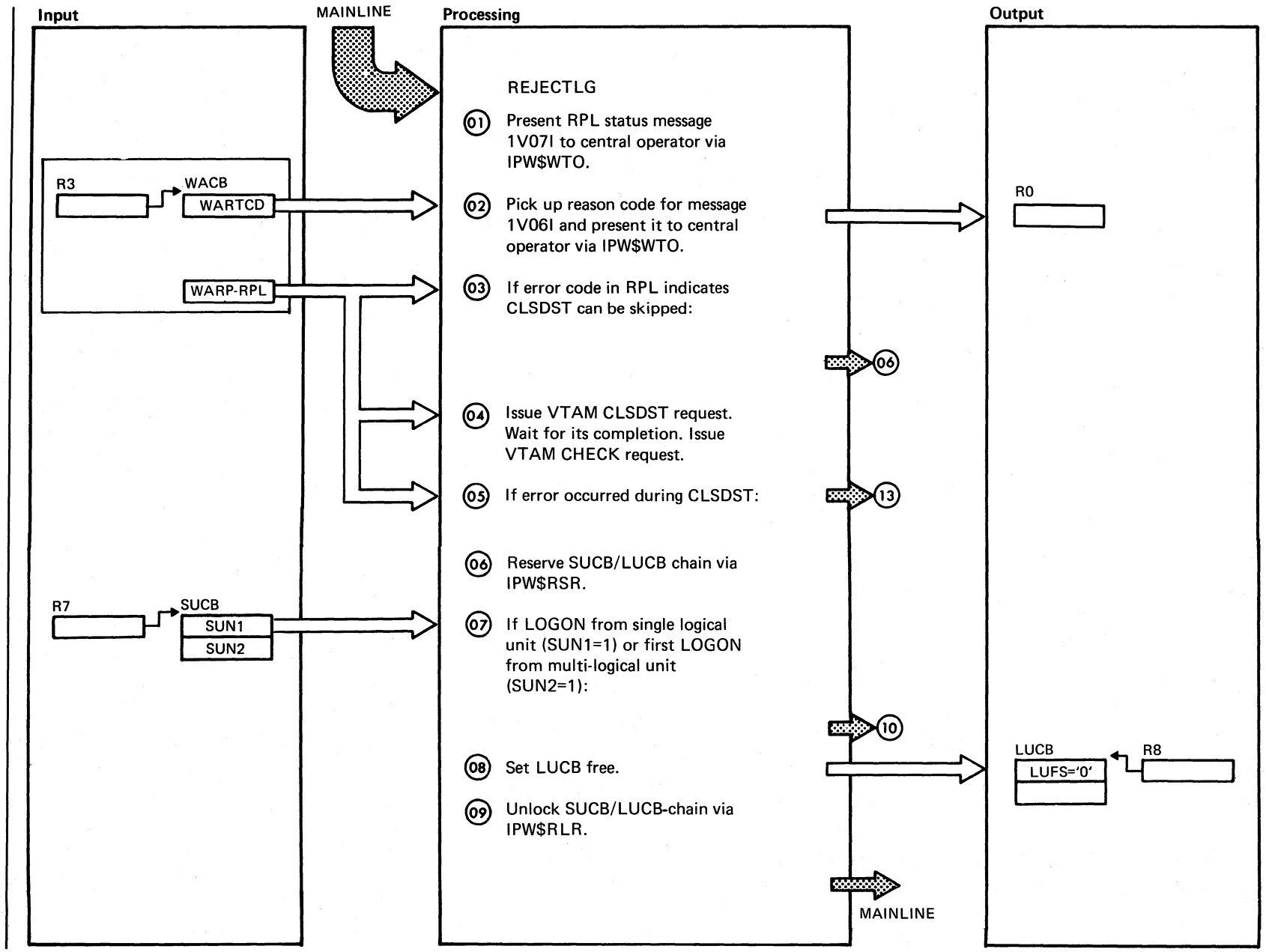
The CID of the terminal has to be known to POWER/VS, before start data traffic is given. The CID is used by the SNA-MANAGER to indentify the session, when the receive any RPL is posted.

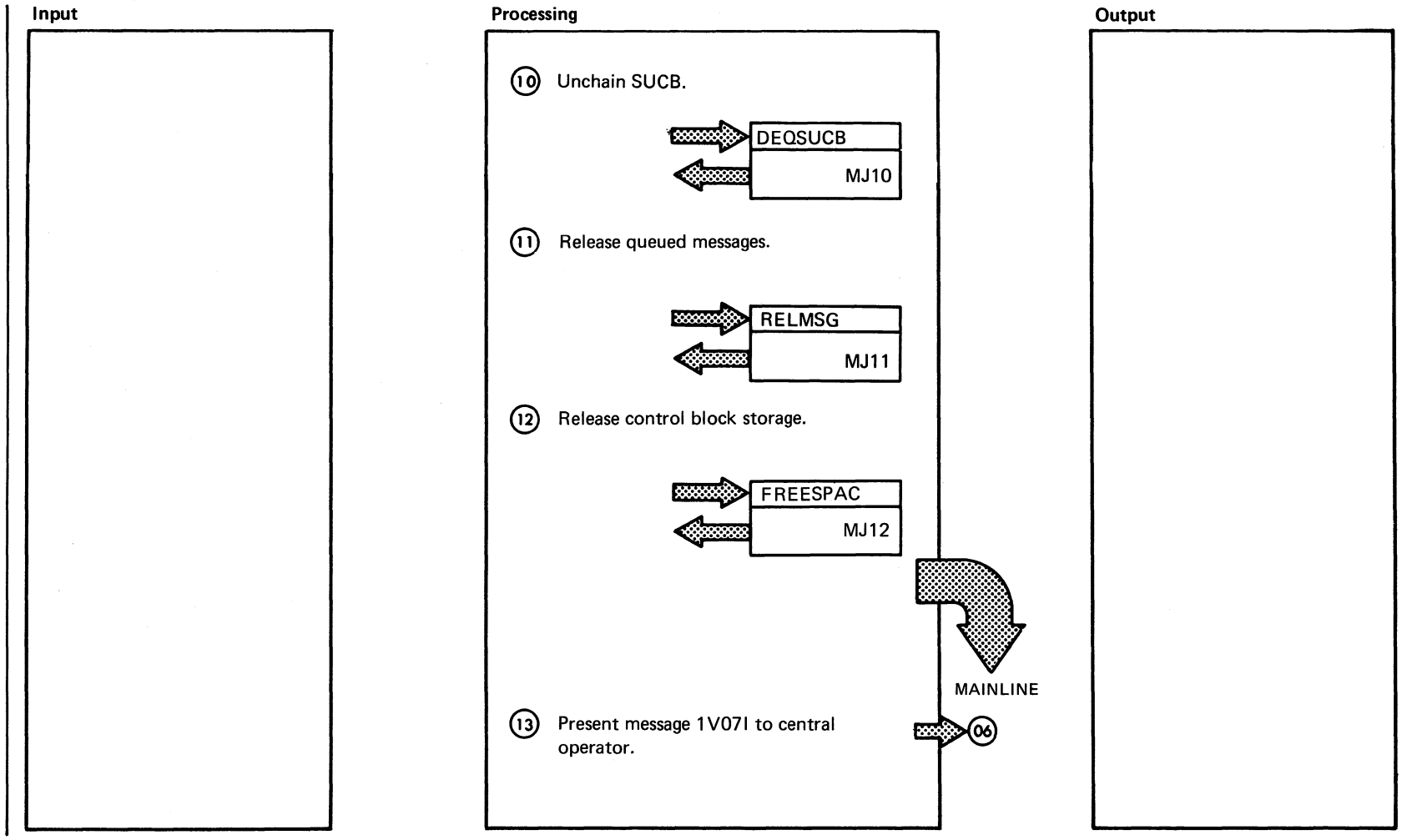

Input
MAINLINE
Processing
Output
SDT
01 Set reason code for message 1V06I, in case SESSIONC fails.
02 Issue VTAM SESSIONC SDT request to start flow of messages and responses between POWER/VS and the logical unit.
03 Wait for SESSIONC completion via IPW$WFC.
04 Issue VTAM CHECK request to check for final completion of SESSIONC SDT.
05 If error occurred on SESSIONC or CHECK:
WACB
WARTCD
R3
RPL
RPLECB
TCB
ECB
R15
MAINLINE
MAINLINE
REJECTLG


Input
MAINLINE
Processing
Output
UPDTSULU
01 Reserve SUCB/LUCB chain via IPW$RSR.
02 Complete information in LUCB.
A. Set TCB address of IPW$$LN to zero to mark LUCB inactive.
B. Indicate LOGON complete.
C. Set SELECT byte.
03 Unlock SUCB/LUCB chain via IPW$RLR.
LUCB
R8
LURT='0'
LUSO='1'
LUSL='S'
MAINLINE

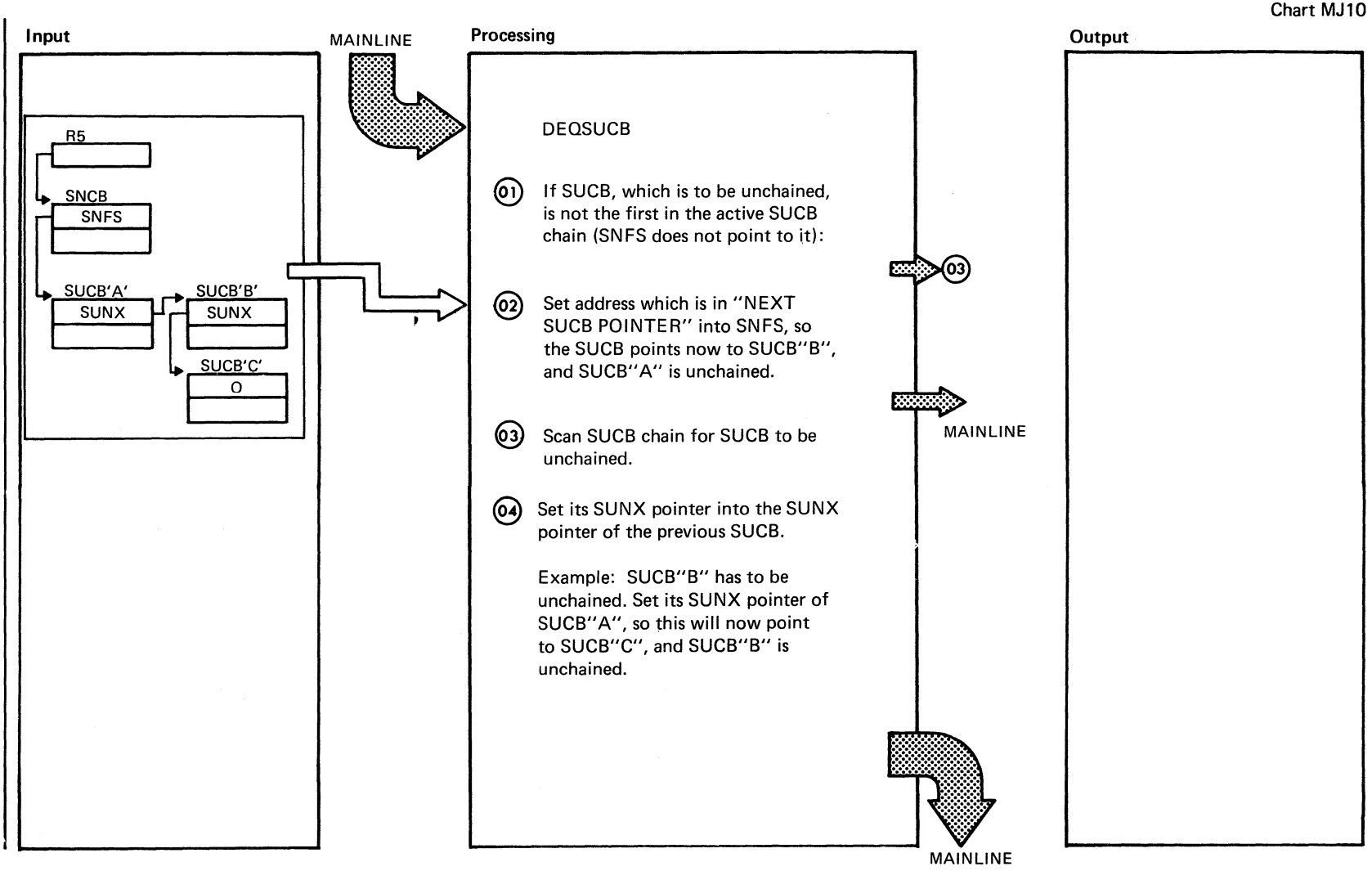


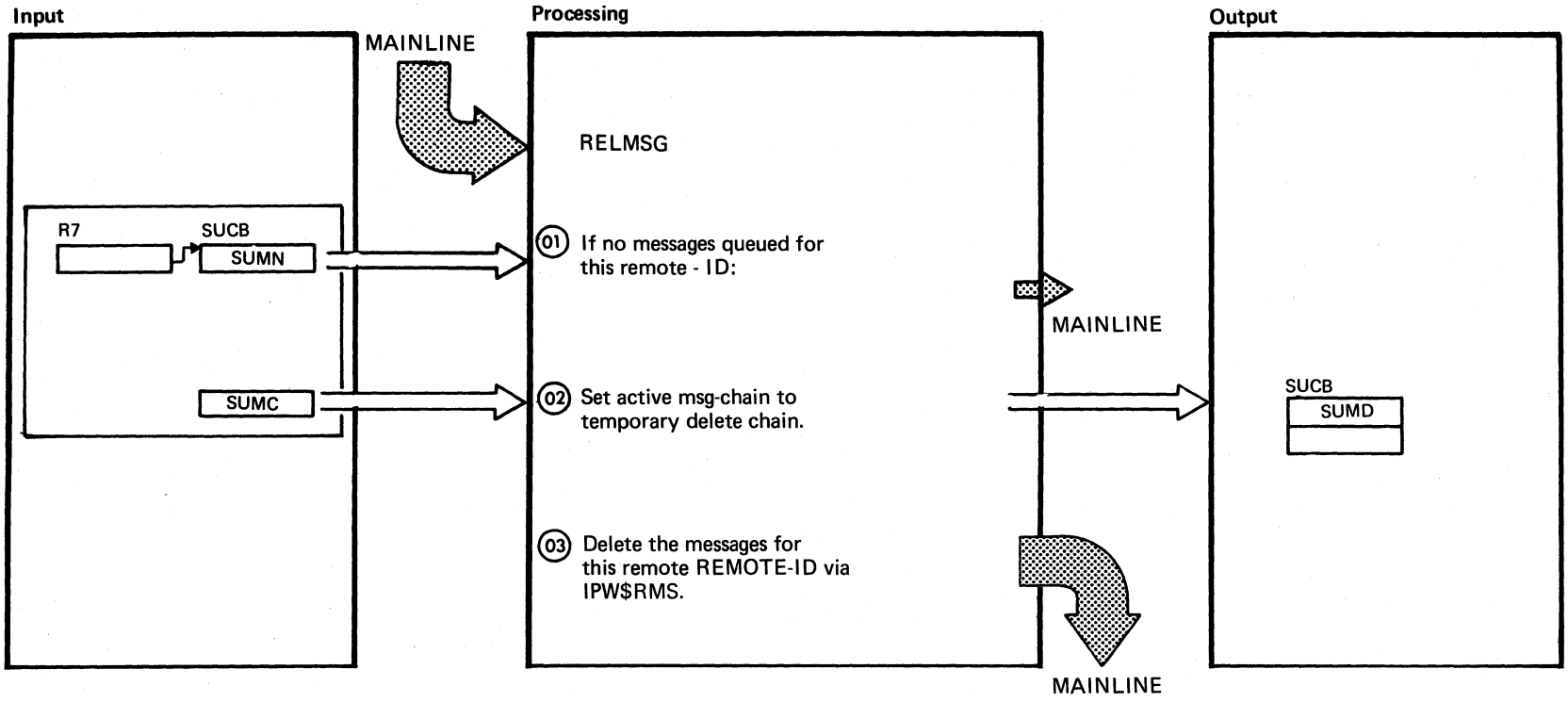
Extended Description	Include Segment	Call Subroutine/ Macro	Chart
01 Message "1V091 REMOTE xxx LOGGED ON TO POWER ON LUNAME" 04 SNA-MANAGER will find a message queued for the REMOTE-ID and attach a message processor to transmit the LOGON message.			


Input
MAINLINE
Processing
Output
REJECTLG
01 Present RPL status message 1V07I to central operator via IPW$WTO.
02 Pick up reason code for message 1V06I and present it to central operator via IPW$WTO.
03 If error code in RPL indicates CLSDST can be skipped:
04 Issue VTAM CLSDST request. Wait for its completion. Issue VTAM CHECK request.
05 If error occurred during CLSDST:
06 Reserve SUCB/LUCB chain via IPW$RSR.
07 If LOGON from single logical unit (SUN1=1) or first LOGON from multi-logical unit (SUN2=1):
08 Set LUCB free.
09 Unlock SUCB/LUCB-chain via IPW$RLR.
R3
WACB
WARTCD
WARP-RPL
R7
SUCB
SUN1
SUN2
R0
LUCB
LUFS='0'
R8
06
13
10
MAINLINE


Input
Processing
Output
10 Unchain SUCB.
DEQSUCB
MJ10
11 Release queued messages.
RELMSG
MJ11
12 Release control block storage.
FREESPAC
MJ12
MAINLINE
13 Present message 1V07I to central operator.
06

Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(01) Message for OPNDST or SESSIONC error: "1V07I ERROR ON REQUEST, RTNCD, FDBK2 = xxxx, SENSE = xxx ON LUNAME". (02) Message "1V06I UNABLE TO LOGON LUNAME RC = xxx". (03) If (RPLRTNCD=16 & RPLFDBK2=10) or (RPLRTNCD = 20 & RPLFDBK2=18) no CLSDST request can be issued. (04) The CLSDST request is processed asynchronously, the initial acceptance by VTAM is checked, and if no error occured, a IPW$WFC macro is issued to wait for final completion, and a check is issued after it. If an error occured on initial acceptance, no wait and no check will be issued. (07) If not first LOGON from multi logical unit, space for the control blocks for the work station cannot be released, because it is in use by other sessions on same work station. (08) In this case only the LUCB for the session in error is set inactive. (13) Message "1V07I ERROR ON REQUEST, RTNCD, FDB2 = xxxx, SENSE = xxx on LUNAME.			


Input
MAINLINE
Processing
Output
R5
SNCB
SNFS
SUCB'A'
SUNX
SUCB'B'
SUNX
SUCB'C'
0
DEQSUCB
01 If SUCB, which is to be unchained, is not the first in the active SUCB chain (SNFS does not point to it):
03
02 Set address which is in "NEXT SUCB POINTER" into SNFS, so the SUCB points now to SUCB"B", and SUCB"A" is unchained.
MAINLINE
03 Scan SUCB chain for SUCB to be unchained.
04 Set its SUNX pointer into the SUNX pointer of the previous SUCB.
Example: SUCB"B" has to be unchained. Set its SUNX pointer of SUCB"A", so this will now point to SUCB"C", and SUCB"B" is unchained.
MAINLINE



Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(01) Messages (i.e. from broadcasts) can be queued for a remote - id, as soon as its SUCB is chained to the active SUCB - chain. As the SUCB was taken out off the chain in routine DEQSUCB, no messages can be queued any more. (02) These messages have to be deleted from the message queue.			

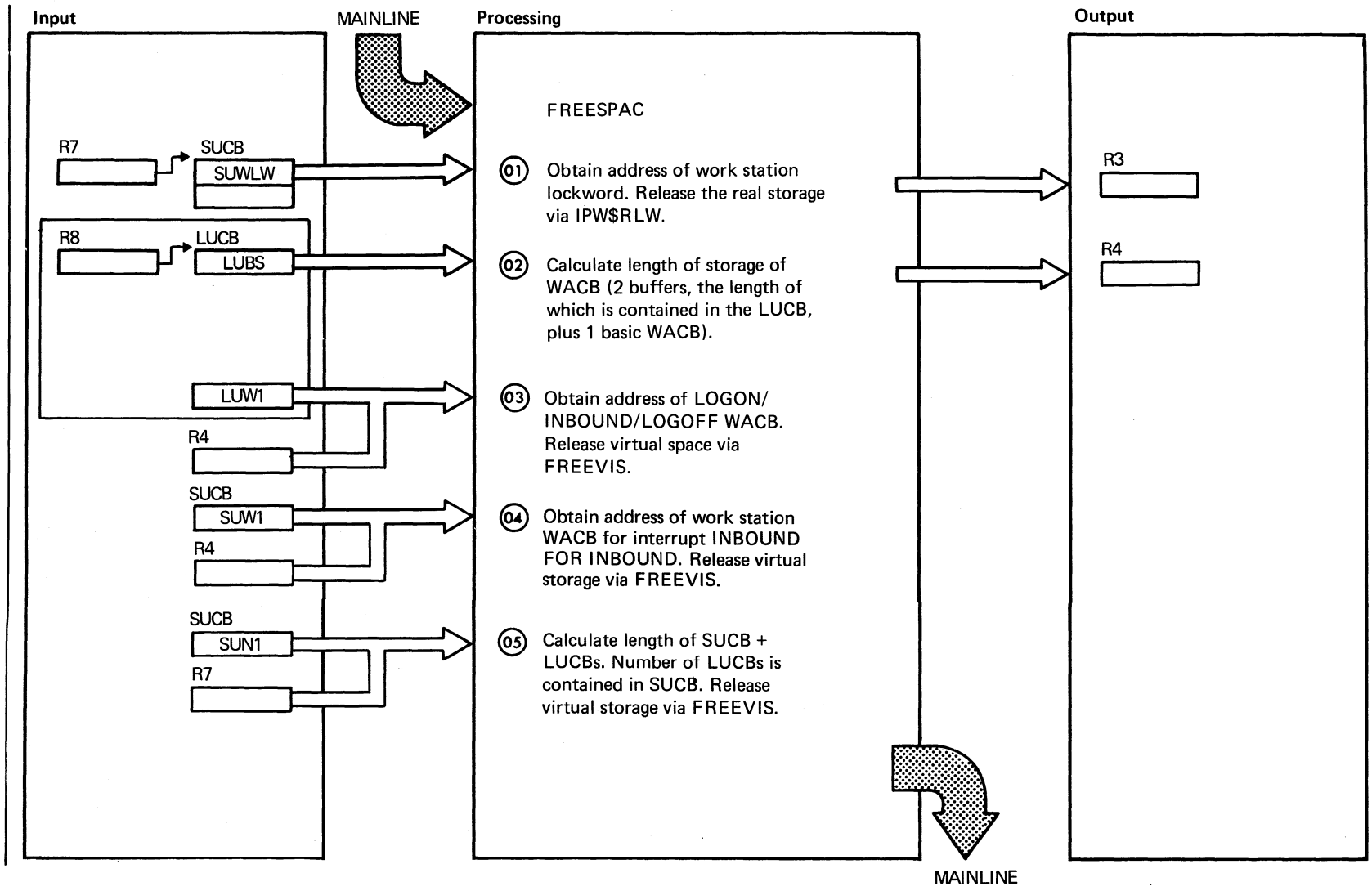
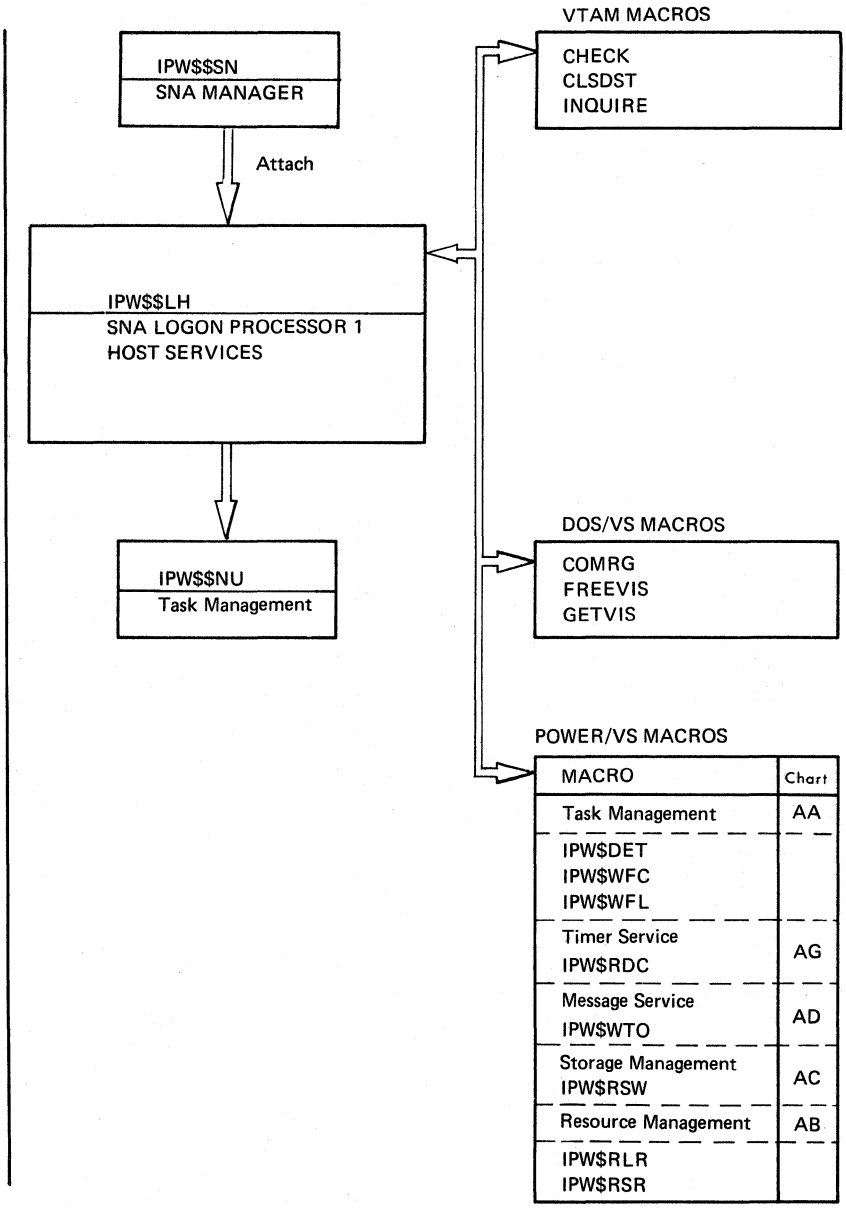

Input
MAINLINE
Processing
Output
FREESPAC
R7
SUCB
SUWLW
R8
LUCB
LUBS
LUW1
R4
SUCB
SUW1
R4
SUCB
SUN1
R7
01 Obtain address of work station lockword. Release the real storage via IPW$RLW.
02 Calculate length of storage of WACB (2 buffers, the length of which is contained in the LUCB, plus 1 basic WACB).
03 Obtain address of LOGON/ INBOUND/LOGOFF WACB. Release virtual space via FREEVIS.
04 Obtain address of work station WACB for interrupt INBOUND FOR INBOUND. Release virtual storage via FREEVIS.
05 Calculate length of SUCB + LUCBs. Number of LUCBs is contained in SUCB. Release virtual storage via FREEVIS.
R3
R4
MAINLINE

CHART MK: IPW$$LH - SNA LOGON PROCESSOR 1 (30 PARTS)

Chart MK00: IPW$$LH - SNA/LOGON PROCESSOR 1, General Flow and Macro Calls



MACRO	Chart
Task Management	AA
IPW$DET IPW$WFC IPW$WFL	
Timer Service IPW$RDC	AG
Message Service IPW$WTO	AD
Storage Management IPW$RSW	AC
Resource Management	AB
IPW$RLR IPW$RSR	

Chart MK1

SNA LOGON PROCESSOR 1 (IPW$$LH) ORGANIZATION

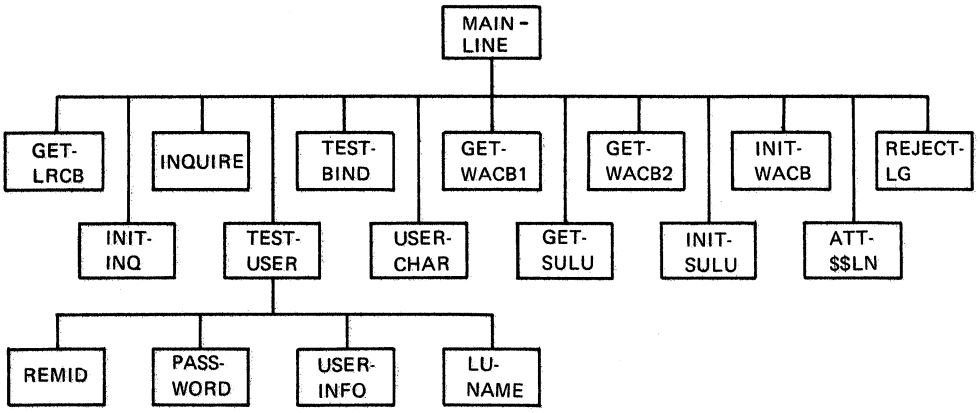
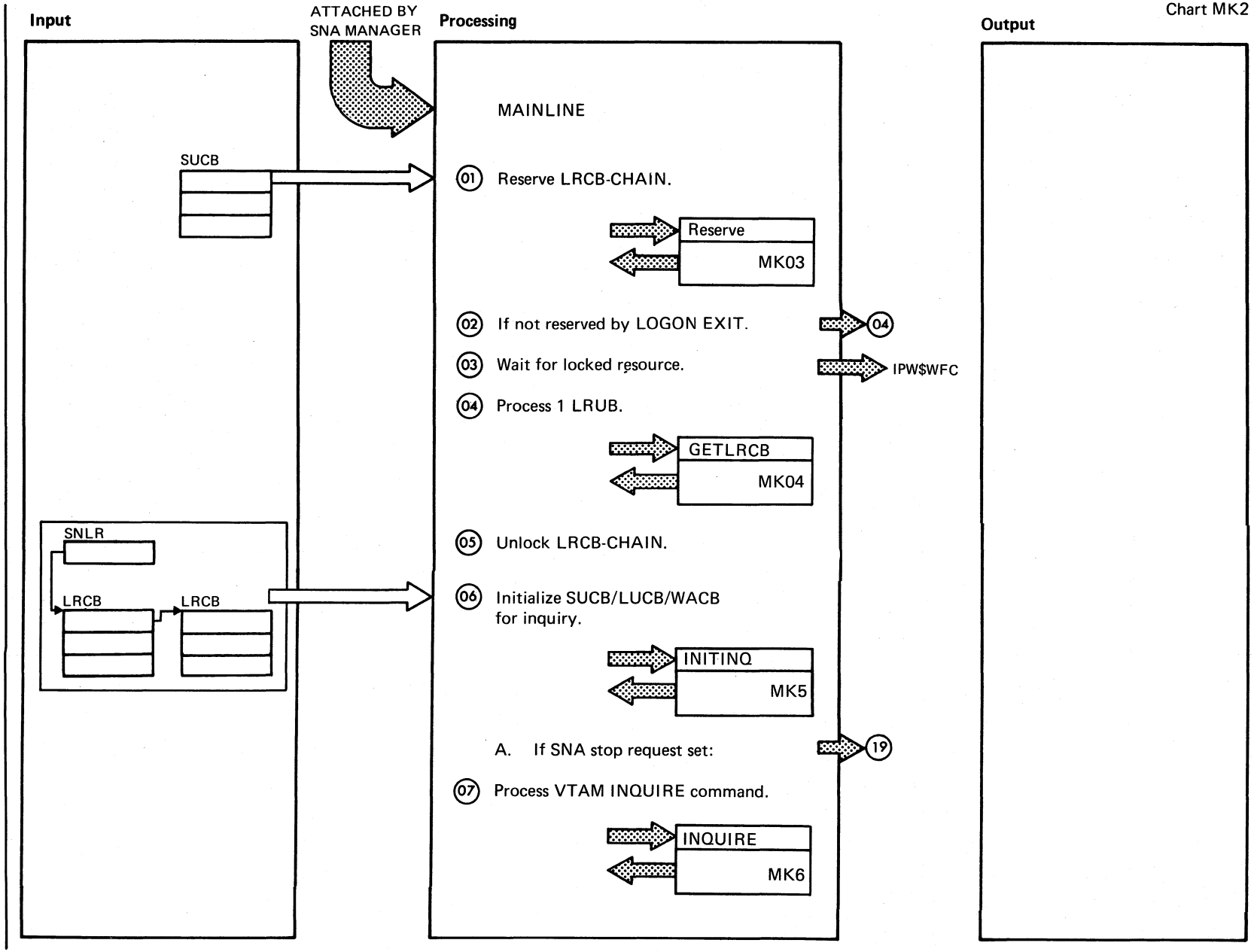
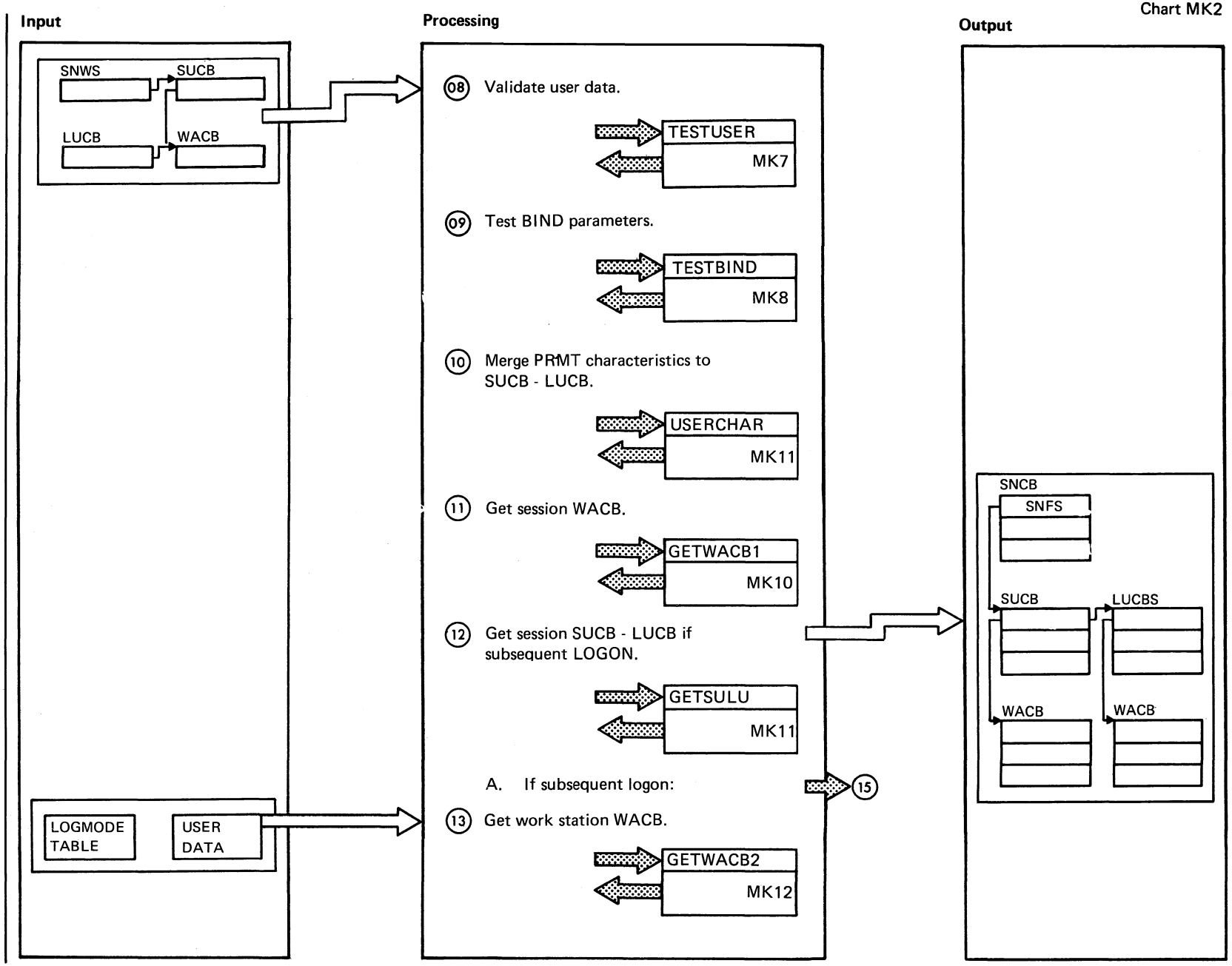
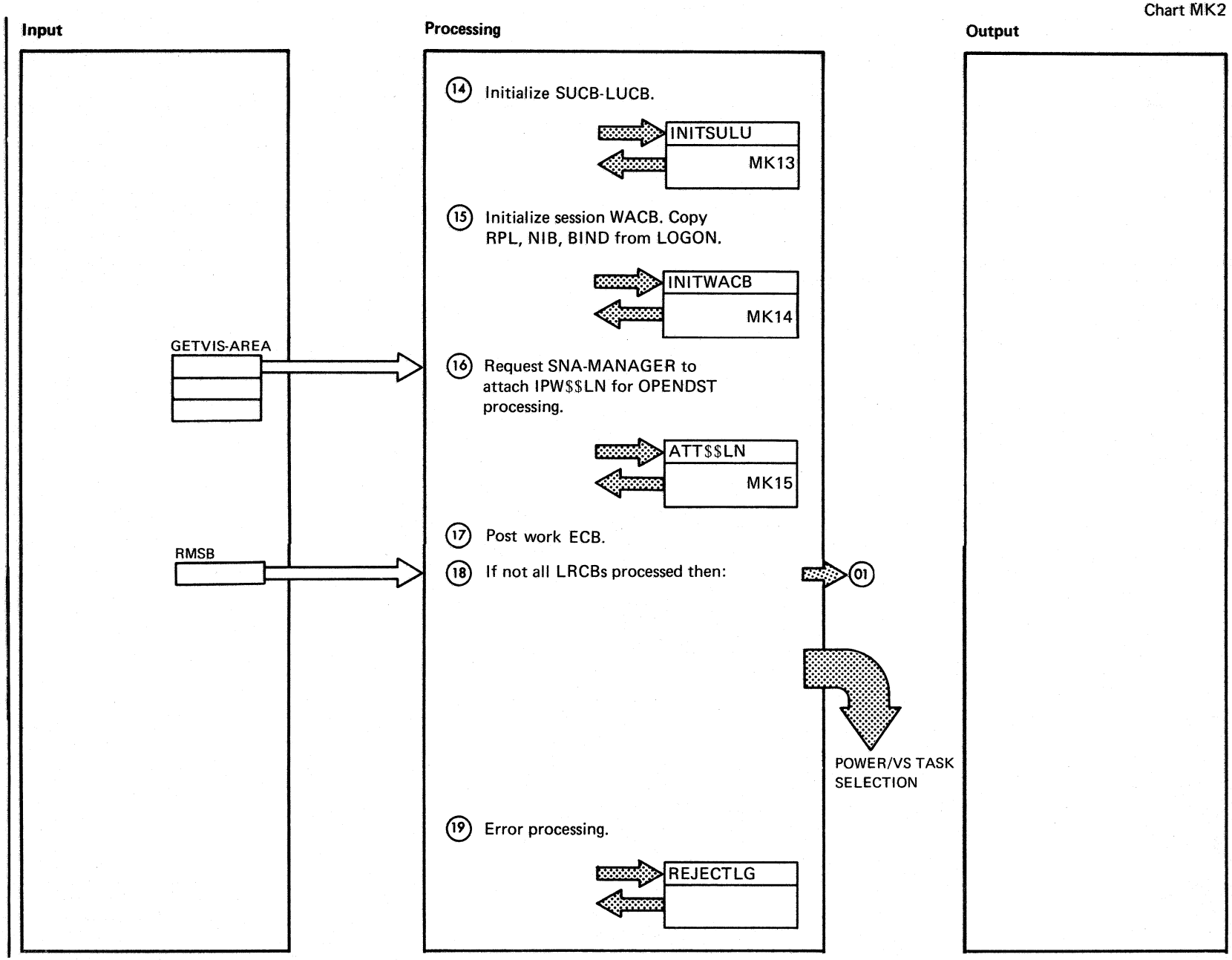


Chart MK1


Input
ATTACHED BY SNA MANAGER
Processing
Output
MAINLINE
SUCB
01 Reserve LRCB-CHAIN.
Reserve
MK03
02 If not reserved by LOGON EXIT.
04
03 Wait for locked resource.
IPW$WFC
04 Process 1 LRUB.
GETLRCB
MK04
SNLR
LRCB
LRCB
05 Unlock LRCB-CHAIN.
06 Initialize SUCB/LUCB/WACB for inquiry.
INITINQ
MK5
A. If SNA stop request set:
19
07 Process VTAM INQUIRE command.
INQUIRE
MK6


Input
Processing
Output
SNWS
SUCB
LUCB
WACB
LOGMODE TABLE
USER DATA
08 Validate user data.
TESTUSER
MK7
09 Test BIND parameters.
TESTBIND
MK8
10 Merge PRMT characteristics to SUCB - LUCB.
USERCHAR
MK11
11 Get session WACB.
GETWACB1
MK10
12 Get session SUCB - LUCB if subsequent LOGON.
GETSULU
MK11
A. If subsequent logon:
15
13 Get work station WACB.
GETWACB2
MK12
SNCB
SNFS
SUCB
LUCBS
WACB
WACB


Input
Processing
Output
14 Initialize SUCB-LUCB.
INITSULU
MK13
15 Initialize session WACB. Copy RPL, NIB, BIND from LOGON.
INITWACB
MK14
GETVIS-AREA
16 Request SNA-MANAGER to attach IPW$$LN for OPENDST processing.
ATT$$LN
MK15
17 Post work ECB.
RMSB
18 If not all LRCBs processed then:
01
POWER/VS TASK SELECTION
19 Error processing.
REJECTLG

Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(01) A lockword is set to inform the LOGON EXIT that IPW$$LH processes the LRCB-CHAIN			
(02) If LOGON EXIT already processes the LRCB-CHAIN itself, a wait in IPW$$LH is issued, until LOGON EXIT unlocks the CHAIN.			
(03) Etc.			
(04) Scan LRCBs for active LRUB, merge LUNAME to LOGON LUCB, release virtual space if 1 LRCB completely processed.			
(05) Unlock the LRCB-CHAIN.			
(06) Move RPL and NIB image to WACB. Initialize RPL and NIB.			
(07) Issue INQUIRE command to get the session parameters. Issue IPW$WFC to wait until VTAM completes the request, check the RPL for errors and deactivation.			
(08) Validate the user data for REMID, LUNAME defined in PRMT (OPT.), PASSWORD (OPT.) and USERINFO.			
(09) Test the BIND parameters, e.g. for alternate code, COMPACTION, RU-SIZE. Check if they match the POWER profile.			
(11) Get virtual space for one WACB. Length of buffers according to RUSIZE in BIND parameters. This WACB is used for LOGON, INBOUND and LOGOFF. One WACB is attached to each LUCB.			
(12) Get virtual space for the session SUCB - LUCB (S), according to SESSLIM-VALUE.			
(14) Initialize the SUCB-LUCBs.			
(15) The WACB obtained in step 11 is used for LOGON processing in IPW$$LN. The WACB of the session inquire LOGON NIB/RPL is merged to this WACB for OPNDST processing.			
(16) Attach IPW$$LN to establish the session. The session related control blocks SUCB-LUDBS-WACBS are set up. IPW$$LN is reentrant to process one or more OPENDST requests in parallel.			
(19) Storage is freed using the FREEVIS macro, messages for the central operator are queued. This routine is entered, when a GETVIS or a VTAM request fails, when erroneous user data or BIND parameters are detected or a SNA stop state is indicated. In all cases the logon request will be rejected via CLSDST.			

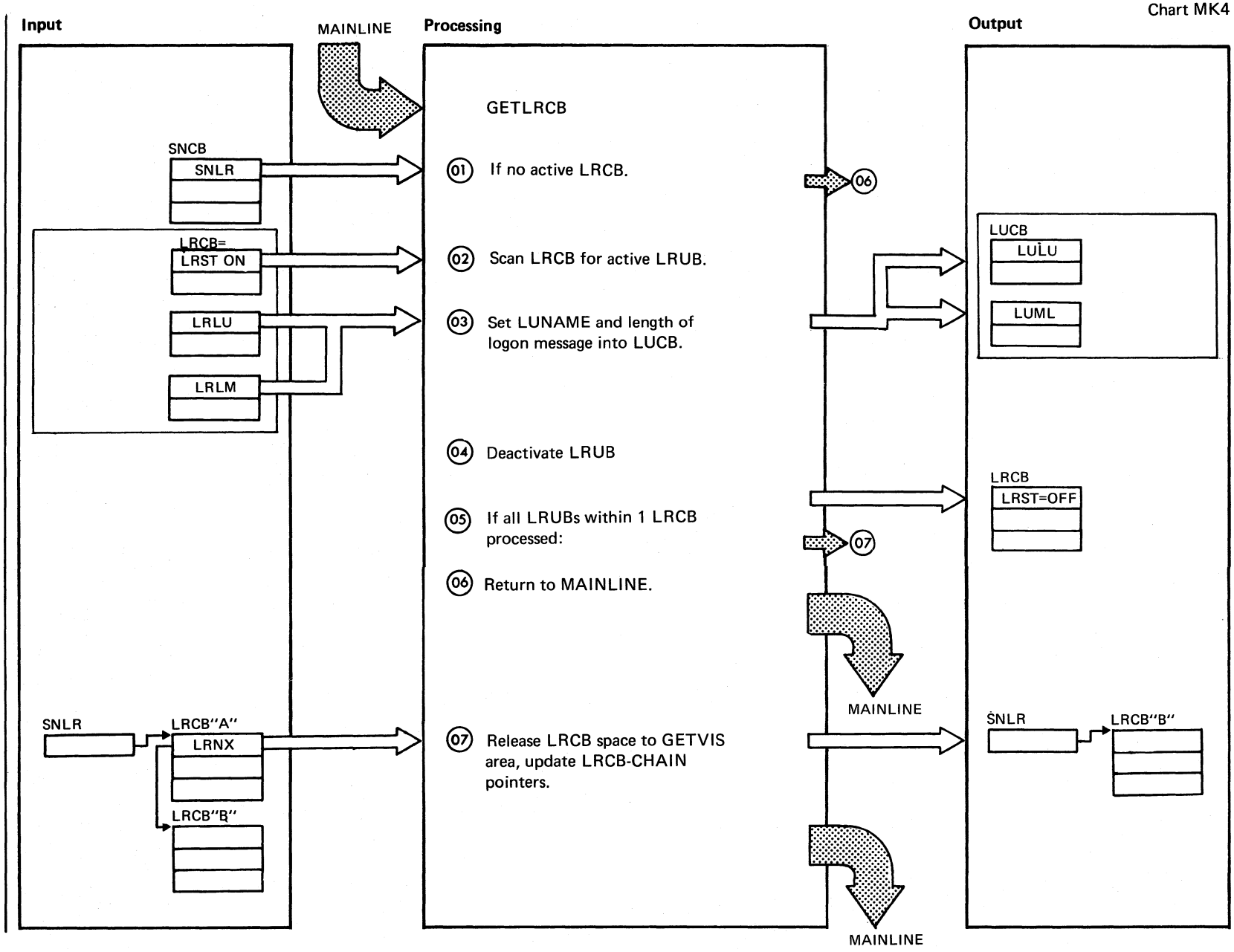
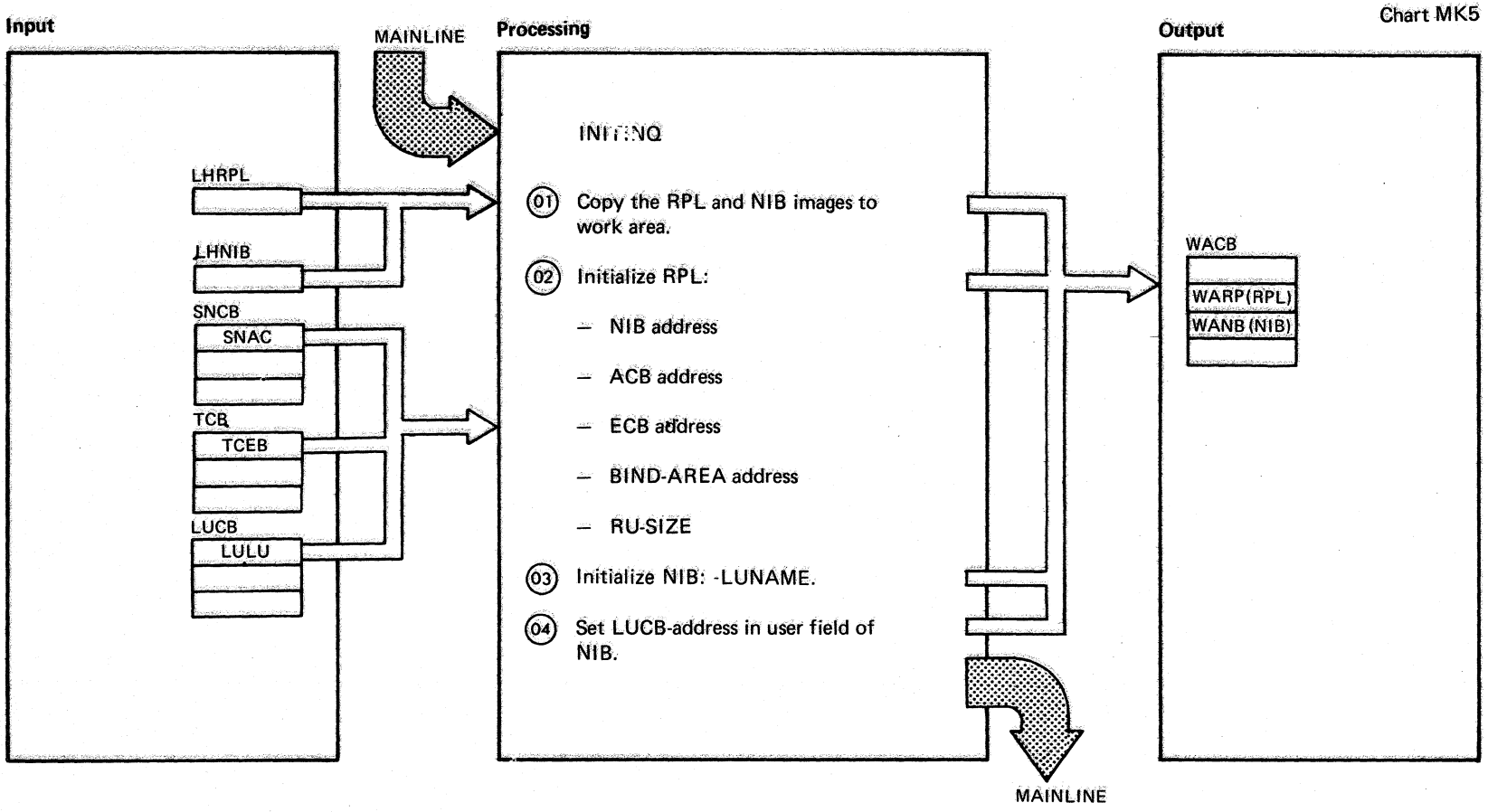


Chart MK4
Part 1 of 2

Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(01) If SNLR contains all zeros, no LRCB is chained. (02) Each LRUB within a LRCB contains a status byte (LRST) which is set to ' ON ' when a LRUB is filled with data by LOGON EXIT. (03) A LRUB entry contains the address of the luname which requests logon, and the length of userdata, entered with the logon request. The LUNAME, pointed to by LRLU is moved into the LUCB (LULU), and the message length is saved into LUCB (LUCD). (04) The LRUB space is marked deactive (RLST = OFF). LOGON EXIT can use this entry within the LRCB again. (05) If all LRUBs within a LRCB are processed, the LRCB can be released back to the GETVIS area. The LRCB-CHAIN is updated. (06) No LOGON requests are pending, and mainline detaches !PW$$LH. (07) See text of process No. 5.			

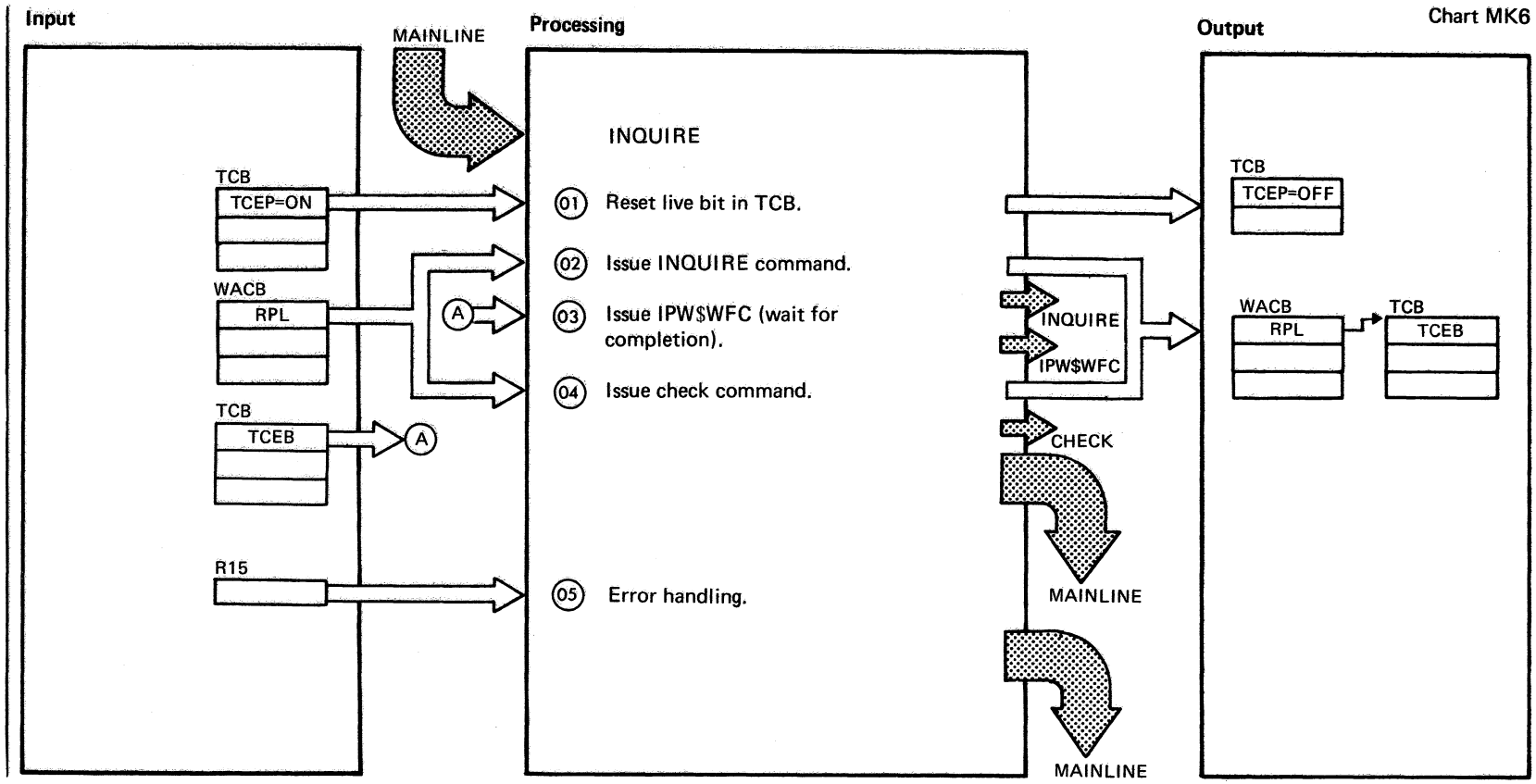


Extended Description

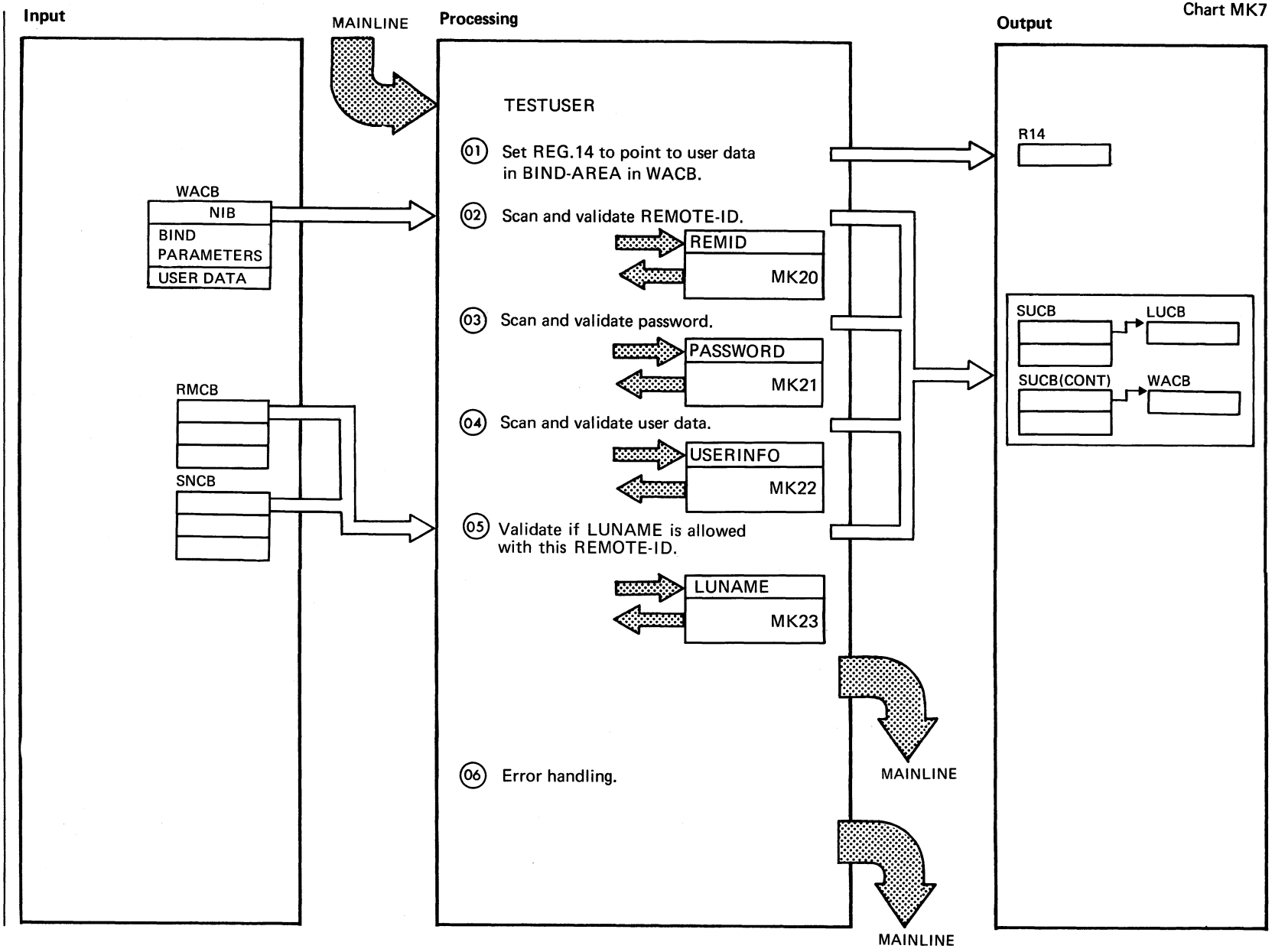
Include Segment	Call Subroutine/ Macro	Chart

(01) At assembly time a RPL and NIB image is build within IPW$$LH. These images are copied to the LOGON WACB's RPL and NIB area.

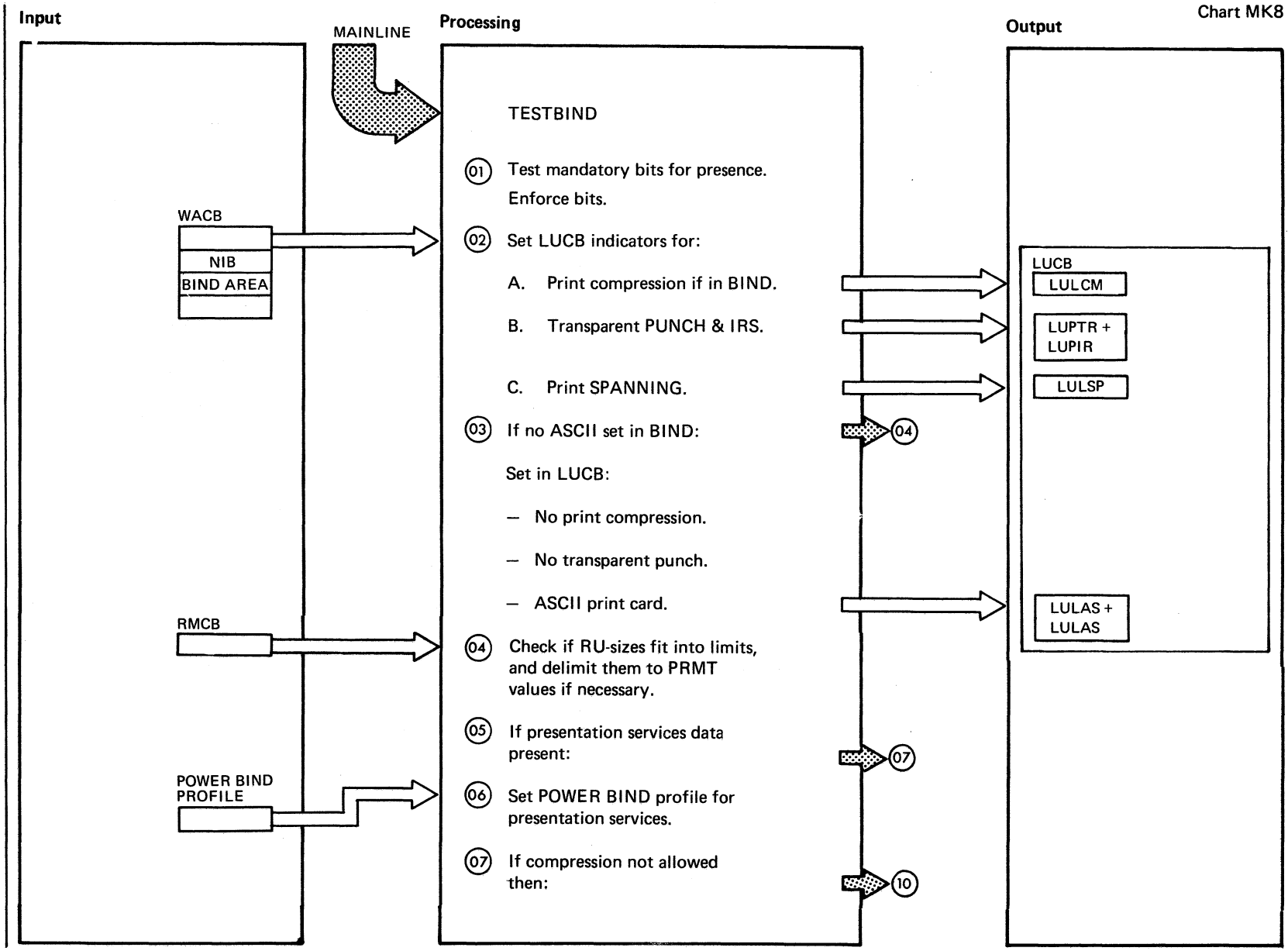
(03) The LUNAME is accessed in the GETLRCB routine and stored in the LUCB. Now the LUNAME is moved from the LUCB to the NIB.

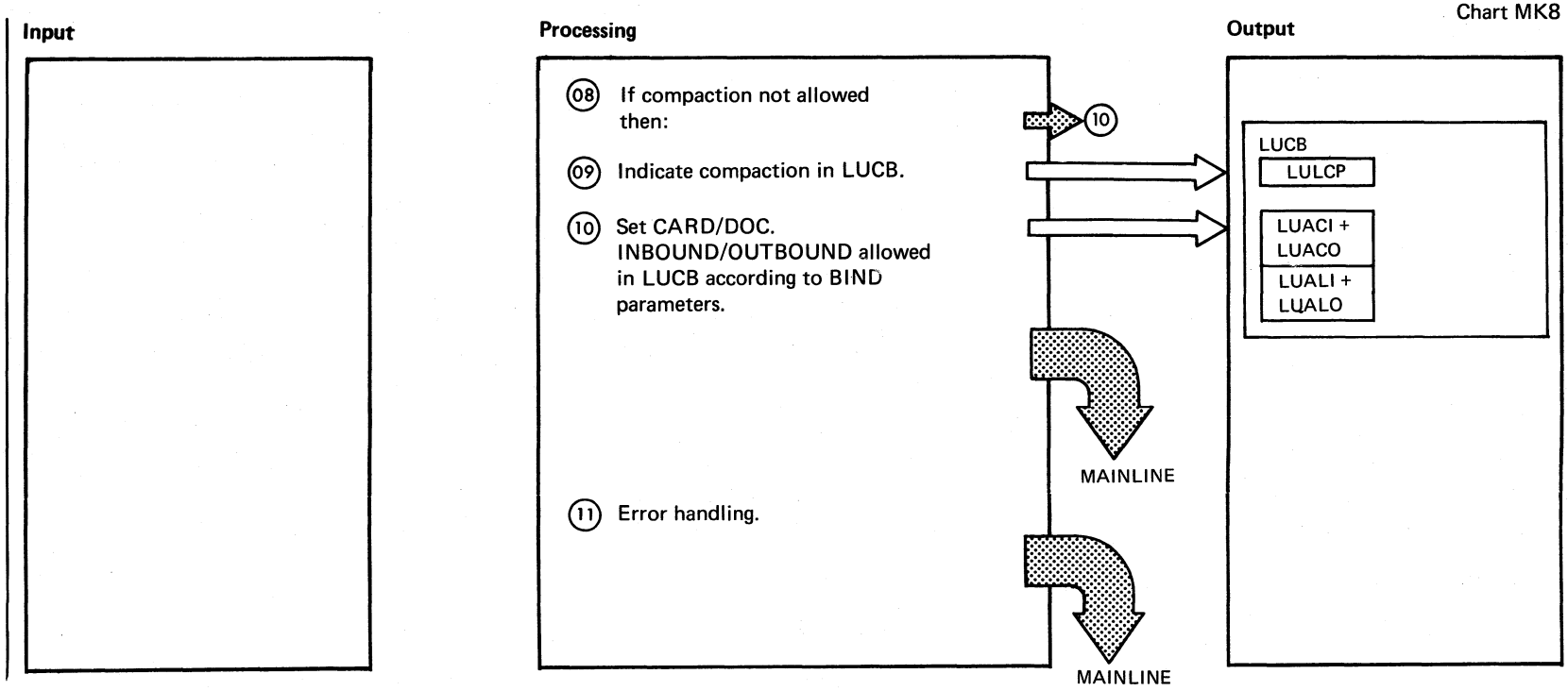

Input
MAINLINE
Processing
Output
Chart MK6
INQUIRE
TCB
TCEP=ON
WACB
RPL
TCB
TCEB
R15
A
A
01 Reset live bit in TCB.
02 Issue INQUIRE command.
03 Issue IPW$WFC (wait for completion).
04 Issue check command.
05 Error handling.
INQUIRE
IPW$WFC
CHECK
MAINLINE
MAINLINE
TCB
TCEP=OFF
WACB
RPL
TCB
TCEB

Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(01) The INQUIRE command is processed asynchroneously by VTAM. Upon final completion, a ECB, pointed to by the RPL, and located in the TCB, is posted, the traffic bit has to be reset before issuing the request. (02) Upon initial completion of the inquire command (acceptance by VTAM), the return-code in REG. 15 is checked. (03) A wait for completion is issued on the ECB for VTAM's final completion of the inquire command. Upon final completion, the ECB is posted, and IPW$$LH gets control. The execution return-code in REG. 15 ist checked. (04) A VTAM check macro is issued to deactivate the asynchroneously processed RPL. (05) Present message "1V07I ERROR ON INQUIRE RTNCD, FDBK2=xx" followed by message "1V06I UNABLE TO LOGON LU" to central operator. Mainline will call error processing.			

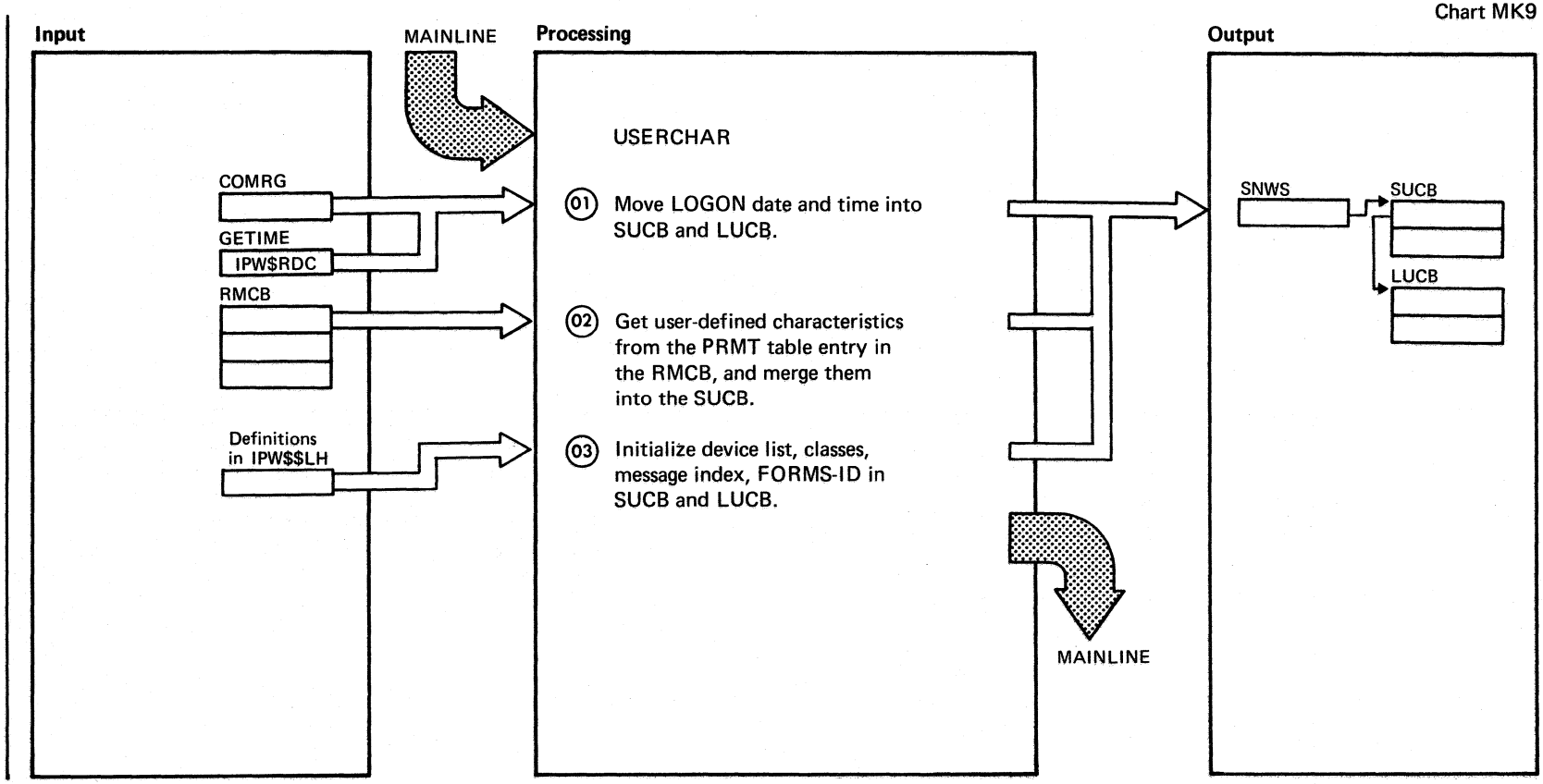

Input
MAINLINE
Processing
Output
Chart MK7
TESTUSER
R14
WACB
NIB
BIND PARAMETERS
USER DATA
01 Set REG.14 to point to user data in BIND-AREA in WACB.
02 Scan and validate REMOTE-ID.
REMID
MK20
03 Scan and validate password.
PASSWORD
MK21
SUCB
LUCB
SUCB(CONT)
WACB
RMCB
SNCB
04 Scan and validate user data.
USERINFO
MK22
05 Validate if LUNAME is allowed with this REMOTE-ID.
LUNAME
MK23
06 Error handling.
MAINLINE
MAINLINE

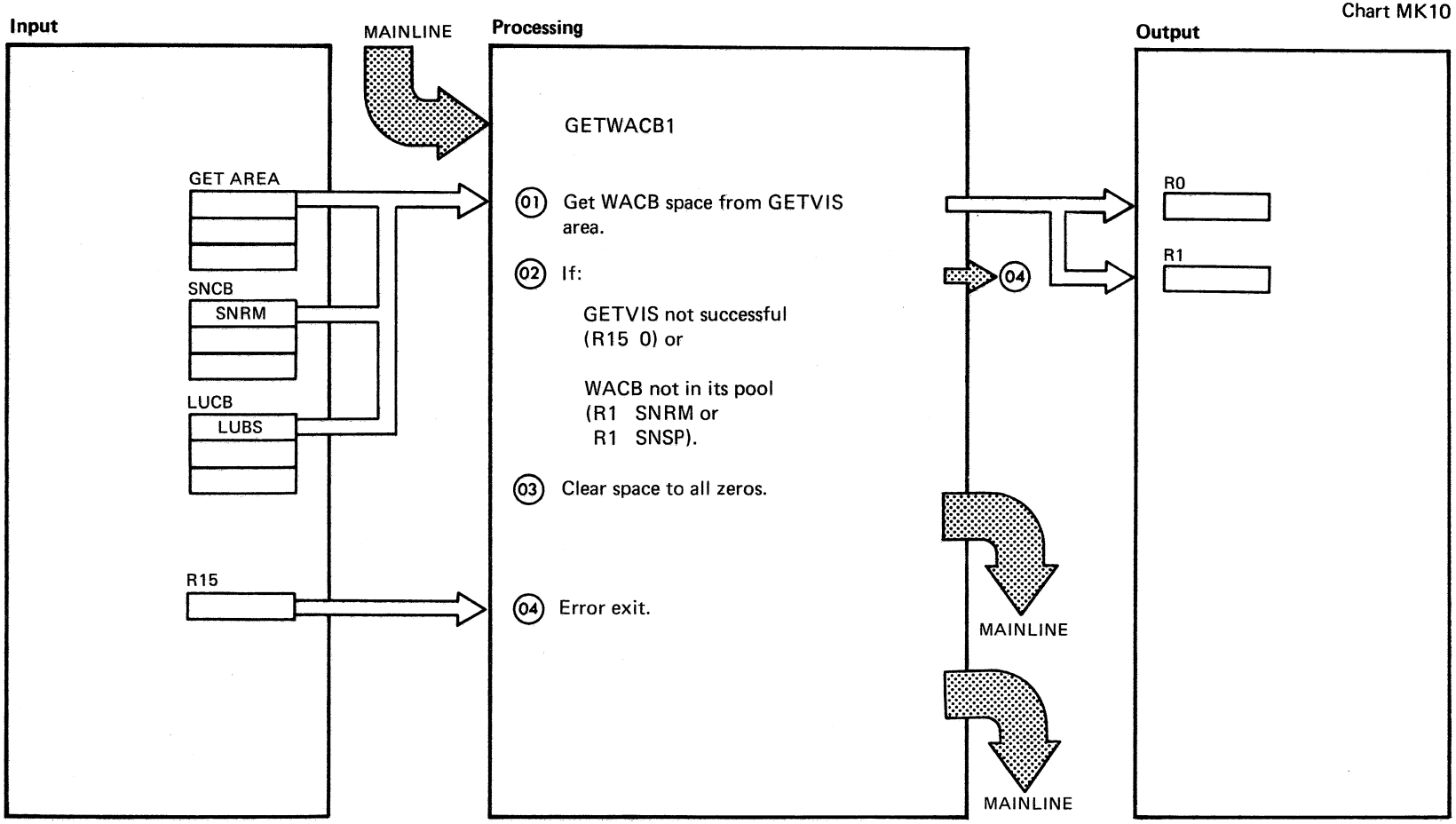
Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(02) Routine name = REMID (03) Routine name = PASSWORD. (04) Routine name = USERINFO. (05) Routine name = LUNAME. (06) Message "1V26I INVALID REMOTE-ID, PASSWORD OR LUNAME", followed by message" 1V06I UNABLE TO LOGON LU" is presented to the central operator. Reason code for message 1V26I: RC = 30 Invalid REMOTE-ID. RC = 31 PASSWORD error. RC = 32 LUNAME error.			


Input
MAINLINE
Processing
Output
TESTBIND
01 Test mandatory bits for presence.
Enforce bits.
WACB
NIB
BIND AREA
02 Set LUCB indicators for:
A. Print compression if in BIND.
B. Transparent PUNCH & IRS.
C. Print SPANNING.
LUCB
LULCM
LUPTR +
LUPIR
LULSP
03 If no ASCII set in BIND:
04
Set in LUCB:
– No print compression.
– No transparent punch.
– ASCII print card.
LULAS +
LULAS
RMCB
04 Check if RU-sizes fit into limits, and delimit them to PRMT values if necessary.
05 If presentation services data present:
07
POWER BIND
PROFILE
06 Set POWER BIND profile for presentation services.
07 If compression not allowed then:
10

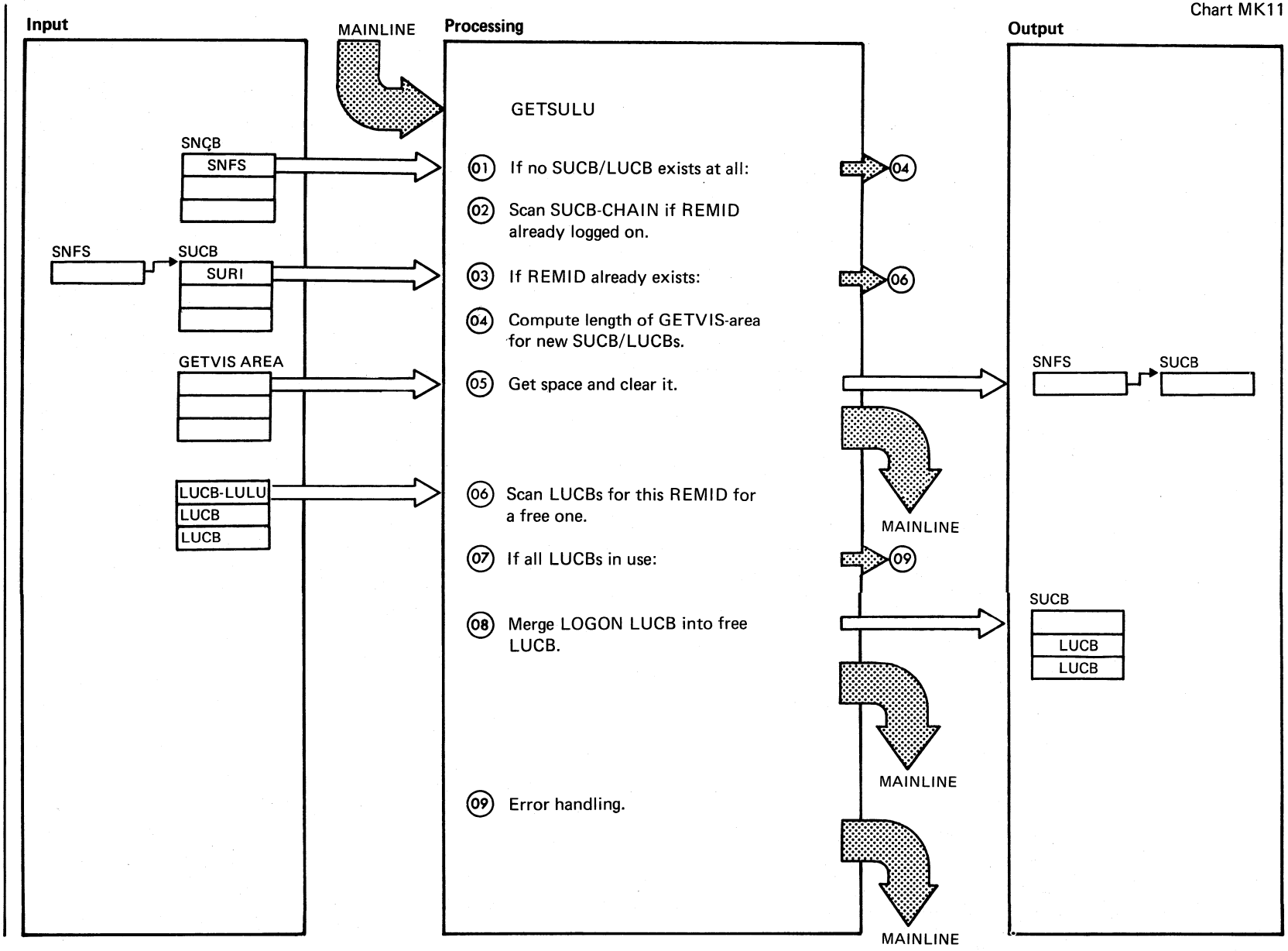

Input
Processing
Output
08 If compaction not allowed then:
09 Indicate compaction in LUCB.
10 Set CARD/DOC. INBOUND/OUTBOUND allowed in LUCB according to BIND parameters.
11 Error handling.
10
LUCB
LULCP
LUACI + LUACO
LUALI + LUALO
MAINLINE
MAINLINE

Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(01) If mandatory bits are not set, BIND data is invalid, and LOGON REQUEST is rejected. Enforced bits are set according to the specifications. SLU will test them with the OPNDST command. Variable bits cause indicators to be set in the LUCB according to the specifications. If presentation service data is ommited in the BIND data, following POWER profile is set, starting with byte 14 (BINLUP) of the BIND data: Byte / Hex. Value 14 01 15 10 16 00 17 00 18 F1 19 00 20 C0 21 00 22 00 23 01 24 00 25 40 26 00 RU size for PLU and SLU is, if not X'85' enforced to X'85', which means RU size of 256 bytes (8* 2** 5=256). (02) Present messages "1V34I BIND DATA", "1V08I ERROR ON BIND" "1V06I UNABLE TO LOGON LU" to central operator.			

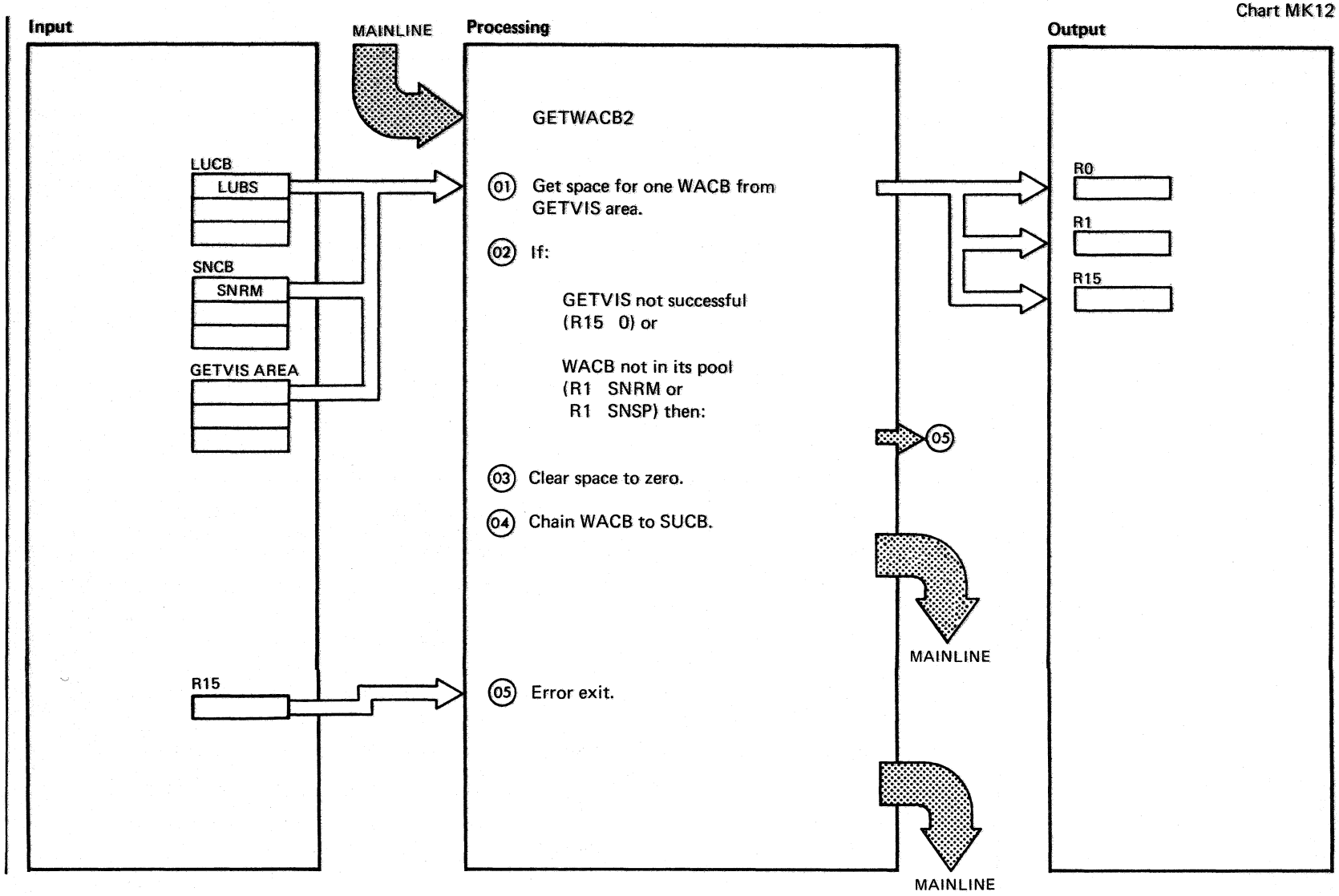

Input
MAINLINE
Processing
Output
USERCHAR
COMRG
GETIME
IPW$RDC
RMCB
Definitions
in IPW$$LH
01 Move LOGON date and time into SUCB and LUCB.
02 Get user-defined characteristics from the PRMT table entry in the RMCB, and merge them into the SUCB.
03 Initialize device list, classes, message index, FORMS-ID in SUCB and LUCB.
SNWS
SUCB
LUCB
MAINLINE



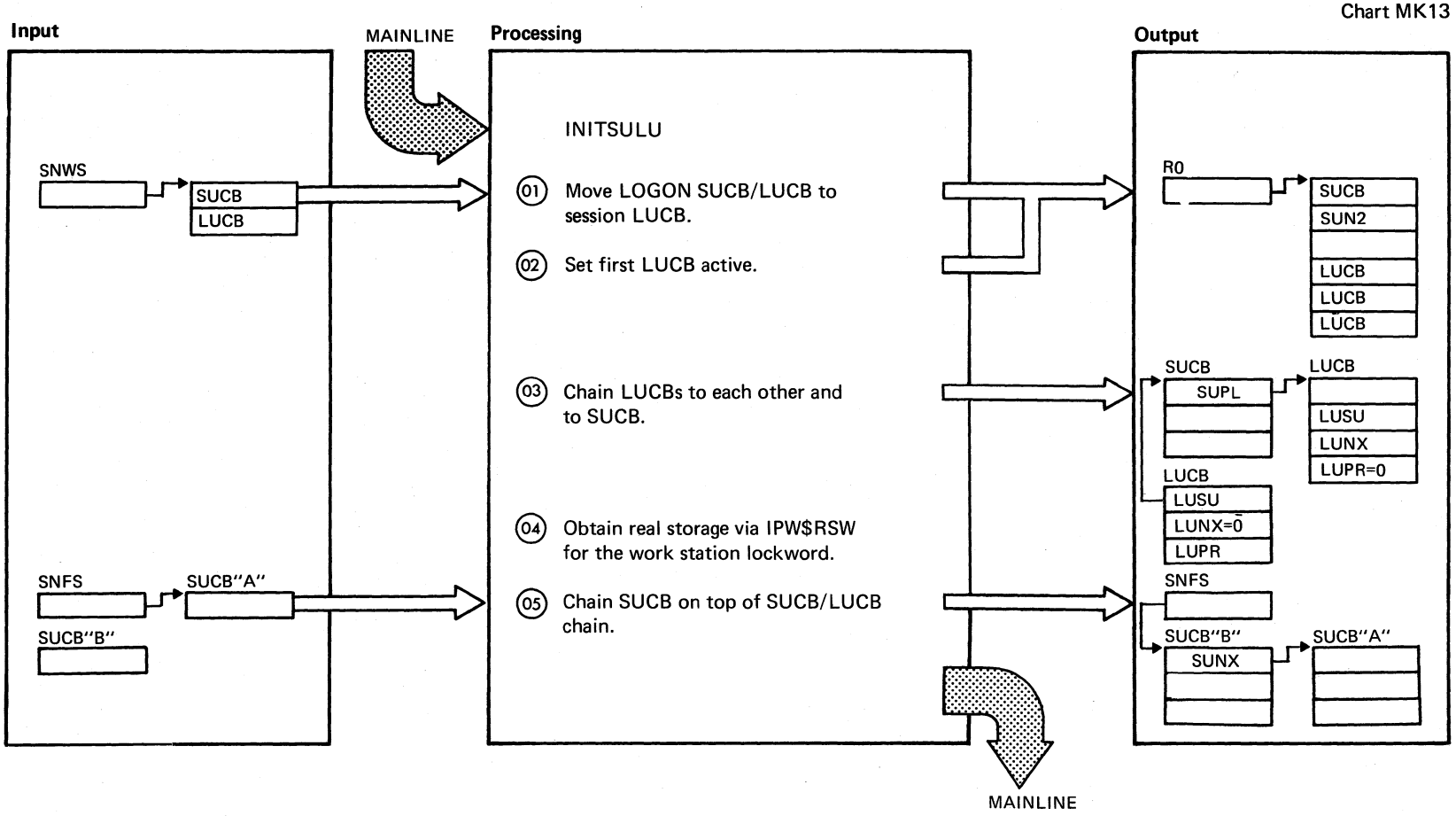
Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(01) This WACB is attached to the LUCB and is used for LOGON, INBOUND and LOGOFF processing. The space is obtained from the WACB pool, and the buffersize is according to the requested RU-size, specified in the PRMT and in the BIND parameters. (03) MESSAGE "1V06I UNABLE TO LOGON LU" is presented to the central operator.			


Input
MAINLINE
Processing
Output
GETSULU
SNCB
SNFS
SNFS
SUCB
SURI
GETVIS AREA
LUCB-LULU
LUCB
LUCB
01 If no SUCB/LUCB exists at all:
02 Scan SUCB-CHAIN if REMID already logged on.
03 If REMID already exists:
04 Compute length of GETVIS-area for new SUCB/LUCBs.
05 Get space and clear it.
06 Scan LUCBs for this REMID for a free one.
07 If all LUCBs in use:
08 Merge LOGON LUCB into free LUCB.
09 Error handling.
04
06
09
MAINLINE
MAINLINE
MAINLINE
SNFS
SUCB
SUCB
LUCB
LUCB

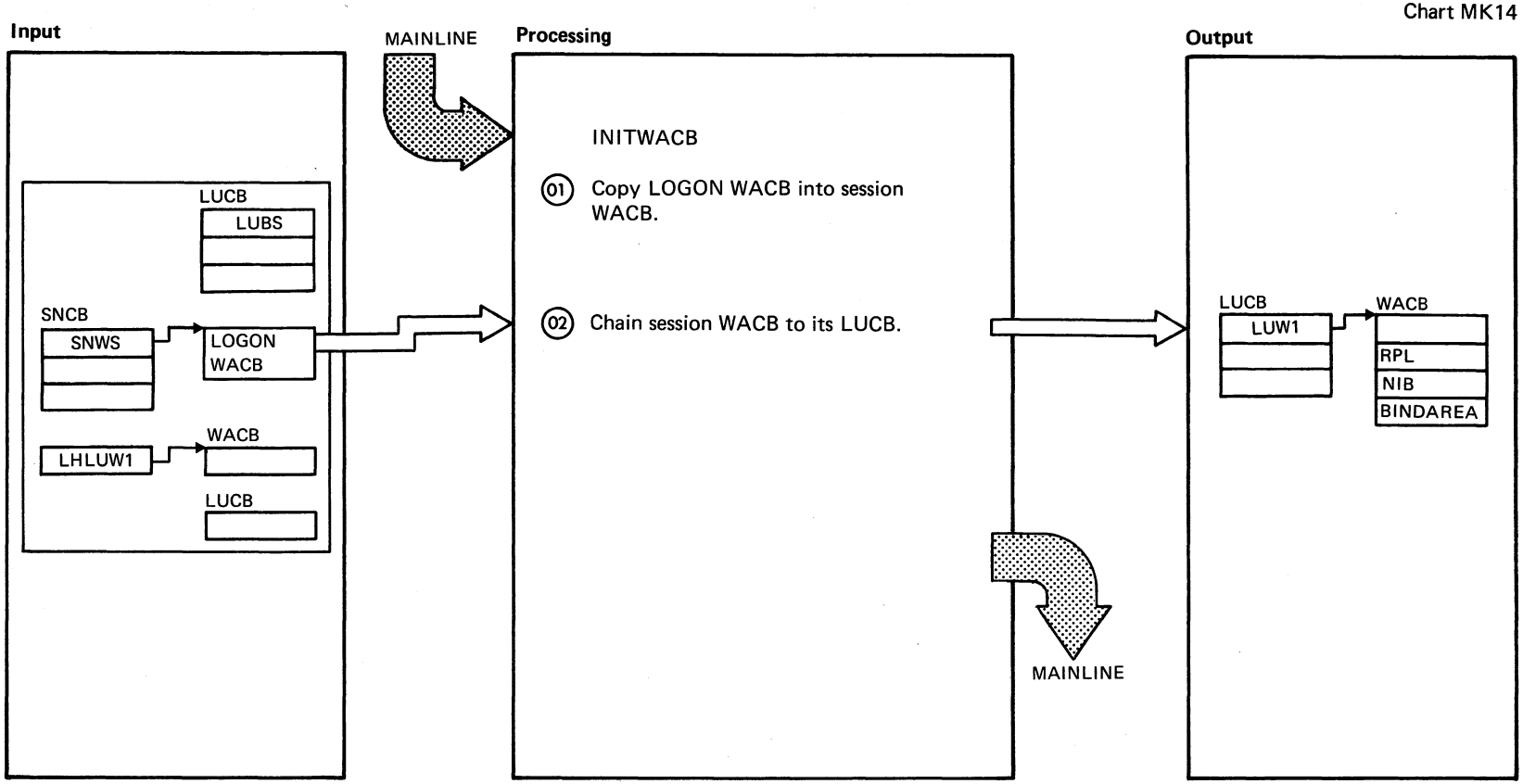
Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(01) Upon entry, lock SUCB/LUCB-CHAIN by issuing a IPW$RSR macro. SNFS in the SNCB contains the address of the first active SUCB. If SNFS is zero, no SUCB exists at all.			
(02) Check SUCB/LUCB-CHAIN, if REMOTE - ID of LOGON SUCB already logged on. If not, a complete SUCB - LUCB - WACB structure has to be obtained from GETVIS area, and has to be initialized. If already logged on, only the LOGON LUCB has to be merged to a free LUCB of the REMOTE-ID.			
(04) The length of the SUCB/LUCB space is computed according to following facts: Length of one SUCB, plus length of one or more LUCBs. The number of LUCBs is defined with the sesslim-parameter in in the PRMT macro, and set into the RMCB.			
(05) Get the SUCB/LUCB space from the GETVIS area, and clear the space to hex zero.			
(06) Scan the LUCB chain for this REMID to find a free one. A free LUCB is indicated, when LUF1 = OFF.			
(07) If all LUCBs for this REMID are occupied, the LOGON request has to be rejected.			
(08) Unlock SUCB/LUCB - CHAIN with IPW$RLR macro.			
(09) Free the previously obtained space for the LOGON/INBOUND/ LOGOFF WACB, using the FREEVIS - macro. Present message "1V06I UNABLE TO LOGON LU" to central operator.			


Input
MAINLINE
Processing
Output
GETWACB2
LUCB
LUBS
SNCB
SNRM
GETVIS AREA
R15
01 Get space for one WACB from GETVIS area.
02 If:
GETVIS not successful (R15 0) or
WACB not in its pool (R1 SNRM or R1 SNSP) then:
05
03 Clear space to zero.
04 Chain WACB to SUCB.
05 Error exit.
MAINLINE
MAINLINE
R0
R1
R15

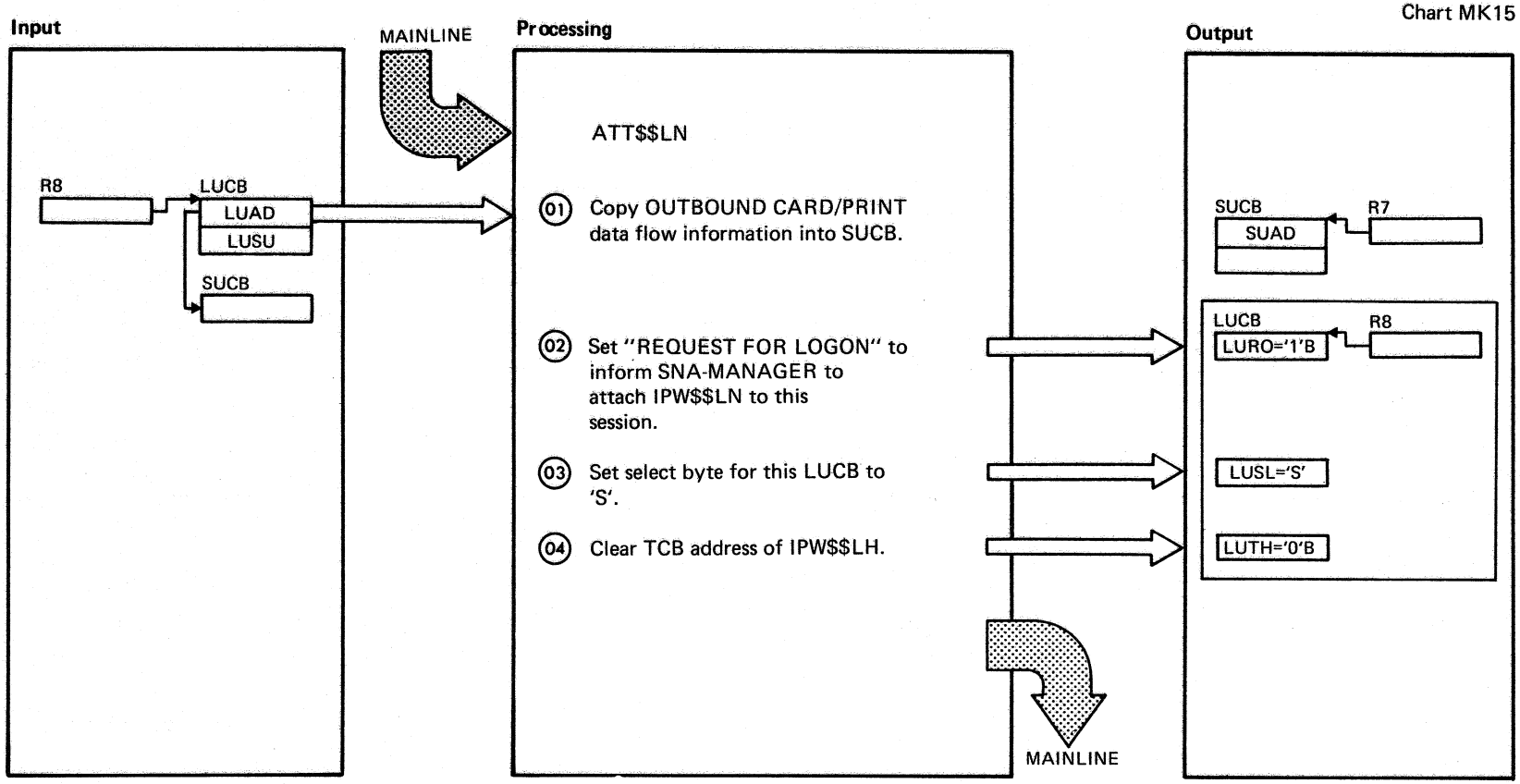
Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(01) This WACB is attached to the SUCB, and can be attached to any LUCB within this workstation. It is then marked to be in use, and can be used by the session to process inbound requests. (05) Free the previously obtained virtual space for LOGON/INBOUND/LOGOFF WACB and SUCB/LUCB using the FREEVIS macro. Present message "1V06I UNABLE TO LOGON LU" to central operator.			



Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(01) The length for the move is the length of 1 SUCB plus 1 LUCB. (03) Each LUCB has a pointer to its SUCB, and each LUCB has a pointer to the next, and a pointer to the previous LUCB. For the first LUCB, the previous pointer is zero, for the last LUCB, the next pointer is zero. (05) The new SUCB is chained on top of the SUCB/LUCB CHAIN, using the compare and swap instruction.			

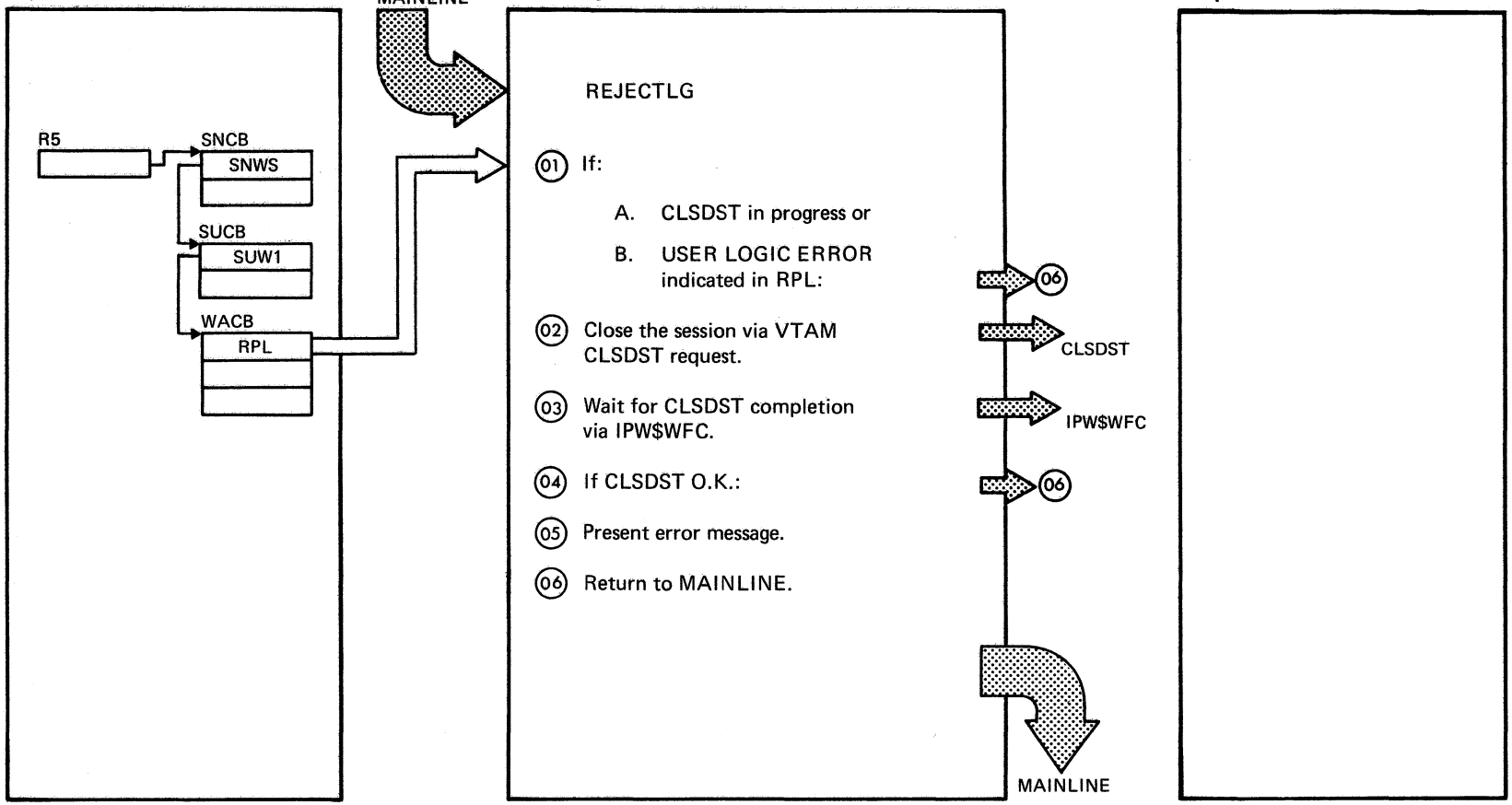


Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(01) The LOGON WACB contains RPL, NIB and the BIND parameters. This WACB is copied into the session WACB for opendst processing in IPW$$LN.			

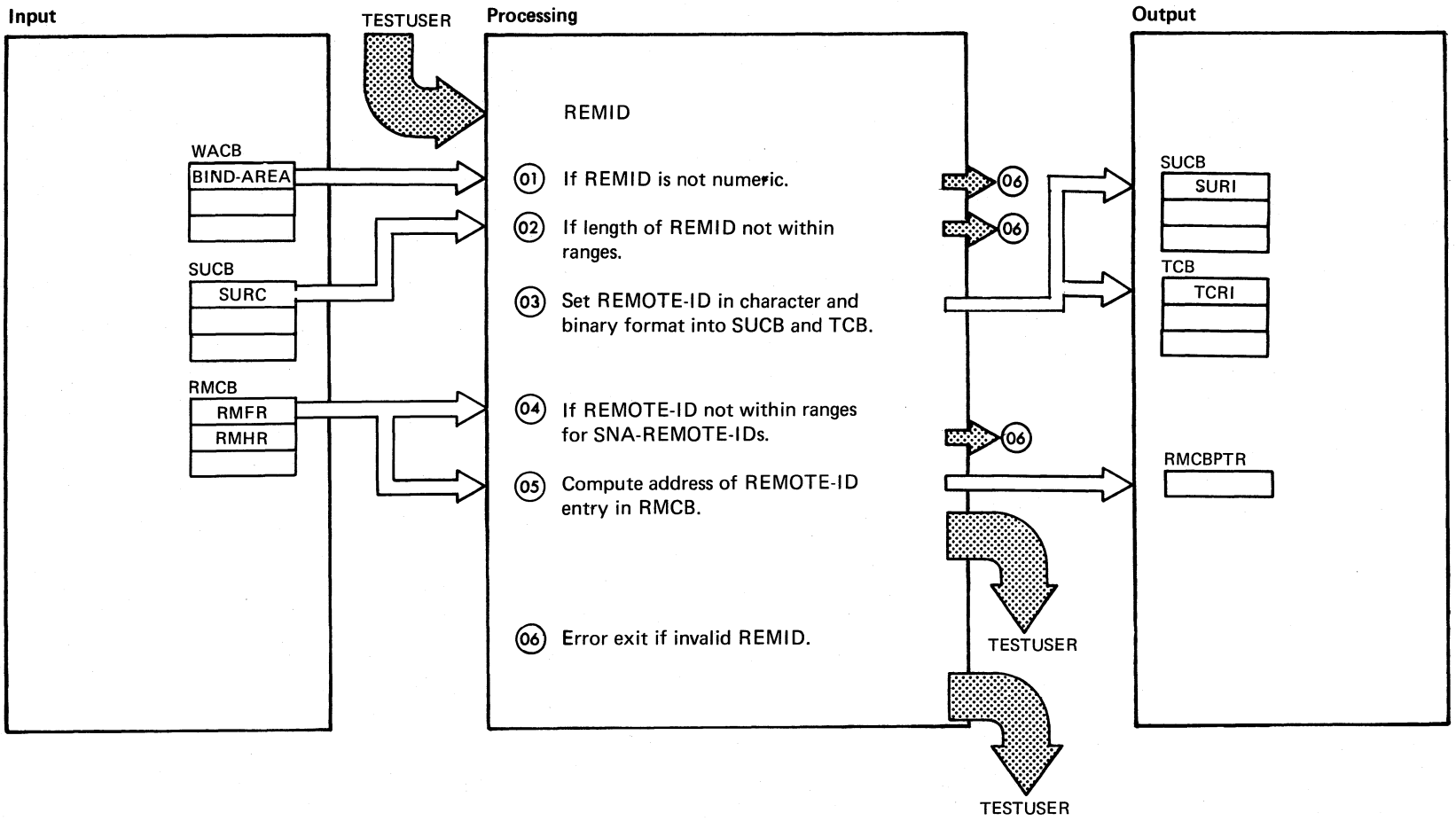


Extended Description

	Include Segment	Call Subroutine/ Macro	Chart
(01) The SNCB contains a summary of outbound CARD/PRINT dataflow, which is indicated in the bind data. This summary is used to check in the SNA - MANAGER if there is at least one session BOUND which allows card or print outbound data flow for a workstation with a MLU concept. (03) SNA MANAGER's LOOK FOR WORK" routine scans all LUCBs for the select indicator 'S' in LUSL. This major scan argument indicates that an action for this LUCB has to be started. Further analysis of the action bytes in the LUCB will cause the appropriate task to be attached. If LURO is set, SNA MANAGER will attach LOGON PROCESSOR2 IPW$$LN to process stage2 of the LOGON process.			



Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(01) RPLRTNCD | X '10' | X '14' | RPLFDB2 | X '0A" | X '12' | In both cases do not issue a CLSDST request to VTAM. (05) Present MSG "1V07I ERROR ON CLSDST", to central operator.			



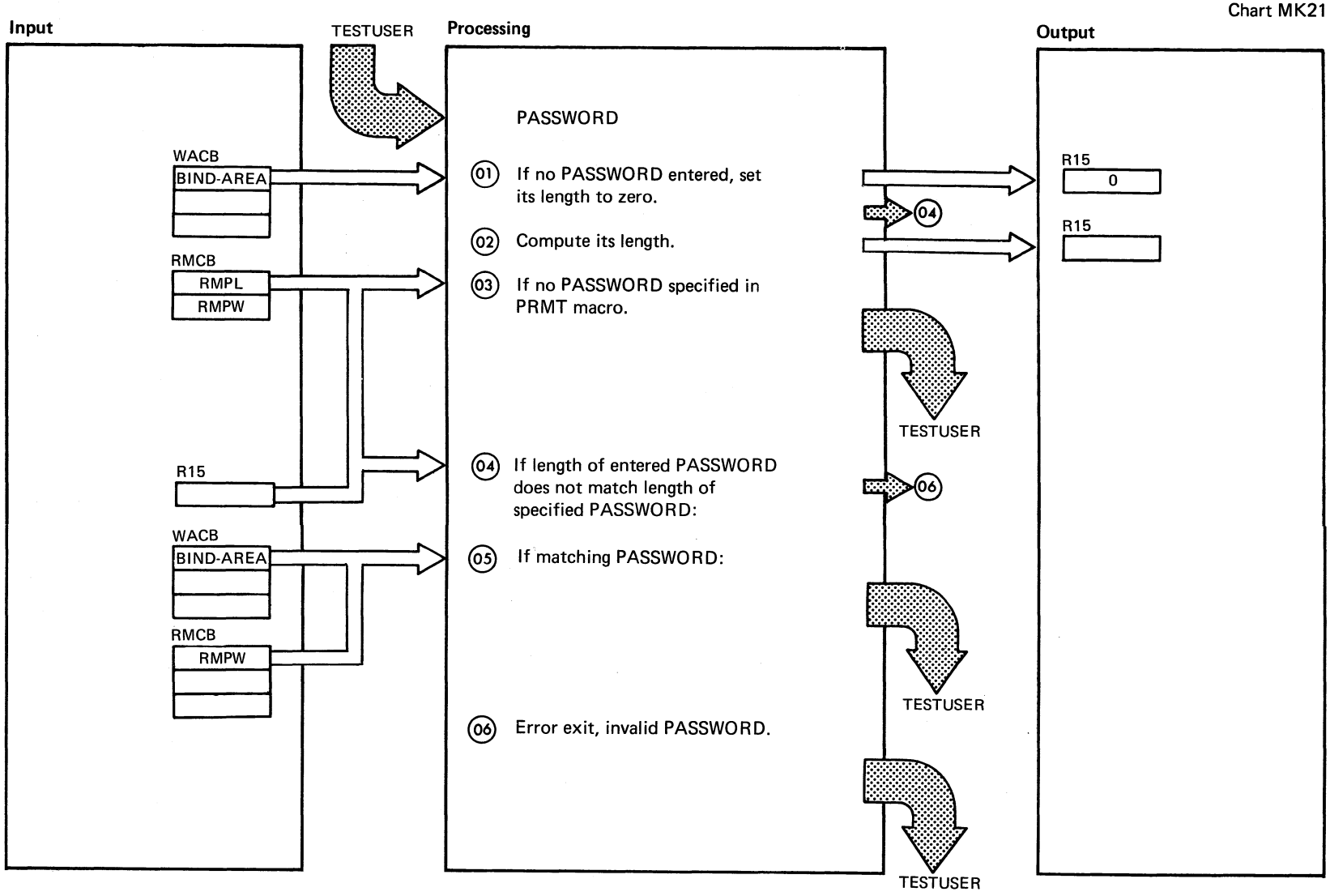
Extended Description

Include Segment	Call Subroutine/ Macro	Chart

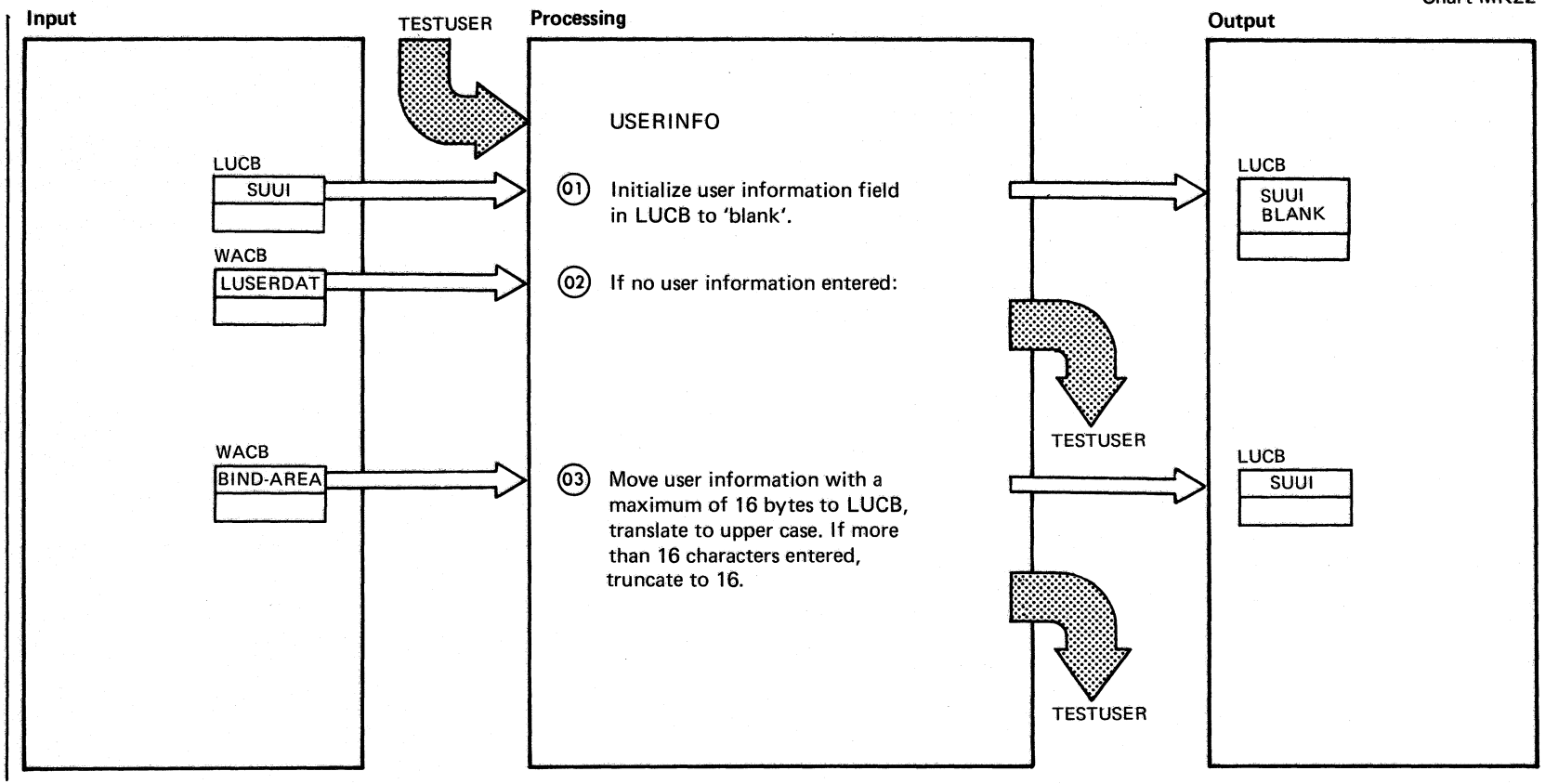
(05) This entry contains all information specified in the PRMT macro, and is used for further checking of password and LUNAMES, and to set up the SUCB/LUCB with the user characteristics.

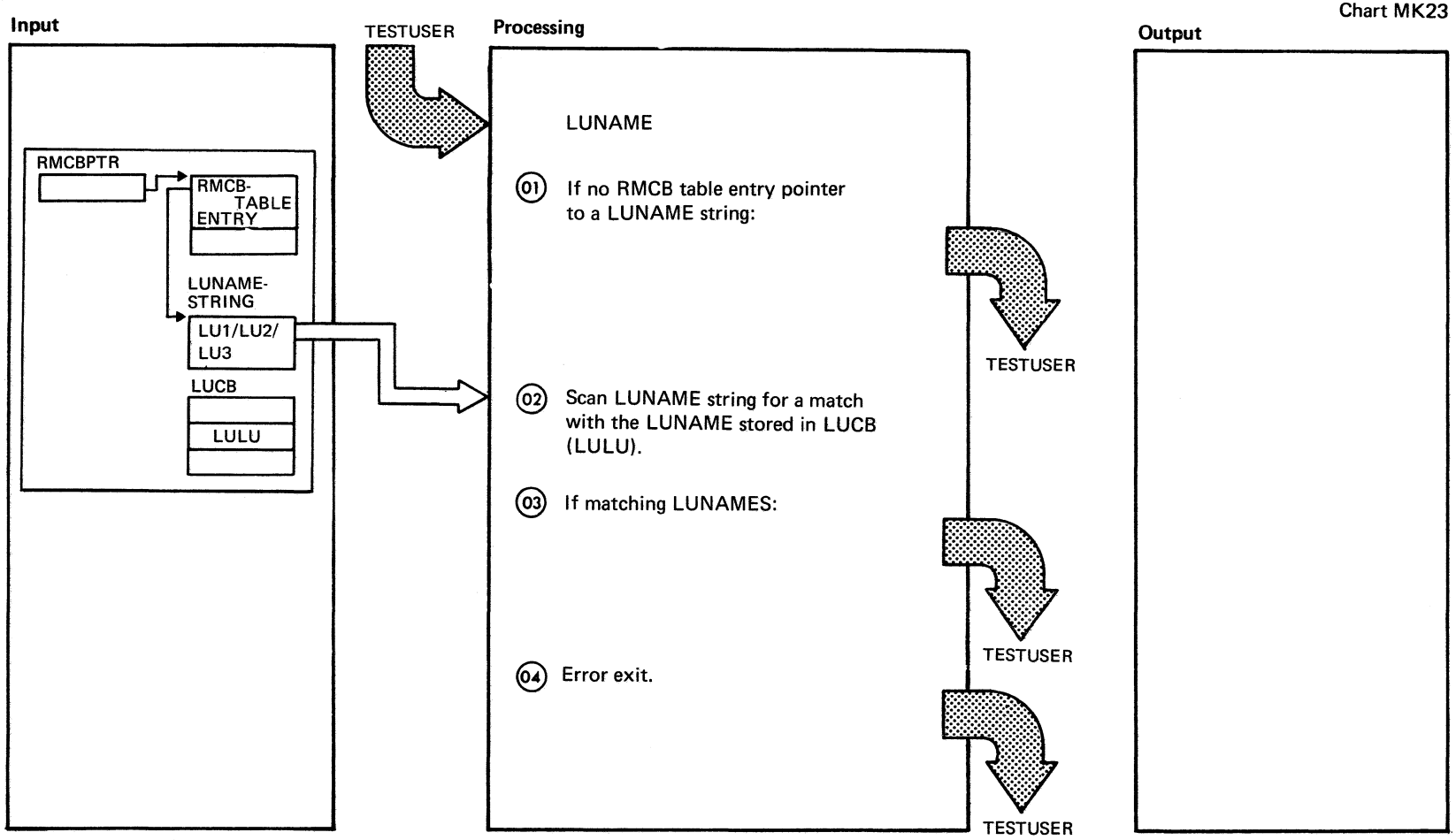
(06) Testuser error messages:

1V26I RC 30, 1V00I



Extended Description	Include Segment	Call Subroutine/ Macro	Chart
05 Error message in TESTUSER: 1V26I RC = 31, 1v06I			


Input
TESTUSER
Processing
Output
USERINFO
LUCB
SUUI
WACB
LUSERDAT
WACB
BIND-AREA
01 Initialize user information field in LUCB to 'blank'.
02 If no user information entered:
03 Move user information with a maximum of 16 bytes to LUCB, translate to upper case. If more than 16 characters entered, truncate to 16.
TESTUSER
TESTUSER
LUCB
SUUI
BLANK
LUCB
SUUI



Extended Description

	Include Segment	Call Subroutine/ Macro	Chart
01 No LUNAME checking required for this REMOTE-ID. 04 Present error message in TESTUSER. 1V26I RC = 32, 1V06I			

CHART ML: IPW$$OC - SNA OUTBOUND COMPACTION MANAGER (15 PARTS)

Chart ML00: IPW$$OC - SNA Outbound Compaction Manager, General Flow and Macro Calls

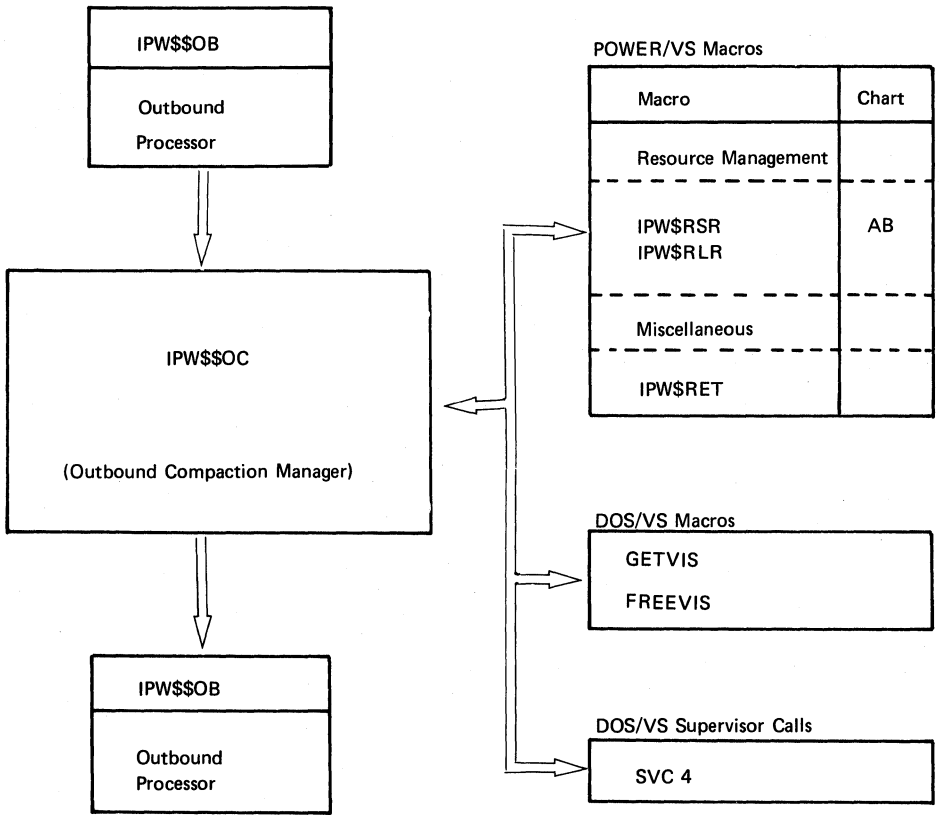
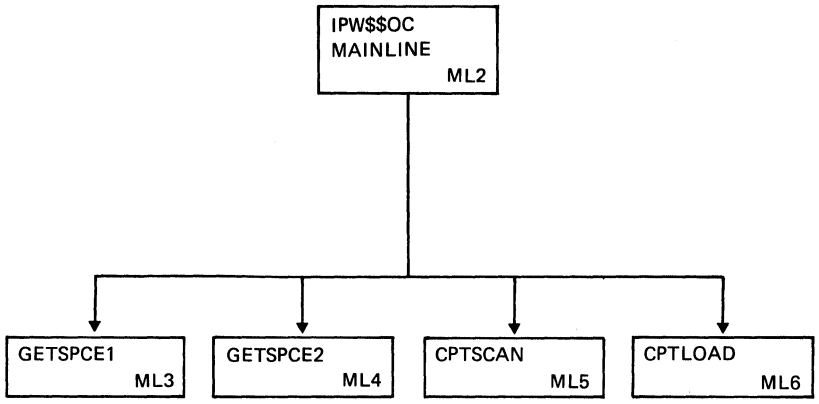


Chart ML1: IPW$$OC - SNA Compaction Processor, Organization

COMPACTION PROCESSOR (IPW$$OC) ORGANIZATION



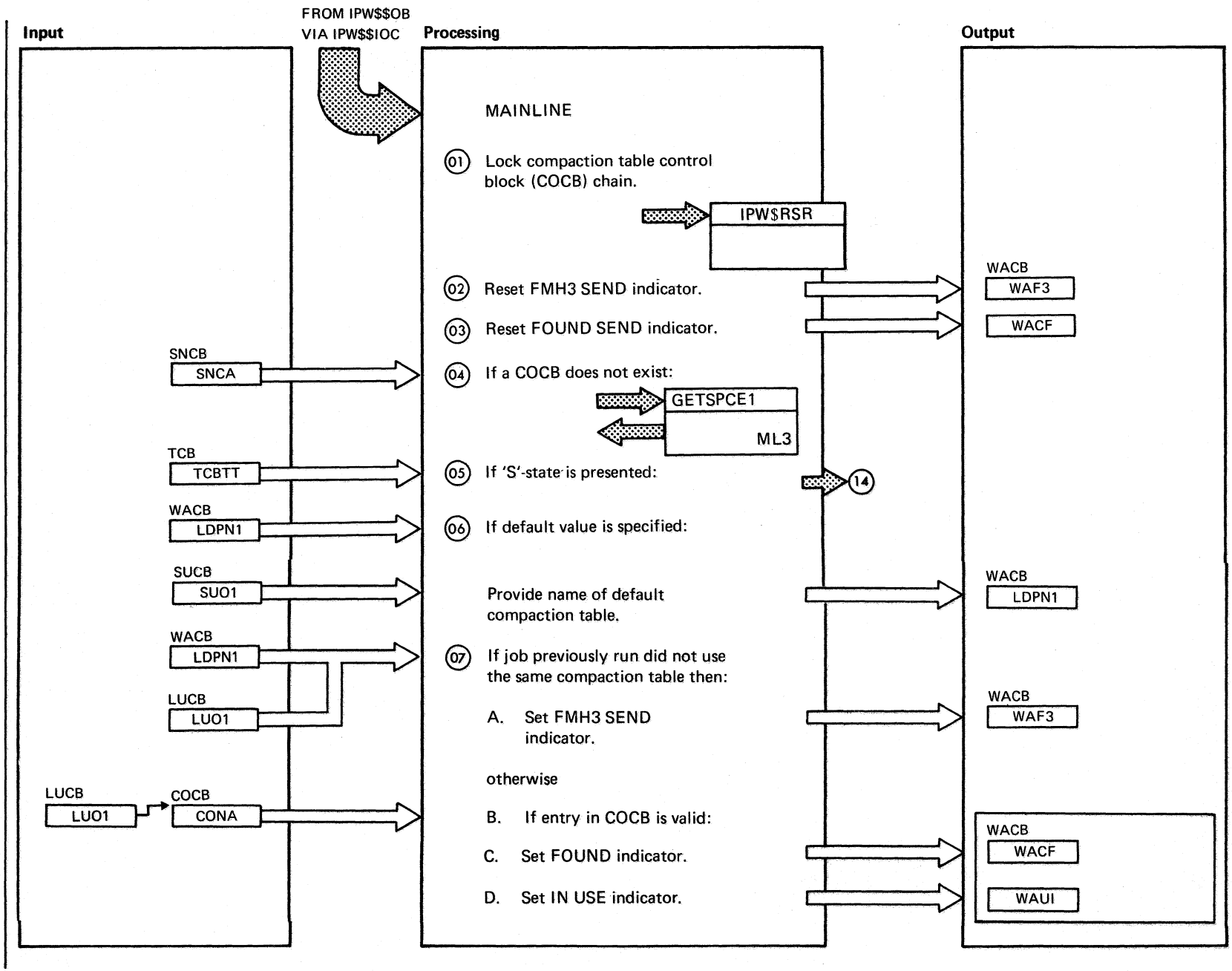
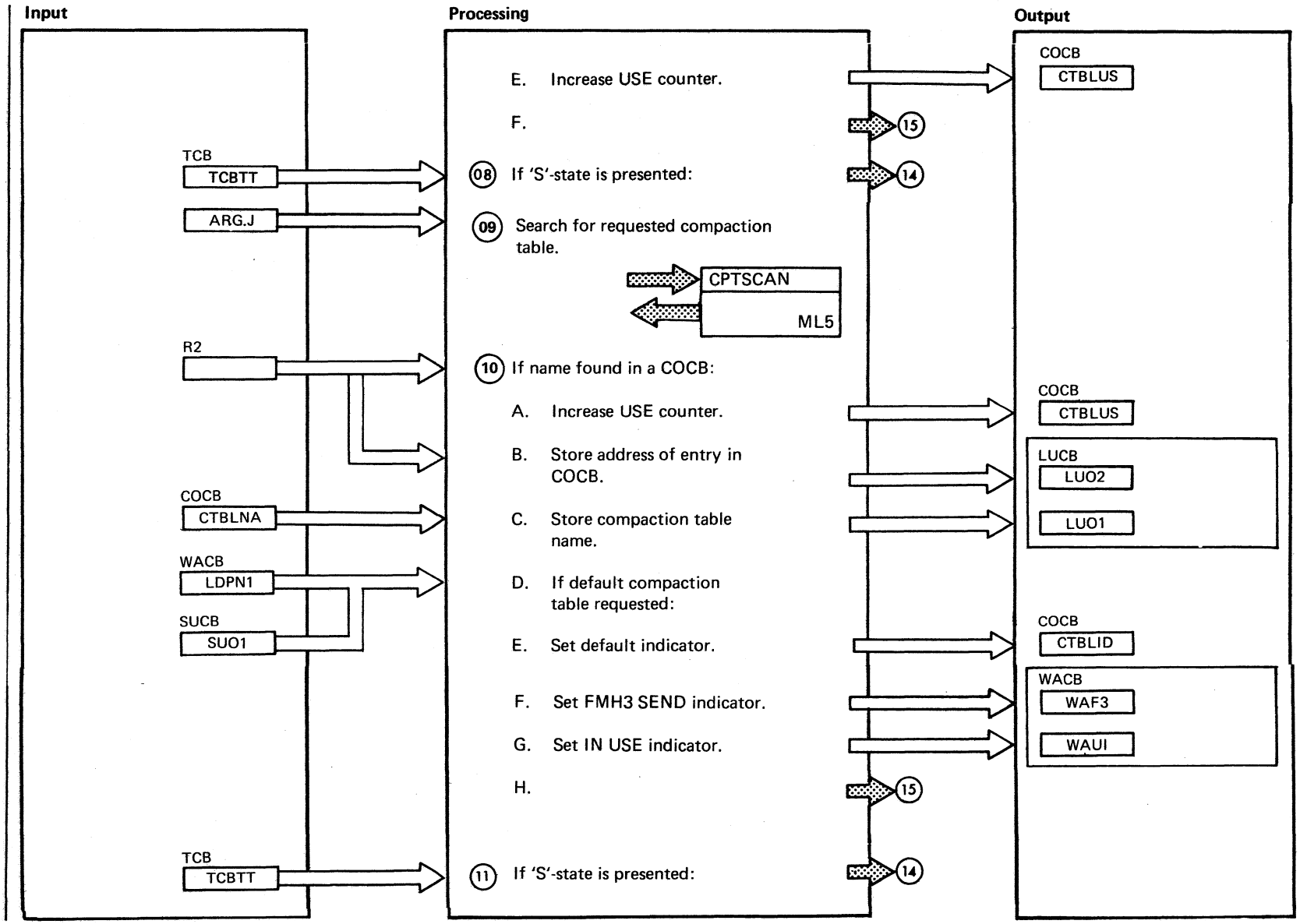
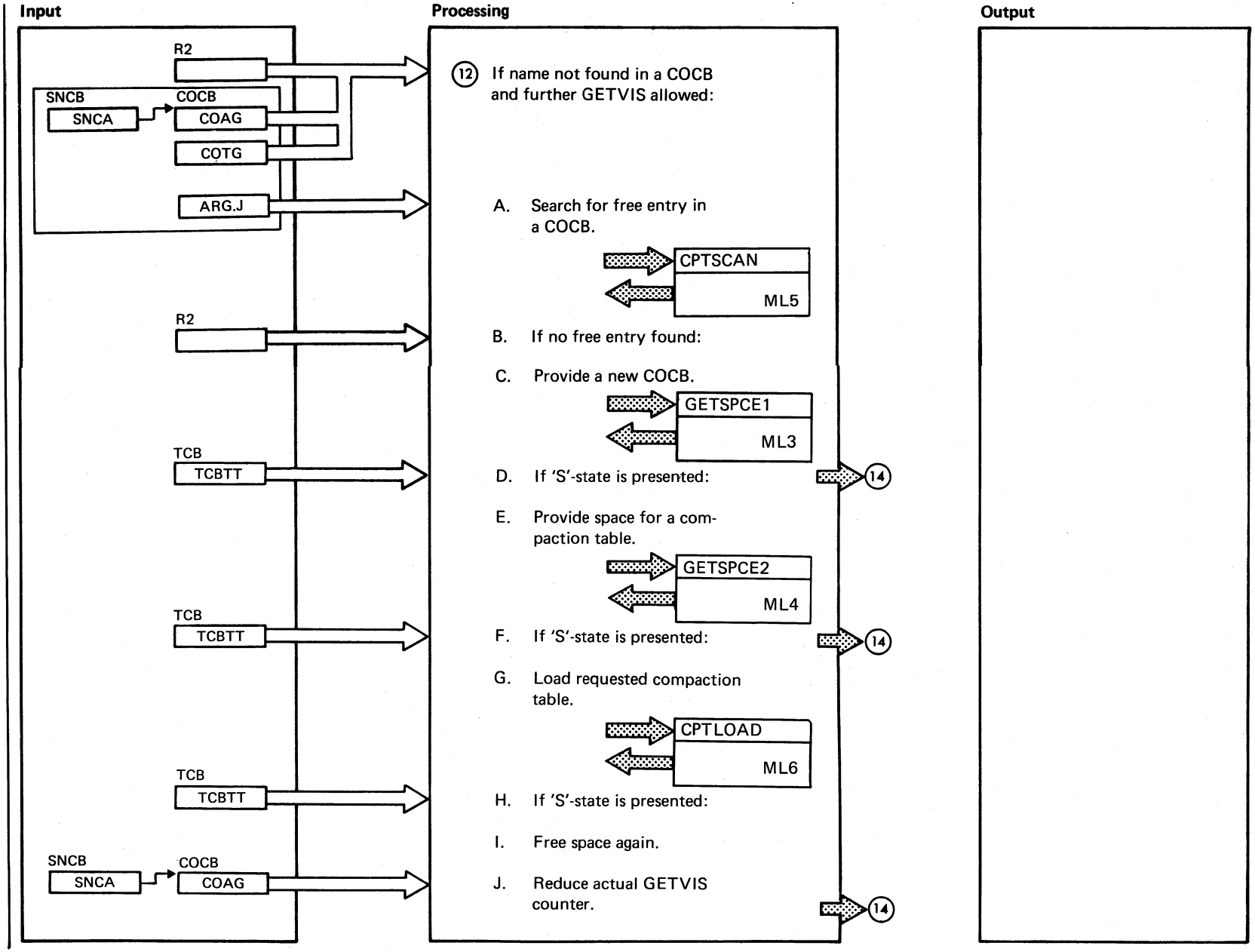
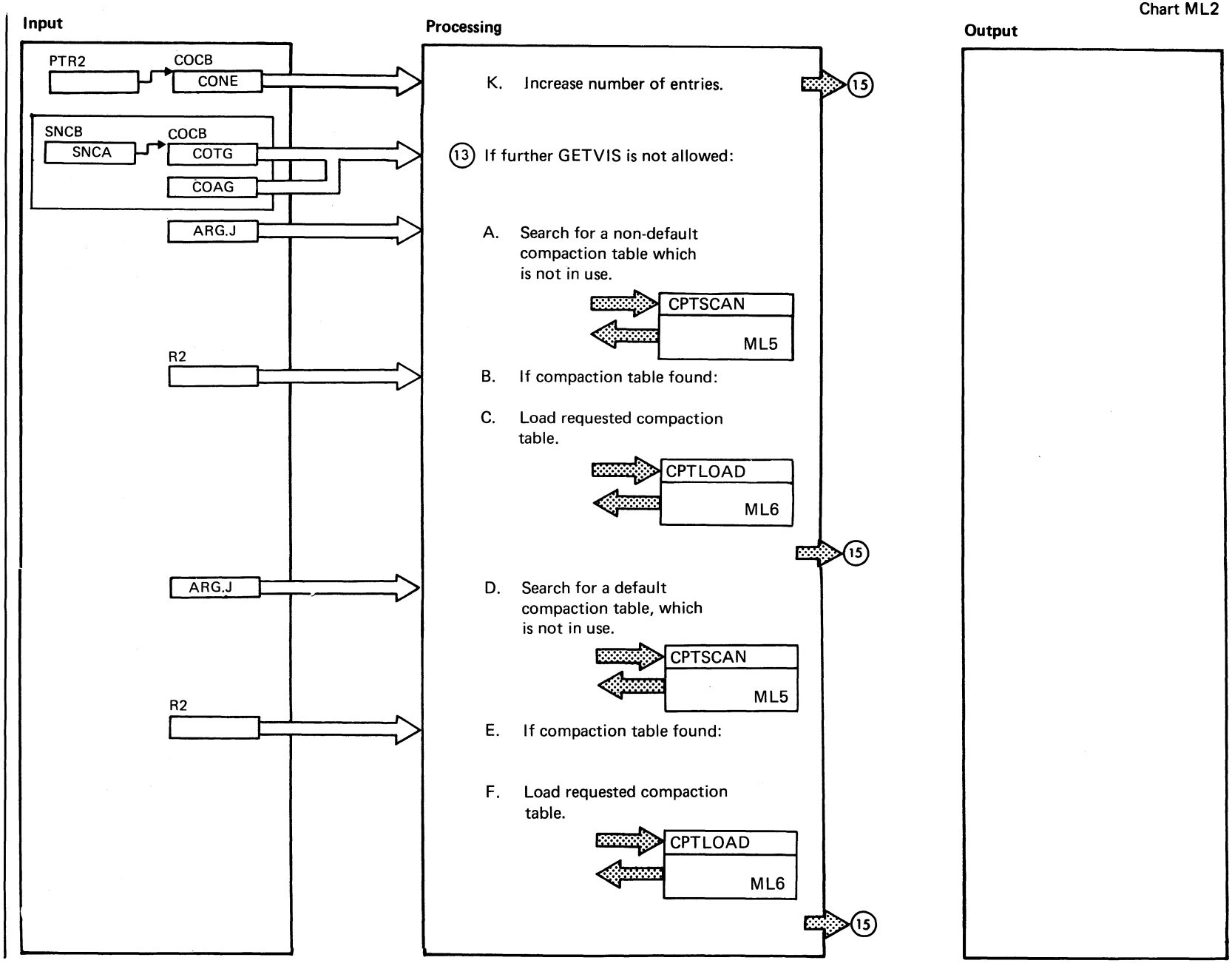
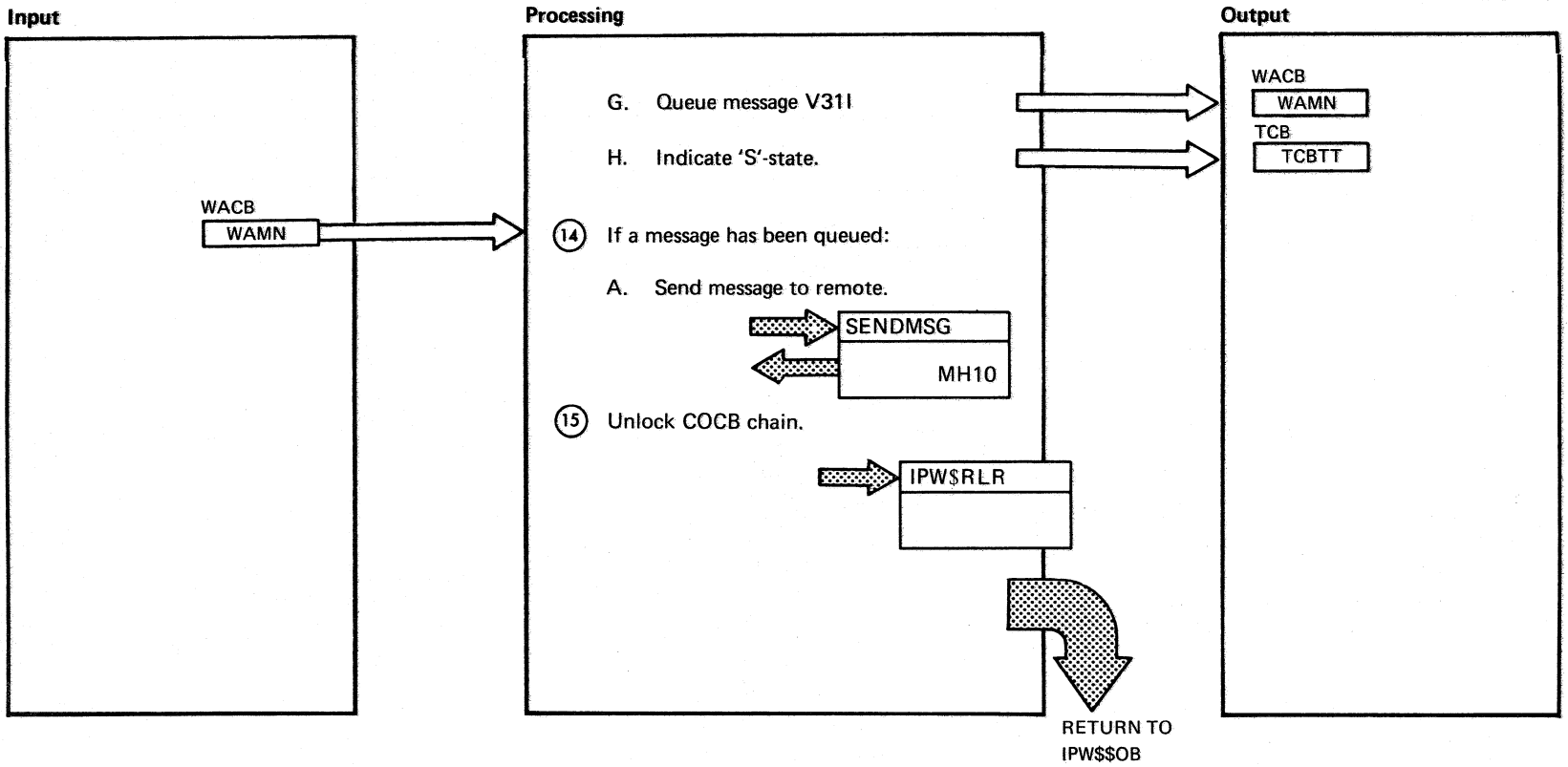


Chart ML2
Part 1 of 6


Input
Processing
Output
TCB
TCBTT
ARG.J
R2
COCB
CTBLNA
WACB
LDPN1
SUCB
SUO1
TCB
TCBTT
E. Increase USE counter.
F.
15
08 If 'S'-state is presented:
14
09 Search for requested compaction table.
CPTSCAN
ML5
10 If name found in a COCB:
A. Increase USE counter.
B. Store address of entry in COCB.
C. Store compaction table name.
D. If default compaction table requested:
E. Set default indicator.
F. Set FMH3 SEND indicator.
G. Set IN USE indicator.
H.
15
11 If 'S'-state is presented:
14
COCB
CTBLUS
COCB
CTBLUS
LUCB
LUO2
LUO1
COCB
CTBLID
WACB
WAF3
WAUI


Input
Processing
Output
R2
SNCB
SNCA
COCB
COAG
COTG
ARG.J
R2
TCB
TCBTT
TCB
TCBTT
TCB
TCBTT
SNCB
SNCA
COCB
COAG
(12) If name not found in a COCB and further GETVIS allowed:
A. Search for free entry in a COCB.
CPTSCAN
ML5
B. If no free entry found:
C. Provide a new COCB.
GETSPCE1
ML3
D. If 'S'-state is presented:
(14)
E. Provide space for a compaction table.
GETSPCE2
ML4
F. If 'S'-state is presented:
(14)
G. Load requested compaction table.
CPTLOAD
ML6
H. If 'S'-state is presented:
I. Free space again.
J. Reduce actual GETVIS counter.
(14)


Input
Processing
Output
PTR2
COCB
CONE
SNCB
SNCA
COCB
COTG
COAG
ARG.J
R2
ARG.J
R2
K. Increase number of entries.
15
13 If further GETVIS is not allowed:
A. Search for a non-default compaction table which is not in use.
CPTSCAN
ML5
B. If compaction table found:
C. Load requested compaction table.
CPTLOAD
ML6
15
D. Search for a default compaction table, which is not in use.
CPTSCAN
ML5
E. If compaction table found:
F. Load requested compaction table.
CPTLOAD
ML6
15



Extended Description

Include Segment	Call Subroutine/ Macro	Chart

(02) WAF3 is set whenever a management header type 3 (FMH3) has to be transmitted to the workstation. FMH3 represents the compaction table which has been used for a job.

(03) WACF is set whenever a valid compaction table (that means the table set generated by use of PCPTAB - macro) is available for a job.

(04) SNCA contains the address of the first COCB in chain. If SNCA = 0 means that no COCB exists and therefore the first COCB in chain has to be generated.

Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(05) If GETSPCE1 returns with 'S' - state, that means that providing an CPCB was not successful. A branch back to the outbound processor IPW$$OB is made where the CLOSE routine will detach the outbound processor. (06) If the CMPACT parameter in $$LST statement is omitted the default compaction table specified in SUCB has to be used for the job. LDPN1 and LDPN2 are used as fields in a local directory list for a later SVC4. (07) LUO1 and LUO2 in LUCB contain the name and the address of the currently active compaction table. If the last job used the same compaction table FMH3 must not send to the workstation. In this case WAF3 still remains not set and a check has to be performed, whether the compaction table name in LUO1 matches the name in COCB pointed to by LUO2. This check is necessary, because in the meantime the COCB entry might have been overridden. (09) After return from CPTSCAN, R2 will contain the address of the entry in a COCB if the search was successful, that means the compaction table name was found in a COCB. If R2 contains zeroes, the search was unsuccessful. ARG contains the name of the requested compaction table and J sets the position in each entry where to compare with search argument as following figure shows: C O C B C O C B - header I ↓ C O C B - entry name | address | identifier | use count | length 1 9 J → In this case J is set to the position of name in a COCB entry. (12) R2 contains zeroes after return from CPTSCAN. Further GETVIS is allowed if the total GETVIS counter COTG is greater than the actual GETVIS counter COAG. After each successful GETVIS COAG is increased by one. For the whole COCB chain COTG and COAG are maintained only in the first COCB. ARG contains zeroes, J is set to the position of the USE count in a COCB entry. (13) Further GETVIS is not allowed if COTG <= COAG.			

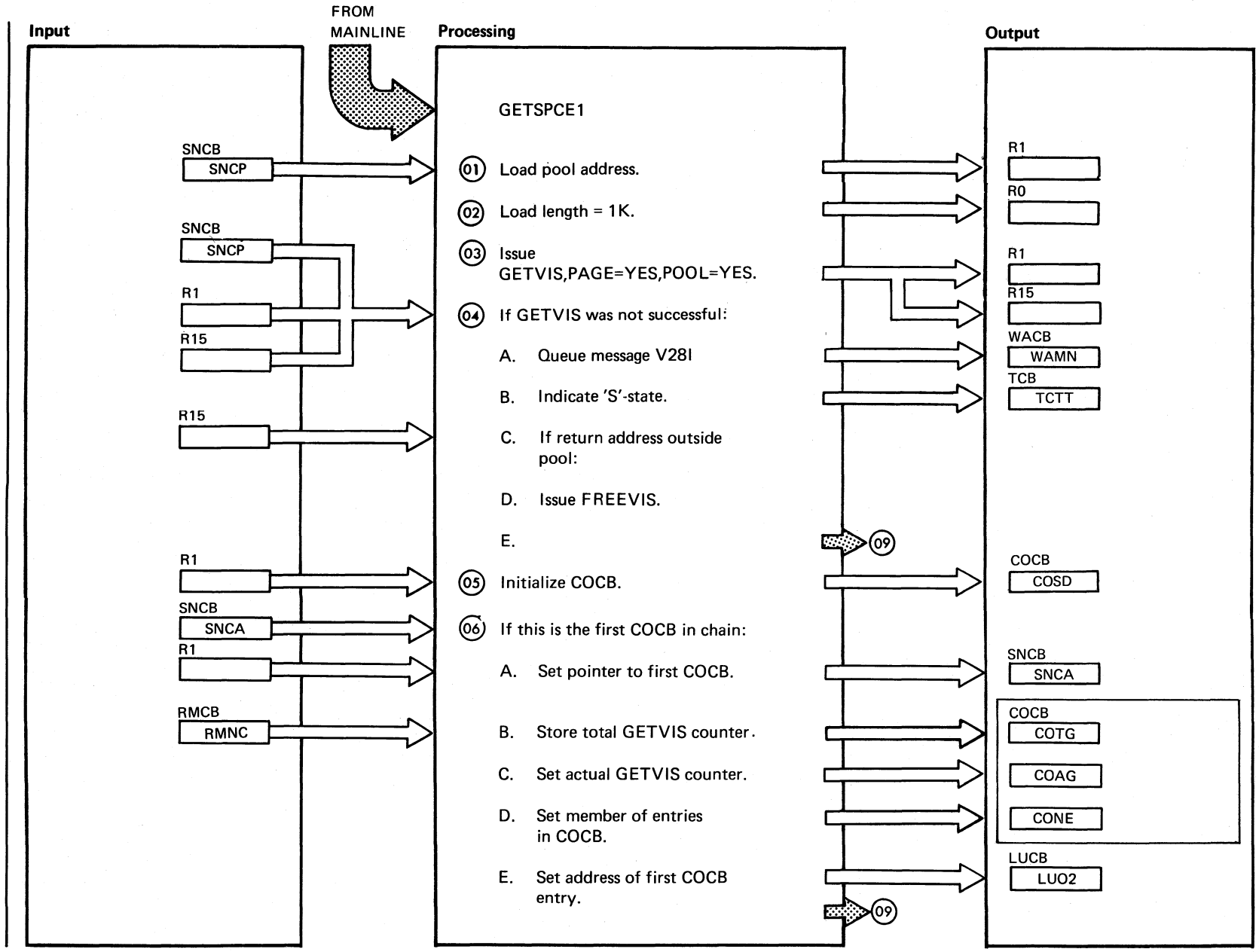
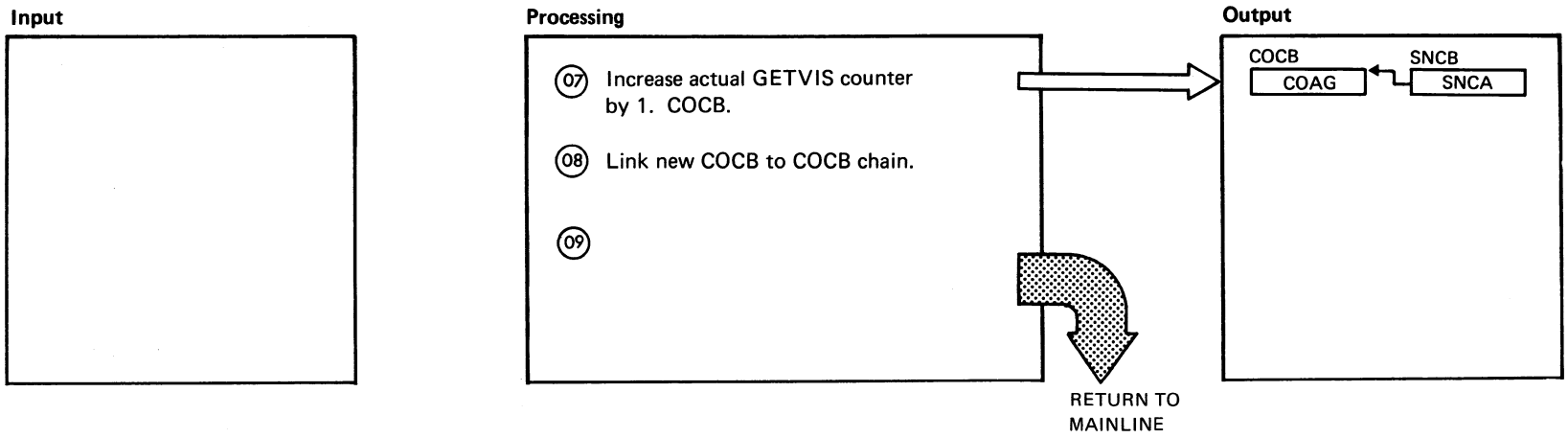
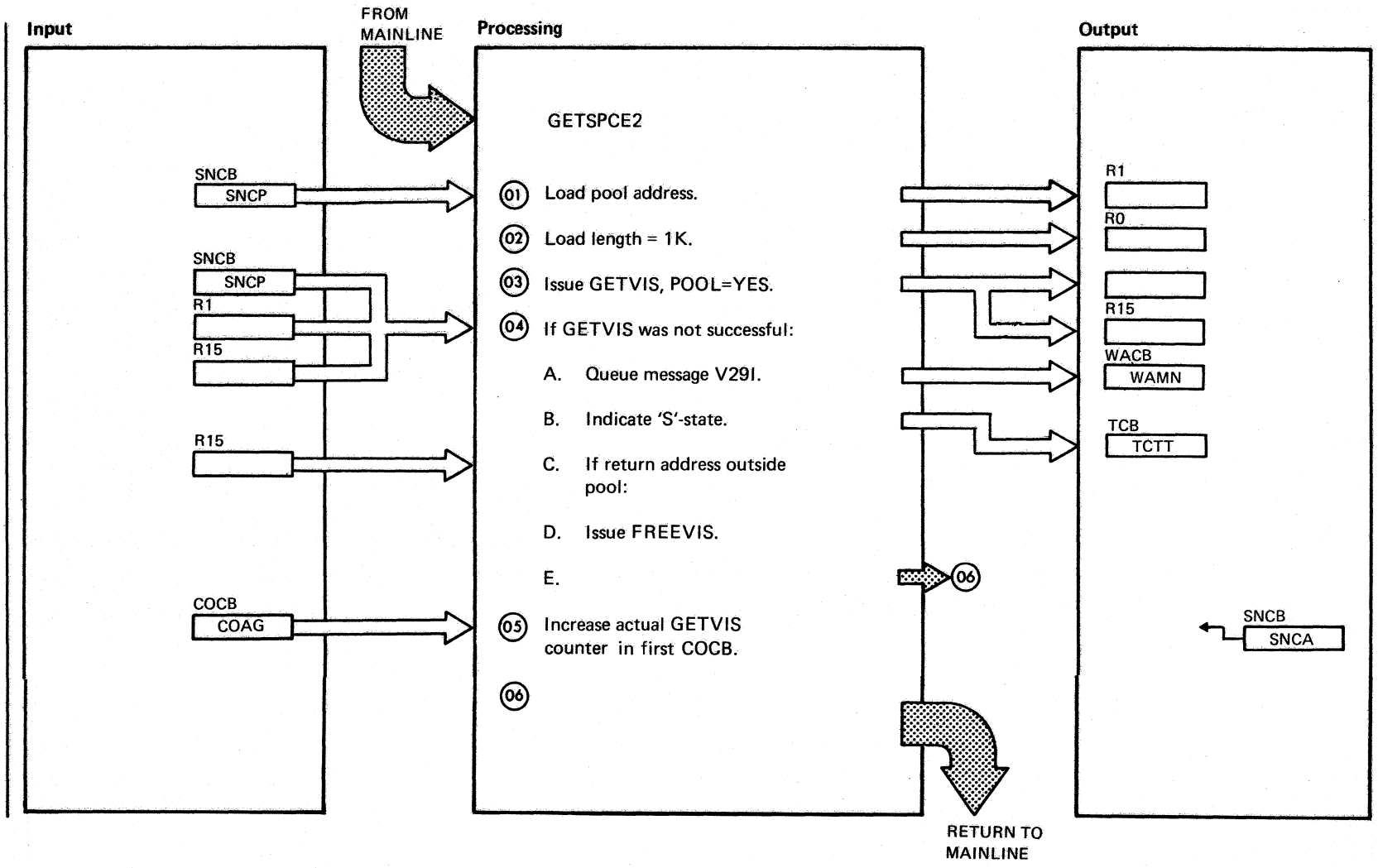


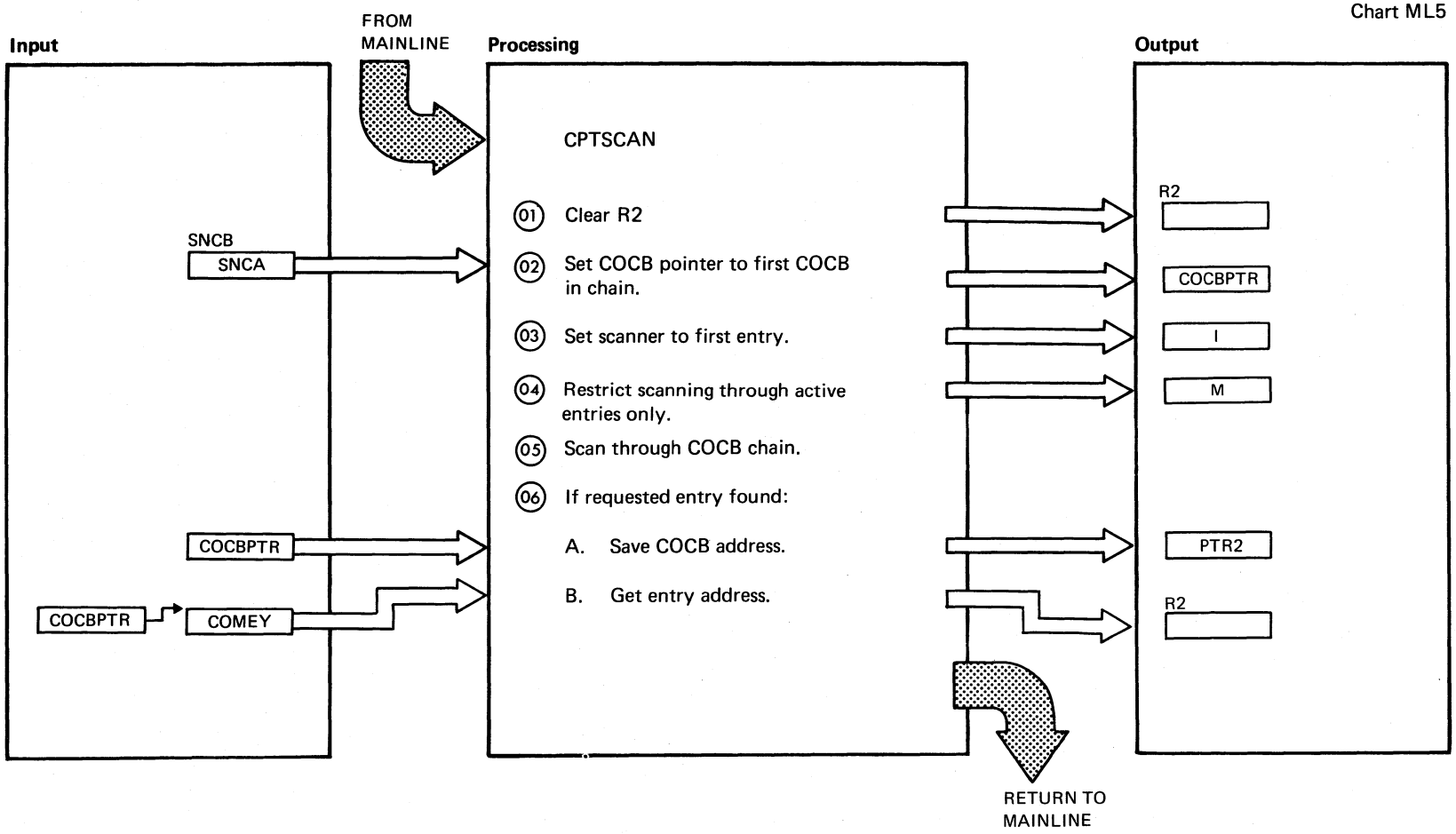
Chart ML3
Part 1 of 2



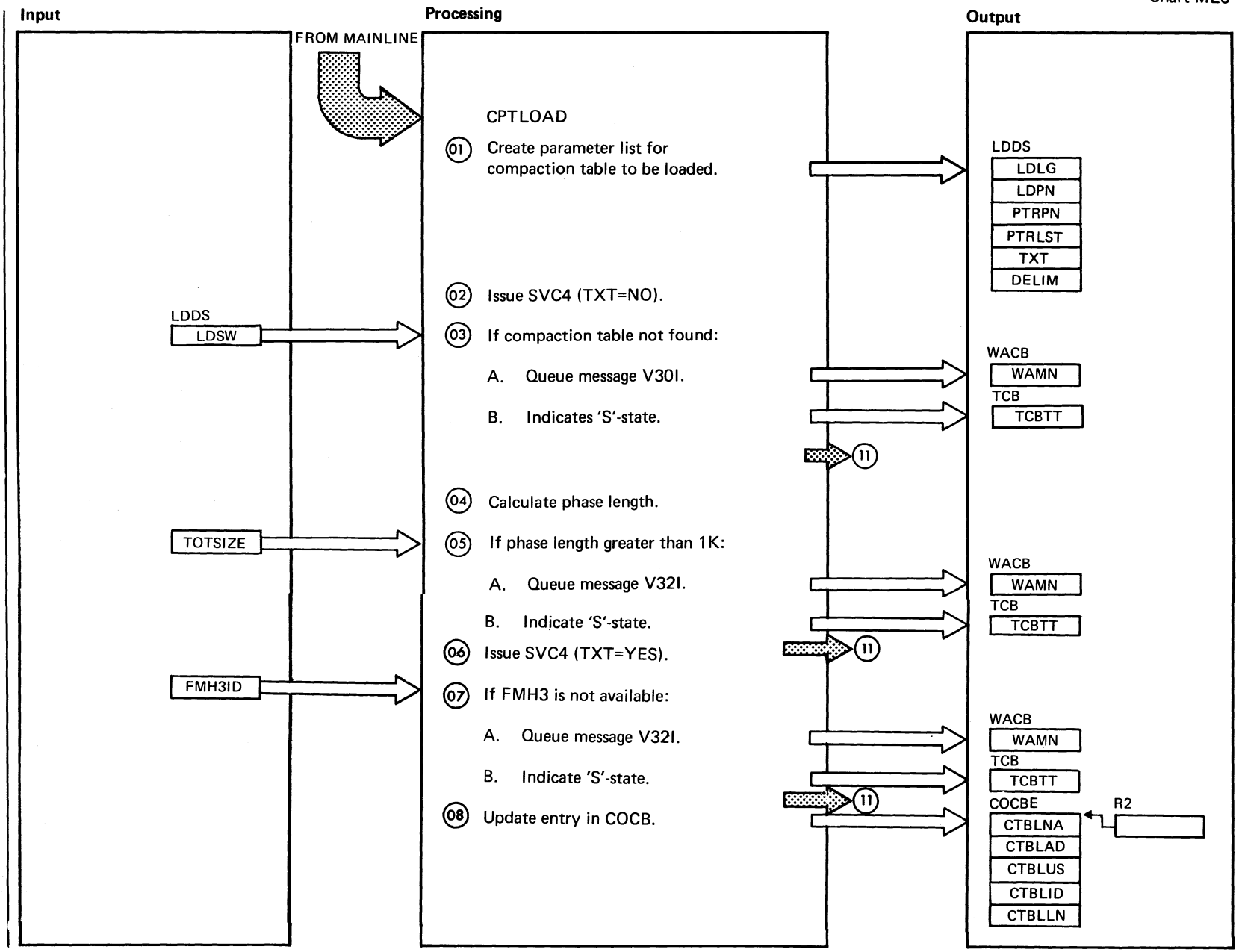
Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(03) The requested storage area starts on a page boundary and is located within the compaction table pool. (04) That means R15 contains a return code other than zero, or R1 contains an address outside the compaction table pool (R1 < SNCP). (04A) Message V28I: 1V28I GETVIS FOR COMPACTION TABLE CONTROL BLOCK FAILED (06) That means that SNCA contains zeroes. For the whole COCB chain total GETVIS counter COTG and actual GETVIS counter COAG are maintained in the first COCB only.			

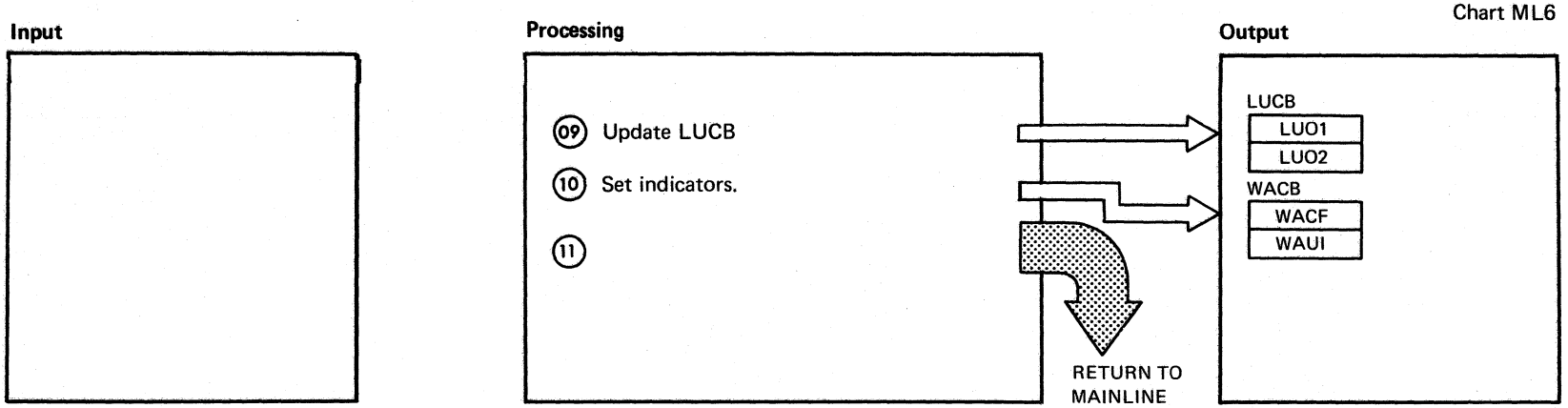

Input
FROM MAINLINE
Processing
Output
GETSPCE2
SNCB
SNCP
SNCB
SNCP
R1
R15
R15
COCB
COAG
01 Load pool address.
02 Load length = 1K.
03 Issue GETVIS, POOL=YES.
04 If GETVIS was not successful:
A. Queue message V29I.
B. Indicate 'S'-state.
C. If return address outside pool:
D. Issue FREEVIS.
E.
05 Increase actual GETVIS counter in first COCB.
06
06
R1
R0
R15
WACB
WAMN
TCB
TCTT
SNCB
SNCA
RETURN TO MAINLINE

Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(04) That means R15 contains a return code other than zero, or R1 contains an address outside the compaction table pool (R1 < SNCP). (04A) Message V29I: 1V29I: GETVIS FOR COMPACTION TABLE FAILED. (05) For the whole COCB chain total GETVIS counter COTG and actual GETVIS counter COAG are maintained in the first COCB only.			



Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(05) For scanning thru COCB chain the search argument is passed by ARG, the position in COCB entry by J from MAINLINE routine.			


Input
Processing
Output
FROM MAINLINE
CPTLOAD
01 Create parameter list for compaction table to be loaded.
LDDS
LDLG
LDPN
PTRPN
PTRLST
TXT
DELIM
02 Issue SVC4 (TXT=NO).
LDDS
LDSW
03 If compaction table not found:
A. Queue message V301.
B. Indicates 'S'-state.
WACB
WAMN
TCB
TCBTT
11
04 Calculate phase length.
TOTSIZE
05 If phase length greater than 1K:
A. Queue message V321.
B. Indicate 'S'-state.
WACB
WAMN
TCB
TCBTT
06 Issue SVC4 (TXT=YES).
11
FMH3ID
07 If FMH3 is not available:
A. Queue message V321.
B. Indicate 'S'-state.
WACB
WAMN
TCB
TCBTT
11
08 Update entry in COCB.
COCBE
R2
CTBLNA
CTBLAD
CTBLUS
CTBLID
CTBLLN



Extended Description	Include Segment	Call Subroutine/ Macro	Chart
03A Message V301: 1V301: COMPACTION TABLE NOT FOUND. 04 This check is necessary because user might have specified unintentionally a phase name, which does not represent a compaction table. As the compaction table generated by PCPTAB macro is always 1K bytes long, phases with a length of 1K bytes are considered as compaction tables for the first. 05A Message V321: 07A 1V321: INVALID COMPACTION TABLE. 07 After loading the phase this check has to be done to be sure, that this phase represents a compaction table, generated by use of PCPTAB macro.			

CHART MV: IPW$$VE - VTAM EXIT MODULE (20 PARTS)

Chart MV00: IPW$$VE - VTAM Exit Module, General Flow and Macro Calls

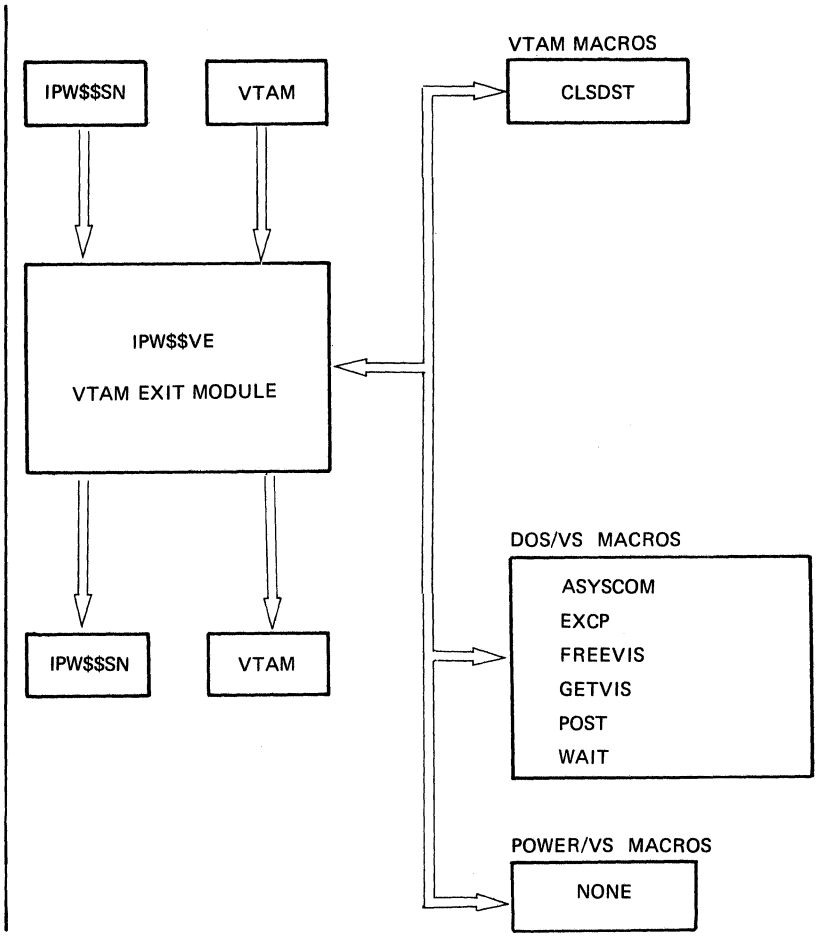


Chart MV1: IPW$$VE - VTAM Exit Module, Organization

VTAM EXIT MODULE (IPW$$VE) ORGANIZATION

Chart MV1

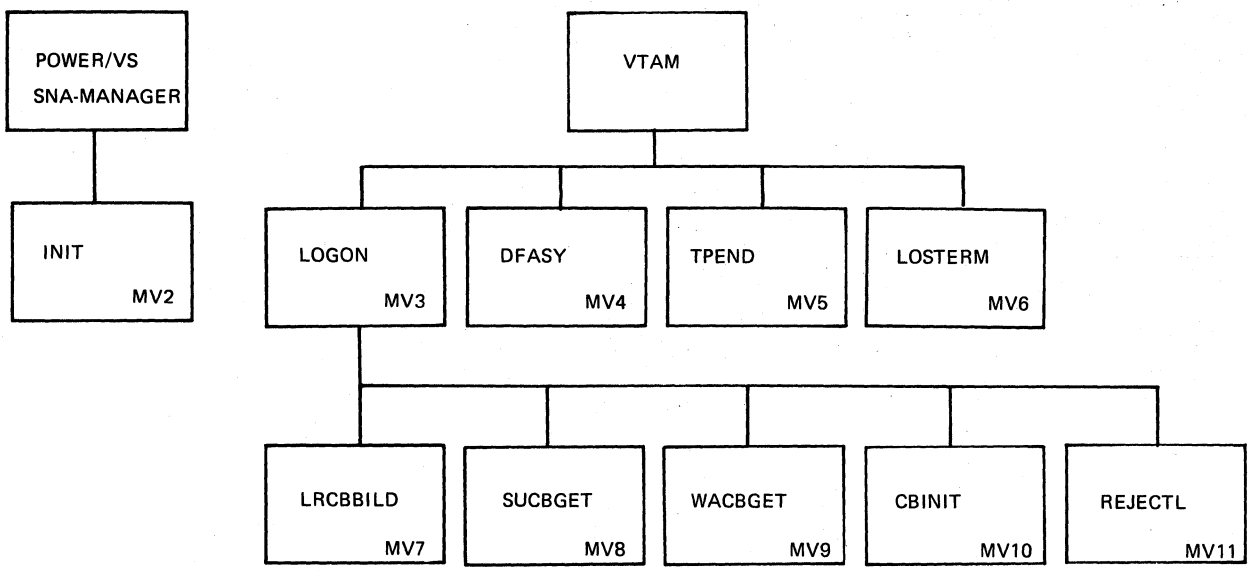
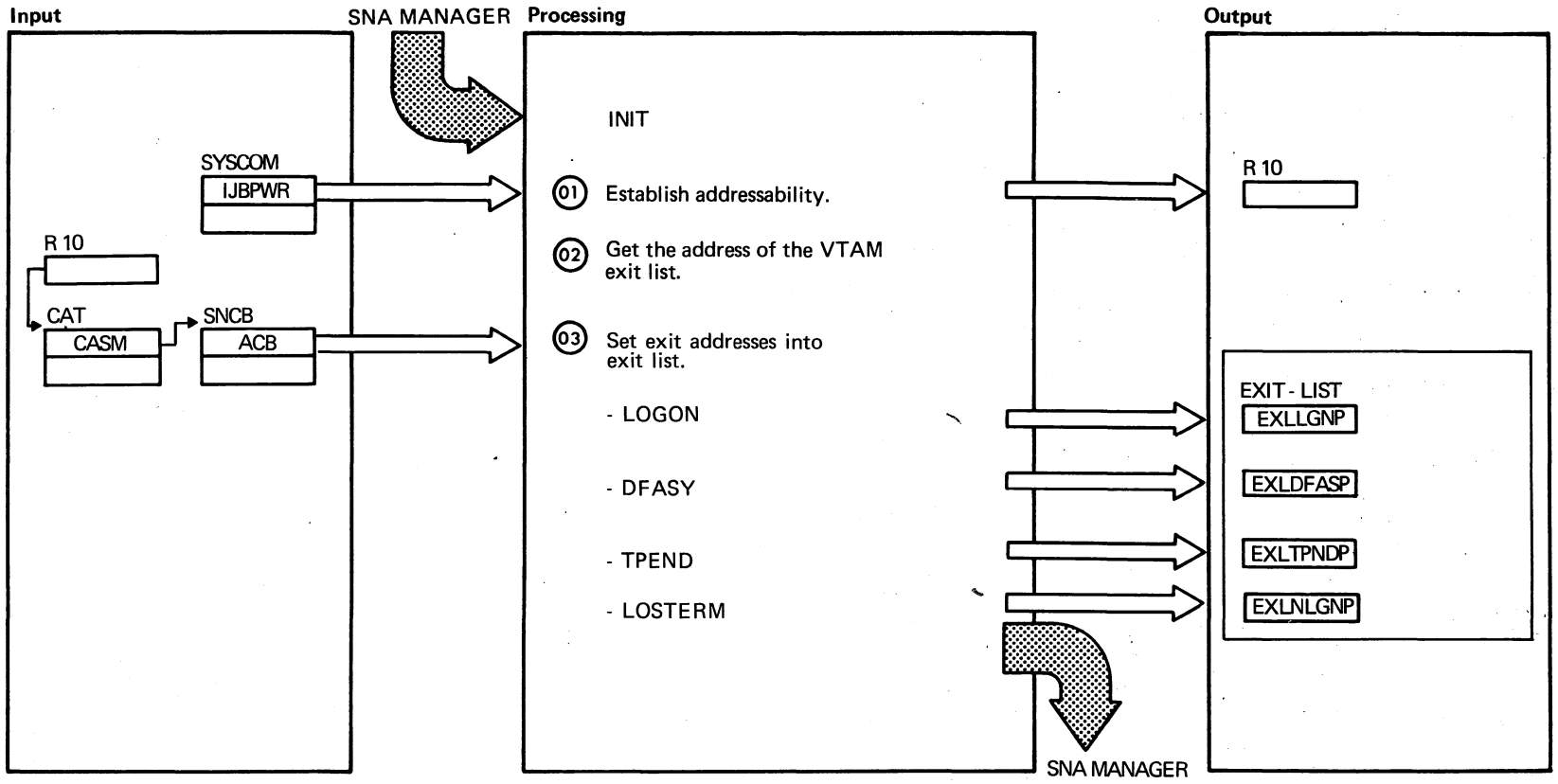


Chart MV1

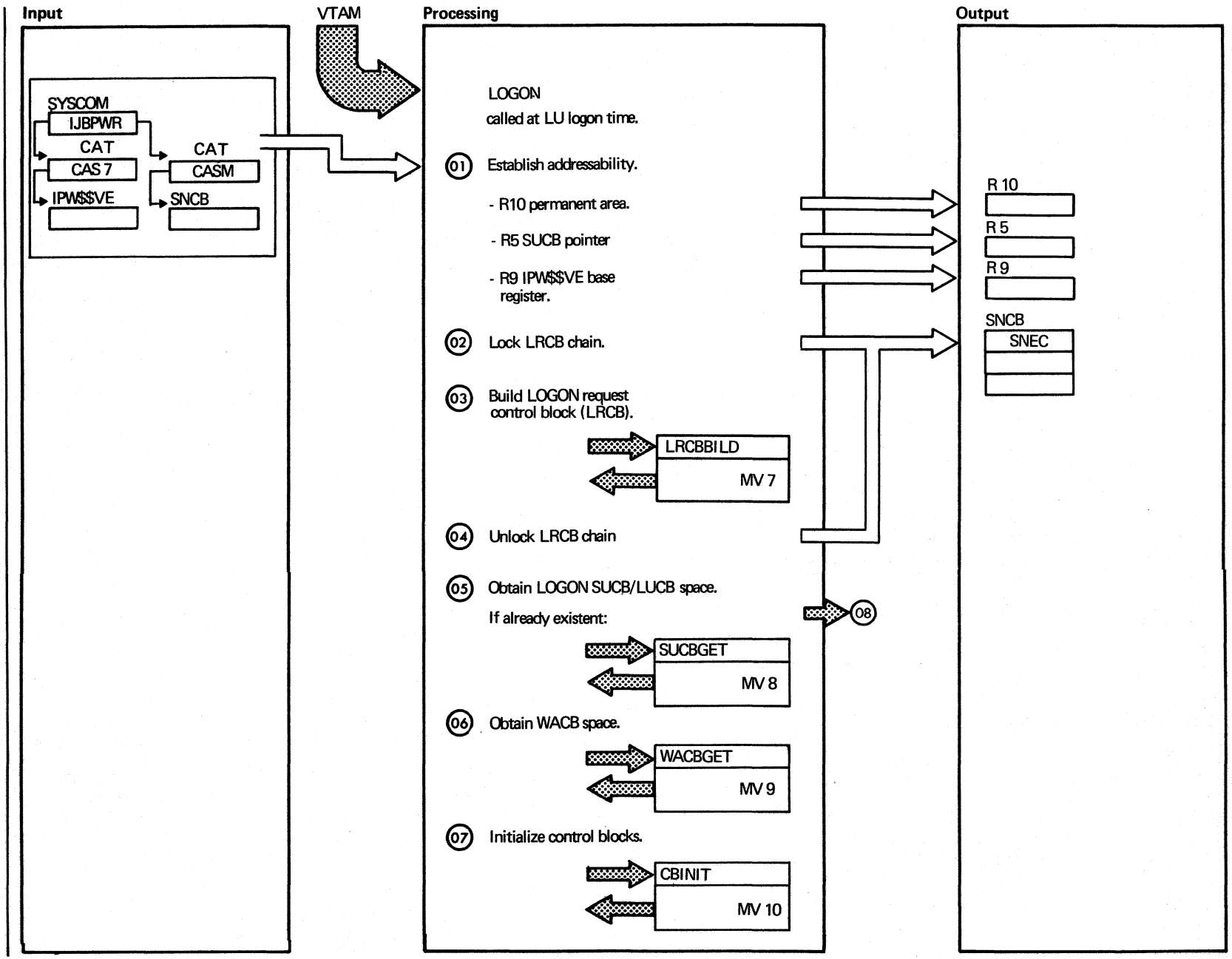


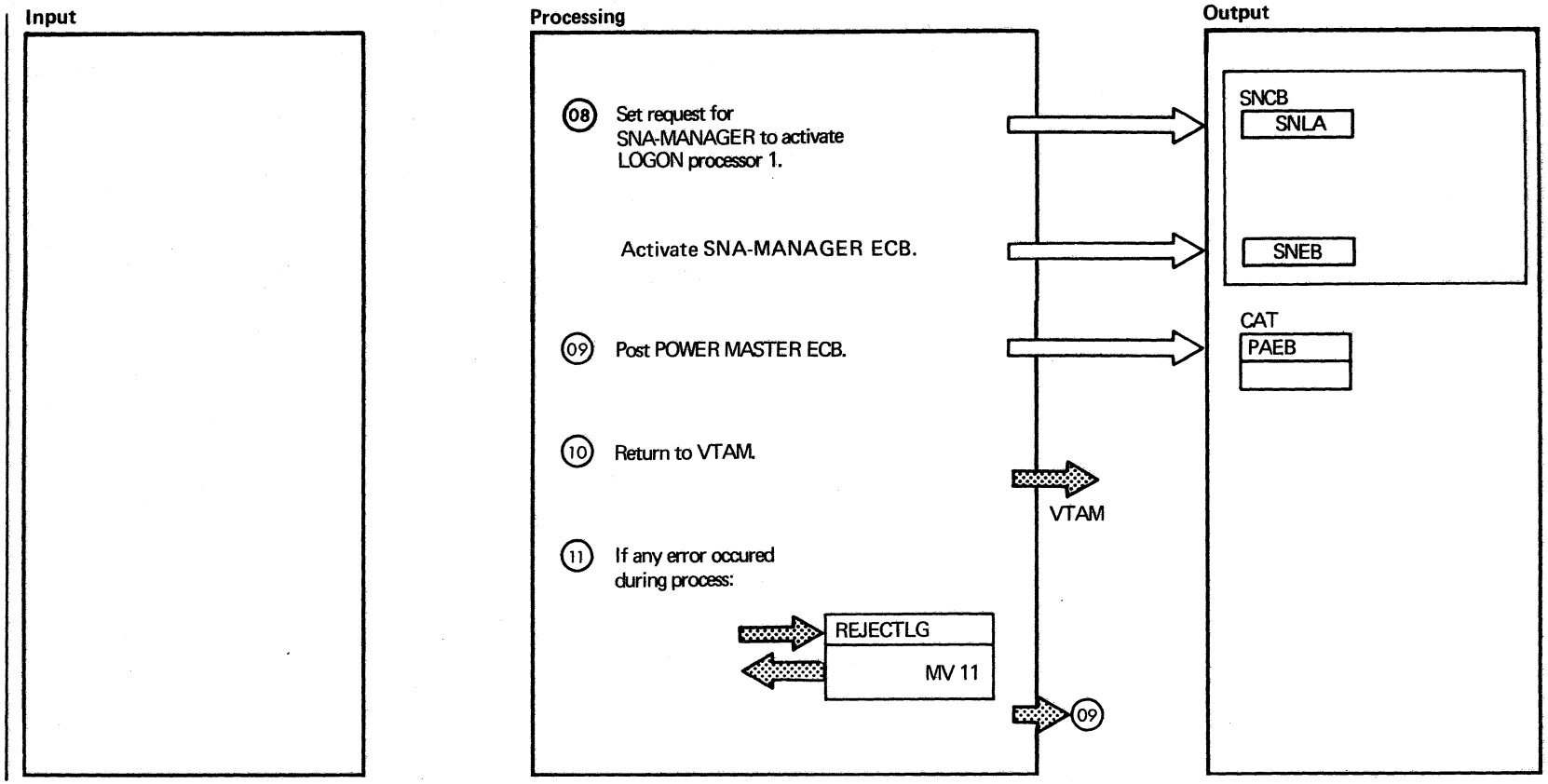
Extended Description

Include Segment	Call Subroutine/ Macro	Chart

(01) Upon entry, R15 is setup to contain the address of IPW$$VE, and is used as base-register for the initialisation routine. The initialisation routine is entered from the DOS/VS subtask in the SNA-MANAGER, and will return to it.

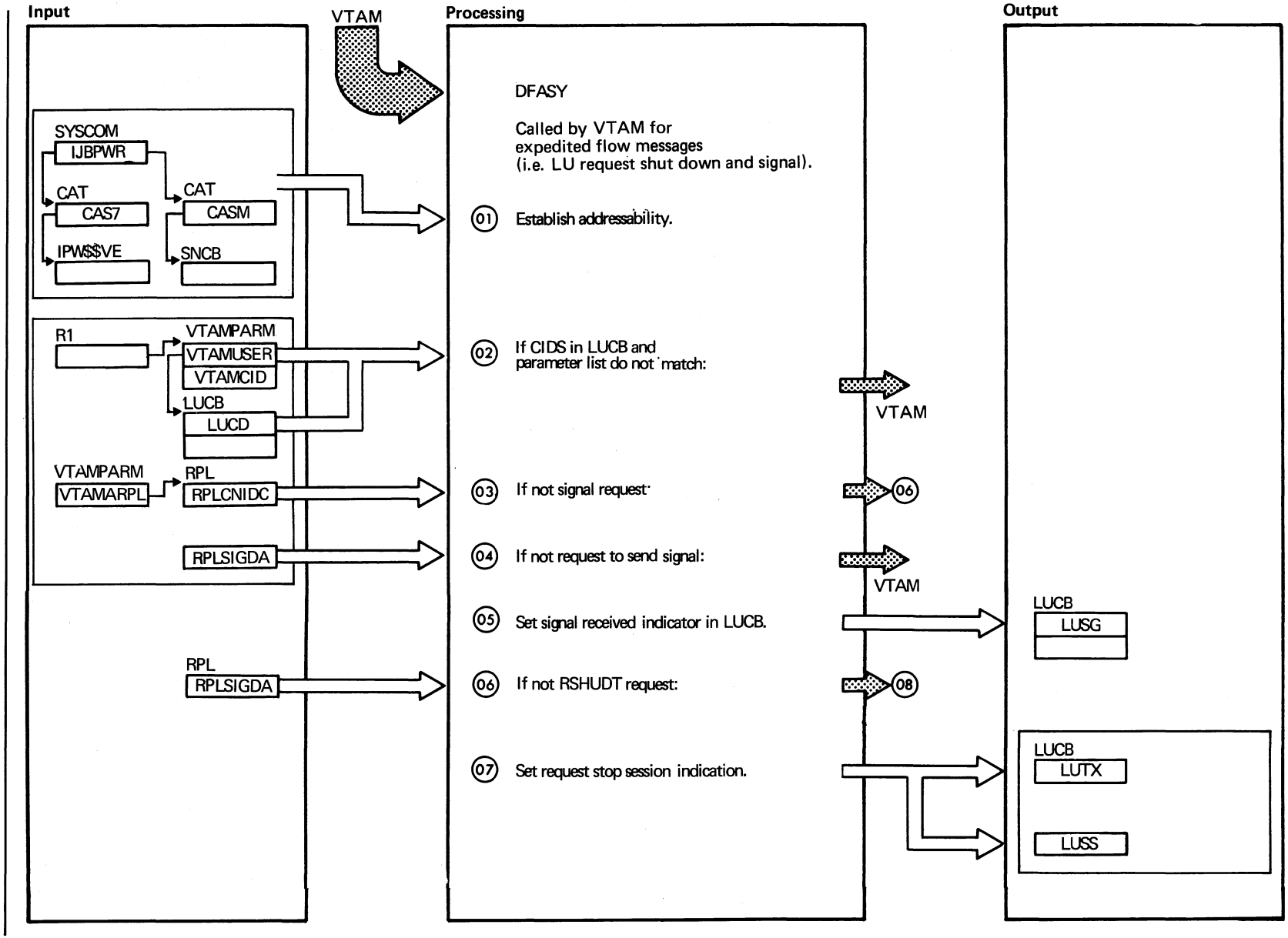
(02) The address of the VTAM exit list is set into the VTAM ACB, located in the SNCB, prior to entering the INIT routine. The exit list itself is located in the SNA-MANAGER.

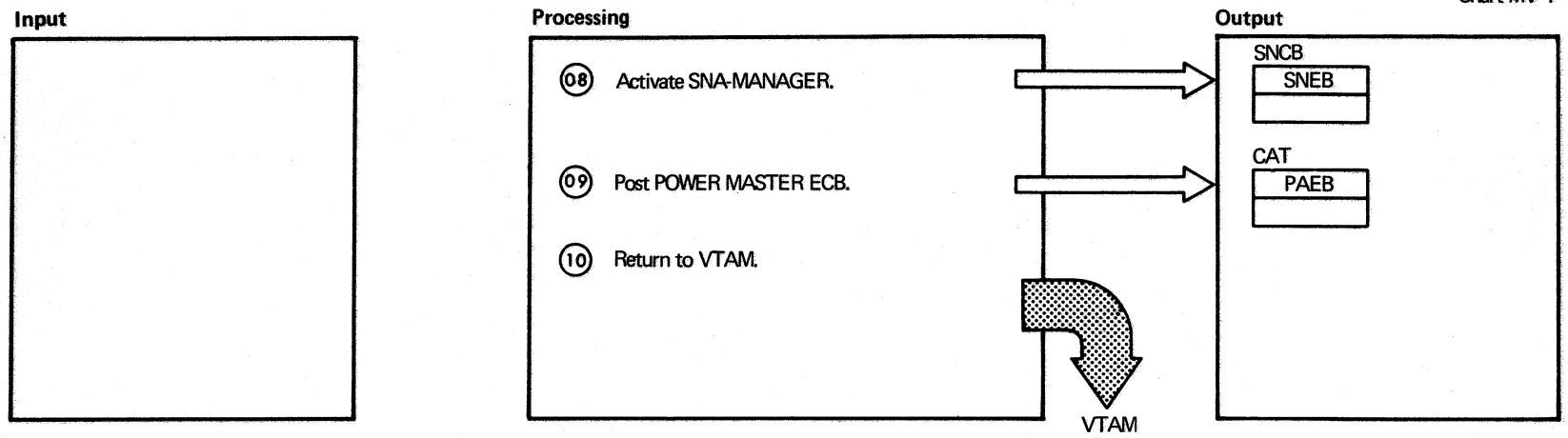

Input
VTAM
Processing
Output
SYSCOM
IJBPWR
CAT
CAS 7
IPW$$VE
CAT
CASM
SNCB
LOGON
called at LU logon time.
01 Establish addressability.
- R10 permanent area.
- R5 SUCB pointer
- R9 IPW$$VE base register.
02 Lock LRCB chain.
03 Build LOGON request control block (LRCB).
LRCBBILD
MV 7
04 Unlock LRCB chain
05 Obtain LOGON SUCB/LUCB space.
If already existent:
08
SUCBGET
MV 8
06 Obtain WACB space.
WACBGET
MV 9
07 Initialize control blocks.
CBINIT
MV 10
R 10
R 5
R 9
SNCB
SNEC


Input
Processing
Output
08 Set request for SNA-MANAGER to activate LOGON processor 1.
Activate SNA-MANAGER ECB.
09 Post POWER MASTER ECB.
10 Return to VTAM.
VTAM
11 If any error occured during process:
REJECTLG
MV 11
09
SNCB
SNLA
SNEB
CAT
PAEB

Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(01) On entry to this routine, R1 contains the address of a 4—word parameter list (ACB—addr, addr of LUNAME, reserved, length of LOGON message). R14 contains the address to VTAM. (02) The LRCB chain is locked to prevent a simultaneously UPDATE of one LRCB by LOGON exit and LOGON processor IPW$$LH. This might occur, when LOGON EXIT is interrupted due to a page fault, and POWER is activated by its PHO exit. (03) A check is made if a LRCB already exists. If not, a GETVIS request to DOS/VS is issued to obtain virtual storage to build a LRCB. It is looked for a free entry in the LRCB (if already existing). The VTAM parameter list is saved in the LRCB. (05) The first LOGON from a LU after SNA processing ist started. It causes the LOGON exit to obtain virtual storage via GETVIS request for the LOGON SUCB, LUCB and WACB. This storage is maintained during SNA processing and is used to handle further LOGON requests. (11) An error occured on GETVIS for a control block. Dependent on the type of error, the storage is released and the LOGON request is rejected. As the LOGON processor 1 might wait for the locked LRCB-chain, POWER has to be posted to reactivate the LOGON processor in case no other action takes place within POWER.			

Chart MV 4


Input
VTAM
Processing
Output
DFASY
Called by VTAM for
expedited flow messages
(i.e. LU request shut down and signal).
SYSCOM
IJBPWR
CAT
CAS7
CAT
CASM
IPW$$VE
SNCB
R1
VTAMPARM
VTAMUSER
VTAMCID
LUCB
LUCD
VTAMPARM
VTAMARPL
RPL
RPLCNIDC
RPLSIGDA
RPL
RPLSIGDA
01 Establish addressability.
02 If CIDS in LUCB and parameter list do not match:
VTAM
03 If not signal request
06
04 If not request to send signal:
VTAM
05 Set signal received indicator in LUCB.
LUCB
LUSG
06 If not RSHUDT request:
08
07 Set request stop session indication.
LUCB
LUTX
LUSS



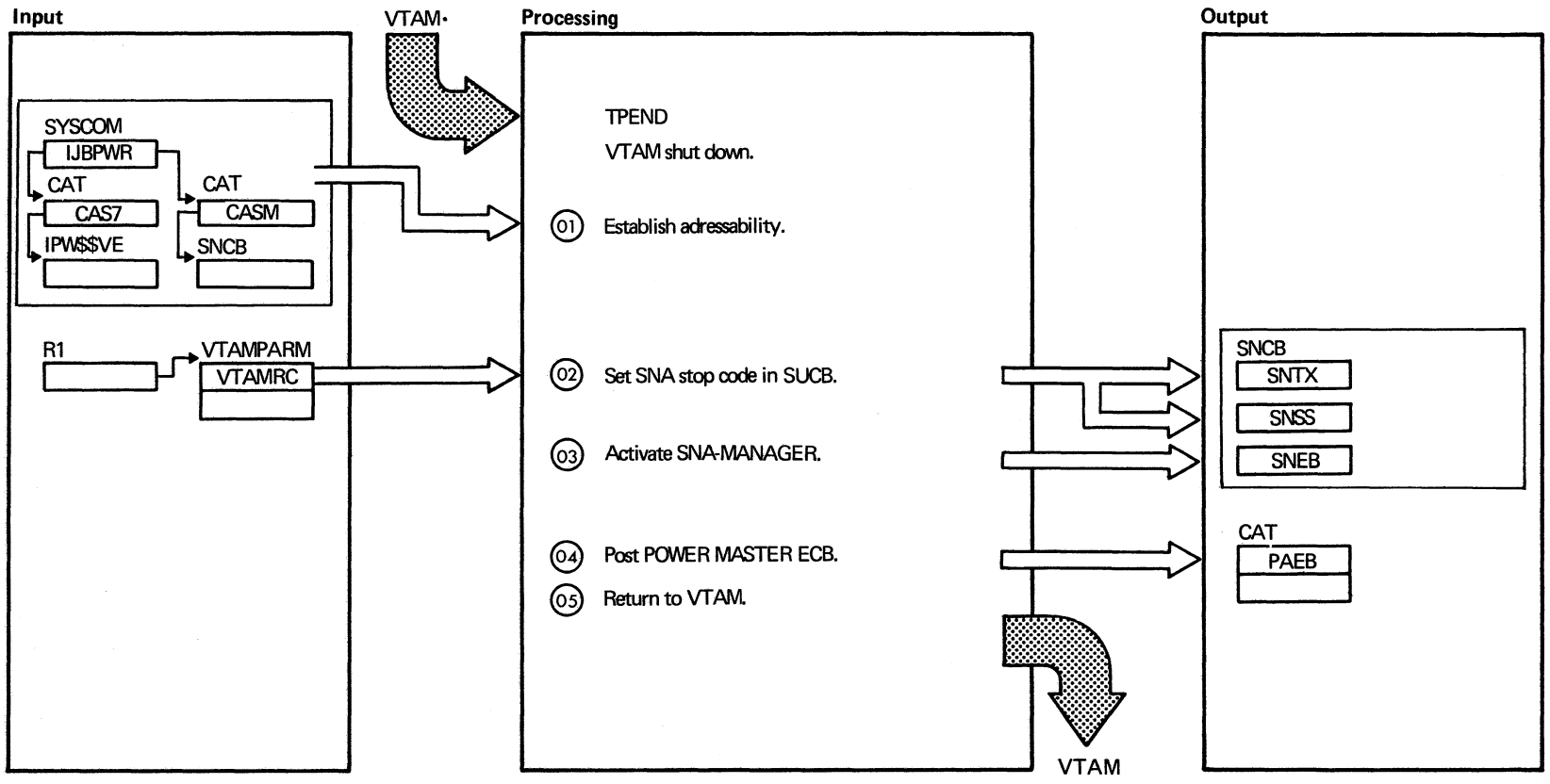
Extended Description

Include Segment	Call Subroutine/ Macro	Chart

(01) On entry to this routine, R1 contains the address of a 5-word parameter list (ACB-address, CID of terminal, user information (here address of a LUCB), number of bytes received and address of a read-only RPL). R14 contains the return address to VTAM.

(05) If a request to send signal is received, an information bit is set in the LUCB. This bit is periodically checked by inbound and outbound processor, running on this session. If the bit is found to be set, the processor will suspend to allow data-flow inbound.

(06) The second type of signal which is accepted, is request for shutdown. All other signals are ignored.



Extended Description

	Include Segment	Call Subroutine/ Macro	Chart
(01) On entry to this routine, R1 contains the address of a 2-word parameter list (ACB-adress, reason for shutdown (0 = NORMAL, 4 = QUICK HALT)). R14 contains the return address to VTAM. (02) Dependent on the VTAM reason for shutdown, a HALT or a HALT IMMEDIATE indication is set in the SNCB.			

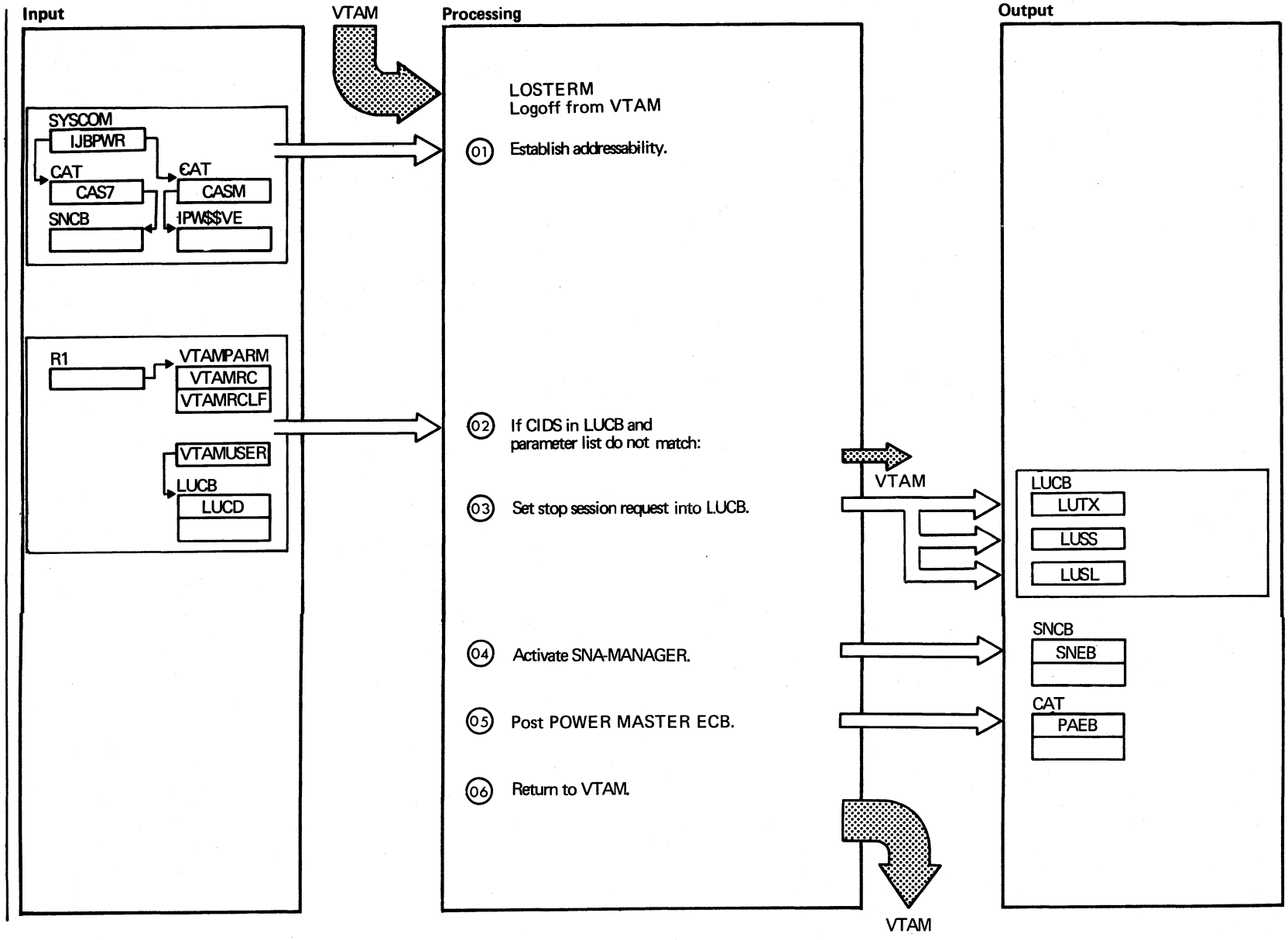


Chart MV6
Part 1 of 2

Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(01) On entry to this routine, R1 contains the address of a 4-word parameter list (ACB-address, CID of terminal, user information (here: address of a LUCB), reason for LOGOFF (32 = CONDITIONAL LOGOFF)). R 14 contains the return address to VTAM. (03) Dependent on the LOGOFF reason, a HALT or HALT IMMEDIATE indication is set in the LUCB.			

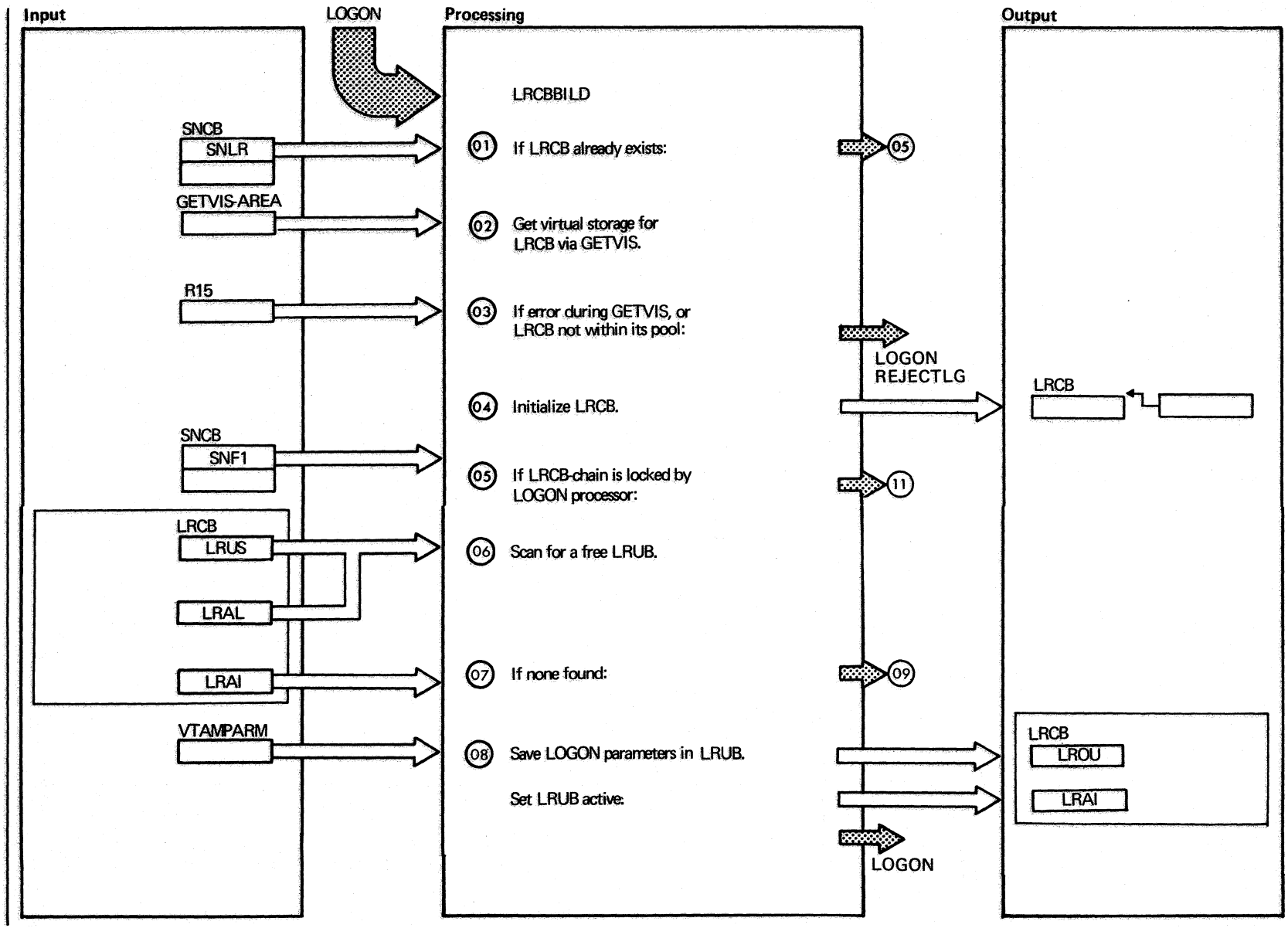
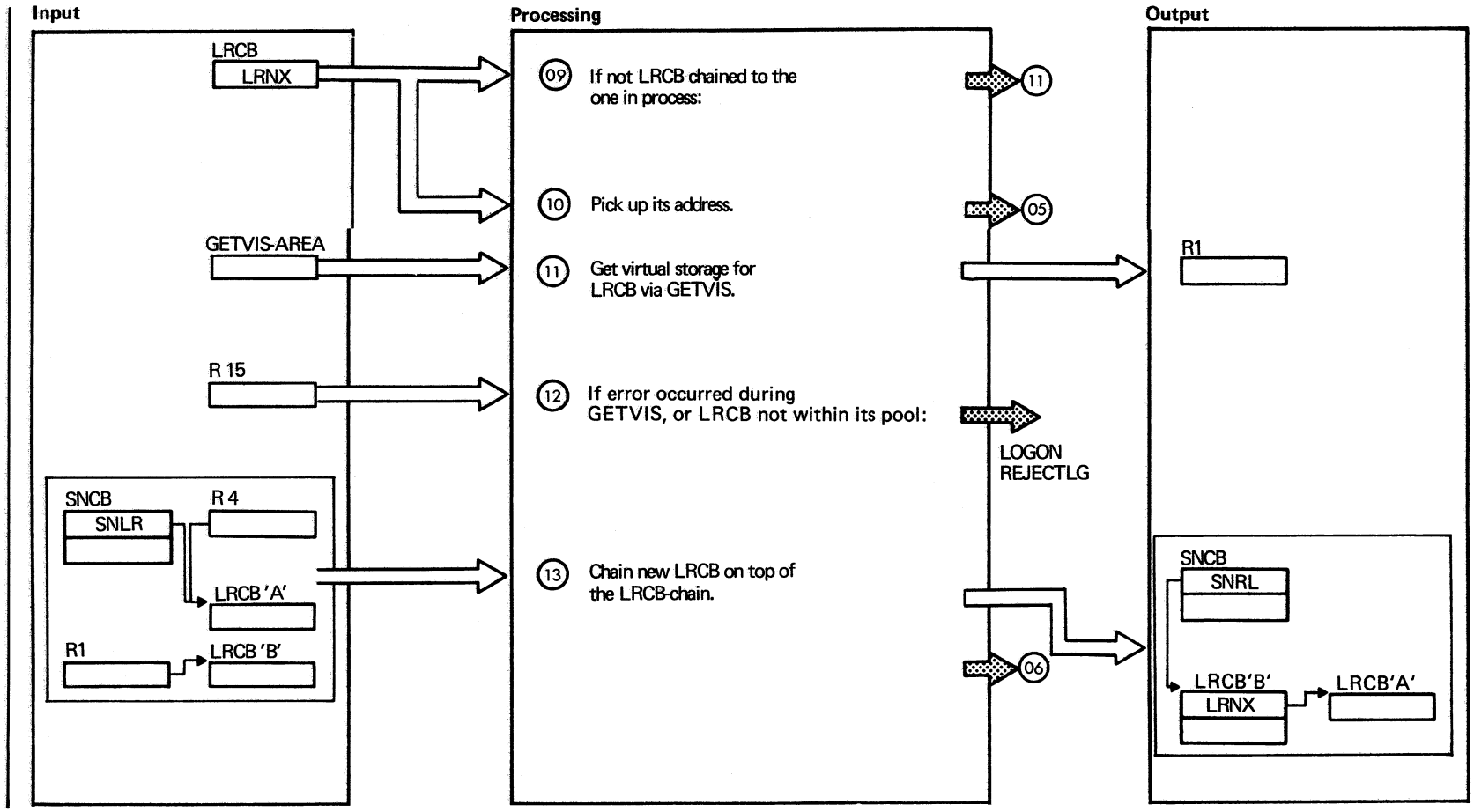
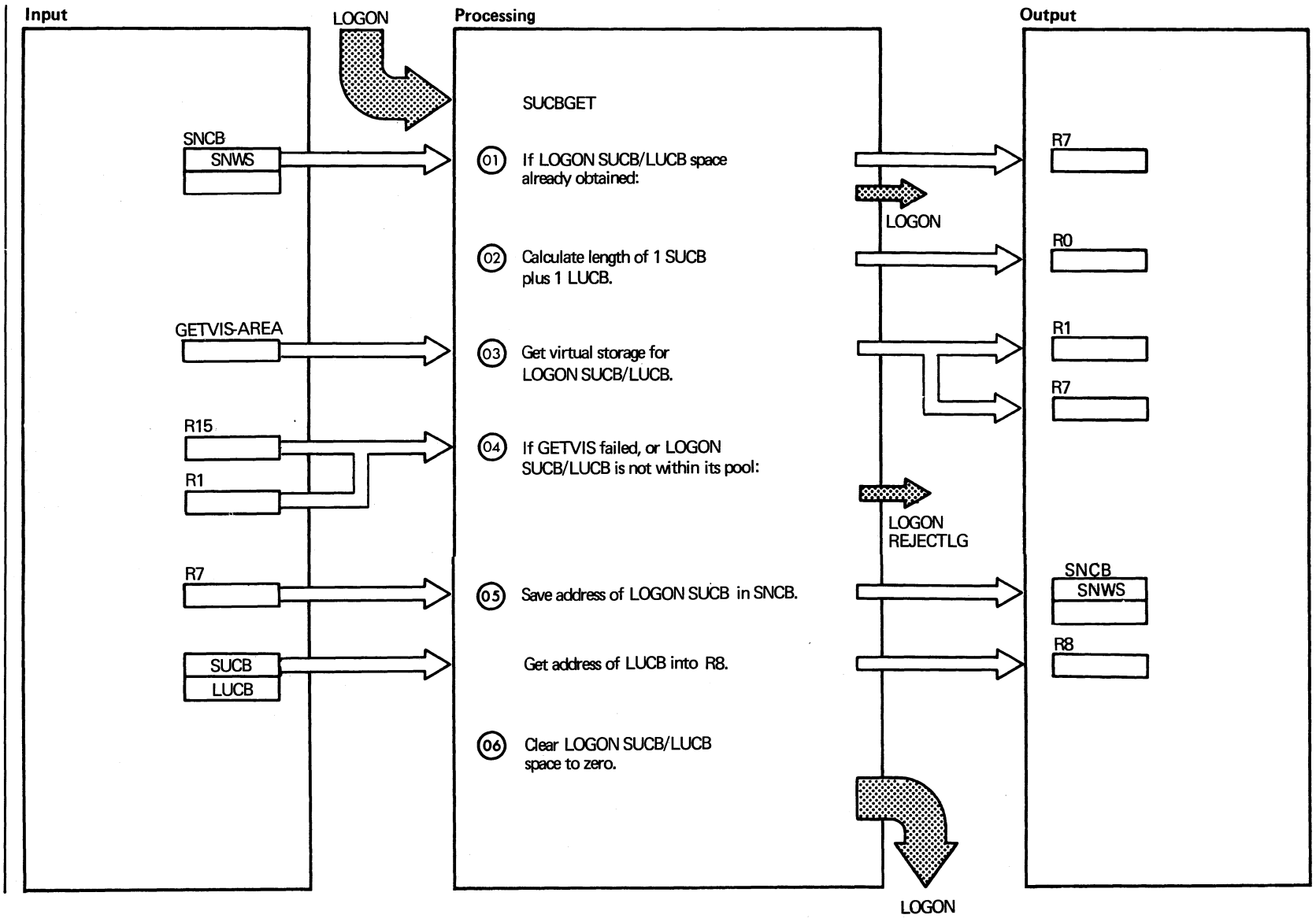


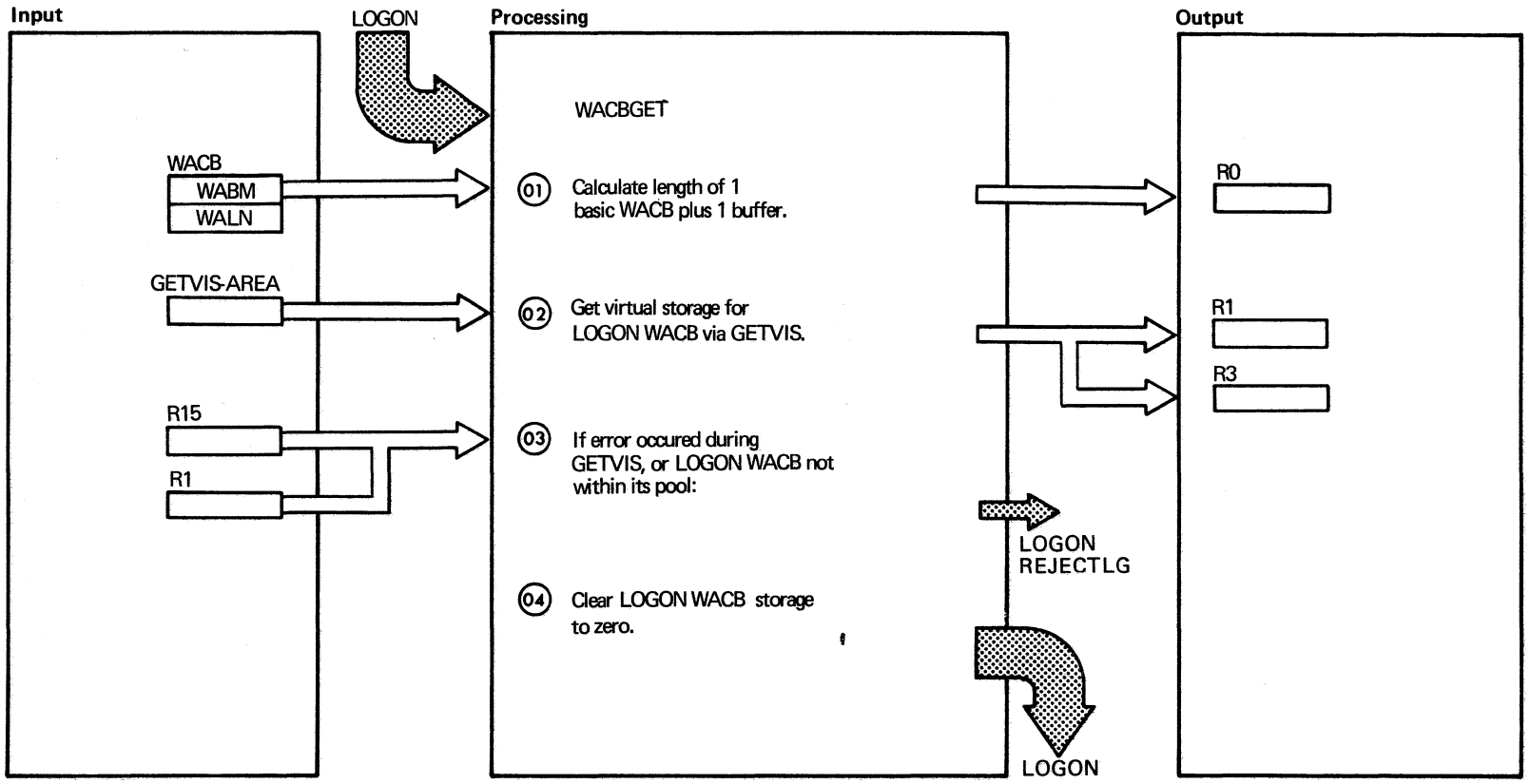
Chart MV7
Part 1 of 3


Input
Processing
Output
LRCB
LRNX
GETVIS-AREA
R 15
SNCB
SNLR
R 4
LRCB 'A'
R1
LRCB 'B'
09 If not LRCB chained to the one in process:
10 Pick up its address.
11 Get virtual storage for LRCB via GETVIS.
12 If error occurred during GETVIS, or LRCB not within its pool:
13 Chain new LRCB on top of the LRCB-chain.
11
05
LOGON
REJECTLG
06
R1
SNCB
SNRL
LRCB'B'
LRNX
LRCB'A'

Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(01) Pointer SNLR in SNCB contains the address of the first LRCB. (02) Get virtual storage from LOGON storage pool. (03) The GETVIS requested failed, or the piece of virtual storage obtained, is not within the LOGON storage pool. In the latter case, the storage must be freed, using the FREEVIS macro. (04) Clear LRCB storage to zero. Set storage descriptor and initial values. (05) The LOGON processor 1 (IPW$$LH) is active and processing the LRCB-chain. To prevent simultaneously UPDATE of a LRCB, the LOGON exit chains a new element on top of the chain. This element is in use only by the LOGON exit. (06) A free LRUB is found, when the active indicator is not on. (09) The LRNX-pointer in the LRCB contains an address, if another LRCB is chained. (11) R1 contains the address of the virtual storage for the LRCB. (13) LRCB"B" is the one to be chained. Using the "COMPARE AND SWAP" instruction R1 points to LRCB"B", R4 points to LRCB"A" which is also pointed bo by the SNLR, the hook in the SUCB to the LRCBs.			


Input
LOGON
Processing
Output
SUCBGET
SNCB
SNWS
01 If LOGON SUCB/LUCB space already obtained:
R7
LOGON
02 Calculate length of 1 SUCB plus 1 LUCB.
R0
GETVIS-AREA
03 Get virtual storage for LOGON SUCB/LUCB.
R1
R7
R15
R1
04 If GETVIS failed, or LOGON SUCB/LUCB is not within its pool:
LOGON REJECTLG
R7
05 Save address of LOGON SUCB in SNCB.
SNCB
SNWS
SUCB
LUCB
Get address of LUCB into R8.
R8
06 Clear LOGON SUCB/LUCB space to zero.
LOGON

Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(01) The SNWS pointer in the SUCB contains the address of the LOGON SUCB/LUCB space. (03) R1 points to the virtual storage for the SUCB/LUCB. (04) If no space for the LOGON SUCB/LUCB can be obtained from the LOGON space pool, the LOGON request has to be rejected. (05) The LOGON SUCB is based on R7, the LUCB is based on R8.			

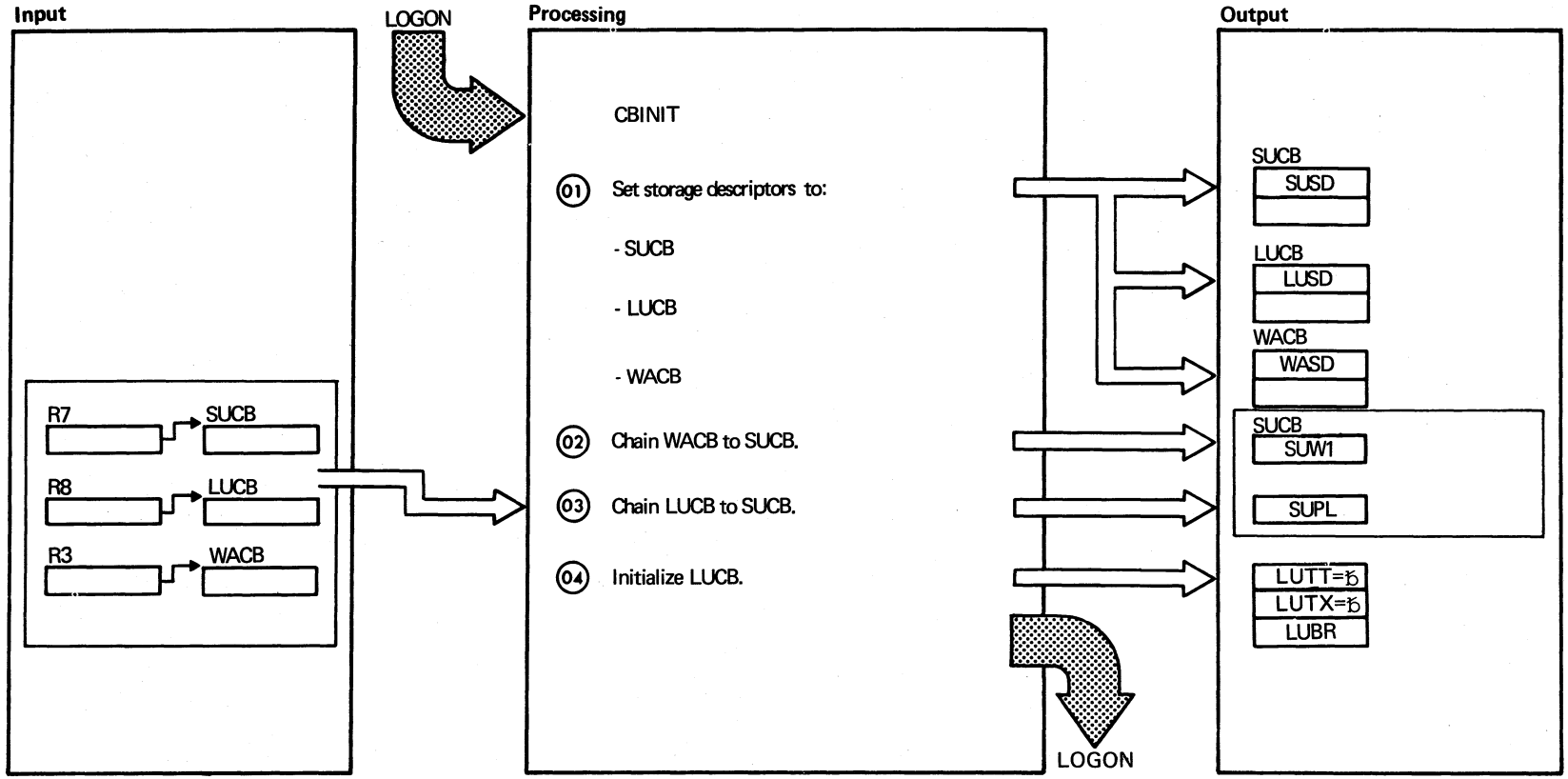


Extended Description

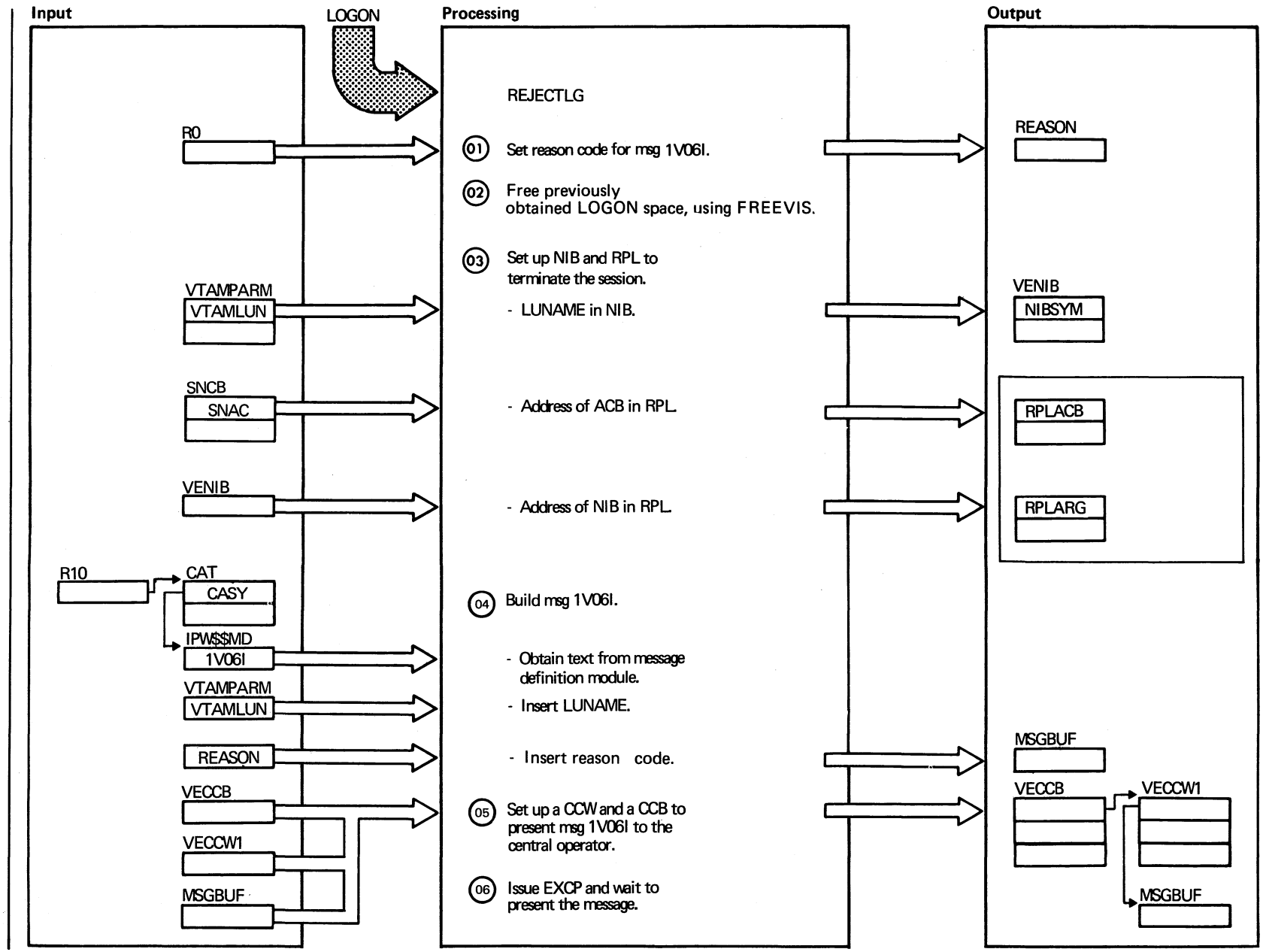
Include Segment	Call Subroutine/ Macro	Chart

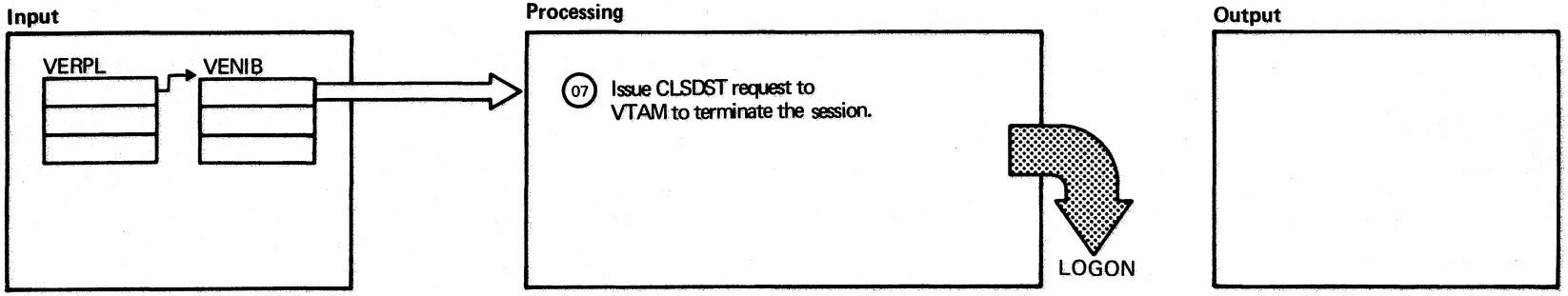
(01) The LOGON WACB consists of one basic WACB plus one buffer. The length of the buffer is according to the maximum buffer size, indicated in the WACB.

(02) The WACB is based on R3.



Extended Description	Include Segment	Call Subroutine/ Macro	Chart
01 Upon entry, R3, R7 and R8 were set up to point to the control blocks.			

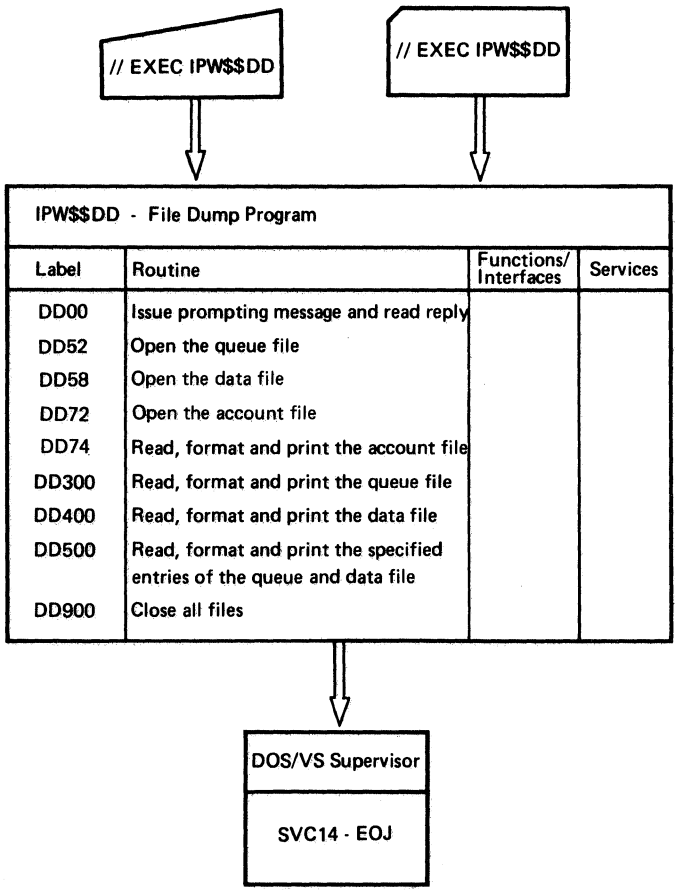

Input
LOGON
Processing
Output
REJECTLG
R0
01 Set reason code for msg 1V06I.
02 Free previously obtained LOGON space, using FREEVIS.
03 Set up NIB and RPL to terminate the session.
- LUNAME in NIB.
- Address of ACB in RPL.
- Address of NIB in RPL.
04 Build msg 1V06I.
- Obtain text from message definition module.
- Insert LUNAME.
- Insert reason code.
05 Set up a CCW and a CCB to present msg 1V06I to the central operator.
06 Issue EXCP and wait to present the message.
REASON
VTAMPARM
VTAMLUN
SNCB
SNAC
VENIB
R10
CAT
CASY
IPW$$MD
1V06I
VTAMPARM
VTAMLUN
REASON
VECCB
VECCW1
MSGBUF
VENIB
NIBSYM
RPLACB
RPLARG
MSGBUF
VECCB
VECCW1
MSGBUF



Extended Description	Include Segment	Call Subroutine/ Macro	Chart
(01) Message "1V06I UNABLE TO LOGON LUNAME RC = xxx". (02) If space for a LOGON control block (LRCB, SUCB/LUCB or WACB) is not within the LOGON storage pool, this space has to be freed, using the FREEVIS macro. If the GETVIS for the first LRCB, or the SUCB/LUCB, or the WACB fails, any previously obtained virtual storage for LOGON control blocks has to be freed, using the FREEVIS macro. If for example the GETVIS for the LOGON WACB fails, the previously obtained storage for the LOGON/SUCB/LUCB and the LRCB are freed, using the FREEVIS macro. If for a subsequent LRCB the GETVIS fails, the storage for LOGON SUCB/LUCB and WACB will not be freed, using the FREEVIS macro. (03) The LOGON request has to be rejected. A RPL and a NIB are build to issue a CLSDST request to VTAM. (04) The message text for message 1V06I is copied from the message definition module IPW$$MD into a message area (MSGBUF) within IPW$$VE. Then the copied message is modified with the LUNAME of the failing session, and the reason code for the failure. (05) The CCW and CCB, generated at assembly time, are relocated and contain the length of the message. The CCB points to the CCW, the CCW points to the message text.			

CHART NA: IPW$$DD - FILE DUMP PROGRAM (14 PARTS)

Chart NA00: IPW$$DD - File Dump Program, General Flow and Macro Calls



Labels	Chart NA01: IPW$$DD - File Dump Program		Modified Data Fields	Reg. Usage	Calls
	This program is executed when a // EXEC IPW$$DD statement is entered on the console.				
	Register usage:				
	0: Service work register			R0	
	1: Service work register			R1	
	2: Service work register			R2	
	3: Service work register			R3	
	4: Service work register			R4	
	5: Pointer to I/O buffer			R5	
	6: Service work register			R6	
	7: Service work register			R7	
	8: Service work register			R8	
	9: Service work register			R9	
	10: Pointer to I/O buffer			R10	
	11: Return address to caller			R11	
	12: Second base register			R12	
	13: Not used			R13	
	14: Return address to caller			R14	
	15: Function base register			R15	
	Establish addressability, Issue Prompting Message, and Check Reply				
DD00	Set up base register.			R15	
DD06	Set up second base register.			R12	
	Relocate CCBs and CCWs using register 1 as a work register.			R1	
DD08	Load the address of the communications region into register 1 to get DOS/VS line count.		DOSLCT	R1	COMRG
DD10	Blank the reply area. Load the address of the CCB in register 1.		INREP	R1	
	Write message DUMP FUNCTION= and wait for reply.				WAIT
	Clear work registers.			R3-R5 R8,R9	
	If blank reply, branch to..... >	DD50			
DD12	Check the bytes of the reply area. If the byte contains a comma. branch to..... >	DD16			
	If the byte contains a blank, branch to..... >	DD20			
DD14	Address the next byte of the reply. If 18 bytes have been read, branch to..... >	DD50			
	Otherwise, branch to..... >	DD12			

Labels	Chart NA02: IPW$$DD - File Dump Program		Modified Data Fields	Reg. Usage	Calls
DD16	If the comma is not the first one in the reply, branch to..... >	DD18			
	If it is the first comma, store its address in register 3 and branch to >	DD14		R3	
DD18	If the comma is not the second one, the reply is invalid, branch to.... >	DD50			
	If it is the second comma, store its address in register 4 and branch to >	DD14		R4	
DD20	Store the address of the blank character in register 9.			R9	
	If register 3 contains the address of the first comma, branch to..... >	DD30			
	If the length of the reply is not 1 byte, branch to..... >	DD22			
	Move the contents of the reply area (INREP) into the dump function field.		FDMP		
	If the reply is D (dump of the data file requested), branch to..... >	DD58			
	If the reply is A (dump of the account file requested), branch to. >	DD72			
	If the reply is Q (dump of the queue file requested), branch to..... >	DD52			
	If the character is not D, A, or Q, it must be a job name.				
	Set the job name length field to 1.		JBNL		
	Reset the dump function to blank.		FDMP		
	Branch to..... >	DD52			
DD22	If the reply is not EOJ, branch to. >	DD24			
	Otherwise, terminate the job.				SVC 14
DD24	Establish the job name.				
DD26	If a blank character is found, branch to..... >	DD28			
DD28	If the job name is longer than 8 characters, branch to..... >	DD46			
	Otherwise, save the length of the job name.		JBNL		
	Branch to..... >	DD52			

Labels	Chart NA03: IPW$$DD - File Dump Program		Modified Data Fields	Reg. Usage	Calls
DD30	Establish the length of the job name. If it is more than 8 characters, branch to..... >	DD46			
	Otherwise, store the length of the job name.		JBNL		
	If the reply does not contain a second comma, branch to..... >	DD38			
	Otherwise, calculate the length of the job number.				
	If the job number is longer than 6 characters, branch to..... >	DD44			
DD32	If no job number is specified, branch to..... >	DD40			
DD34	If the specified job number is invalid, branch to..... >	DD44			
	Otherwise, save the job number and branch to..... >	DD40	JBNR		
DD38	Establish the length of the job number and branch to..... >	DD32			
DD40	If no queue option is specified in the reply, branch to..... >	DD52			
	If the queue option specified is not L, P, or R (for list, punch, or read queue), branch to..... >	DD42			
	Otherwise, branch to..... >	DD48			
DD42	Save the specified queue option and branch to..... >	DD52	JOBFCT		
DD44	Move message INVALID JOBNUMBER into the message area.		MSGAR		
	To print the message on the console, branch and link to..... >	MSGPRTC			
	Clear the message area and branch to..... >	DD10	MSGAR		
DD46	Move message INVALID JOBNAME into the message area.		MSGAR		
	To print the message on the console, branch and link to..... >	MSGPRTC			
	Clear the message area and branch to..... >	DD10	MSGAR		
DD48	Move message INVALID THIRD OPERAND into the message area.		MSGAR		
	To print the message on the console, branch and link to..... >	MSGPRTC			

Labels	Chart NA04: IPW$$DD - File Dump Program		Modified Data Fields	Reg. Usage	Calls
	Clear the message area and branch to..... >	DD10	MSGAR		
DD50	Move message INVALID REPLY into the message area.		MSGAR		
	To print the message on the console, branch and link to..... >	MSGPRTC			
	Clear the message area and branch to..... >	DD10	MSGAR		
	Open the Queue File				
DD52	Set up register 4 as base register for the queue file (IJQFILE).			R4	
	For setting up the DTFPH, branch and link to..... >	DD62			
	Establish addressability for the module control block and the disk management block in registers 6 and 7			R6,R7	
	Open the queue file.				OPENR
	Move the disk address from the MCB to the seek address of the queue file and set the record number to 1.		QCCHH QR		
	If only a dump of the queue file is required, branch to..... >	DD300			
	Branch and link to read a master queue record..... >	DD700			
	Establish addressability of the master record in register 5.			R5	
	Using registers 2 and 3 as work registers, calculate the number of track groups per cylinder to initialize the translate table.			R2,R3	
	Branch to open the data file..... >	DD200			
	Open the Data File				
DD58	Set up register 4 as a base register for the data file (IJDFILE).			R4	
	For setting up the DTFPH, branch and link to..... >	DD62			
	Establish addressability for the MCB in register 6.			R6	
	Open the data file.				OPENR

Labels	Chart NA05: IPW$$DD - File Dump Program		Modified Data Fields	Reg. Usage	Calls
	Move the address from the MCB to the seek address of the data file and set the record number to 1.		DCCHH DR		
	Store the maximum block length in the MCB, using register 1 as a work register.		MCBL	R1	
	Switch off the SLI bit.		READ+4		
	For reading the first data block, branch and link to..... >	DD700			
	Switch on the SLI bit.		READ+4		
	Calculate the length of the data block and store it in the MCB, using register 3 as a work register.		MCBL	R3	
	Calculate the number of records per track, using registers 3, 8, and 9 as work registers.			R3,R8, R9	
DD60	Store the number of records per track in the MCB.		MCRT		
	If there are more MCBs to be filled, branch to..... >	DD60			
	If only a dump of the data file is requested (D specified), branch to. >	DD400			
	Otherwise, branch to..... >	DD500			
	Set Up the DTFPH				
DD62	Load the address of the communications region to register 1			R1	COMRG
	Find displacements in PUB and LUB table for the specified logical unit in the DTFPH.			R6,R7, R4	
	If no logical unit is assigned, branch to..... >	DD66			
	Get device table address.			R2	
	Get number of entries.			R3	
DD64	Check the device table for a matching entry, using registers 2 and 3 as work registers.			R2,R3	
	If a matching logical unit is found, branch to..... >	DD68			
DD66	Issue message INVALID LOGICAL UNIT and branch to..... >	DD900			
DD68	Insert the DTF device type code and return.		PHDT		

Labels	Chart NA06: IPW$$DD - File Dump Program		Modified Data Fields	Reg. Usage	Calls
DD70	Move lower extents and number of tracks to MCBs.		MCLO,MCDF,MCLU, MCNT (IPW$DMC)	R1,R7, R6,R5, R4	
	Store DTFPH pointer in MCBs, move logical unit number to MCB and return to open.				
	Open the Account File				
DD150	Set up register 4 as a base register for the account file (IJAFILE).			R4	
	For setting up the DTFPH, branch and link to..... >	DD62			
	Establish addressability for the MCB in register 6.			R6	
	Open the account file.				OPENR
	Move the disk address from the MCB to the seek address of the account file and set the record number to 1. Store the maximum block length in the MCB, using register 1 as a work register.		ACCHH AR MCBL	R1	
	Read the Account File				
	To repeat message DUMP FUNCTION= reply on SYSLST and to start a new page on SYSLST, branch and link to. >	SKIP			
DD74	For reading the next record, branch and link to..... >	DD700			
	Calculate the block length, using register 3 as work register.			R3	
	If no data was read in, branch to.. >	DD900			
DD285	To format and print the account record, branch and link to..... >	DSKDEBL			
	To get the address of the next record, branch and link to..... >	DD600		R1	
	If all records have been read, unchain read count and branch to... >	DD900			
	Otherwise, branch to..... >	DD74			
	Read the Queue File				
DD300	To repeat message DUMP FUNCTION= reply and to start a new page on SYSLST, branch and link to..... >	SKIP			
DD310	Set up register 5 as base register for the queue file.			R5	
	To read the next queue record, branch and link to..... >	DD700			
	To format and print the record, branch and link to..... >	DSKDEBL			

Labels	Chart NA07: IPW$$DD - File Dump Program		Modified Data Fields	Reg. Usage	Calls
	If all records are read, branch to. >	DD900			
	To get the address of the next record, branch and link to..... >	DD600			
	If there are more records to be read, branch to..... >	DD310			
	Otherwise, branch to..... >	DD900			
	Read the Data File				
DD400	To start a new page on SYSLST, branch and link to..... >	SKIP			
DD410	Set up register 5 as base register.			R5	
	To check whether the record contains data, branch and link to..... >	TESTZERO			
	To format and print the record, branch and link to..... >	DSKDEBL			
	To get the address of the next record, branch and link to..... >	DD600			
	If the block length is not exceeded, branch to..... >	DD460			
	Otherwise, update the model ID, using register 1 as work register.		DMI	R1	
	Get the address of the next MCB, using register 6 as work register.			R6	
	If the next MCB is not available, branch to..... >	DD900			
	Move the disk address from the MCB to the seek address of the data file, and set the record number to 1.		DCCHH DR		
DD460	Load register 1 with the seek address of the data file.			R1	
	To read the next block, branch and link to..... >	DD700			
	To continue the dump, branch to.... >	DD410			
	Job Name Specified: Read Queue and Data File				
DD500	Set up register 5 as base register.			R5	
	Indicate that first record has to be printed.		SKIPSW		
	Blank out record-found indicator.		SAVCCHHR		
	Branch to..... >	DD513			

Labels	Chart NA08: IPW$$DD - File Dump Program		Modified Data Fields	Reg. Usage	Calls
DD505	Load the seek address of the queue file in register 2.			R2	
	Load the address of the queue file MCB into register 6 for use as a base register.			R6	
	To get the address of the next record, branch and link to..... >	DD600			
	If the end of the extent has not been reached, branch to..... >	DD515			
	If no record is found, branch to... >	DD545			
	Branch to..... >	DD900			
DD513	Load the address of the queue file MCB in register 6 for use as base register.			R6	
	Move the disk address from the MCB to the seek address of the queue file and set the record number to 1.		QCCHH QR		
DD515	For reading the next record, branch and link to..... >	DD700			
	Establish the length of the job name and check whether it matches the name specified, using register 8 as work register.			R8	
	If the job name of the record does not match, branch to..... >	DD505			
	Check whether a job number was specified in the request, using register 1 as work register.			R1	
	If no job number was specified, branch to..... >	DD520			
	If the job number of the record does not match the number specified, branch to..... >	DD505			
DD520	If the record identifier does not match the requested option (L, P, or R), branch to..... >	DD505			
	If the record is not the first of a queue set, branch to..... >	DD505			
	Otherwise, save the seek address of this queue set and branch to..... >	DD529	SAVCCHHR		
CNAME	Compare the job name in the record with the name specified in the request.				

Labels	Chart NA09: IPW$$DD - File Dump Program		Modified Data Fields	Reg. Usage	Calls
DD529	If any record has been printed already, branch to..... >	DD530			
	Otherwise, to start a new page on SYSLST, branch and link to..... >	SKIP			
DD530	For formatting and printing the queue record, branch and link to..... >	DSKDEBL			
	Move the address of the first data block from the DMB.		DSKADDR+1		
	Move the address of the next record in the queue set from the DMB.		QSKADDR+1		
DD540	Set up register 5 as base register.			R5	
	Load the address of the data file in register 1.			R1	
	To read a data record, branch and link to..... >	DD700			
	Check if the first two bytes are zeros. If not, branch to..... >	DD542			
	Otherwise, to check if all bytes in the record are zeros, branch and link to..... >	TESTZERO			
DD542	To format and print the record, branch and link to..... >	DSKBEBL			
	Load the address of the data file in register 2.			R2	
	To get the address of the next data record, branch and link to..... >	DD600			
	If the end of the trackgroup has not been reached, branch to..... >	DD540			
	If the last trackgroup has been read, branch to..... >	DD544			
	Load the seek address of the queue file in register 1.			R1	
	To read the next queue record in this queue set, branch and link to..... >	DD700			
	Branch to..... >	DD530			
DD544	Get the saved seek address of the queue set currently processed.		QCCHH		
	Branch to..... >	DD505			
DD545	Move message JOB ENTRY NOT FOUND into the message area.		MSGAR		
	To print the message on the console, branch and link to..... >	MSGPRTC			

Labels	Chart NA10: IPW$$DD - File Dump Program		Modified Data Fields	Reg. Usage	Calls
	Clear the message area and branch to..... >	DD900	MSGAR		
	Update Seek Address				
DD600	Calculate the address of the next record and store it in register 1.			R1	
	If the account file is not the file to be dumped, branch to..... >	DD610			
	If the record count is not 1, return to the calling routine.				
	If the record count is 1, branch to..... >	DD615			
DD610	If the next record is on the same track, return to the calling routine.				
DD615	Update the track number in register 1.			R1	
	Set the record number to 1.				
	If it is the QFILE, branch to..... >	DD618			
	If 'jobname,jobnumber' is specified in the request, branch to..... >	DD640			
DD618	If the next track is in the samecylinder, branch to..... >	DD620			
	Set the track number to 1.				
	Calculate the next cylinder number and store it in register 1.			R1	
DD620	If the cylinder number does not exceed the range of the extent, return to the calling routine.				
	If the extent is exceeded, clear register 1 and return to the calling routine.			R1	
DD640	If the track number is within the track group, return to the calling routine.				
	If the track group is exceeded, clear register 1 and return to the calling routine.			R1	

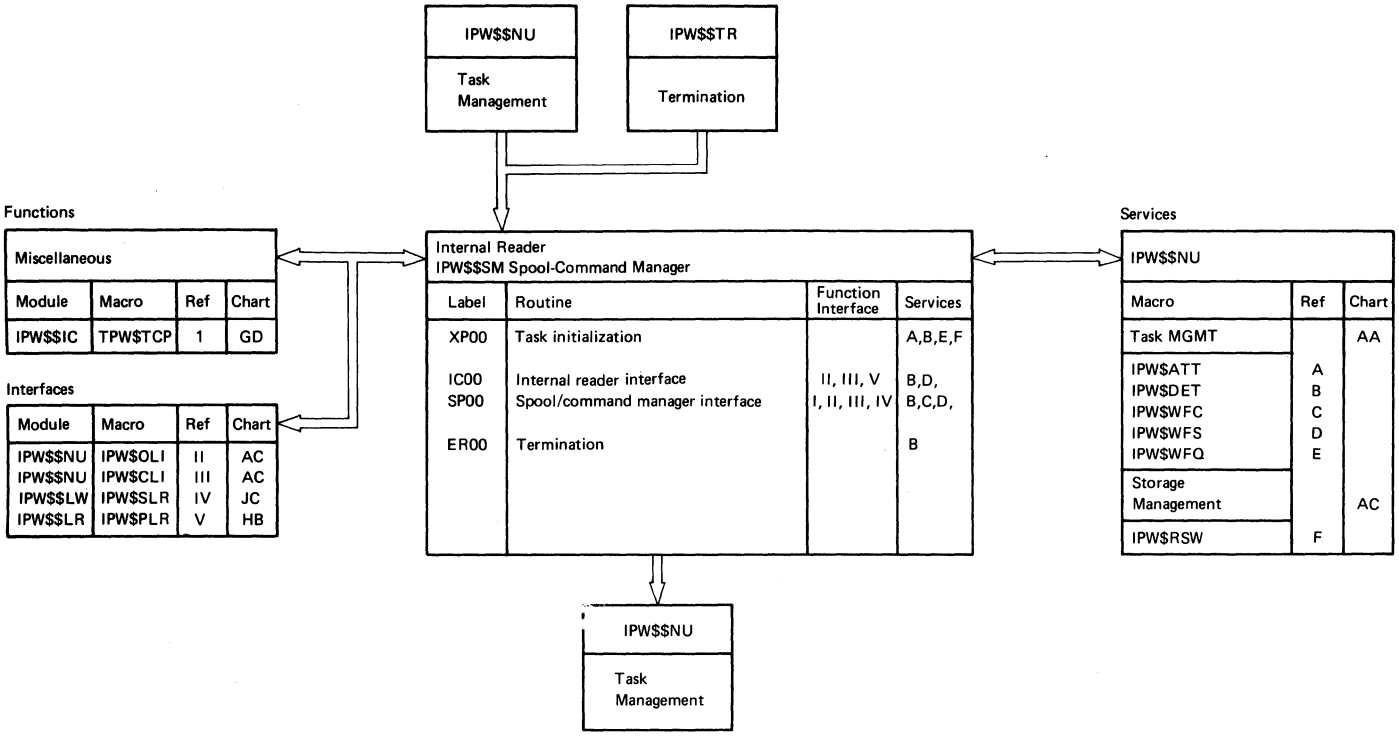
Labels	Chart NA11: IPW$$DD - File Dump Program		Modified Data Fields	Reg. Usage	Calls
	Read a Record from Disk				
DD700	Get the seek address of the file.		SKADR		
	Load the MCB address in register 6.			R6	
	Get the CCB pointer and set up register 1 as base register.			R1	
	Move the logical unit number from the MCB to the CCB.		PHCB+6		
	Move the block length from the MCB to the CCW.		READ+6		
	Read a record and wait.				WAIT
	Load the block length in register 3 and save it.		R7SAV	R3	
	Return to the calling routine.				
	Close All Files				
DD900	Load the address of the queue file DTFPH in register 4.			R4	
	To close the queue file, branch and link to..... >	DD990			
	Load the address of the data file DTFPH in register 4.			R4	
	To close the data file, branch and link to..... >	DD990			
	Load the address of the account file in register 4.			R4	
	To close the account file, branch and link to..... >	DD990			
	To repeat the prompting message, branch to..... >	DD003			
DD990	If the file is not open, return to the calling routine.				
	Otherwise, close the file and return to the calling routine.				$$BCLOSE
	Prepare Headings for the Listing				
DSKDEBL	Set up addressability.			R10	
	Load the cylinder number into register 7.			R7	
	To convert it to decimal, branch and link to..... >	DECCVT			
	Store the converted cylinder number.		DCYL+5		

Labels	Chart NA12: IPW$$DD - File Dump Program		Modified Data Fields	Reg. Usage	Calls
	Load the track number into register 7.			R7	
	To convert it to decimal, branch and link to..... >	DECCVT			
	Store the converted track number.		DHED+6		
	Load the record number into register 7.			R7	
	To convert it to decimal, branch and link to..... >	DECCVT			
	Store the converted record number.		DREC+4		
	Load the block length into register 7.			R7	
	To convert it to decimal, branch and link to..... >	DECCVT			
	Store the converted block length.		DATH+4		
	Move cylinder number, track number, record number, and block length into the heading of the first record.		HDG1, HDG2, HDG3		
	Load the block length into register 3.			R3	
	To set up print length, branch and link to..... >	DEBLOK			
	To print a data record, branch and link to..... >	PRNT			
	Return to the calling routine.				
	Convert to Printable Characters				
DECCVT	Convert binary numbers to decimal, using registers 0 and 1 as work registers.			R0,R1	
	Return to the calling routine when finished.				
	Set up Correct Print Length				
DEBLOK	Load the maximum print length (100 characters) into register 4.			R4	
	If the record does not contain all zeros, branch to..... >	DB1A			
	Otherwise, load the maximum print length in register 3, and move the message THIS RECORD CONTAINS BINARY ZEROS ON DISK into the print area.				

Labels	Chart NA13: IPW$$DD - File Dump Program		Modified Data Fields	Reg. Usage	Calls
DB1A	If the record length is not greater than 100, branch to ..... >	LSBLK			
DB2	Otherwise, to convert the binary record, branch and link to..... >	DECCVT			
	To print the record, branch and link to..... >	HEXDP			
	Clear the heading area.		HDG1		
	Increment the input pointer.				R10
	Branch to..... >	DB1			
LSBLK	If the record length is zero, return to the calling routine.				
	Otherwise, load the length of the remaining record in register 4 and clear register 3.			R4 R3	
	Branch to..... >	DB2			

CHART PA: IPW$$SM - SPOOL MANAGER

Chart PA00: IPW$$SM - Spool Manager, General Flow and Macro Calls



Labels	Chart PA01: IPW$$SM - Spool Manager		Modified Data Fields	Reg. Usage	Calls
	This routine is entered from task selection. Normal register usage.				
SMCS	The first 16 bytes constitute the storage descriptor: 'SMCS V7M0'				
	TASK ACTIVATION				
SM05	Spool/command manager request (GETSPOOL/CTLSPOOL)?				
	If yes..... >	SP00			
	Internal reader request (PUTSPOOL)? If yes..... >	IC00			
XP00	Determine which action to take:				
	• If TCB's already initialized, return to task selection..... >	XP70			
	• Construct internal reader or spool/command manager TCB				
XP50	Reserve space for TCB.			R0,R1	IPW$RSW Chart AC
	Initialize task control block:				
	• Descriptor		TCSD(IPW$DTC)		
	• Class		TCCT(IPW$DTC)		
	• Task save area		TCRD(IPW$DTC)		
	• TCB address		TCSV(IPW$DTC)		
	• Spool parameter list address		TCPL(IPW$DTC)		
	• Job name		TCJN(IPW$DTC)		
	• Task identifier		TCTI(IPW$DTC)		
	• Task CUU		TCCU(IPW$DTC)		
	• Error exit routine address		TCXA(IPW$DTC)		
	Register 3 points to SM05 where task is to receive control.				
XP60	Attach new task.			R1	IPW$ATT Chart AA

Labels	Chart PA02: IPW$$SM - Spool Manager		Modified Data Fields	Reg. Usage	Calls
XP70	Return here after attaching new task.				
	POWER/VS terminating?				
	If yes..... >	XP80			
	Return master spool management to task selection.				
	• If POWER/VS initialization task: wait on both POWER/VS cross-partition XECB's.				
	• If internal reader task: wait on spool/command manager XECB to create TCB.				
	• If spool/command manager task: wait on internal reader XECB to create TCB.			R1,R11	IPW$WFQ Chart AA
XP80	Detach master spool task.			R11	IPW$DET Chart AA
	Internal Reader Interface Subroutine to Process PUTSPOOL Request				
IC00	Open interface to logical reader.				IPW$OLI Chart AC
	Validate SPL addresses if not previously checked. If invalid SPL address..... >	IC90			
	If invalid addr/size on job boundary>	IC45			
	If invalid address/size not on job boundary..... >	IC40			
IC05	Validate SPL parameters:				
	If POWER is shutting down..... >	IC90			
	Validate job name:				
	If invalid..... >	IC90			
	Continuing same job..... >	IC25			
	Validate class and disposition.		TCJN(IPW$DTC)		
	If invalid..... >	IC90			
IC08	Supply JECL statements if not user-supplied.			R0,R1	IPW$PLR Chart HB

Labels	Chart PA03: IPW$$SM - Spool Manager		Modified Data Fields	Reg. Usage	Calls
IC30	Pass user record from user's buffer area to POWER/VS RDR queue across logical reader interface.			R0,R1	IPW$PLR Chart HB
	Error during logical reader processing? If yes, error return..... >	IC38			
	End of user's buffer chain? If no..... >	IC30			
IC35	End of PUTSPOOL job? If no..... >	IC46			
IC40	Supply POWER/VS EOJ card if not given.			R0,R1	IPW$PLR Chart HB
	Supply DOS/VS EOJ card if LR still not on job boundary.				
IC45	Set job name to blanks to indicate PUTSPOOL job boundary.		TCJN(IPW$DTC)		
IC46	Release task and return any errors.		TCPL(IPW$DTC)		
	Errors or end of job? If no..... >	IC50			
	If LR is not on a job boundary then pass a POWER or DOS EOJ card to logical reader.				
	Close logical reader interface.				IPW$CLI Chart AC
IC50	XPOST the cross-partition XECB of the caller that issued the PUTSPOOL.			R0,R1 R14,R15	XPOST
	Address POWER/VS cross-partition XECB and return to task selection.			R1	IPW$WFS Chart AA
	Return here from task selection; determine if user still active: Active? If yes..... >	IC70		R0,R1, R14,R15	XECBTAB
	Previous job complete? If yes..... >	IC55			
	Close-up logical reader job.				IPW$PLR Chart HB

Labels	Chart PA04: IPW$$SM - Spool Manager		Modified Data Fields	Reg. Usage	Calls
IC55	Close logical reader interface update wait list for task selection.		ICWL(IPW$DPA)		IPW$CLI Chart AC
IC60	Detach task.			R11	IPW$DET Chart AA
IC70	Initialize error return, unpost XECB's, and clear user's PIK/TIK in DOS/VS XECBTAB.		ICXP(IPW$DPA)		
	Validate SPL addresses to be referenced:				
	If invalid SPL address..... >	IC90			
	If invalid address/size on job boundary..... >	IC45			
	If invalid address/size not on job boundary..... >	IC40			
	Beginning of new job?				
	If yes..... >	IC00			
	Send new internal reader/writer job >	IC05			
	Complete previous job and close logical reader interface..... >	IC00		R0,R1	IPW$PLR Chart HB IPW$CLI Chart AC
	Spool/Command Manager Interface Subroutine to Process GETSPOOL/CTLSPOOL Requests				
SP00	Open interface to logical writer... >	SP57			IPW$OLI Chart AC
SP05	Valid class given on GETSPOOL?				
	If no, error return..... >	SP30			
	Construct class list valid job name if request for a new job.		TCCT(IPW$DTC)		
	If no, error return..... >	SP40			
	Position on line number request?				
	If no..... >	SP15			
	Establish relative line number for positioning.		TCRS(IPW$DTC)		
SP15	Set logical writer switch.		TCSS(IPW$DTC)		

Labels	Chart PA05: IPW$$SM - Spool Manager		Modified Data Fields	Reg. Usage	Calls
SP23	Go to logical writer to retrieve a list record.				IPW$GLR Chart JC
	Error during logical writer processing? If yes..... >	SP30			
	End of list output job? If yes..... >	SP33		R1	
	End of list output data? If yes..... >	SP23		R0	
	Output data record? If yes..... >	SP27			
	Does user want non-zero control characters? If no..... >	SP23			
SP27	Is buffer area large enough? If yes..... >	SP35			
SP30	Close logical writer.		TCSS(IPW$DTC)	R0,R1	IPW$GLR Chart JC
SP33	Indicate logical writer job boundary. Error return..... >	SP40	TCJN(IPW$DTC)		
SP35	Return print record and/or non-zero print control character to user; return a buffer of printable blanks if only print control character is returned.				
SP40	Release task and return any errors.		TCPL(IPW$DTC)		
	XPOST the cross-partition XECB of the caller that issued the GETSPOOL or CTLSPOOL.			R0,R1 R14,R15	XPOST
	Address POWER/VS cross-partition XECB and return to task selection.			R1	IPW$WFS Chart AA
	Return here from task selection; determine if user still active: If active..... >	SP55		R0,R1, R15	XECBTAB

Labels	Chart PA06: IPW$$SM - Spool Manager		Modified Data Fields	Reg. Usage	Calls
	Previous job complete?				
	If yes..... >	SP47			
	Close-up logical writer.				IPW$GLR Chart JC
SP47	Close logical writer interface.				IPW$OLI Chart AC
	Update wait list for task selection.		SPWL(IPW$DPA)		
SP50	Detach task.				IPW$DET Chart AA
SP55	Initialize error return.		SMXP(IPW$DPA)		
SP57	Unpost XECB's, and clear user's PIK/TIK in DOS/VS XECBTAB.				
SP58	Validate SPL addresses, buffer length and address to be referenced.				
	If invalid, error return..... >	SP30			
	POWER/VS terminating?				
	If yes..... >	SP40			
	Load only request? If yes..... >	SP05			
	GETSPOOL request in-process?				
	If yes..... >	SP62			
	Close logical writer job.				IPW$GLR Chart JC
	Process CTLSPOOL Request				
SP62	User supplied own command?				
	If no..... >	SP63			
SP64	Check if command is supported.				
	If yes..... >	SP85			
	If no, error return..... >	SP40	SPER		
SP63	Validate job name in SPL:				
	If invalid, error return..... >	SP40			
	Construct POWER/VS command for CTLSPOOL in user-supplied buffer according to the CTLSPOOL request.				

Labels	Chart PA07: IPW$$SM - Spool Manager		Modified Data Fields	Reg. Usage	Calls
SP85	Pass POWER/VS command to the command processor by invoking the auxiliary command function.			R0,R1	IPW$ICP Chart GD
	Wait for command processing to complete.			R1	IPW$WFC Chart AA
	User supplied own command?				
	If yes, return..... >	SP40			
	Status request?				
	If yes..... >	SP90			
	Command successful?				
	If yes..... >	SP40			
	Return with error message in buffer..... >	SP40			
SP90	Job found?				
	If yes..... >	SP40			
	LST queue just scanned:				
	Not found on LST nor RDR queue, error return..... >	SP40			
	Go scan RDR queue for job..... >	SP63			
	Internal Reader Task Error Exit				
	The following error exit subroutines receive control from POWER/VS termination (IPW$$TR) or when POWER's XECB is XPOSTED by the DOS/VS Supervisor's clean-up function as determined by several checks throughout the IPW$$SM phase.				
ER00	Check if PUTSPOOL user is active:			R0,R1 R14,R15	XECBTAB
	If not..... >	ER20			
	XPOST user's cross-partition XECB.			R0,R1 R14,R15	XPOST

Labels	Chart PA08: IPW$$SM - Spool Manager		Modified Data Fields	Reg. Usage	Calls
ER20	Clear POWER/VS cross-partition XECB, update wait list for task selection, and detach internal reader task. Spool/Command Manager Task Error Exit		ICXP(IPW$DPA) ICWL(IPW$DPA)	R11	IPW$DET Chart AA
ER50	Check if GETSPOOL/CTLSPOOL user is active: If not..... >	ER70		R0,R1 R14,15	XECBTAB
	XPOST user's cross-partition XECB.			R0,R1 R14,R15	XPOST
ER70	Clear POWER/VS cross-partition XECB, update wait list for task selection, and detach spool/command manager LST task.		SMXP(IPW$DPA) SPWL(IPW$DPA)	R11	IPW$DET Chart AA

DOS/VS POWER/VS Logic Part 2 Printed in U.S.A. SY33-8577-1

IBM

IBM World Trade Corporation
821 United Nations Plaza
New York, New York 10017
U.S.A.

READER'S COMMENT FORM

DOS/VS POWER/VS
Logic Part 2

SY33-8577-1

Please comment on the usefulness and readability of this publication, suggest additions and deletions, and list specific errors and omissions (give page numbers). All comments and suggestions become the property of IBM. If you wish a reply, be sure to include your name and address.

COMMENTS

THANK YOU FOR YOUR COOPERATION
PLEASE FOLD ON TWO LINES, STAPLE AND MAIL

YOUR COMMENTS, PLEASE

Your comments on the other side of this form will help us improve future editions of this publication. Each reply will be carefully reviewed by the persons and department responsible for writing and publishing this material.

Please note that requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or the IBM branch office serving your locality.

IBM Germany
Publications Department
Schwertstrasse 58

D-7032 Sindelfingen

Germany

DOS/VS POWER/VS Logic Part 2 Printed in U.S.A. SY33-8577-1

IBM

IBM World Trade Corporation
821 United Nations Plaza
New York, New York 10017
U.S.A.